aws

Panduan Pengguna

Amazon ECR

Versi API 2015-09-21

Amazon ECR: Panduan Pengguna

Table of Contents

Apa Itu Amazon Elastic Container Registry?

Amazon Elastic Container Registry (Amazon ECR) adalah AWS layanan registri gambar kontainer terkelola yang aman, terukur, dan andal. Amazon ECR mendukung repositori pribadi dengan izin berbasis sumber daya menggunakan IAM. AWS Hal ini agar pengguna tertentu atau instans Amazon EC2 dapat mengakses repositori kontainer dan citra. Anda dapat menggunakan CLI pilihan Anda untuk mendorong, menarik, dan mengelola citra Docker, citra Open Container Initiative (OCI), dan artefak yang kompatibel dengan OCI.

> **Note**
>
> Amazon ECR juga mendukung repositori citra kontainer publik. Untuk informasi selengkapnya, lihat Apa itu Amazon ECR Public di Panduan Pengguna Amazon ECR Publik.

Tim layanan AWS kontainer memelihara peta jalan publik di GitHub. Ini berisi informasi tentang apa yang sedang dikerjakan tim dan memungkinkan semua AWS pelanggan kemampuan untuk memberikan umpan balik langsung. Untuk informasi selengkapnya, lihat AWS Panduan (roadmap) Kontainer.

Konsep dan komponen Amazon ECR

Amazon ECR adalah layanan registri kontainer Docker yang dikelola sepenuhnya yang disediakan oleh. AWS Ini memungkinkan Anda untuk menyimpan, mengelola, dan menyebarkan gambar kontainer Docker dengan aman dan andal. Konsep dan komponen ini bekerja sama untuk menyediakan layanan registri kontainer Docker yang aman, terukur, dan andal di dalamnya AWS, memungkinkan Anda mengelola dan menerapkan aplikasi kontainer Anda secara efisien.

Berikut adalah beberapa konsep dan komponen kunci Amazon ECR:

Registri

Registri Amazon ECR adalah repositori pribadi yang disediakan untuk setiap AWS akun, tempat Anda dapat membuat satu atau lebih repositori. Repositori ini memungkinkan Anda untuk menyimpan dan mendistribusikan gambar Docker, gambar Open Container Initiative (OCI), dan OCI-compatible artefak lainnya di lingkungan Anda. AWS Untuk informasi selengkapnya, lihat Registri pribadi Amazon ECR.

Token otorisasi

Klien Anda harus mengautentikasi ke registri pribadi Amazon ECR sebagai AWS pengguna sebelum dapat mendorong dan menarik gambar. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

Repositori

Repositori di Amazon ECR adalah koleksi logis tempat Anda dapat menyimpan gambar Docker, gambar Open Container Initiative (OCI), dan artefak lainnya. OCI-compatible Dalam satu registri Amazon ECR, Anda dapat memiliki beberapa repositori untuk mengatur gambar kontainer Anda. Untuk informasi selengkapnya, lihat Repositori pribadi Amazon ECR.

Kebijakan repositori

Anda dapat mengontrol akses ke repositori Anda dan konten di dalamnya dengan kebijakan repositori. Untuk informasi selengkapnya, lihat Kebijakan repositori pribadi di Amazon ECR.

Gambar

Anda dapat mendorong dan menarik citra kontainer ke repositori Anda. Anda dapat menggunakan citra ini secara lokal pada sistem pengembangan Anda, atau Anda dapat menggunakannya dalam definisi tugas Amazon ECS dan spesifikasi pod Amazon EKS. Untuk informasi selengkapnya, lihat Menggunakan citra Amazon ECR dengan Amazon ECS dan Menggunakan Citra Amazon ECR dengan Amazon EKS.

Kebijakan Siklus Hidup

Kebijakan siklus hidup Amazon ECR memungkinkan Anda mengelola siklus hidup gambar dengan menetapkan aturan pemangkasan dan kedaluwarsa gambar lama atau yang tidak digunakan. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.

Pemindaian Gambar

Amazon ECR menyediakan kemampuan pemindaian gambar terintegrasi yang membantu mengidentifikasi kerentanan perangkat lunak dalam gambar kontainer Anda. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.

Kontrol Akses

Amazon ECR menggunakan IAM untuk mengontrol akses ke repositori Anda. Anda dapat membuat pengguna, grup, dan peran IAM dengan izin khusus untuk mendorong, menarik, atau

mengelola repositori Amazon ECR. Untuk informasi selengkapnya, lihat Amazon Elastic Container Registry.

Cross-account dan Cross-region replikasi

Amazon ECR mendukung replikasi gambar di beberapa AWS akun dan wilayah untuk meningkatkan ketersediaan dan mengurangi latensi. Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR.

Enkripsi

Amazon ECR mendukung enkripsi sisi server dari gambar Docker Anda saat istirahat menggunakan. AWS KMS Untuk informasi selengkapnya, lihat Perlindungan data dalam Amazon ECR.

AWS Command Line Interface Integrasi

AWS CLI Ini menyediakan perintah untuk berinteraksi dengan repositori Amazon ECR, seperti membuat, mencantumkan, mendorong, dan menarik gambar.

Konsol Manajemen AWS

Amazon ECR juga dapat dikelola melalui Konsol Manajemen AWS, menyediakan antarmuka web yang ramah pengguna untuk bekerja dengan repositori dan gambar Anda.

AWS CloudTrail

Amazon ECR terintegrasi dengan AWS CloudTrail, memungkinkan Anda untuk mencatat dan mengaudit panggilan API yang dilakukan ke Amazon ECR untuk tujuan keamanan dan kepatuhan. Untuk informasi selengkapnya, lihat Mencatat tindakan Amazon ECR dengan AWS CloudTrail.

Amazon CloudWatch

Amazon ECR menyediakan metrik dan log yang dapat dipantau menggunakan Amazon CloudWatch, memungkinkan Anda melacak kinerja dan penggunaan repositori Amazon ECR Anda. Untuk informasi selengkapnya, lihat Metrik repositori Amazon ECR.

Penandatanganan terkelola

Penandatanganan terkelola secara otomatis menghasilkan tanda tangan kriptografi saat gambar didorong ke Amazon ECR, menyederhanakan penandatanganan gambar kontainer. Untuk informasi selengkapnya, lihat Penandatanganan terkelola.

Kasus penggunaan umum di Amazon ECR

Amazon ECR adalah layanan registri kontainer Docker yang dikelola sepenuhnya yang ditawarkan oleh. AWS Ini menyediakan repositori yang aman dan terukur untuk menyimpan dan mendistribusikan gambar kontainer Docker, menjadikannya komponen penting dalam penerapan aplikasi kontainer. Amazon ECR menyederhanakan proses membangun, mendistribusikan, dan menjalankan aplikasi kontainer di berbagai AWS layanan dan lingkungan lokal.

Berikut adalah beberapa kasus penggunaan utama untuk Amazon ECR:

Penyimpanan dan Distribusi Gambar Kontainer

Amazon ECR berfungsi sebagai repositori terpusat untuk menyimpan dan mendistribusikan gambar kontainer Docker dalam suatu organisasi atau untuk konsumsi publik. Pengembang dapat mendorong gambar kontainer mereka ke Amazon ECR dan kemudian menariknya dari lingkungan komputasi apa pun di dalamnya AWS, seperti Amazon EC2, AWS Fargate, atau Amazon EKS. Untuk informasi selengkapnya, lihat Repositori pribadi Amazon ECR.

Integrasi Berkelanjutan dan Penerapan Berkelanjutan () CI/CD

Amazon ECR terintegrasi secara mulus dengan AWS CodeBuild,, dan CI/CD alat lainnya AWS CodePipeline, memungkinkan pembuatan, pengujian, dan penyebaran aplikasi kontainer secara otomatis. Gambar kontainer dapat secara otomatis didorong ke Amazon ECR sebagai bagian dari CI/CD pipeline, memastikan penerapan yang konsisten dan andal di berbagai lingkungan.

Arsitektur Microservices

Amazon ECR sangat cocok untuk arsitektur layanan mikro, di mana aplikasi dipecah menjadi layanan terpisah yang lebih kecil yang dikemas sebagai wadah. Setiap layanan mikro dapat memiliki gambar kontainer sendiri yang disimpan di Amazon ECR, memungkinkan pengembangan, penyebaran, dan penskalaan layanan individual secara independen.

Hybrid dan Multi-Cloud Deployment

Amazon ECR mendukung kemampuan untuk menarik gambar kontainer dari pendaftar kontainer lain, seperti Docker Hub atau pendaftar pihak ketiga. Hal ini memungkinkan organisasi untuk mempertahankan model penerapan yang konsisten di seluruh lingkungan hybrid atau multi-cloud, menggunakan Amazon ECR sebagai repositori pusat untuk gambar kontainer.

Kontrol Akses dan Keamanan

Amazon ECR menyediakan mekanisme kontrol akses berbutir halus, memungkinkan organisasi untuk mengontrol siapa yang dapat mendorong atau menarik gambar kontainer dari registri.

Ini juga terintegrasi dengan AWS Identity and Access Management otentikasi dan otorisasi, memastikan akses aman ke gambar kontainer. Untuk informasi selengkapnya, lihat Amazon Elastic Container Registry.

Pemindaian Kerentanan Gambar

Amazon ECR menawarkan pemindaian otomatis gambar kontainer untuk kerentanan perangkat lunak dan potensi kesalahan konfigurasi, membantu menjaga lingkungan kontainer yang aman dan sesuai. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.

Registri Kontainer Pribadi

Untuk organisasi dengan persyaratan keamanan atau kepatuhan yang ketat, Amazon ECR dapat digunakan sebagai registri kontainer pribadi, memastikan bahwa gambar kontainer sensitif tidak terpapar ke pendaftar publik dan hanya dapat diakses dalam lingkungan organisasi. AWS Untuk informasi selengkapnya, lihat Registri pribadi Amazon ECR.

Penyebaran Aplikasi Terdistribusi Secara Global dengan Replikasi Amazon ECR

Memanfaatkan kemampuan replikasi Amazon ECR, Anda dapat memusatkan gambar aplikasi web kontainer Anda di repositori utama, memungkinkan distribusi otomatis di beberapa AWS wilayah, memastikan penerapan global yang konsisten dengan latensi rendah di seluruh dunia dan mengurangi beban operasional. Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR

Pembersihan Otomatis Gambar Kontainer Basi

Kebijakan siklus hidup Amazon ECR memungkinkan pembersihan otomatis gambar kontainer basi berdasarkan aturan yang ditentukan seperti usia, jumlah, atau tag, mengoptimalkan biaya penyimpanan, memelihara registri yang terorganisir, meningkatkan keamanan dan kepatuhan, dan merampingkan alur kerja pengembangan melalui otomatisasi. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR

Fitur-Fitur Amazon ECR

Amazon ECR menyediakan fitur-fitur berikut:

- Kebijakan siklus hidup membantu mengelola siklus hidup citra di repositori Anda. Anda menentukan aturan yang mengakibatkan pembersihan citra yang tidak terpakai. Anda dapat

menguji aturan sebelum menerapkannya ke repositori Anda. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.

- Pemindaian citra membantu dalam mengidentifikasi kerentanan dalam citra kontainer Anda. Setiap repositori dapat dikonfigurasi untuk pindai saat mendorong. Hal ini memastikan bahwa setiap citra baru yang didorong ke repositori dipindai. Anda kemudian dapat mengambil hasil pemindaian citra. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.
- Cross-Region dan replikasi lintas akun memudahkan Anda untuk memiliki gambar di tempat yang Anda butuhkan. Ini dikonfigurasi sebagai pengaturan registri dan berdasarkan per Wilayah. Untuk informasi selengkapnya, lihat Pengaturan registri pribadi di Amazon ECR.
- Tarik melalui aturan cache menyediakan cara untuk menyimpan repositori di registri upstream di registri ECR Amazon pribadi Anda. Menggunakan aturan cache pull through, Amazon ECR akan secara berkala menjangkau registri hulu untuk memastikan gambar yang di-cache di registri pribadi Amazon ECR Anda mutakhir. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR.
- Template pembuatan repositori memungkinkan Anda menentukan pengaturan untuk repositori yang dibuat oleh Amazon ECR atas nama Anda selama pull through cache, create on push, atau tindakan replikasi. Anda dapat menentukan kekekalan tag, konfigurasi enkripsi, kebijakan repositori, kebijakan siklus hidup, dan tag sumber daya untuk repositori yang dibuat secara otomatis. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.
- Penandatanganan terkelola secara otomatis menghasilkan tanda tangan kriptografi saat gambar didorong ke Amazon ECR, menyederhanakan penandatanganan gambar kontainer. Untuk informasi selengkapnya, lihat Penandatanganan terkelola.

Mendaftar untuk Akun AWS

Untuk memulai AWS, Anda membutuhkan Akun AWS. Untuk informasi tentang membuat Akun AWS, lihat Memulai dengan Akun AWS di Panduan AWS Account Management Referensi.

Cara memulai dengan Amazon ECR

Jika Anda menggunakan Amazon Elastic Container Service (Amazon ECS) Service Elastic Container (Amazon ECS) atau Amazon Elastic Kubernetes Service (Amazon EKS), perhatikan bahwa

pengaturan untuk kedua layanan tersebut mirip dengan pengaturan untuk Amazon ECR karena Amazon ECR adalah perpanjangan dari kedua layanan tersebut.

Saat menggunakan AWS Command Line Interface dengan Amazon ECR, gunakan versi AWS CLI yang mendukung fitur Amazon ECR terbaru. Jika Anda tidak melihat dukungan untuk fitur Amazon ECR di AWS CLI, tingkatkan ke versi terbaru. AWS CLI Untuk informasi tentang menginstal versi terbaru AWS CLI, lihat Menginstal atau memperbarui ke versi terbaru dari AWS CLI dalam Panduan AWS Command Line Interface Pengguna.

Untuk mempelajari cara mendorong image container ke repositori Amazon ECR pribadi menggunakan AWS CLI dan Docker, lihat. Memindahkan gambar melalui siklus hidupnya di Amazon ECR

Harga untuk Amazon ECR

Dengan Amazon ECR, Anda membayar jumlah data yang Anda simpan di repositori, transfer data dari push dan pull gambar Anda, dan tindakan gambar yang Anda pilih seperti penandatanganan dan replikasi gambar. Untuk informasi selengkapnya, lihat Harga Amazon ECR.

Memindahkan gambar melalui siklus hidupnya di Amazon ECR

Jika Anda menggunakan Amazon ECR untuk pertama kalinya, gunakan langkah-langkah berikut dengan CLI Docker dan AWS CLI untuk membuat gambar sampel, mengautentikasi ke registri default, dan membuat repositori pribadi. Kemudian dorong gambar ke dan tarik gambar dari repositori pribadi. Setelah selesai dengan gambar sampel, hapus gambar sampel dan repositori.

Untuk menggunakan Konsol Manajemen AWS alih-alih AWS CLI, lihatthe section called “Membuat repositori untuk menyimpan gambar”.

Untuk informasi selengkapnya tentang alat lain yang tersedia untuk mengelola AWS sumber daya Anda, termasuk AWS SDK yang berbeda, toolkit IDE, dan alat baris PowerShell perintah Windows, lihat. http://aws.amazon.com/tools/

Prasyarat

Jika Anda tidak memiliki yang terbaru AWS CLI dan Docker diinstal dan siap digunakan, gunakan langkah-langkah berikut untuk menginstal kedua alat ini.

Instal AWS CLI

Untuk menggunakan AWS CLI dengan Amazon ECR, instal AWS CLI versi terbaru. Untuk selengkapnya, lihat Menginstal AWS Command Line Interface di Panduan AWS Command Line Interface Pengguna.

Instal Docker

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

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

Jika Anda sudah menginstal Docker, langsung ke Langkah 1: Buat citra Docker.

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

1. Luncurkan instance dengan AMI Amazon Linux 2023 terbaru. Untuk informasi selengkapnya, lihat Meluncurkan instance di Panduan Pengguna Amazon EC2.
2. Terhubung ke instans Anda. Untuk informasi selengkapnya, lihat Connect to Linux Instance Anda di Panduan Pengguna Amazon EC2.
3. Perbarui paket yang diinstal dan paket cache pada instans Anda.

   ```
   sudo yum update -y
   ```

4. Instal paket Edisi Komunitas Docker terbaru.

   ```
   sudo yum install docker
   ```

5. Mulai layanan Docker.

   ```
   sudo service docker start
   ```

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

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

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

   ```
   docker info
   ```

   > **Note**
   >
   > Dalam beberapa kasus, Anda mungkin perlu melakukan booting ulang pada instans Anda untuk memberikan izin bagi `ec2-user` untuk mengakses daemon Docker. Coba me-reboot instans Anda jika Anda melihat kesalahan berikut:

```
Cannot connect to the Docker daemon. Is the docker daemon running on this
 host?
```

Langkah 1: Buat citra Docker

Pada langkah ini, Anda membuat gambar Docker dari aplikasi web sederhana, dan mengujinya di sistem lokal Anda atau instans Amazon EC2.

Untuk membuat citra Docker dari aplikasi web sederhana

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

   ```
   touch Dockerfile
   ```

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

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

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

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

   # Configure apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
    echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh

   EXPOSE 80

   CMD /root/run_apache.sh
   ```

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

3. Membangun citra Docker dari Dockerfile Anda.

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

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

4. Buat daftar gambar kontainer Anda.

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

Output:

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

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

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

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

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

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

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

   Anda akan melihat halaman web dengan pernyataan “Hello World!” .

7. Hentikan kontainer Docker dengan mengetik Ctrl + c.

Langkah 2: Buat repositori

Kini Anda memiliki citra yang akan didorong ke Amazon ECR, Anda harus membuat repositori untuk menahannya. Dalam contoh ini, Anda membuat repositori yang disebut `hello-repository` yang kemudian Anda mendorong citra `hello-world:latest`. Untuk membuat repositori, jalankan perintah berikut:

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

Langkah 3: Otentikasi ke registri default Anda

Setelah Anda menginstal dan mengkonfigurasi AWS CLI, otentikasi CLI Docker ke registri default Anda. Dengan cara tersebut, perintah docker dapat mendorong dan menarik citra dengan Amazon ECR. AWS CLI Ini menyediakan get-login-password perintah untuk menyederhanakan proses otentikasi.

> **Note**
>
> Kepala IAM Anda harus memiliki `ecr:GetAuthorizationToken` izin untuk mengautentikasi ke registri. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Registry.

Untuk mengotentikasi Docker ke registri Amazon ECR dengan get-login-password, jalankan perintah aws ecr get-login-password. Ketika meneruskan token otorisasi ke perintah docker login, gunakan nilai `AWS` untuk nama pengguna dan tentukan URI registri Amazon ECR yang ingin Anda autentikasi. Jika melakukan autentikasi untuk beberapa registri, Anda harus mengulangi perintah tersebut untuk setiap registri.

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

- get-login-password(AWS CLI)

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

- Get-ECRLoginCommand (AWS Tools for Windows PowerShell)

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Langkah 4: Dorong citra ke Amazon ECR

Sekarang Anda dapat mendorong citra Anda ke repositori Amazon ECR yang Anda buat di bagian sebelumnya. Gunakan docker CLI untuk mendorong gambar setelah prasyarat berikut terpenuhi:

- Versi minimum diinstal: 1.7. docker
- Token otorisasi Amazon ECR telah dikonfigurasi dengan. docker login

- Amazon ECR repositori tersedia dan pengguna memiliki akses untuk mendorong ke repositori.

Setelah persyaratan tersebut terpenuhi, Anda dapat mendorong citra Anda ke repositori yang baru dibuat di registrasi default untuk akun Anda.

Untuk membuat tanda dan mendorong citra ke Amazon ECR

1. Cantumkan citra yang telah Anda simpan secara lokal untuk mengidentifikasi citra yang akan di-tanda dan didorong.

   ```
   docker images
   ```

   Output:

   ```
   REPOSITORY          TAG                 IMAGE ID            CREATED
    VIRTUAL SIZE
   hello-world         latest              e9ffedc8c286        4 minutes ago
    241MB
   ```

2. Tandai citra untuk mendorongnya ke repositori Anda.

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

3. Mendorong citra

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

   Output:

   ```
   The push refers to a repository [aws_account_id.dkr.ecr.region.amazonaws.com/hello-
   repository] (len: 1)
   e9ae3c220b23: Pushed
   a6785352b25c: Pushed
   0998bf8fb9e9: Pushed
   0a85502c06c9: Pushed
   latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
    size: 6774
   ```

Langkah 5: Menarik citra dari Amazon ECR

Setelah gambar Anda didorong ke repositori Amazon ECR Anda, Anda dapat menariknya dari lokasi lain. Gunakan docker CLI untuk menarik gambar setelah prasyarat berikut terpenuhi:

- Versi minimum diinstal: 1.7. docker
- Token otorisasi Amazon ECR telah dikonfigurasi dengan. docker login
- Amazon ECR repositori tersedia dan pengguna memiliki akses untuk melakukan penarikan dari repositori.

Setelah prasyarat tersebut terpenuhi, Anda bisa menarik citra Anda. Untuk menarik citra contoh Anda dari Amazon ECR, jalankan perintah berikut:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
```

Output:

```
latest: Pulling from hello-repository
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
Status: Downloaded newer image for aws_account_id.dkr.region.amazonaws.com/hello-
repository:latest
```

Langkah 6: Menghapus citra

Jika Anda tidak lagi membutuhkan gambar di salah satu repositori Anda, Anda dapat menghapus gambar tersebut. Untuk menghapus gambar, tentukan repositori yang ada di dalamnya dan `imageDigest` nilai `imageTag` atau untuk gambar tersebut. Contoh berikut menghapus gambar di `hello-repository` repositori dengan tag gambar. `latest` Untuk menghapus gambar contoh Anda dari repositori, jalankan perintah berikut:

```
aws ecr batch-delete-image \
      --repository-name hello-repository \
      --image-ids imageTag=latest \
```

```
      --region region
```

Langkah 7: Menghapus repositori

Jika Anda tidak lagi membutuhkan seluruh repositori gambar, Anda dapat menghapus repositori. Contoh berikut menggunakan `--force` bendera untuk menghapus repositori yang berisi gambar. Untuk menghapus repositori yang berisi gambar (dan semua gambar di dalamnya), jalankan perintah berikut:

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

Mengoptimalkan performa untuk Amazon ECR

Anda dapat menggunakan rekomendasi berikut tentang pengaturan dan strategi untuk mengoptimalkan kinerja saat menggunakan Amazon ECR.

Gunakan Docker versi 1.10 ke atas untuk memanfaatkan unggahan lapisan secara simultan

Citra Docker terdiri dari lapisan, yang merupakan tahap pembangunan intermediate dari citra. Setiap baris dalam Dockerfile menghasilkan pembuatan lapisan baru. Ketika Anda menggunakan Docker versi 1.10 ke atas, Docker secara default mendorong sebanyak mungkin lapisan sebagai unggahan simultan ke Amazon ECR, sehingga waktu unggah lebih cepat.

Gunakan citra dasar yang lebih kecil

Citra default yang tersedia melalui Docker Hub mungkin berisi banyak dependensi yang tidak diperlukan aplikasi Anda. Pertimbangkan untuk menggunakan citra yang lebih kecil yang dibuat dan dikelola oleh orang lain di komunitas Docker, atau membangun citra dasar Anda sendiri menggunakan citra scratch minimal Docker. Untuk informasi selengkapnya, lihat Membuat citra dasardalam dokumentasi Docker.

Tempatkan dependensi yang paling tidak berubah sebelumnya di Dockerfile

Docker men-cache lapisan, dan itu mempercepat waktu pembangunan. Jika tidak ada yang berubah pada lapisan sejak pembuatan terakhir, Docker menggunakan versi yang di-cache alih-alih membangun kembali lapisan. Namun, setiap lapisan tergantung pada lapisan yang ada sebelumnya. Jika sebuah lapisan berubah, Docker mengkompilasi ulang tidak hanya lapisan itu, tetapi juga setiap lapisan yang datang setelah lapisan tersebut.

Untuk meminimalkan waktu yang diperlukan untuk membangun kembali Dockerfile dan mengunggah ulang lapisan, pertimbangkan untuk menempatkan dependensi yang paling tidak sering berubah sebelumnya di Dockerfile Anda. Tempatkan dependensi yang berubah dengan cepat (seperti kode sumber aplikasi Anda) nanti di tumpukan.

Perintah rantai untuk menghindari penyimpanan file yang tidak perlu

File-file intermediate yang dibuat pada lapisan tetap menjadi bagian dari lapisan tersebut meskipun dihapus dalam lapisan berikutnya. Pertimbangkan contoh berikut:

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz
RUN wget tar -xvf software.tar.gz
```

```
RUN mv software/binary /opt/bin/myapp
RUN rm software.tar.gz
```

Dalam contoh ini, lapisan yang dibuat oleh perintah RUN pertama dan kedua berisi file .tar.gz asli dan semua isinya yang di-unzip. Ini meskipun file .tar.gz dihapus oleh perintah RUN keempat. Perintah ini dapat dirantai bersama-sama menjadi pernyataan RUN tunggal untuk memastikan bahwa file-file yang tidak perlu ini bukan bagian dari citra Docker akhir:

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz &&\
    wget tar -xvf software.tar.gz &&\
    mv software/binary /opt/bin/myapp &&\
    rm software.tar.gz
```

Gunakan titik akhir regional terdekat

Anda dapat mengurangi latensi dalam menarik citra dari Amazon ECR dengan memastikan bahwa Anda menggunakan titik akhir regional yang terdekat dengan tempat aplikasi Anda berjalan. Jika aplikasi Anda berjalan pada EC2 instance Amazon, Anda dapat menggunakan kode shell berikut untuk mendapatkan wilayah dari Availability Zone instance:

```
REGION=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone
 |\
    sed -n 's/\(\d*\)[a-zA-Z]*$/\1/p')
```

Wilayah dapat diteruskan ke AWS CLI perintah menggunakan --region parameter, atau ditetapkan sebagai wilayah default untuk profil menggunakan aws configure perintah. Anda juga dapat mengatur wilayah saat melakukan panggilan menggunakan AWS SDK. Untuk informasi lebih lanjut, lihat dokumentasi SDK untuk bahasa pemrograman khusus Anda.

Membuat permintaan ke pendaftar Amazon ECR

Anda dapat mendorong, menarik, menghapus, melihat, dan mengelola gambar OCI, gambar Docker, dan artefak yang kompatibel dengan OCI di pendaftar pribadi Amazon ECR menggunakan titik akhir -only IPv4 atau titik akhir tumpukan ganda (dan). IPv4 IPv6 Untuk membuat permintaan dari IPv4 jaringan, Anda dapat menggunakan dual-stack atau IPv4 endpoint. Untuk membuat permintaan dari IPv6 jaringan, gunakan endpoint dual-stack. Untuk informasi selengkapnya tentang membuat permintaan ke pendaftar Publik Amazon ECR yang menggunakan IPv4 dan titik akhir tumpukan ganda, lihat Membuat permintaan ke pendaftar Publik Amazon ECR. Tidak ada biaya tambahan untuk mengakses Amazon ECR. IPv6 Untuk informasi selengkapnya tentang harga, lihat harga Amazon Elastic Container Registry.

Titik akhir Amazon ECR ditetapkan oleh atribut di luar dukungan titik akhir IPv4 -only atau dual-stack endpoint. Atribut ini dapat mencakup:

- Wilayah - Setiap titik akhir khusus untuk Wilayah.
- Jenis - Pemilihan titik akhir tergantung pada apakah Anda menggunakan antarmuka baris perintah AWS SDK atau yang kompatibel dengan OCI dan Docker.
- Keamanan — Di Wilayah tertentu Amazon ECR menawarkan titik akhir yang sesuai dengan FIPS. Untuk informasi selengkapnya tentang daftar titik akhir Amazon ECR yang sesuai dengan FIPS, lihat Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Untuk informasi selengkapnya tentang titik akhir layanan yang didukung oleh IPv4, dual-stack, Docker, dan klien OCI, yang menangani panggilan Amazon ECR API dari AWS CLI dan lihat, titik akhir Layanan. AWS SDKs

Memulai dengan membuat permintaan selesai IPv6

Untuk membuat permintaan ke registri Amazon ECR IPv6, Anda perlu menggunakan endpoint dual-stack. Sebelum mengakses registri Amazon ECR IPv6, verifikasi persyaratan berikut:

- Klien dan jaringan Anda harus mendukung IPv6.
- Amazon ECR mendukung jenis permintaan berikut: IPv6
  - Permintaan klien OCI dan Docker:

```
<registry-id>.dkr-ecr.<aws-region>.on.aws
```

- AWS Permintaan API:

  `ecr.<aws-region>.api.aws`

- Anda harus memperbarui AWS Identity and Access Management (IAM) atau kebijakan registri apa pun yang menggunakan pemfilteran alamat IP sumber untuk menyertakan rentang IPv6 alamat. Untuk informasi selengkapnya, lihat Menggunakan IPv6 alamat dalam kebijakan IAM.
- Saat Anda menggunakan IPv6, log akses server menampilkan `Remote IP` alamat dalam IPv6 format. Perbarui alat, skrip, dan perangkat lunak yang ada untuk mengurai alamat IP yang IPv6 diformat ini.

> **Note**
>
> Jika Anda mengalami masalah yang terkait dengan keberadaan IPv6 alamat dalam file log, hubungi AWS Dukungan.

Menguji kompatibilitas alamat IP

Jika Anda menggunakan use Linux/Unix atau Mac OS X, Anda dapat menguji apakah Anda dapat mengakses titik akhir dual-stack IPv6 dengan menggunakan `curl` perintah seperti yang ditunjukkan pada contoh berikut:

Example

```
curl --verbose https://ecr.us-west-2.api.aws
```

Anda mendapatkan informasi yang serupa dengan contoh berikut. Jika Anda terhubung melalui IPv6 alamat IP yang terhubung akan menjadi IPv6 alamat.

```
* About to connect() to ecr.us-west-2.api.aws port 443 (#0)
* Trying IPv6 address... connected
* Connected to ecr.us-west-2.api.aws (IPv6 address) port 443 (#0)
> Host: ecr.us-west-2.api.aws
* Request completely sent off
```

Jika Anda menggunakan Microsoft Windows 7 atau Windows 10, Anda dapat menguji apakah Anda dapat mengakses titik akhir dual-stack di atas IPv4 atau IPv6 dengan menggunakan `ping` perintah seperti yang ditunjukkan pada contoh berikut.

```
ping ecr.us-west-2.api.aws
```

Membuat permintaan IPv6 dengan menggunakan titik akhir dual-stack

Anda dapat membuat panggilan Amazon ECR API melalui IPv6 menggunakan titik akhir dual-stack. Fungsionalitas dan kinerja operasi Amazon ECR API tetap konsisten apakah Anda menggunakan IPv4 atau IPv6.

Bila Anda menggunakan AWS Command Line Interface (AWS CLI) dan AWS SDKs, Anda dapat mengaktifkan IPv6 baik dengan menggunakan parameter atau flag untuk beralih ke titik akhir dual-stack, atau dengan langsung menentukan titik akhir dual-stack dalam file konfigurasi Anda untuk mengganti endpoint Amazon ECR default. Anda juga dapat membuat perubahan konfigurasi dengan menggunakan perintah, yang disetel `use_dualstack_endpoint` ke true di profil default. Untuk informasi selengkapnya`use_dualstack_endpoint`, lihat Dual-stack dan titik akhir FIPS.

Example Membuat perubahan konfigurasi dengan menggunakan perintah

```
aws configure set default.ecr.use_dualstack_endpoint true
```

Example Membuat permintaan lebih dari IPv6 menggunakan AWS CLI

```
aws ecr describe-repositories --region us-west-2 --endpoint-url https://
ecr.us-west-2.api.aws
```

Menggunakan titik akhir Amazon ECR dari docker CLI

Setelah Anda masuk ke repositori Amazon ECR Anda dan menandai gambar Anda, Anda dapat mendorong dan menarik gambar OCI dan gambar Docker ke dan dari pendaftar Amazon ECR. Contoh berikut menunjukkan perintah docker push dan docker pull dengan kedua titik akhir dual-stack.

Example Mendorong gambar docker menggunakan endpoint IPv4

```
docker push <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example Mendorong gambar docker menggunakan titik akhir dual-stack

```
docker push <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Example Menarik gambar docker menggunakan endpoint IPv4

```
docker pull <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example Menarik gambar docker menggunakan titik akhir dual-stack

```
docker pull <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Menggunakan IPv6 alamat dalam kebijakan IAM

Sebelum Anda mengakses registri menggunakan IPv6, pastikan bahwa pengguna IAM dan kebijakan registri Amazon ECR Anda yang menggunakan pemfilteran alamat IP menyertakan IPv6 rentang alamat. Jika kebijakan pemfilteran alamat IP tidak diperbarui untuk menangani IPv6 alamat, klien mungkin salah kehilangan atau mendapatkan akses ke registri saat mereka mulai menggunakan IPv6. Untuk informasi selengkapnya tentang mengelola izin akses, lihat Identity and Access Management untuk Amazon Elastic Container Registry.

Kebijakan IAM yang memfilter alamat IP menggunakan Operator Kondisi Alamat IP. Contoh kebijakan registri berikut menunjukkan cara mengidentifikasi `54.240.143.*` rentang IPv4 alamat yang diizinkan dengan menggunakan operator kondisi alamat IP. Alamat IP apa pun di luar rentang ini ditolak akses ke registri (`exampleregistry`). Karena semua IPv6 alamat berada di luar rentang yang diizinkan, kebijakan ini mencegah akses IPv6 alamat. `exampleregistry`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "ecr:*",
      "Resource": "arn:aws:ecr:*:*:repository/exampleregistry/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

Untuk mengizinkan rentang alamat IPv4 IPv6 (`54.240.143.0/24`2001:DB8:1234:5678::/64) dan (), ubah elemen Kondisi kebijakan registri seperti yang ditunjukkan pada contoh berikut. Anda dapat menggunakan format `Condition` blok ini untuk memperbarui kebijakan pengguna dan registri IAM Anda.

```
      "Condition": {
        "IpAddress": {
           "aws:SourceIp": [
             "54.240.143.0/24",
              "2001:DB8:1234:5678::/64"
            ]
         }
       }
```

> **Important**
>
> Sebelum menggunakan, IPv6 Anda harus memperbarui semua kebijakan pengguna dan registri IAM yang relevan yang menggunakan pemfilteran alamat IP. Kami tidak menyarankan menggunakan pemfilteran alamat IP dalam kebijakan registri.

Anda dapat meninjau kebijakan pengguna IAM menggunakan konsol IAM di. https://console.aws.amazon.com/iam/ Untuk informasi lebih lanjut tentang IAM, lihat Panduan Pengguna IAM.

Registri pribadi Amazon ECR

Registri pribadi Amazon ECR menghosting gambar kontainer Anda dalam arsitektur yang sangat tersedia dan dapat diskalakan. Anda dapat menggunakan registri pribadi Anda untuk mengelola repositori citra pribadi yang terdiri dari citra dan artefak Docker dan Open Container Initiative (OCI). Setiap akun AWS disediakan registri pribadi default Amazon ECR. Untuk informasi lebih lanjut tentang registri publik Amazon ECR, lihat Registri publik di Panduan pengguna Amazon Elastic Container Registry.

Konsep registri pribadi

- URL untuk registri pribadi default Anda adalah `https://aws_account_id.dkr.ecr.region.amazonaws.com`.
- Secara default, akun Anda telah membaca dan menulis akses ke repositori di registri pribadi Anda. Namun, pengguna memerlukan izin untuk melakukan panggilan ke Amazon ECR API dan untuk mendorong atau menarik gambar ke dan dari repositori pribadi Anda. Amazon ECR menyediakan beberapa kebijakan yang dikelola untuk mengontrol akses pengguna pada berbagai tingkat. Untuk informasi selengkapnya, lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry.
- Anda harus mengautentikasi klien Docker ke registri pribadi Anda sehingga Anda dapat menggunakan perintah docker push dan docker pull untuk mendorong dan menarik citra ke dan dari repositori dalam registri tersebut. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.
- Repositori pribadi dapat dikontrol dengan kebijakan akses pengguna dan kebijakan repositori. Untuk informasi lebih lanjut tentang kebijakan repositori, lihat Kebijakan repositori pribadi di Amazon ECR.
- Repositori di registri pribadi Anda dapat direplikasi di seluruh AWS Wilayah di registri pribadi Anda sendiri dan di seluruh akun terpisah dengan mengonfigurasi replikasi untuk registri pribadi Anda. Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR.

Otentikasi registri pribadi di Amazon ECR

Anda dapat menggunakan Konsol Manajemen AWS, SDK AWS CLI, atau AWS SDK untuk membuat dan mengelola repositori pribadi. Anda juga dapat menggunakan metode tersebut untuk melakukan beberapa tindakan pada citra, seperti mendaftar atau menghapusnya. Klien ini menggunakan metode AWS otentikasi standar. Meskipun Anda dapat menggunakan API Amazon ECR untuk mendorong

dan menarik citra, Anda lebih cenderung menggunakan Docker CLI atau perpustakaan Docker khusus bahasa.

Docker CLI tidak mendukung metode otentikasi IAM asli. Langkah-langkah tambahan harus diambil sehingga Amazon ECR dapat mengautentikasi dan mengotorisasi permintaan dorongan dan tarikan Docker.

Metode autentikasi registri yang dirinci dalam bagian berikut tersedia.

Menggunakan Amazon ECR credential helper

Amazon ECR menyediakan Docker credential helper yang membuatnya lebih mudah untuk menyimpan dan menggunakan kredensial Docker saat mendorong dan menarik citra ke Amazon ECR. Untuk langkah-langkah instalasi dan konfigurasi, lihat Amazon ECR Docker Credential Helper.

> **Note**
>
> Pembantu kredensi Amazon ECR Docker saat ini tidak mendukung otentikasi multi-faktor (MFA).

Menggunakan token otorisasi

Lingkup perizinan token otorisasi sesuai dengan IAM utama yang digunakan untuk mengambil token autentikasi. Token otentikasi digunakan untuk mengakses registri Amazon ECR di mana IAM utama Anda memiliki akses dan berlaku selama 12 jam. Untuk mendapatkan token otorisasi, Anda harus menggunakan operasi GetAuthorizationTokenAPI untuk mengambil token otorisasi berenkode base64 yang berisi nama pengguna dan kata sandi yang dikodekan. `AWS` AWS CLI `get-login-password`Perintah menyederhanakan ini dengan mengambil dan mendekode token otorisasi yang kemudian dapat Anda salurkan ke perintah untuk mengautentikasi. docker login

Untuk mengautentikasi Docker ke registri pribadi Amazon ECR dengan get-login

- Untuk mengotentikasi Docker ke registri Amazon ECR dengan get-login-password, jalankan perintah aws ecr get-login-password. Ketika meneruskan token otorisasi ke perintah docker login, gunakan nilai `AWS` untuk nama pengguna dan tentukan URI registri Amazon ECR yang ingin Anda autentikasi. Jika melakukan autentikasi untuk beberapa registri, Anda harus mengulangi perintah tersebut untuk setiap registri.

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

- get-login-password(AWS CLI)

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

- Get-ECRLoginCommand (AWS Tools for Windows PowerShell)

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-
stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

Menggunakan autentikasi HTTP API

Amazon ECR mendukung HTTP API Registri Docker. Namun, karena Amazon ECR adalah registri pribadi, Anda harus memberikan token otorisasi pada setiap permintaan HTTP. Anda dapat menambahkan header otorisasi HTTP menggunakan `-H` opsi untuk curl dan meneruskan token otorisasi yang disediakan oleh perintah. get-authorization-token AWS CLI

Untuk mengautentikasi dengan HTTP API Amazon ECR

1. Ambil token otorisasi dengan AWS CLI dan atur ke variabel lingkungan.

```
TOKEN=$(aws ecr get-authorization-token --output text --query
 'authorizationData[].authorizationToken')
```

2. Untuk mengautentikasi ke API, berikan variabel `$TOKEN` ke opsi `-H` dari curl. Misalnya, perintah berikut mencantumkan tanda citra dalam repositori Amazon ECR. Untuk informasi lebih lanjut, lihat dokumentasi referensi HTTP API Registri Docker.

```
curl -i -H "Authorization: Basic $TOKEN"
 https://aws_account_id.dkr.ecr.region.amazonaws.com/v2/amazonlinux/tags/list
```

Output adalah sebagai berikut:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
Connection: keep-alive

{"name":"amazonlinux","tags":["2017.09","latest"]}
```

Pengaturan registri pribadi di Amazon ECR

Amazon ECR menggunakan pengaturan registri pribadi untuk mengonfigurasi fitur di tingkat registri. Pengaturan registri pribadi dikonfigurasi secara terpisah untuk setiap Wilayah. Anda dapat menggunakan pengaturan registri pribadi untuk mengonfigurasi fitur-fitur berikut.

- Izin registri — Kebijakan izin registri memberikan kontrol atas replikasi dan menarik izin cache. Untuk informasi selengkapnya, lihat Izin registri pribadi di Amazon ECR.
- Tarik aturan cache — Aturan pull through cache digunakan untuk menyimpan gambar dari registri upstream di registri pribadi Amazon ECR Anda. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR.
- Konfigurasi replikasi - Konfigurasi replikasi digunakan untuk mengontrol apakah repositori Anda disalin di seluruh atau. Wilayah AWS Akun AWS Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR.
- Templat pembuatan repositori — Template pembuatan repositori digunakan untuk menentukan pengaturan standar yang akan diterapkan saat repositori baru dibuat oleh Amazon ECR atas nama Anda. Misalnya, repositori yang dibuat oleh aksi pull through cache, create on push, atau replikasi. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.
- Konfigurasi pemindaian — Secara default, registri Anda diaktifkan untuk pemindaian dasar. Anda dapat mengaktifkan pemindaian yang disempurnakan yang menyediakan mode pemindaian otomatis dan berkelanjutan yang memindai kerentanan paket sistem operasi dan bahasa pemrograman. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.

- Pull-time pengecualian pembaruan - Anda dapat mengonfigurasi pengecualian pembaruan waktu tarik untuk mencegah waktu tarik terakhir diperbarui untuk gambar tertentu saat ditarik. Ini berguna untuk gambar yang digunakan untuk pengujian atau CI/CD tujuan di mana Anda tidak ingin waktu tarik memengaruhi keputusan kebijakan siklus hidup. Untuk informasi selengkapnya, lihat Pengecualian pembaruan waktu tarik.
- Konfigurasi pemasangan gumpalan - Konfigurasi pemasangan gumpalan digunakan untuk mengontrol apakah repositori dalam registri Anda berbagi lapisan umum daripada menyimpan lapisan duplikat. Untuk informasi selengkapnya, lihat Pemasangan gumpalan di Amazon ECR.

Pemasangan gumpalan di Amazon ECR

Amazon ECR mendukung kemampuan yang disebut blob mounting untuk berbagi lapisan gambar umum di seluruh repositori dalam registri. Ketika diaktifkan, repositori dalam satu registri dapat mereferensikan lapisan dari repositori lain dalam registri yang sama alih-alih menyimpan salinan duplikat.

Saat pemasangan gumpalan registri diaktifkan, Amazon ECR memeriksa lapisan yang ada di registri Anda selama operasi push saat parameter pemasangan disertakan. Jika lapisan sudah ada di repositori lain dalam registri yang sama, Amazon ECR akan memasang lapisan yang ada alih-alih mengunggah duplikat.

> **Note**
>
> Klien OCI secara otomatis menyertakan parameter pemasangan jika mereka mendeteksi gumpalan mungkin sudah ada di repositori yang berbeda. Amazon ECR mencoba pemasangan hanya ketika parameter ini ada dalam permintaan POST klien.

Konsep pemasangan gumpalan

- Pemasangan gumpalan hanya berfungsi dalam registri yang sama (akun dan wilayah yang sama).
- Repositori harus menggunakan jenis dan kunci enkripsi yang identik.
- Pemasangan gumpalan tidak didukung untuk gambar yang dibuat melalui pull through cache.
- Jika Anda memutuskan untuk menonaktifkan pemasangan gumpalan, gambar yang ada yang didorong dengan pemasangan gumpalan yang dikonfigurasi akan terus berfungsi dan lapisan akan tetap terpasang.

Konfigurasi pemasangan gumpalan

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk mengkonfigurasi pemasangan gumpalan untuk registri Anda.

> **Note**
>
> Pengguna memerlukan izin `ecr:GetDownloadUrlForLayer` IAM pada repositori untuk memasang lapisan darinya.

Konsol Manajemen AWS

Gunakan langkah-langkah berikut untuk memperbarui konfigurasi pemasangan gumpalan registri Anda menggunakan. Konsol Manajemen AWS

Aktifkan konfigurasi pemasangan gumpalan untuk registri pribadi Anda

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/private-registry/repositories
2. Dari bilah navigasi, pilih Wilayah.
3. Di panel navigasi, pilih Registri pribadi, Fitur & Pengaturan, lalu pilih Pemasangan gumpalan.
4. Pada halaman pemasangan Blob, pilih Aktifkan.

   Tampilan spanduk yang menunjukkan bahwa konfigurasi pemasangan gumpalan telah diperbarui untuk diaktifkan.

AWS CLI

Gunakan perintah berikut untuk memperbarui konfigurasi pemasangan gumpalan registri Anda menggunakan. AWS CLI

- ```
  aws ecr put-account-setting --name BLOB_MOUNTING --value ENABLED
  ```
  ```

Izin registri pribadi di Amazon ECR

Amazon ECR menggunakan kebijakan registri untuk memberikan izin kepada AWS kepala sekolah di tingkat registri pribadi.

Amazon ECR memungkinkan semua tindakan ECR dalam kebijakan dan memberlakukan kebijakan registri di semua permintaan ECR. Anda dapat menggunakan kebijakan registri untuk memberikan izin untuk tindakan seperti konfigurasi replikasi, pembuatan aturan cache pull-through, dan pembuatan repositori. Untuk daftar lengkap tindakan API, lihat Panduan Amazon ECR API. Untuk informasi tentang setelan umum untuk registri pribadi Amazon ECR Anda, lihatPengaturan registri pribadi di Amazon ECR.

> **Note**
>
> Meskipun dimungkinkan untuk menambahkan `ecr:*` tindakan ke kebijakan registri pribadi, dianggap praktik terbaik untuk hanya menambahkan tindakan spesifik yang diperlukan berdasarkan fitur yang Anda gunakan daripada menggunakan wildcard.

Topik

Contoh kebijakan registri pribadi untuk Amazon ECR

Contoh berikut menunjukkan pernyataan kebijakan izin registri yang dapat Anda gunakan untuk mengontrol izin yang dimiliki pengguna atas registri Amazon ECR Anda.

> **Note**
>
> Dalam setiap contoh, jika `ecr:CreateRepository` tindakan dihapus dari kebijakan registri Anda, replikasi masih dapat terjadi. Namun, agar replikasi berhasil, Anda perlu membuat repositori dengan nama yang sama dalam akun Anda.

Contoh: Izinkan semua prinsipal IAM di akun sumber untuk mereplikasi semua repositori

Kebijakan izin registri berikut memungkinkan semua prinsipal IAM (pengguna dan peran) di akun sumber untuk mereplikasi semua repositori.

Perhatikan hal-hal berikut:

- Penting: Saat Anda menentukan Akun AWS ID sebagai prinsipal dalam kebijakan, Anda memberikan akses ke semua pengguna dan peran IAM dalam akun tersebut, bukan hanya pengguna root. Ini memberikan akses luas di seluruh akun.
- Pertimbangan Keamanan: Account-level izin memberikan akses ke semua entitas IAM di akun yang ditentukan. Untuk akses yang lebih ketat, tentukan pengguna IAM individu, peran, atau gunakan pernyataan kondisi untuk membatasi akses lebih lanjut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:444455556666:repository/*"
            ]
        }
    ]
}
```

Contoh: Izinkan prinsipal IAM dari beberapa akun

Kebijakan izin registri berikut memiliki dua pernyataan. Setiap pernyataan memungkinkan semua prinsipal IAM (pengguna dan peran) dalam akun sumber untuk mereplikasi semua repositori.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount1",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:123456789012:repository/*"
            ]
        },
        {
            "Sid":"ReplicationAccessCrossAccount2",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::444455556666:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:123456789012:repository/*"
            ]
        }
    ]
}
```

Contoh: Izinkan semua prinsipal IAM di akun sumber untuk mereplikasi semua repositori dengan awalan. **`prod-`**

Kebijakan izin registri berikut memungkinkan semua prinsipal IAM (pengguna dan peran) di akun sumber untuk mereplikasi semua repositori yang dimulai dengan. `prod-`

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:444455556666:repository/prod-*"
            ]
        }
    ]
}
```

Memberikan izin registri untuk replikasi lintas akun di Amazon ECR

Jenis kebijakan lintas akun digunakan untuk memberikan izin kepada AWS prinsipal, memungkinkan replikasi repositori dari registri sumber ke registri Anda. Secara default, Anda memiliki izin untuk mengonfigurasi replikasi antar wilayah dalam registri Anda sendiri. Anda hanya perlu mengonfigurasi kebijakan registri jika Anda memberikan izin akun lain untuk mereplikasi konten ke registri Anda.

Kebijakan registri harus memberikan izin untuk tindakan API `ecr:ReplicateImage`. API ini adalah API Amazon ECR internal yang dapat mereplikasi citra antar Wilayah atau akun. Anda juga dapat memberikan izin untuk `ecr:CreateRepository`, yang memungkinkan Amazon ECR untuk membuat repositori di registri Anda jika belum ada. Jika izin `ecr:CreateRepository`

tidak diberikan, repositori dengan nama yang sama dengan repositori sumber harus dibuat secara manual di registri Anda. Jika keduanya tidak dilakukan, replikasi gagal. Setiap tindakan yang gagal `CreateRepository` atau `ReplicateImage` API muncul di CloudTrail.

Untuk mengonfigurasi kebijakan izin untuk replikasi ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi kebijakan registri Anda.
3. Di panel navigasi, pilih Registri pribadi, pilih Fitur & Pengaturan, lalu pilih Izin.
4. Pada halaman Izin registri, pilih Hasilkan pernyataan.
5. Menyelesaikan langkah-langkah berikut untuk menentukan pernyataan kebijakan Anda menggunakan kebijakan generator.
   a. Untuk jenis Kebijakan, pilih Replikasi - lintas akun.
   b. Untuk ID Pernyataan, masukkan ID pernyataan unik. Bidang ini digunakan sebagai `Sid` pada kebijakan registri.
   c. Untuk Akun, masukkan ID akun untuk setiap akun yang ingin Anda berikan izin. Saat menentukan beberapa ID akun, pisahkan dengan koma.
6. Pilih Simpan.

Untuk mengonfigurasi kebijakan izin untuk replikasi ()AWS CLI

1. Buat file bernama `registry_policy.json` dan isi dengan kebijakan registri.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::111122223333:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
```

```
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:444455556666:repository/*"
            ]
        }
    ]
}
```

2. Buat kebijakan registri menggunakan file kebijakan.

```
aws ecr put-registry-policy \
      --policy-text file://registry_policy.json \
      --region us-west-2
```

3. Ambil kebijakan untuk registri Anda untuk mengonfirmasi.

```
aws ecr get-registry-policy \
      --region us-west-2
```

Memberikan izin registri untuk menarik cache di Amazon ECR

Izin registri pribadi Amazon ECR dapat digunakan untuk mencakup izin entitas IAM individu untuk menggunakan cache pull through. Jika entitas IAM memiliki lebih banyak izin yang diberikan oleh kebijakan IAM daripada yang diberikan oleh kebijakan izin registri, kebijakan IAM akan diutamakan.

Untuk membuat kebijakan izin registri pribadi ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pernyataan izin registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, pilih Fitur & Pengaturan, lalu pilih Izin.
4. Pada halaman Izin registri, pilih Hasilkan pernyataan.
5. Untuk setiap pull through pernyataan kebijakan izin cache yang ingin Anda buat, lakukan hal berikut.

    a. Untuk jenis Policy, pilih Pull through cache policy.

    b. Untuk id Pernyataan, berikan nama untuk kebijakan pernyataan cache tarik melalui.

    c. Untuk entitas IAM, tentukan pengguna, grup, atau peran yang akan disertakan dalam kebijakan.

d. Untuk namespace Cache, pilih aturan pull through cache untuk mengaitkan kebijakan dengan.

e. Untuk nama Repositori, tentukan nama dasar repositori untuk menerapkan aturan. Misalnya, jika Anda ingin menentukan repositori Amazon Linux di Amazon ECR Public, nama repositori akan menjadi. `amazonlinux`

Repositori pribadi Amazon ECR

Repositori pribadi Amazon ECR berisi gambar Docker Anda, gambar Open Container Initiative (OCI), dan artefak yang kompatibel dengan OCI. Anda dapat membuat, memantau, dan menghapus repositori gambar dan mengatur izin yang mengontrol siapa yang dapat mengaksesnya dengan menggunakan operasi Amazon ECR API atau bagian Repositori dari konsol Amazon ECR. Amazon ECR juga terintegrasi dengan CLI Docker, sehingga Anda dapat mendorong dan menarik gambar dari lingkungan pengembangan Anda ke repositori Anda.

Topik

Konsep repositori pribadi

- Secara default, akun Anda telah membaca dan menulis akses ke repositori di registri default Anda (*aws_account_id*.dkr.ecr.*region*.amazonaws.com). Namun, pengguna memerlukan izin untuk melakukan panggilan ke Amazon ECR APIs dan mendorong atau menarik gambar ke dan dari repositori Anda. Amazon ECR menyediakan beberapa kebijakan yang dikelola untuk mengontrol akses pengguna pada berbagai tingkat. Untuk informasi selengkapnya, lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry.
- Repositori dapat dikontrol dengan kebijakan akses pengguna dan kebijakan repositori individu. Untuk informasi selengkapnya, lihat Kebijakan repositori pribadi di Amazon ECR.
- Nama repositori dapat mendukung namespace, yang dapat Anda gunakan untuk mengelompokkan repositori serupa. Misalnya, jika ada beberapa tim yang menggunakan registri yang sama, Tim A dapat menggunakan namespace `team-a`, dan Tim B dapat menggunakan namespace `team-b`. Dengan melakukan ini, setiap tim memiliki citra mereka sendiri yang disebut `web-app` pada setiap citra yang diawali dengan namespace tim. Konfigurasi ini memungkinkan citra-citra pada setiap tim ini untuk digunakan secara bersamaan tanpa gangguan. Citra Tim A adalah `team-a/web-app`, dan citra Tim B adalah `team-b/web-app`.

- Citra Anda dapat direplikasi ke repositori lain di seluruh Wilayah di registri Anda sendiri dan di seluruh akun. Anda dapat melakukan ini dengan menentukan konfigurasi replikasi dalam pengaturan registri Anda. Untuk informasi selengkapnya, lihat Pengaturan registri pribadi di Amazon ECR.
- Saat pemasangan gumpalan diaktifkan di tingkat registri, repositori dapat berbagi lapisan gambar yang umum.

Membuat repositori pribadi Amazon ECR untuk menyimpan gambar

> **Important**
>
> Enkripsi sisi server dua lapis dengan AWS KMS (DSSE-KMS) hanya tersedia di Wilayah. AWS GovCloud (US)

Buat repositori pribadi Amazon ECR, lalu gunakan repositori untuk menyimpan gambar kontainer Anda. Gunakan langkah-langkah berikut untuk membuat repositori pribadi menggunakan file. Konsol Manajemen AWS

Untuk membuat repositori ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah untuk membuat repositori Anda.
3. Pilih Repositori pribadi, lalu pilih Buat repositori.
4. Untuk Nama repositori, masukkan nama yang unik untuk repositori Anda. Nama repositori dapat ditentukan sendiri (misalnya `nginx-web-app`). Atau, dapat ditambahkan dengan namespace untuk mengelompokkan repositori ke dalam suatu kategori (misalnya `project-a/nginx-web-app`).

   > **Note**
   >
   > Nama repositori dapat berisi karakter maksimum. 256 Nama harus dimulai dengan huruf dan hanya dapat berisi huruf kecil, angka, tanda hubung, garis bawah, titik dan garis miring ke depan. Menggunakan garis miring ganda tidak didukung.

5. Untuk kekekalan tag Gambar, pilih salah satu pengaturan mutabilitas tag berikut untuk repositori.

- Mutable - Pilih opsi ini jika Anda ingin tag gambar ditimpa. Direkomendasikan untuk repositori yang menggunakan tindakan pull through cache untuk memastikan Amazon ECR dapat memperbarui gambar yang di-cache. Selain itu, untuk menonaktifkan pembaruan tag untuk beberapa tag yang dapat berubah, masukkan nama tag atau gunakan wildcard (*) untuk mencocokkan beberapa tag serupa di kotak teks pengecualian tag Mutable.
- Immutable - Pilih opsi ini jika Anda ingin mencegah tag gambar ditimpa, dan itu berlaku untuk semua tag dan pengecualian di repositori saat mendorong gambar dengan tag yang ada. Amazon ECR mengembalikan `ImageTagAlreadyExistsException` jika Anda mencoba mendorong gambar dengan tag yang ada. Selain itu, untuk mengaktifkan pembaruan tag untuk beberapa tag yang tidak dapat diubah, masukkan nama tag atau gunakan wildcard (*) untuk mencocokkan beberapa tag serupa di kotak teks pengecualian tag Immutable.

> **Note**
>
> Setelan mutabilitas tag individual tidak didukung.

6. Untuk konfigurasi Enkripsi, pilih antara AES-256 atau. AWS KMS Untuk informasi selengkapnya, lihat Enkripsi saat diam.

   a. Jika AWS KMS dipilih, pilih antara enkripsi Single-layer dan Dual-layer encryption. Ada biaya tambahan untuk menggunakan AWS KMS atau enkripsi Dual-layer. Untuk informasi selengkapnya, lihat Harga Layanan Amazon ECR.

   b. Secara default, kunci AWS terkelola dengan alias `aws/ecr` dipilih. Kunci ini dibuat di akun Anda saat pertama kali Anda membuat repositori dengan AWS KMS enkripsi diaktifkan. Pilih Kunci terkelola pelanggan (lanjutan) untuk memilih AWS KMS kunci Anda sendiri. AWS KMS Kuncinya harus berada di Region yang sama dengan cluster. Pilih Buat AWS KMS kunci untuk menavigasi ke AWS KMS konsol untuk membuat kunci Anda sendiri.

7. Untuk pengaturan pemindaian gambar, sementara Anda dapat menentukan pengaturan pemindaian di tingkat repositori untuk pemindaian dasar, ini adalah praktik terbaik untuk menentukan konfigurasi pemindaian di tingkat registri pribadi. Mengkonfigurasi pengaturan pemindaian di tingkat registri pribadi memungkinkan Anda memilih antara pemindaian yang ditingkatkan atau pemindaian dasar, dan juga memungkinkan Anda menentukan filter untuk menentukan repositori mana yang harus dipindai.

8. Pilih Buat.

Untuk membuat repositori ()AWS CLI

1. Anda dapat membuat repositori menggunakan perintah AWS CLI with the. aws ecr create-repository

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

2. Jika Anda memiliki template pembuatan repositori yang ditentukan, Anda dapat membuat repositori dengan mendorong gambar Anda menggunakan perintah push Amazon ECR yang sudah dikenal dengan nama repositori yang Anda inginkan. Amazon ECR akan secara otomatis membuat repositori untuk Anda menggunakan pengaturan yang telah ditentukan dari template pembuatan repositori Anda. Jika Anda belum memiliki template pembuatan repositori yang ditentukan, permintaan Anda ke repositori gambar Anda yang tidak ada akan gagal.

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/prefix/my-new-
repository:tag
```

Langkah selanjutnya

Untuk melihat langkah-langkah untuk mendorong gambar ke repositori Anda, pilih repositori dan pilih Lihat perintah push. Untuk informasi selengkapnya tentang mendorong gambar ke repositori Anda, lihat. Mendorong gambar ke repositori pribadi Amazon ECR

Melihat konten dan detail repositori pribadi di Amazon ECR

Setelah Anda membuat repositori pribadi, Anda dapat melihat detail tentang repositori di: Konsol Manajemen AWS

- Citra mana yang disimpan dalam repositori
- Detail tentang setiap gambar yang disimpan dalam repositori, termasuk ukuran dan intisari SHA untuk setiap gambar
- Frekuensi pemindaian yang ditentukan untuk isi repositori
- Apakah repositori memiliki aturan pull through cache aktif yang terkait dengannya
- Pengaturan enkripsi untuk repositori

> **Note**
>
> Dimulai dengan Docker versi 1.9, client Docker mengompresi lapisan citra sebelum mendorongnya ke registri V2 Docker. Output dari perintah docker images menunjukkan ukuran citra yang tidak terkompresi. Oleh karena itu, perlu diingat bahwa Docker mungkin mengembalikan gambar yang lebih besar daripada citra yang ditampilkan dalam Konsol Manajemen AWS.

Untuk melihat informasi repositori (Konsol Manajemen AWS)

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori untuk dilihat.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih tab Private dan kemudian repositori untuk dilihat.
5. Pada halaman detail repositori, konsol default ke tampilan Gambar. Gunakan menu navigasi untuk melihat informasi lain tentang repositori.
   - Pilih Ringkasan untuk melihat detail repositori dan tarik data hitungan untuk repositori.
   - Pilih Gambar untuk melihat informasi tentang tag gambar di repositori. Untuk melihat informasi lebih lanjut tentang gambar, pilih tag gambar. Untuk informasi selengkapnya, lihat Melihat detail gambar di Amazon ECR.

     Jika ada citra yang tidak ditandai yang ingin Anda hapus, Anda dapat memilih kotak di sebelah kiri repositori untuk menghapus dan pilih Hapus. Untuk informasi selengkapnya, lihat Menghapus gambar di Amazon ECR.
   - Pilih Izin untuk melihat kebijakan repositori yang diterapkan ke repositori. Untuk informasi selengkapnya, lihat Kebijakan repositori pribadi di Amazon ECR.
   - Pilih Kebijakan siklus hidup untuk melihat aturan kebijakan siklus hidup yang diterapkan ke repositori. Histori peristiwa siklus hidup juga dilihat di sini. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.
   - Pilih Tanda untuk melihat tanda metadata yang diterapkan ke repositori.

Menghapus repositori pribadi di Amazon ECR

Jika Anda selesai menggunakan repositori, Anda dapat menghapusnya. Ketika Anda menghapus repositori di Konsol Manajemen AWS, semua gambar yang terkandung dalam repositori juga dihapus; ini tidak dapat dibatalkan.

> **Important**
>
> Gambar di repositori yang dihapus juga dihapus. Anda tidak dapat membatalkan operasi ini.

Untuk menghapus repositori ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori untuk dihapus.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih tab Private dan kemudian pilih repositori yang akan dihapus dan pilih Delete.
5. Di ***repository_name*** jendela Hapus, verifikasi bahwa repositori yang dipilih harus dihapus dan pilih Hapus.

Kebijakan repositori pribadi di Amazon ECR

Amazon ECR menggunakan izin berbasis sumber daya untuk mengontrol akses ke repositori. Izin berbasis sumber daya memungkinkan Anda menentukan pengguna atau peran mana yang memiliki akses ke repositori dan tindakan apa yang dapat mereka lakukan di repositori. Secara default, hanya AWS akun yang membuat repositori yang memiliki akses ke repositori. Anda dapat menerapkan kebijakan repositori yang memungkinkan akses tambahan ke repositori Anda.

Topik

- Kebijakan repositori vs kebijakan IAM
- Contoh kebijakan repositori pribadi di Amazon ECR
- Menyetel pernyataan kebijakan repositori pribadi di Amazon ECR

Kebijakan repositori vs kebijakan IAM

Kebijakan repositori Amazon ECR adalah bagian dari kebijakan IAM yang memiliki lingkup untuk, dan secara khusus digunakan untuk, mengendalikan akses ke repositori Amazon ECR individu. Kebijakan IAM umumnya digunakan untuk menerapkan izin untuk seluruh layanan Amazon ECR tetapi juga dapat digunakan untuk mengontrol akses ke sumber daya tertentu.

Kebijakan repositori Amazon ECR dan kebijakan IAM digunakan saat menentukan tindakan yang mungkin dilakukan pengguna atau peran tertentu pada repositori. Jika pengguna atau peran diperbolehkan untuk melakukan tindakan melalui kebijakan repositori tetapi tidak diberi izin melalui kebijakan IAM (atau sebaliknya) maka tindakan akan ditolak. Agar tindakan diizinkan, pengguna atau peran hanya perlu diberi izin untuk tindakan baik melalui kebijakan repositori atau kebijakan IAM, tetapi tidak keduanya.

> **Important**
>
> Amazon ECR mengharuskan pengguna memiliki izin untuk melakukan panggilan ke API `ecr:GetAuthorizationToken` melalui kebijakan IAM sebelum mereka dapat mengautentikasi ke registri dan mendorong atau menarik citra dari repositori Amazon ECR. Amazon ECR menyediakan beberapa kebijakan IAM terkelola untuk mengontrol akses pengguna pada berbagai tingkatan. Untuk informasi selengkapnya, lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry.

Anda dapat menggunakan salah satu jenis kebijakan ini untuk mengontrol akses ke repositori Anda, seperti yang ditampilkan dalam contoh berikut.

Contoh ini menunjukkan kebijakan repositori Amazon ECR, yang memungkinkan pengguna tertentu untuk mendeskripsikan repositori dan gambar dalam repositori.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ECRRepositoryPolicy",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::111122223333:user/username"},
```

```
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh ini menunjukkan kebijakan IAM yang mencapai tujuan yang sama seperti di atas, dengan lingkup kebijakan pada repositori (ditentukan oleh ARN penuh repositori) menggunakan parameter sumber daya. Untuk informasi lebih lanjut tentang format Amazon Resource Name (ARN), lihat Sumber daya.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeRepoImage",
            "Effect": "Allow",
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": ["arn:aws:ecr:us-
east-1:111122223333:repository/repository-name"]
        }
    ]
}
```

Contoh kebijakan repositori pribadi di Amazon ECR

> **Important**
>
> Contoh kebijakan repositori di halaman ini dimaksudkan untuk diterapkan ke repositori pribadi Amazon ECR. Mereka tidak akan berfungsi dengan baik jika digunakan dengan prinsipal IAM secara langsung kecuali dimodifikasi untuk menentukan repositori Amazon ECR sebagai

sumber daya. Untuk informasi selengkapnya tentang pengaturan kebijakan repositori, lihat. Menyetel pernyataan kebijakan repositori pribadi di Amazon ECR

Kebijakan repositori Amazon ECR adalah bagian dari kebijakan IAM yang memiliki lingkup untuk, dan secara khusus digunakan untuk, mengendalikan akses ke repositori Amazon ECR individu. Kebijakan IAM umumnya digunakan untuk menerapkan izin untuk seluruh layanan Amazon ECR tetapi juga dapat digunakan untuk mengontrol akses ke sumber daya tertentu. Untuk informasi selengkapnya, lihat Kebijakan repositori vs kebijakan IAM.

Contoh kebijakan repositori berikut menunjukkan pernyataan izin yang dapat Anda gunakan untuk mengontrol akses ke repositori pribadi Amazon ECR Anda.

Important

Amazon ECR mengharuskan pengguna memiliki izin untuk melakukan panggilan ke API `ecr:GetAuthorizationToken` melalui kebijakan IAM sebelum mereka dapat melakukan autentikasi ke registrasi dan mendorong atau menarik citra dari repositori Amazon ECR. Amazon ECR menyediakan beberapa kebijakan IAM terkelola untuk mengontrol akses pengguna pada berbagai tingkatan. Untuk informasi selengkapnya, lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry.

Contoh: Izinkan satu atau lebih pengguna

Kebijakan repositori berikut memungkinkan satu atau lebih pengguna untuk mendorong dan menarik gambar ke dan dari repositori.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPushPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/push-pull-user-1",
```

```
                    "arn:aws:iam::111122223333:user/push-pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:GetDownloadUrlForLayer",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh: Izinkan akun lain

Kebijakan repositori berikut memungkinkan akun tertentu untuk mendorong citra.

> **Important**
>
> Akun yang Anda berikan izin harus memiliki Wilayah yang Anda beri kebijakan repositori yang diaktifkan, jika tidak, kesalahan akan terjadi.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPush",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
```

```
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan repositori berikut memungkinkan beberapa pengguna untuk menarik gambar (*pull-user-1*dan*pull-user-2*) sambil memberikan akses penuh ke yang lain (*admin-user*).

> **Note**
>
> Untuk kebijakan repositori yang lebih rumit yang saat ini tidak didukung di Konsol Manajemen AWS, Anda dapat menerapkan kebijakan dengan perintah. set-repository-policy AWS CLI

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/pull-user-1",
                    "arn:aws:iam::111122223333:user/pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowAll",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/admin-user"
            },
            "Action": [
                "ecr:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh: Tolak semua

Kebijakan repositori berikut menolak semua pengguna di semua akun untuk menarik citra.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPull",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh: Membatasi akses ke alamat IP tertentu

Contoh berikut menolak izin kepada pengguna mana pun untuk melakukan operasi ECR Amazon apa pun saat diterapkan ke repositori dari rentang alamat tertentu.

Kondisi dalam pernyataan ini mengidentifikasi `54.240.143.*` kisaran alamat IP Internet Protocol versi 4 (IPv4) yang diizinkan.

`Condition`Blok menggunakan `NotIpAddress` kondisi dan kunci `aws:SourceIp` kondisi, yang merupakan kunci kondisi AWS-wide. Untuk informasi lebih lanjut tentang kunci syarat ini, lihat AWS Kunci Konteks Syarat Global. `aws:sourceIp` IPv4 Nilai-nilai menggunakan notasi CIDR standar. Untuk informasi lebih lanjut, lihat Operator Syarat Alamat IP dalam Panduan Pengguna IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ECRPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "ecr:*",
            "Resource": "*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "54.240.143.0/24"
                }
            }
        }
    ]
}
```

Contoh: Izinkan AWS layanan

Kebijakan repositori berikut memungkinkan AWS CodeBuild akses ke tindakan Amazon ECR API yang diperlukan untuk integrasi dengan layanan tersebut. Saat menggunakan contoh berikut, Anda harus menggunakan kunci `aws:SourceArn` dan `aws:SourceAccount` kondisi untuk cakupan sumber daya mana yang dapat mengasumsikan izin ini. Untuk informasi selengkapnya, lihat contoh Amazon ECR CodeBuild di Panduan AWS CodeBuild Pengguna.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Resource": "*",
         "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:us-
east-1:123456789012:project/project-name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

Menyetel pernyataan kebijakan repositori pribadi di Amazon ECR

Anda dapat menambahkan pernyataan kebijakan akses ke repositori Konsol Manajemen AWS dengan mengikuti langkah-langkah di bawah ini. Anda dapat menambahkan beberapa pernyataan kebijakan per repositori. Untuk contoh kebijakan, lihat Contoh kebijakan repositori pribadi di Amazon ECR.

> **Important**
>
> Amazon ECR mengharuskan pengguna memiliki izin untuk melakukan panggilan ke API `ecr:GetAuthorizationToken` melalui kebijakan IAM sebelum mereka dapat mengautentikasi ke registri dan mendorong atau menarik citra dari repositori Amazon ECR. Amazon ECR menyediakan beberapa kebijakan IAM terkelola untuk mengontrol akses pengguna pada berbagai tingkatan. Untuk informasi selengkapnya, lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry.

Untuk menetapkan pernyataan kebijakan repositori

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori yang akan ditetapkan pernyataan kebijakan.
3. Di panel navigasi, pilih Repositori.
4. Pada Repositori, pilih repositori yang akan ditetapkan pernyataan kebijakan untuk melihat isi repositori.
5. Dari tampilan daftar citra repositori, di panel navigasi, pilih Izin, Edit.

> **Note**
>
> Jika Anda tidak melihat opsi Izin di panel navigasi, pastikan bahwa Anda berada di tampilan daftar citra repositori.

6. Pada halaman Edit izin, pilih Tambah pernyataan.
7. Untuk Nama pernyataan, masukkan nama untuk pernyataan tersebut.
8. Untuk Efek, pilih apakah pernyataan kebijakan tersebut akan menghasilkan izin atau penolakan eksplisit.
9. Untuk Utama, pilih ruang lingkup untuk penerapan pernyataan kebijakan. Untuk informasi lebih lanjut, lihat Elemen Kebijakan JSON AWS : Utama dalam Panduan Pengguna IAM.

   - Anda dapat menerapkan pernyataan ke semua AWS pengguna yang diautentikasi dengan memilih kotak centang Semua orang (*).
   - Untuk Prinsipal layanan, tentukan nama prinsipal layanan (misalnya, `ecs.amazonaws.com`) untuk menerapkan pernyataan ke layanan tertentu.
   - Untuk AWS Akun IDs, tentukan nomor AWS akun (misalnya,`111122223333`) untuk menerapkan pernyataan ke semua pengguna di bawah AWS akun tertentu. Beberapa akun dapat ditentukan dengan menggunakan daftar yang dipisahkan koma.

> **Important**
>
> Akun yang Anda berikan izin harus memiliki Wilayah yang Anda beri kebijakan repositori yang diaktifkan, jika tidak, kesalahan akan terjadi.

- Untuk Entitas IAM, pilih peran atau pengguna di bawah AWS akun Anda untuk menerapkan pernyataan tersebut.

> **Note**
>
> Untuk kebijakan repositori yang lebih rumit yang saat ini tidak didukung di Konsol Manajemen AWS, Anda dapat menerapkan kebijakan dengan perintah. set-repository-policy AWS CLI

10. Untuk Tindakan, pilih ruang lingkup operasi API Amazon ECR yang akan diterapkan pernyataan kebijakan dari daftar operasi API individu.
11. Setelah selesai, pilih Simpan untuk menetapkan kebijakan.
12. Ulangi langkah sebelumnya untuk setiap kebijakan repositori yang ditambahkan.

Menandai repositori pribadi di Amazon ECR

Untuk membantu mengelola repositori Amazon ECR, Anda dapat menetapkan metadata Anda sendiri ke repositori Amazon ECR baru atau yang sudah ada dengan menggunakan tag sumber daya. AWS Misalnya, Anda dapat menentukan satu set tag untuk repositori Amazon ECR akun Anda yang membantu Anda melacak pemilik setiap repositori.

Dasar-dasar tag

Tanda tidak memiliki makna semantik bagi Amazon ECR dan diterjemahkan sebagai serangkaian karakter saja. Selain itu, tag tidak dapat ditetapkan secara otomatis ke sumber daya Anda. Anda dapat mengedit kunci dan nilai tanda, dan dapat menghapus tanda dari sumber daya kapan saja. Anda dapat mengatur nilai tanda ke string kosong, tetapi tidak dapat mengatur nilai tanda ke null. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang telah ada pada sumber daya tersebut, nilai yang baru akan menimpa nilai yang lama. Jika sumber daya dihapus, semua tanda untuk sumber daya tersebut juga akan dihapus.

Anda dapat bekerja dengan tag menggunakan konsol Amazon ECR, the AWS CLI, dan Amazon ECR API.

Menggunakan AWS Identity and Access Management (IAM), Anda dapat mengontrol pengguna mana di AWS akun Anda yang memiliki izin untuk membuat, mengedit, atau menghapus tag. Untuk informasi tentang tag dalam kebijakan IAM, lihatthe section called “Menggunakan Tag-Based Access Control”.

Penandaan sumber daya Anda untuk penagihan

Tanda yang Anda tambahkan ke repositori Amazon ECR Anda sangat membantu ketika meninjau alokasi biaya setelah tanda diaktifkan dalam Laporan Biaya & Penggunaan Anda. Untuk informasi selengkapnya, lihat Laporan penggunaan Amazon ECR.

Untuk melihat biaya sumber daya gabungan, Anda dapat mengatur informasi penagihan berdasarkan sumber daya yang memiliki nilai kunci tanda yang sama. Misalnya, Anda dapat memberi tanda pada beberapa sumber daya dengan nama aplikasi tertentu, lalu organisir informasi penagihan Anda untuk melihat total biaya aplikasi tersebut pada beberapa layanan. Untuk informasi selengkapnya tentang pengaturan laporan alokasi biaya dengan tanda, lihat Laporan Alokasi Biaya Bulanan dalam Panduan Pengguna AWS Billing .

> **Note**
>
> Jika Anda baru saja mengaktifkan pelaporan, data untuk bulan yang berjalan dapat dilihat setelah 24 jam.

Menambahkan tag ke repositori pribadi di Amazon ECR

Anda dapat menambahkan tag ke repositori pribadi.

Untuk informasi tentang nama dan praktik terbaik untuk tag, lihat Batas dan persyaratan penamaan tag dan Praktik terbaik di Panduan Pengguna AWS Sumber Daya Penandaan.

Menambahkan tag ke repositori ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih kotak centang di sebelah repositori yang ingin Anda tag.
5. Dari menu Tindakan, pilih Tag repositori.
6. Pada halaman tag Repositori, pilih Tambahkan tag, Tambahkan tag.
7. Pada halaman Edit tag repositori, tentukan kunci dan nilai untuk setiap tag, lalu pilih Simpan.

Menambahkan tag ke repositori (AWS CLI atau API)

Anda dapat menambahkan atau menimpa satu atau beberapa tag dengan menggunakan AWS CLI atau API.

- AWS CLI - tag-sumber daya
- Tindakan API - TagResource

Contoh berikut menunjukkan cara menambahkan tag menggunakan AWS CLI.

Contoh 1: Menandai repositori

Perintah berikut menandai repositori.

```
aws ecr tag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tags Key=stack,Value=dev
```

Contoh 2: Tag repositori dengan beberapa tag

Perintah berikut menambahkan tiga tag ke repositori.

```
aws ecr tag-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
     --tags Key=key1,Value=value1 Key=key2,Value=value2 Key=key3,Value=value3
```

Contoh 3: Daftar tag untuk repositori

Perintah berikut mencantumkan tag yang terkait dengan repositori.

```
aws ecr list-tags-for-resource \
     --resource-arn arn:aws:ecr:region:account_id:repository/repository_name
```

Contoh 4: Buat repositori dan tambahkan tag

Perintah berikut membuat repositori bernama `test-repo` dan menambahkan tanda dengan kunci `team` dan nilai `devs`.

```
aws ecr create-repository \
     --repository-name test-repo \
```

```
    --tags Key=team,Value=devs
```

Menghapus tag dari repositori pribadi di Amazon ECR

Anda dapat menghapus tag dari repositori pribadi.

Untuk menghapus tag dari repositori pribadi ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk digunakan.
3. Pada halaman Repositori, pilih kotak centang di sebelah repositori tempat Anda ingin menghapus tag.
4. Dari menu Tindakan, pilih Tag repositori.
5. Pada halaman tag Repositori, pilih Edit.
6. Pada halaman Edit tag repositori, pilih Hapus untuk setiap tag yang ingin Anda hapus, dan pilih Simpan.

Untuk menghapus tag dari repositori pribadi ()AWS CLI

Anda dapat menghapus satu atau beberapa tag dengan menggunakan AWS CLI atau API.

- AWS CLI - untag-sumber daya
- Tindakan API - UntagResource

Contoh berikut menunjukkan cara menghapus tag dari repositori menggunakan. AWS CLI

```
aws ecr untag-resource \
    --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
    --tag-keys tag_key
```

Gambar pribadi di Amazon ECR

Amazon ECR menyimpan gambar Docker, gambar Open Container Initiative (OCI), dan artefak yang kompatibel dengan OCI di repositori pribadi. Anda dapat menggunakan CLI Docker, atau klien pilihan Anda, untuk mendorong dan menarik gambar ke dan dari repositori Anda.

Dengan dukungan Amazon ECR untuk OCI v1.1, Anda dapat menyimpan dan mengelola artefak referensi yang ditentukan oleh OCI Referrers API. Artefak termasuk tanda tangan, Software Bill of Materials (SBoMs), bagan helm, hasil pemindaian, dan pengesahan. Satu set artefak untuk gambar kontainer ditransfer dengan wadah itu dan disimpan sebagai gambar terpisah yang dihitung sebagai gambar yang dikonsumsi untuk repositori Anda.

Menghapus tanda tangan dan artefak lainnya dari repositori pribadi Amazon ECRHalaman Menandatangani gambar di Amazon ECR dan halaman memberikan contoh cara menggunakan artefak terkait tanda tangan. Untuk informasi selengkapnya tentang penandatanganan gambar kontainer, lihat Menandatangani gambar kontainer di Panduan AWS Signer Pengembang.

Topik

Mendorong gambar ke repositori pribadi Amazon ECR

Anda dapat mendorong gambar Docker, daftar manifes, dan gambar Open Container Initiative (OCI) dan artefak yang kompatibel ke repositori pribadi Anda.

Amazon ECR menyediakan cara untuk mereplikasi gambar Anda ke repositori lain. Dengan menentukan konfigurasi replikasi dalam pengaturan registri pribadi Anda, Anda dapat mereplikasi di seluruh Wilayah di registri Anda sendiri dan di berbagai akun. Untuk informasi selengkapnya, lihat Pengaturan registri pribadi di Amazon ECR.

> **Note**
>
> Jika Anda mendorong gambar yang saat ini diarsipkan, gambar itu akan secara otomatis dipulihkan dan dihapus dari arsip. Untuk informasi selengkapnya tentang pengarsipan dan pemulihan gambar, lihat. Mengarsipkan gambar di Amazon ECR

Saat pemasangan gumpalan registri diaktifkan dan parameter pemasangan disertakan, Amazon ECR secara otomatis memeriksa lapisan yang ada di registri Anda selama operasi push. Jika lapisan sudah ada di repositori lain dalam registri yang sama, Amazon ECR akan memasang lapisan yang ada alih-alih mengunggah duplikat. Untuk informasi selengkapnya, lihat Pemasangan gumpalan di Amazon ECR.

Topik

- Izin IAM untuk mendorong gambar ke repositori pribadi Amazon ECR
- Mendorong gambar Docker ke repositori pribadi Amazon ECR
- Mendorong gambar multi-arsitektur ke repositori pribadi Amazon ECR
- Mendorong bagan Helm ke repositori pribadi Amazon ECR

Izin IAM untuk mendorong gambar ke repositori pribadi Amazon ECR

Pengguna memerlukan izin IAM untuk mendorong gambar ke repositori pribadi Amazon ECR. Mengikuti praktik terbaik pemberian hak istimewa paling sedikit, Anda dapat memberikan akses ke repositori tertentu. Anda juga dapat memberikan akses ke semua repositori.

Pengguna harus mengautentikasi ke setiap registri Amazon ECR yang ingin mereka dorong gambar dengan meminta token otorisasi. Amazon ECR menyediakan beberapa kebijakan AWS terkelola

untuk mengontrol akses pengguna pada berbagai tingkatan. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Registry.

Anda juga dapat membuat kebijakan IAM Anda sendiri. Kebijakan IAM berikut memberikan izin yang diperlukan untuk mendorong gambar ke repositori tertentu. Untuk membatasi izin untuk repositori tertentu, gunakan Nama Sumber Daya Amazon (ARN) lengkap dari repositori.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:us-
east-1:111122223333:repository/repository-name"
        },
        {
            "Effect": "Allow",
            "Action": "ecr:GetAuthorizationToken",
            "Resource": "*"
        }
    ]
}
```

Kebijakan IAM berikut memberikan izin yang diperlukan untuk mendorong gambar ke semua repositori.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                    "ecr:CompleteLayerUpload",
                    "ecr:GetAuthorizationToken",
                    "ecr:UploadLayerPart",
                    "ecr:InitiateLayerUpload",
                    "ecr:BatchCheckLayerAvailability",
                    "ecr:PutImage",
                    "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:us-west-2:111122223333:repository/*"
        }
    ]
}
```

Mendorong gambar Docker ke repositori pribadi Amazon ECR

Anda dapat mendorong gambar kontainer Anda ke repositori Amazon ECR dengan perintah. docker push

Amazon ECR juga mendukung pembuatan dan mendorong daftar manifes Docker yang digunakan untuk gambar multi-arsitektur. Untuk informasi, lihat Mendorong gambar multi-arsitektur ke repositori pribadi Amazon ECR.

Untuk mendorong gambar Docker ke repositori Amazon ECR

Repositori Amazon ECR harus ada sebelum Anda mendorong gambar, atau Anda harus memiliki template pembuatan repositori yang ditentukan. Untuk informasi selengkapnya, lihat Membuat repositori pribadi Amazon ECR untuk menyimpan gambar dan Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.

1. Autentikasi klien Docker Anda ke registrasi Amazon ECR di mana Anda berniat untuk mendorong citra Anda. Token autentikasi harus diperoleh untuk setiap registrasi yang digunakan, dan token berlaku selama 12 jam. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

   Untuk mengautentikasi Docker ke registri Amazon ECR, jalankan perintah. aws ecr get-login-password Ketika meneruskan token otorisasi ke perintah docker login, gunakan nilai `AWS` untuk nama pengguna dan tentukan URI registri Amazon ECR yang ingin Anda autentikasi. Jika melakukan autentikasi untuk beberapa registri, Anda harus mengulangi perintah tersebut untuk setiap registri.

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

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

2. Jika repositori gambar Anda belum ada di registri yang ingin Anda dorong, dan Anda memiliki templat pembuatan repositori yang ditentukan, Anda dapat mendorong gambar Anda menggunakan awalan template pembuatan repositori dan nama repositori yang Anda inginkan. ECR akan secara otomatis membuat repositori untuk Anda menggunakan pengaturan yang telah ditentukan dari template pembuatan repositori Anda.

   Jika Anda tidak memiliki template pembuatan repositori yang cocok ditentukan, Anda harus membuat repositori. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi atau Membuat repositori pribadi Amazon ECR untuk menyimpan gambar.

3. Identifikasi gambar lokal untuk didorong. Jalankan docker images perintah untuk membuat daftar gambar kontainer di sistem Anda.

   ```
   docker images
   ```

   Anda dapat mengidentifikasi gambar dengan *repository:tag* nilai atau ID gambar dalam output perintah yang dihasilkan.

4. Tandai citra Anda dengan registrasi Amazon ECR, repositori, dan kombinasi nama tanda citra opsional untuk digunakan. Format registri adalah *aws_account_id*.dkr.ecr.*region*.amazonaws.com. Nama repositori harus sesuai dengan repositori yang Anda buat untuk citra Anda. Jika Anda menghilangkan tanda citra, maka kami berasumsi bahwa tandanya adalah`latest`.

   Contoh berikut menandai gambar lokal dengan ID *e9ae3c220b23* sebagai*aws_account_id*.dkr.ecr.*region*.amazonaws.com/my-repository:tag.

```
docker tag e9ae3c220b23 aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:tag
```

5. Mendorong citra menggunakan perintah docker push:

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag
```

6. (Opsional) Terapkan tanda tambahan untuk citra Anda dan dorong tanda tersebut ke Amazon ECR dengan mengulangi Step 4 dan Step 5.

Mendorong gambar multi-arsitektur ke repositori pribadi Amazon ECR

Anda dapat mendorong gambar multi-arsitektur ke repositori Amazon ECR dengan membuat dan mendorong daftar manifes Docker. Daftar manifes adalah daftar citra yang dibuat dengan menentukan satu atau lebih nama citra. Dalam kebanyakan kasus, daftar manifes dibuat dari gambar yang melayani fungsi yang sama tetapi untuk sistem operasi atau arsitektur yang berbeda. Daftar manifes yang tidak diperlukan. Untuk informasi selengkapnya, lihat manifes docker.

Sebuah daftar manifes dapat ditarik atau direferensikan dalam definisi tugas Amazon ECS atau Amazon EKS pod spec seperti citra Amazon ECR lainnya.

Prasyarat

- Di CLI Docker Anda, aktifkan fitur eksperimental. Untuk informasi tentang fitur eksperimental, lihat Fitur eksperimental dalam dokumentasi Docker.
- Repositori Amazon ECR harus tersedia sebelum Anda mendorong citra. Untuk informasi selengkapnya, lihat the section called “Membuat repositori untuk menyimpan gambar”.
- Gambar harus didorong ke repositori Anda sebelum Anda membuat manifes Docker. Untuk informasi tentang cara mendorong citra, lihat Mendorong gambar Docker ke repositori pribadi Amazon ECR.

Untuk mendorong gambar Docker multi-arsitektur ke repositori Amazon ECR

1. Autentikasi klien Docker Anda ke registrasi Amazon ECR di mana Anda berniat untuk mendorong citra Anda. Token autentikasi harus diperoleh untuk setiap registrasi yang digunakan, dan token berlaku selama 12 jam. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

Untuk mengautentikasi Docker ke registri Amazon ECR, jalankan perintah. aws ecr get-login-password Ketika meneruskan token otorisasi ke perintah docker login, gunakan nilai `AWS` untuk nama pengguna dan tentukan URI registri Amazon ECR yang ingin Anda autentikasi. Jika melakukan autentikasi untuk beberapa registri, Anda harus mengulangi perintah tersebut untuk setiap registri.

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

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

2. Masukkan citra di repositori Anda, konfirmasikan tanda citra.

```
aws ecr describe-images --repository-name my-repository
```

3. Buat daftar manifes Docker. Perintah `manifest create` memverifikasi bahwa citra yang direferensikan sudah berada di repositori Anda dan menciptakan manifes lokal.

```
docker manifest create aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:image_one_tag aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:image_two
```

4. (Opsional) Periksa daftar manifes Docker. Hal ini memungkinkan Anda untuk mengonfirmasi ukuran dan digest untuk setiap manifes citra yang direferensikan dalam daftar manifes.

```
docker manifest inspect aws_account_id.dkr.ecr.region.amazonaws.com/my-repository
```

5. Dorong daftar manifes Docker ke repositori Amazon ECR Anda.

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

Mendorong bagan Helm ke repositori pribadi Amazon ECR

Anda dapat mendorong artefak Open Container Initiative (OCI) ke repositori Amazon ECR. Untuk melihat contoh fungsi ini, gunakan langkah-langkah berikut untuk mendorong bagan Helm ke Amazon ECR.

Untuk informasi tentang menggunakan bagan Helm yang dihosting Amazon ECR Anda dengan Amazon EKS, lihat. Memasang bagan Helm di kluster Amazon EKS

Untuk mendorong grafik Helm ke repositori Amazon ECR

1. Instal versi terbaru klien Helm. Langkah-langkah ini ditulis menggunakan versi `3.18.6` Helm. Untuk kompatibilitas dengan versi Kubernetes yang didukung Amazon EKS, gunakan Helm versi 3.9 atau yang lebih baru. Untuk informasi selengkapnya, lihat Helm.
2. Lakukan langkah-langkah berikut untuk membuat grafik tes Helm. Untuk informasi lebih lanjut, lihat Helm Docs - Memulai.

   a. Buat grafik Helm bernama `helm-test-chart` dan hapus isi direktori `templates`.

   ```
   helm create helm-test-chart
   rm -rf ./helm-test-chart/templates/*
   ```

   b. Buat `ConfigMap` di `templates` folder.

   ```
   cd helm-test-chart/templates
   cat <<EOF > configmap.yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: helm-test-chart-configmap
   data:
     myvalue: "Hello World"
   EOF
   ```

3. Package grafik. Outputnya akan berisi nama file bagan paket yang Anda gunakan saat mendorong bagan Helm.

   ```
   cd ../..
   helm package helm-test-chart
   ```

   Output

```
Successfully packaged chart and saved it to: /Users/username/helm-test-
chart-0.1.0.tgz
```

4. Buat repositori untuk menyimpan grafik Helm Anda. Nama repositori Anda harus sesuai dengan nama yang Anda gunakan saat membuat bagan Helm di langkah 2. Untuk informasi selengkapnya, lihat Membuat repositori pribadi Amazon ECR untuk menyimpan gambar.

```
aws ecr create-repository \
     --repository-name helm-test-chart \
     --region us-west-2
```

5. Autentikasi Helm client Anda ke registrasi Amazon ECR di mana Anda berniat untuk mendorong citra Anda. Token autentikasi harus diperoleh untuk setiap registrasi yang digunakan, dan token berlaku selama 12 jam. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

```
aws ecr get-login-password \
     --region us-west-2 | helm registry login \
     --username AWS \
     --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

6. Dorong bagan Helm menggunakan helm push perintah. Outputnya harus mencakup URI repositori Amazon ECR dan SHA digest.

```
helm push helm-test-chart-0.1.0.tgz
 oci://aws_account_id.dkr.ecr.region.amazonaws.com/
```

7. Deskripsikan grafik Helm Anda.

```
aws ecr describe-images \
     --repository-name helm-test-chart \
     --region us-west-2
```

Outputnya, verifikasi bahwa parameter `artifactMediaType` menunjukkan jenis artefak yang tepat.

```
{
    "imageDetails": [
        {
            "registryId": "aws_account_id",
```

```
            "repositoryName": "helm-test-chart",
            "imageDigest":
 "sha256:dd8aebdda7df991a0ffe0b3d6c0cf315fd582cd26f9755a347a52adEXAMPLE",
            "imageTags": [
                "0.1.0"
            ],
            "imageSizeInBytes": 1620,
            "imagePushedAt": "2021-09-23T11:39:30-05:00",
            "imageManifestMediaType": "application/vnd.oci.image.manifest.v1+json",
            "artifactMediaType": "application/vnd.cncf.helm.config.v1+json"
        }
    ]
}
```

8. (Opsional) Untuk langkah tambahan, instal Helm `ConfigMap` dan mulai dengan Amazon EKS. Untuk informasi selengkapnya, lihat Memasang bagan Helm di kluster Amazon EKS.

Menghapus tanda tangan dan artefak lainnya dari repositori pribadi Amazon ECR

Anda dapat menggunakan klien ORAS untuk membuat daftar dan menghapus tanda tangan dan artefak jenis referensi lainnya dari repositori pribadi Amazon ECR. Menghapus tanda tangan dan artefak referensi lainnya mirip dengan cara gambar dihapus (lihat). Menghapus gambar di Amazon ECR Berikut adalah cara membuat daftar artefak dan menghapus tanda tangan:

Untuk mengelola artefak gambar menggunakan ORAS CLI

1. Instal dan konfigurasikan klien ORAS.

   Untuk informasi tentang menginstal dan mengonfigurasi klien ORAS, lihat Instalasi di dokumentasi ORAS.

2. Untuk mencantumkan artefak yang tersedia untuk gambar Amazon ECR, gunakan`oras discover`, diikuti dengan nama gambar:

   ```
   oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
   ```

   Outputnya akan terlihat serupa dengan ini:

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
### application/vnd.cncf.notary.signature
    ### sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
    ### sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

3. Untuk menghapus tanda tangan menggunakan CLI ORAS, diberikan contoh sebelumnya, jalankan perintah berikut:

```
oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

   Outputnya akan terlihat serupa dengan ini:

```
Are you sure you want to delete the manifest "111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and
 all tags associated with it? [y/N] y
```

4. Tekan y. Artefak harus dihapus.

Untuk memecahkan masalah penghapusan artefak

Jika penghapusan tanda tangan, seperti yang baru saja ditampilkan, akan gagal, output yang mirip dengan berikut ini akan muncul.

```
Error response from registry: failed to delete 111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42:
unsupported: Requested image referenced by manifest list:
 [sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b]
```

Kegagalan ini dapat terjadi saat menghapus gambar yang didorong sebelum peluncuran OCI 1.1. Seperti disebutkan dalam kesalahan, Anda harus menghapus manifes yang merujuk gambar sebelum Anda dapat menghapus gambar sebagai berikut:

1. Untuk menghapus manifes yang terkait dengan tanda tangan yang ingin Anda hapus, ketik:

```
 oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b
```

Outputnya akan terlihat serupa dengan ini:

```
Are you sure you want to delete the manifest
 "sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b" and all
 tags associated with it? [y/N] y
```

2. Tekan y. Manifes harus dihapus.
3. Dengan manifes hilang, Anda dapat menghapus tanda tangan:

```
 oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

Outputnya akan terlihat mirip dengan ini. Tekan y.

```
Are you sure you want to delete the manifest
 "sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and all
 tags associated with it? [y/N] y
Deleted [registry] 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

4. Untuk melihat bahwa tanda tangan telah dihapus, ketik:

```
oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
```

Outputnya akan terlihat serupa dengan ini:

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
### application/vnd.cncf.notary.signature
```

```
### sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

Melihat detail gambar di Amazon ECR

Setelah Anda mendorong gambar ke repositori Anda, Anda dapat melihat informasi tentangnya. Detail yang disertakan adalah sebagai berikut:

- URI citra
- Tanda citra
- Tipe media artefak
- Tipe manifes citra
- Status pemindaian
- Ukuran citra dalam MB
- Ketika citra telah didorong ke repositori
- Status replikasi

Untuk melihat detail citra (Konsol Manajemen AWS)

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositories.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori berisi citra Anda.
3. Di panel navigasi, di bawah Registri pribadi, pilih Repositori.
4. Pada halaman Private repositories, pilih repositori untuk dilihat.
5. Pada *repository_name* halaman Repositori:, pilih gambar untuk melihat detailnya.

Menarik gambar ke lingkungan lokal Anda dari repositori pribadi Amazon ECR

Jika Anda ingin menjalankan gambar Docker yang tersedia di Amazon ECR, Anda dapat menariknya ke lingkungan lokal Anda dengan perintah docker pull. Anda dapat melakukan ini baik dari registri default Anda atau dari registri yang terkait dengan AWS akun lain.

Untuk menggunakan image Amazon ECR dalam definisi tugas Amazon ECS, lihat. Menggunakan citra Amazon ECR dengan Amazon ECS

> **Important**
>
> Anda tidak dapat menarik gambar yang diarsipkan. Gambar yang diarsipkan harus dipulihkan sebelum dapat ditarik. Untuk informasi selengkapnya tentang pengarsipan dan pemulihan gambar, lihat. Mengarsipkan gambar di Amazon ECR

> **Important**
>
> Amazon ECR mengharuskan pengguna memiliki izin untuk melakukan panggilan ke API `ecr:GetAuthorizationToken` melalui kebijakan IAM sebelum mereka dapat melakukan autentikasi ke registrasi dan mendorong atau menarik citra dari repositori Amazon ECR. Amazon ECR menyediakan beberapa kebijakan AWS terkelola untuk mengontrol akses pengguna pada berbagai tingkatan. Untuk informasi tentang kebijakan AWS terkelola untuk Amazon ECR, lihatAWS kebijakan terkelola untuk Amazon Elastic Container Registry.

Untuk menarik gambar Docker dari repositori Amazon ECR

1. Autentikasi klien Docker Anda ke registrasi Amazon ECR di mana Anda berniat untuk menarik citra Anda. Token autentikasi harus diperoleh untuk setiap registrasi yang digunakan, dan token berlaku selama 12 jam. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.
2. (Opsional) Identifikasi citra untuk menarik.
   - Anda dapat memasukkan repositori di registrasi dengan perintah aws ecr describe-repositories:

     ```
     aws ecr describe-repositories
     ```

     Contoh registri di atas memiliki repositori yang disebut. `amazonlinux`
   - Anda dapat mendeskripsikan citra dalam repositori dengan perintah aws ecr describe-images:

     ```
     aws ecr describe-images --repository-name amazonlinux
     ```

Contoh repositori di atas memiliki citra yang ditandai sebagai `latest` dan `2016.09`, dengan digest `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807` citra.

3. Tarik citra menggunakan perintah docker pull. Format nama citra harus *registry*`/`*repository*` [:`*tag*`]` untuk menariknya dengan tanda atau *registry*`/`*repository*`[@`*digest*`]` untuk menariknya dengan digest.

```
docker pull aws_account_id.dkr.ecr.us-west-2.amazonaws.com/amazonlinux:latest
```

> **Important**
>
> Jika Anda menerima pesan kesalahan *repository-url* `not found: does not exist or no pull access`, Anda mungkin perlu mengautentikasi klien Docker Anda dengan Amazon ECR. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

Menarik gambar wadah Amazon Linux

Citra kontainer Amazon Linux dibangun dari komponen perangkat lunak yang sama yang termasuk dalam Amazon Linux AMI. Gambar wadah Amazon Linux tersedia untuk digunakan di lingkungan apa pun sebagai gambar dasar untuk beban kerja Docker. Jika Anda menggunakan Amazon Linux AMI untuk aplikasi di Amazon EC2, Anda dapat mengkontainerisasi aplikasi Anda dengan image container Amazon Linux.

Anda dapat menggunakan image container Amazon Linux di lingkungan pengembangan lokal Anda dan kemudian mendorong aplikasi Anda untuk AWS menggunakan Amazon ECS. Untuk informasi selengkapnya, lihat Menggunakan citra Amazon ECR dengan Amazon ECS.

Gambar wadah Amazon Linux tersedia di Amazon ECR Public dan di Docker Hub. Untuk dukungan untuk image container Amazon Linux, buka forum AWS pengembang.

Untuk menarik gambar wadah Amazon Linux dari Amazon ECR Public

1. Otentikasi klien Docker Anda ke registri Publik Amazon Linux. Token autentikasi berlaku selama 12 jam. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

> **Note**
>
> ecr-publicPerintah tersedia di AWS CLI awal dengan versi`1.18.1.187`, namun kami sarankan untuk menggunakan versi terbaru dari versi AWS CLI. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface .

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin public.ecr.aws
```

Output adalah sebagai berikut:

```
Login succeeded
```

2. Tarik citra kontainer Amazon Linux menggunakan perintah docker pull. Untuk melihat gambar penampung Amazon Linux di Galeri Publik Amazon ECR, lihat Galeri Publik Amazon ECR - amazonlinux.

```
docker pull public.ecr.aws/amazonlinux/amazonlinux:latest
```

3. (Opsional) Jalankan kontainer secara lokal.

```
docker run -it public.ecr.aws/amazonlinux/amazonlinux /bin/bash
```

Untuk menarik citra kontainer Amazon Linux dari Docker Hub

1. Tarik citra kontainer Amazon Linux menggunakan perintah docker pull.

```
docker pull amazonlinux
```

2. (Opsional) Jalankan kontainer secara lokal.

```
docker run -it amazonlinux:latest /bin/bash
```

Menghapus gambar di Amazon ECR

Jika Anda selesai menggunakan citra, Anda dapat menghapusnya dari repositori. Jika Anda selesai dengan repositori, Anda dapat menghapus seluruh repositori dan semua citra di dalamnya. Untuk informasi selengkapnya, lihat Menghapus repositori pribadi di Amazon ECR.

Sebagai alternatif untuk menghapus citra secara manual, Anda dapat membuat kebijakan siklus hidup repositori yang memberikan kontrol lebih atas manajemen siklus hidup citra dalam repositori Anda. Kebijakan siklus hidup mengotomatisasi proses ini untuk Anda. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.

> **Note**
>
> Jika repositori Anda memiliki campuran gambar, beberapa di antaranya didorong sebelum Amazon ECR mendukung OCI v1.1, beberapa tanda tangan akan memiliki indeks gambar atau daftar manifes yang menunjuk ke sana. Akibatnya, saat Anda menghapus gambar pra-OCI v1.1, Anda mungkin perlu menghapus daftar manifes yang mereferensikan gambar secara manual untuk menghapus artefak.

Untuk menghapus citra (Konsol Manajemen AWS)

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositories.
2. Dari bilah navigasi, pilih Wilayah yang berisi citra yang akan dihapus.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih repositori yang berisi citra yang akan dihapus.
5. Pada ***repository_name*** halaman Repositori: pilih kotak di sebelah kiri gambar yang akan dihapus dan pilih Hapus.
6. Di kotak dialog Hapus citra, verifikasi bahwa citra yang dipilih perlu dihapus kemudian pilih Hapus.

Untuk menghapus citra (AWS CLI)

1. Masukkan citra di repositori Anda. Citra yang ditandai akan memiliki kedua digest citra serta daftar tanda terkait. Citra yang tidak tertagged hanya akan memiliki digest citra.

```
aws ecr list-images \
     --repository-name my-repo
```

2. (Opsional) Hapus tanda yang tidak diinginkan untuk citra tersebut dengan menentukan tanda yang terkait dengan citra yang ingin Anda hapus. Bila tanda terakhir dihapus dari sebuah citra, maka citra tersebut juga akan dihapus.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageTag=tag1 imageTag=tag2
```

3. Hapus citra yang ditandai atau tidak ditandai dengan menentukan digest citra. Ketika Anda menghapus citra dengan referensi digestnya, maka citra dan semua tandanya akan dihapus.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
```

Untuk menghapus beberapa citra, Anda dapat menentukan beberapa tanda citra atau digest citra dalam permintaan tersebut.

```
aws ecr batch-delete-image \
     --repository-name my-repo \
     --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
 imageDigest=sha256:f5t0e245ssffc302b13e25962d8f7a0bd304EXAMPLE
```

Mengarsipkan gambar di Amazon ECR

Apa kelas penyimpanan arsip ECR?

Kelas penyimpanan arsip Amazon ECR adalah kelas penyimpanan baru yang menyediakan penyimpanan jangka panjang berbiaya rendah untuk gambar kontainer. Amazon ECR menawarkan dua kelas penyimpanan:

- Kelas penyimpanan standar ECR — Kelas penyimpanan default untuk gambar aktif yang diakses secara teratur.

- Kelas penyimpanan arsip ECR — Kelas penyimpanan berbiaya rendah untuk gambar yang jarang diakses tetapi perlu dipertahankan untuk kepatuhan atau referensi jangka panjang. Kelas penyimpanan arsip memberikan penghematan biaya untuk sejumlah besar gambar dibandingkan dengan kelas penyimpanan Standar untuk retensi gambar jangka panjang. Untuk informasi harga terperinci, lihat harga Amazon ECR.

Untuk mengarsipkan gambar, Anda memiliki dua opsi. Pertama, Anda dapat mengonfigurasi aturan siklus hidup untuk mengarsipkan gambar secara otomatis berdasarkan:

- Waktu sejak gambar didorong
- Waktu sejak gambar terakhir ditarik
- Jumlah gambar dalam repositori

Anda juga dapat mengonfigurasi pengaturan untuk menghapus gambar secara permanen setelah diarsipkan untuk jangka waktu tertentu. Lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR untuk informasi lebih lanjut.

Anda juga dapat mengarsipkan gambar menggunakan konsol Amazon ECR atau AWS CLI. Lihat Mengarsipkan gambar untuk informasi lebih lanjut.

Saat Anda perlu menggunakan gambar yang diarsipkan lagi, Anda dapat mengembalikannya kembali ke kelas penyimpanan ECR Standard. Anda dapat mengharapkan ECR untuk mengembalikan gambar dalam waktu 20 menit. Gambar yang dipulihkan berperilaku seperti gambar yang baru didorong dan segera tersedia untuk digunakan saat pemulihan selesai. Gambar yang dipulihkan tunduk pada kebijakan pemindaian, replikasi, dan siklus hidup repositori. Lihat Memulihkan gambar untuk informasi lebih lanjut.

Mengarsipkan gambar

Anda dapat mengarsipkan gambar secara manual menggunakan konsol Amazon ECR atau AWS CLI, atau secara otomatis menggunakan kebijakan siklus hidup. Saat gambar diarsipkan:

- Gambar dipindahkan ke kelas penyimpanan arsip.
- Gambar yang diarsipkan tidak dapat ditarik. Permintaan untuk menarik gambar yang diarsipkan akan gagal dengan kesalahan 404.

- Meskipun gambar tidak dapat ditarik, itu masih dapat dijelaskan menggunakan describe-images perintah, atau terdaftar menggunakan list-images perintah. Status gambar akan ditampilkan sebagai`ARCHIVED`.
- Gambar yang diarsipkan memiliki durasi penyimpanan minimum 90 hari. Anda tidak dapat mengonfigurasi kebijakan siklus hidup yang menghapus gambar yang telah berada di arsip kurang dari 90 hari. Jika Anda harus menghapus gambar yang telah diarsipkan kurang dari 90 hari, Anda harus menggunakan batch-delete-image API, tetapi Anda akan dikenakan biaya untuk durasi penyimpanan minimum 90 hari.
- Gambar muncul di tab Gambar yang diarsipkan dalam tampilan repositori (tab ini hanya akan muncul jika setidaknya satu gambar diarsipkan di repositori).
- Gambar dapat dipulihkan sebagai gambar aktif dengan memilihnya secara manual untuk dipulihkan atau dengan mendorong kembali gambar ke repositori.
- Gambar akan dihapus jika repositori memiliki kebijakan siklus hidup yang menghapus gambar dengan kriteria seperti waktu dalam arsip.

Konsol Manajemen AWS

Untuk mengarsipkan gambar

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositories.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori dengan gambar yang ingin Anda arsipkan.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih repositori yang berisi gambar yang ingin Anda arsipkan.
5. Pilih gambar yang ingin Anda arsipkan. Anda akan melihat Detail Gambar.
6. Untuk mengarsipkan gambar, pilih tombol Arsip dan pilih Konfirmasi saat Anda diminta.
7. Jika ini adalah gambar arsip pertama di repositori, tab Gambar yang diarsipkan baru muncul dengan gambar yang baru diarsipkan. Jika ada gambar arsip lainnya, gambar ini akan ditambahkan ke tab itu.

AWS CLI

Untuk mengarsipkan gambar

- Gunakan update-image-storage-class perintah untuk mengarsipkan gambar dengan memperbarui kelas penyimpanannya keARCHIVE:

```
aws ecr update-image-storage-class \
    --repository-name my-repository \
    --image-id
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
    --target-storage-class ARCHIVE
```

Untuk mengarsipkan gambar menggunakan kebijakan siklus hidup

- Anda dapat mengonfigurasi aturan arsip untuk repositori menggunakan kebijakan siklus hidup untuk mengarsipkan gambar secara otomatis. Kebijakan siklus hidup memungkinkan Anda mengarsipkan gambar secara otomatis berdasarkan kriteria seperti:
  - Waktu sejak gambar didorong
  - Waktu sejak gambar terakhir ditarik
  - Jumlah maksimum gambar untuk tetap aktif

  Anda juga dapat mengonfigurasi kebijakan siklus hidup untuk menghapus gambar secara permanen setelah diarsipkan untuk jangka waktu tertentu. Untuk informasi selengkapnya dan contoh kebijakan siklus hidup dengan tindakan arsip, lihat. Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR

  > **Note**
  >
  > Gambar yang diarsipkan memiliki durasi penyimpanan minimum 90 hari. Anda tidak dapat mengonfigurasi kebijakan siklus hidup yang menghapus gambar yang telah berada di arsip kurang dari 90 hari. Jika Anda harus menghapus gambar yang telah diarsipkan kurang dari 90 hari, Anda harus menggunakan batch-delete-image API, tetapi Anda akan dikenakan biaya untuk durasi penyimpanan minimum 90 hari.

Saat Anda mendeskripsikan gambar menggunakan describe-images perintah, gambar yang diarsipkan memiliki file`image-status`. `ARCHIVED` Anda dapat memfilter gambar dengan `image-status` hanya melihat gambar yang diarsipkan atau hanya gambar aktif.

Memulihkan gambar

Saat Anda mengembalikan gambar yang diarsipkan, gambar tersebut dipindahkan dari kelas penyimpanan Arsip ECR kembali ke kelas penyimpanan Standar ECR. Gambar yang dipulihkan dibebankan pada tingkat penyimpanan standar. Proses pemulihan melakukan tindakan serupa yang terjadi saat gambar baru dibuat:

- Gambar menjadi tersedia untuk ditarik saat pemulihan selesai. Pemulihan biasanya memakan waktu hingga 20 menit, meskipun mungkin selesai lebih cepat.
- Jika pemindaian pada push diaktifkan untuk repositori, gambar yang dipulihkan akan dipindai. Perhatikan bahwa hasil pemindaian sebelumnya dari sebelum gambar diarsipkan tidak akan tersedia.
- Jika replikasi dikonfigurasi untuk repositori, gambar yang dipulihkan akan direplikasi jika replikasi diaktifkan pada saat pemulihan.
- Gambar yang dipulihkan muncul di daftar gambar aktif.

Memulihkan gambar biasanya membutuhkan waktu hingga 20 menit, meskipun mungkin selesai lebih cepat. Selama proses pemulihan, gambar tetap dalam keadaan diarsipkan dan tidak dapat ditarik sampai pemulihan selesai.

Konsol Manajemen AWS

Untuk mengembalikan gambar yang diarsipkan

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositories.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori dengan gambar yang diarsipkan yang ingin Anda pulihkan.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih repositori yang berisi gambar yang diarsipkan.
5. Pilih tab Gambar yang diarsipkan.
6. Pilih gambar yang diarsipkan yang ingin Anda pulihkan.
7. Pilih Pulihkan dan konfirmasikan tindakan pemulihan.

8. Tunggu hingga pemulihan selesai. Gambar akan muncul di daftar gambar aktif setelah restorasi selesai.

AWS CLI

Untuk mengembalikan gambar yang diarsipkan

- Gunakan update-image-storage-class perintah untuk memulihkan gambar yang diarsipkan dengan memperbarui kelas penyimpanannya ke`STANDARD`:

```
aws ecr update-image-storage-class \
    --repository-name my-repository \
    --image-id
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
    --target-storage-class STANDARD
```

Saat Anda mendeskripsikan gambar menggunakan describe-images perintah, gambar yang sedang dipulihkan memiliki `image-status` file`ACTIVATING`. Anda dapat memfilter gambar `image-status` dengan nilai `ACTIVATING` untuk melihat gambar yang sedang dipulihkan.

Metode alternatif untuk mengembalikan gambar yang diarsipkan adalah dengan mendorong kembali gambar ke repositori. Saat Anda mendorong gambar yang saat ini diarsipkan, gambar itu akan segera dipulihkan dan dihapus dari arsip.

Menandai ulang gambar di Amazon ECR

Dengan citra Docker Image Manifest V2 Skema 2, Anda dapat menggunakan opsi `--image-tag` printah put-image untuk menandai ulang citra yang tersedia. Anda dapat menandai ulang tanpa menarik atau mendorong citra dengan Docker. Untuk citra yang lebih besar, proses ini menghemat sejumlah besar bandwidth jaringan dan waktu yang diperlukan untuk menandai ulang citra.

Untuk menandai ulang citra (AWS CLI)

Untuk menandai ulang gambar dengan AWS CLI

1. Gunakan batch-get-image perintah untuk mendapatkan manifes gambar untuk gambar untuk retag dan menulis ke file. Dalam contoh ini, manifes untuk gambar dengan tag*latest*,, di repositori*amazonlinux*, ditulis ke variabel lingkungan bernama. *MANIFEST*

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --image-ids
 imageTag=latest --output text --query 'images[].imageManifest')
```

2. Gunakan opsi `--image-tag` perintah put-image untuk menempatkan manifes citra ke Amazon ECR dengan tanda baru. Dalam contoh ini, gambar ditandai sebagai*2017.03*.

> **Note**
>
> Jika `--image-tag` opsi tidak tersedia di versi Anda AWS CLI, tingkatkan ke versi terbaru. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface .

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 --image-
manifest "$MANIFEST"
```

3. Verifikasi bahwa tanda citra baru Anda terpasang pada citra Anda. Pada output berikut, citra memiliki tanda `latest` dan `2017.03`.

```
aws ecr describe-images --repository-name amazonlinux
```

Outputnya adalah sebagai berikut:

```
{
    "imageDetails": [
        {
            "imageSizeInBytes": 98755613,
            "imageDigest":
 "sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a26EXAMPLE",
            "imageTags": [
                "latest",
                "2017.03"
            ],
            "registryId": "aws_account_id",
            "repositoryName": "amazonlinux",
            "imagePushedAt": 1499287667.0
        }
    ]
}
```

Untuk menandai ulang citra (AWS Tools for Windows PowerShell)

Untuk menandai ulang gambar dengan AWS Tools for Windows PowerShell

1. Gunakan Get-ECRImageBatch cmdlet untuk mendapatkan deskripsi gambar untuk retag dan menuliskannya ke variabel lingkungan. Dalam contoh ini, gambar dengan tag, *latest*, di repositori*amazonlinux*, ditulis ke variabel lingkungan,. *$Image*

   > **Note**
   >
   > Jika Anda tidak memiliki yang Get-ECRImageBatch cmdlet tersedia di sistem Anda, lihat Menyiapkan AWS Tools for Windows PowerShell di Panduan Alat AWS untuk PowerShell Pengguna.

   ```
   $Image = Get-ECRImageBatch -ImageId @{ imageTag="latest" } -
   RepositoryName amazonlinux
   ```

2. Tulis manifes gambar ke variabel *$Manifest* lingkungan.

   ```
   $Manifest = $Image.Images[0].ImageManifest
   ```

3. Gunakan `-ImageTag` opsi Write-ECRImage cmdlet untuk menempatkan manifes gambar ke Amazon ECR dengan tag baru. Dalam contoh ini, gambar ditandai sebagai*2017.09*.

   ```
   Write-ECRImage -RepositoryName amazonlinux -ImageManifest $Manifest -
   ImageTag 2017.09
   ```

4. Verifikasi bahwa tanda citra baru Anda terpasang pada citra Anda. Pada output berikut, citra memiliki tanda `latest` dan `2017.09`.

   ```
   Get-ECRImage -RepositoryName amazonlinux
   ```

   Output adalah sebagai berikut:

   ```
   ImageDigest                                                             ImageTag
   -----------                                                             --------
   sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 latest
   sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 2017.09
   ```

Mencegah tag gambar ditimpa di Amazon ECR

Anda dapat mencegah tag gambar ditimpa dengan mengaktifkan kekekalan tag di repositori. Setelah kekekalan tag dihidupkan, `ImageTagAlreadyExistsException` kesalahan dikembalikan jika Anda mendorong gambar dengan tag yang sudah ada di repositori. Kekekalan tag mempengaruhi semua tag. Anda tidak dapat membuat beberapa tag tidak dapat diubah sementara yang lain tidak.

Anda dapat menggunakan Konsol Manajemen AWS dan AWS CLI alat untuk mengatur mutabilitas tag gambar untuk repositori baru atau untuk repositori yang ada. Untuk membuat repositori menggunakan langkah-langkah konsol, lihat. Membuat repositori pribadi Amazon ECR untuk menyimpan gambar

Mengatur mutabilitas tag gambar ()Konsol Manajemen AWS

Untuk mengatur mutabilitas tag gambar

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositories.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori untuk diedit.
3. Di panel navigasi, pilih Repositori di bawah Registri pribadi.

   Jika Anda tidak melihat Repositori, pilih Registri pribadi untuk memperluas menu dan kemudian pilih Repositori.
4. Pada halaman Repositori pribadi, pilih tombol radio sebelum nama repositori yang ingin Anda atur pengaturan mutabilitas tag gambar.
5. Pilih Tindakan dan kemudian pilih Repositori di bawah Edit.
6. Untuk kekekalan tag Gambar, pilih salah satu pengaturan mutabilitas tag berikut untuk repositori.

   - Mutable - Pilih opsi ini jika Anda ingin tag gambar ditimpa. Direkomendasikan untuk repositori yang menggunakan tindakan pull through cache untuk memastikan Amazon ECR dapat memperbarui image yang tersimpan dalam cache. Selain itu, untuk menonaktifkan pembaruan tag untuk beberapa tag yang dapat berubah, masukkan nama tag atau gunakan wildcard (*) untuk mencocokkan beberapa tag serupa di kotak teks pengecualian tag Mutable.
   - Immutable - Pilih opsi ini jika Anda ingin mencegah tag gambar ditimpa, dan itu berlaku untuk semua tag dan pengecualian di repositori saat mendorong gambar dengan tag yang ada. Amazon ECR mengembalikan pesan `ImageTagAlreadyExistsException` jika Anda mencoba melakukan push image dengan tag yang ada. Selain itu, untuk mengaktifkan pembaruan tag untuk beberapa tag yang tidak dapat diubah, masukkan nama tag atau

gunakan wildcard (*) untuk mencocokkan beberapa tag serupa di kotak teks pengecualian tag Immutable.

7. Untuk pengaturan pemindaian Gambar, sementara Anda dapat menentukan pengaturan pemindaian di tingkat repositori untuk pemindaian dasar, praktik terbaik adalah menentukan konfigurasi pemindaian di tingkat registri pribadi. Tentukan pengaturan pemindaian di registri pribadi memungkinkan Anda mengaktifkan pemindaian yang ditingkatkan atau pemindaian dasar serta menentukan filter untuk menentukan repositori mana yang dipindai. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.
8. Untuk pengaturan Enkripsi, ini adalah bidang tampilan saja karena pengaturan enkripsi untuk repositori tidak dapat diubah setelah repositori dibuat.
9. Pilih Simpan untuk memperbarui pengaturan repositori.

Mengatur mutabilitas tag gambar ()AWS CLI

Untuk membuat repositori dengan tanda tetap terkonfigurasi

Gunakan salah satu dari perintah berikut untuk membuat repositori citra baru dengan tanda tetap terkonfigurasi.

- create-repository ()AWS CLI dengan mutabilitas tag gambar

```
aws ecr create-repository --repository-name name --image-tag-mutability IMMUTABLE --
region us-east-2
```

- create-repository (AWS CLI) dengan filter pengecualian mutabilitas tag gambar

```
aws ecr create-repository --repository-name name --image-tag-
mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters
 filterType=WILDCARD,filter=filter-text --region us-east-2
```

- New-ECRRepository(AWS Tools for Windows PowerShell) dengan mutabilitas tag gambar

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE -Region us-
east-2 -Force
```

- New-ECRRepository(AWS Tools for Windows PowerShell) dengan filter pengecualian mutabilitas tag gambar

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE_WITH_EXCLUSION
 -ImageTagMutabilityExclusionFilter @{FilterType=WILDCARD Filter=filter-text} -
Region us-east-2 -Force
```

Untuk memperbarui pengaturan mutabilitas tag gambar untuk repositori

Gunakan salah satu dari perintah berikut untuk memperbarui pengaturan ketetapan tanda citra untuk repositori yang tersedia.

- put-image-tag-mutability () dengan mutabilitas tag gambar AWS CLI

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE --region us-east-2
```

- put-image-tag-mutability ()AWS CLI dengan filter pengecualian mutabilitas tag gambar

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters
 filterType=WILDCARD,filter=latest --region us-east-2
```

- Write-ECRImageTagMutability(AWS Tools for Windows PowerShell) dengan mutabilitas tag gambar

```
Write-ECRImageTagMutability -RepositoryName name -ImageTagMutability IMMUTABLE -
Region us-east-2 -Force
```

- Write-ECRImageTagMutability(AWS Tools for Windows PowerShell) dengan filter pengecualian mutabilitas tag gambar

```
Write-ECRImageTagMutability -RepositoryName name -
ImageTagMutability IMMUTABLE_WITH_EXCLUSION -ImageTagMutabilityExclusionFilter
 @{FilterType=WILDCARD Filter=latest}
```

Dukungan format manifes gambar kontainer di Amazon ECR

Amazon ECR mendukung format manifes citra kontainer berikut:

- Docker Image manifes V2 Skema 1 (digunakan dengan Docker versi 1.9 dan lebih tua)

- Docker Image Manifest V2 Skema 2 (digunakan dengan Docker versi 1.10 dan yang lebih baru)
- Spesifikasi Open Container Initiative (OCI) (v1.0 dan v1.1)

Support untuk Docker Image Manifest V2 Skema 2 menyediakan fungsionalitas berikut:

- Kemampuan untuk menggunakan beberapa tanda untuk citra tunggal.
- Support untuk menyimpan citra kontainer Windows.

Konversi manifes citra Amazon ECR

Ketika Anda mendorong dan menarik citra ke dan dari Amazon ECR, klien mesin kontainer Anda (misalnya, Docker) berkomunikasi dengan registrasi untuk menyetujui format manifes yang dipahami oleh klien dan registrasi yang akan digunakan untuk citra.

Ketika Anda mendorong citra ke Amazon ECR dengan Docker versi 1.9 atau sebelumnya, format manifes citra disimpan sebagai Docker Image Manifest V2 Skema 1. Ketika Anda mendorong citra ke Amazon ECR dengan Docker versi 1.10 atau yang lebih baru, format manifes citra disimpan sebagai Docker Image Manifest V2 Skema 2.

Ketika Anda menarik citra dari Amazon ECR dengan tanda, maka Amazon ECR mengembalikan format manifes citra yang disimpan dalam repositori. Format akan dikembalikan hanya jika format tersebut dipahami oleh klien. Jika format manifes citra yang disimpan tidak dipahami oleh klien, maka Amazon ECR akan mengkonversi manifes citra tersebut ke dalam format yang dipahami. Sebagai contoh, jika klien Docker 1.9 meminta manifes citra yang disimpan sebagai Docker Image Manifest V2 Skema 2, maka Amazon ECR mengembalikan manifes dalam format Docker Image Manifest V2 Skema 1. Tabel berikut mendeskripsikan konversi yang tersedia dan didukung oleh Amazon ECR ketika citra ditarik dengan tanda:

Skema yang diminta oleh klien	Didorong ke ECR sebagai V2, skema 1	Didorong ke ECR sebagai V2, skema 2	Didorong ke ECR sebagai OCI
V2, skema 1	Tidak memerlukan penerjemahan	Diterjemahkan ke V2, skema 1	Penerjemahan tidak tersedia
V2, skema 2	Tidak ada terjemahan yang tersedia, klien	Tidak memerlukan penerjemahan	Diterjemahkan ke V2, skema 2

Skema yang diminta oleh klien	Didorong ke ECR sebagai V2, skema 1	Didorong ke ECR sebagai V2, skema 2	Didorong ke ECR sebagai OCI
	dikembalikan ke V2, skema 1		
OCI	Penerjemahan tidak tersedia	Diterjemahkan ke OCI	Tidak memerlukan penerjemahan

Important

Jika Anda menarik citra berdasarkan digest, maka tidak ada terjemahan yang tersedia. Klien Anda harus memahami format manifes citra yang disimpan di Amazon ECR. Jika Anda meminta citra Docker Image Manifest V2 Skema 2 berdasarkan digest pada Docker 1.9 atau klien yang lebih lama, maka penarikan citra gagal. Untuk informasi selengkapnya, lihat Kompabilitas registrasi dalam dokumentasi Docker.

Pada contoh ini, jika Anda meminta citra yang sama berdasarkan tanda, maka Amazon ECR akan menerjemahkan manifes citra ke dalam format yang dapat dipahami klien. Penarikan citra berhasil.

Menggunakan citra Amazon ECR dengan Amazon ECS

Anda dapat menggunakan repositori pribadi Amazon ECR Anda untuk meng-host gambar kontainer dan artefak yang mungkin ditarik oleh tugas Amazon ECS Anda. Agar ini berfungsi, Amazon ECS, atau Fargate, agen kontainer harus memiliki izin untuk membuat,, `ecr:BatchGetImage` dan`ecr:GetDownloadUrlForLayer`. `ecr:GetAuthorizationToken` APIs

Izin IAM yang diperlukan

Tabel berikut menunjukkan peran IAM yang akan digunakan, untuk setiap jenis peluncuran, yang memberikan izin yang diperlukan untuk tugas Anda untuk menarik dari repositori pribadi Amazon ECR. Amazon ECS menyediakan kebijakan IAM terkelola yang menyertakan izin yang diperlukan.

Jenis peluncuran	IAM Role	AWS kebijakan IAM terkelola
Amazon ECS di instans Amazon EC2	Gunakan peran IAM instance container, yang dikaitkan dengan instans Amazon EC2 yang terdaftar ke cluster Amazon ECS Anda. Untuk informasi selengkapnya, lihat peran IAM instance Container di Panduan Pengembang Layanan Kontainer Elastis Amazon.	`AmazonEC2ContainerServiceforEC2Role` Untuk informasi selengkapnya, lihat AmazonEC2ContainerServiceforEC2Roledi Panduan Pengembang Layanan Amazon Elastic Container
Amazon ECS di Fargate	Gunakan peran IAM eksekusi tugas yang Anda referensikan dalam definisi tugas Amazon ECS Anda. Untuk informasi selengkapnya, lihat Peran IAM eksekusi tugas di Panduan Pengembang Layanan Amazon Elastic Container.	`AmazonECSTaskExecutionRolePolicy` Untuk informasi selengkapnya, lihat AmazonECSTaskExecutionRolePolicydi Panduan Pengembang Layanan Amazon Elastic Container.
Amazon ECS pada instans eksternal	Gunakan peran IAM instance container, yang dikaitkan dengan server lokal atau mesin virtual (VM) yang terdaftar ke cluster Amazon ECS Anda. Untuk informasi selengkapnya, lihat Peran Amazon ECS instance Container di Panduan Pengembang Layanan Kontainer Elastis Amazon.	`AmazonEC2ContainerServiceforEC2Role` Untuk informasi selengkapnya, lihat AmazonEC2ContainerServiceforEC2Roledi Panduan Pengembang Layanan Amazon Elastic Container.

Important

Kebijakan IAM AWS terkelola berisi izin tambahan yang mungkin tidak Anda perlukan untuk digunakan. Dalam hal ini, ini adalah izin minimum yang diperlukan untuk menarik dari repositori pribadi Amazon ECR.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Menentukan gambar Amazon ECR dalam definisi tugas Amazon ECS

Saat membuat definisi tugas Amazon ECS, Anda dapat menentukan gambar kontainer yang dihosting di repositori pribadi Amazon ECR. Dalam definisi tugas, pastikan Anda menggunakan `registry/repository:tag` penamaan lengkap untuk gambar Amazon ECR Anda. Misalnya, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-repository`*`:`*`latest`*.

Kutipan definisi tugas berikut menunjukkan sintaks yang akan Anda gunakan untuk menentukan citra kontainer yang dihost oleh Amazon ECR dalam definisi tugas Amazon ECS Anda.

```
{
    "family": "task-definition-name",
    ...
    "containerDefinitions": [
        {
            "name": "container-name",
```

```
            "image": "aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:latest",
            ...
        }
    ],
    ...
}
```

Menggunakan Citra Amazon ECR dengan Amazon EKS

Anda dapat menggunakan gambar Amazon ECR Anda dengan Amazon EKS.

Ketika mereferensikan citra dari Amazon ECR, Anda harus menggunakan penamaan penuh `registry/repository:tag`untuk citra. Misalnya, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-repository*`:`*latest*.

Izin IAM yang diperlukan

Jika Anda memiliki beban kerja Amazon EKS yang dihosting di node terkelola, node yang dikelola sendiri, atau AWS Fargate, tinjau hal-hal berikut:

- Beban kerja Amazon EKS yang dihosting di node yang dikelola atau dikelola sendiri: Peran IAM node pekerja Amazon EKS diperlukan`NodeInstanceRole`. Peran IAM node pekerja Amazon EKS harus berisi izin kebijakan IAM berikut untuk Amazon ECR.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

> **Note**
>
> Jika Anda menggunakan `eksctl` atau CloudFormation templat di Memulai Amazon EKS untuk membuat grup node cluster dan pekerja, izin IAM ini diterapkan ke peran IAM node pekerja Anda secara default.

- Beban kerja Amazon EKS yang di-host di AWS Fargate: Gunakan peran eksekusi pod Fargate, yang memberikan izin Pod Anda untuk menarik gambar dari repositori Amazon ECR pribadi. Untuk informasi selengkapnya, lihat Membuat peran eksekusi pod Fargate.

Memasang bagan Helm di kluster Amazon EKS

Bagan helm yang dihosting di Amazon ECR dapat diinstal di kluster Amazon EKS Anda.

Prasyarat

- Instal versi terbaru klien Helm. Langkah-langkah ini ditulis menggunakan versi `3.9.0` Helm. Untuk informasi selengkapnya, lihat Helm.
- Anda memiliki setidaknya versi `1.23.9` atau `2.6.3` yang AWS CLI diinstal pada komputer Anda. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui versi terbaru AWS CLI.
- Anda telah mendorong grafik Helm untuk repositori Amazon ECR Anda. Untuk informasi selengkapnya, lihat Mendorong bagan Helm ke repositori pribadi Amazon ECR.
- Anda telah mengkonfigurasi`kubectl` untuk menggunakan Amazon EKS. Untuk informasi selengkapnya, lihat Buat `kubeconfig`untuk Amazon EKSdalamPanduan Pengguna Amazon EKS. Jika perintah berikut berhasil diterapkan pada klaster Anda, berarti Anda telah melakukan konfigurasi dengan benar.

```
kubectl get svc
```

Untuk menginstal bagan Helm di kluster Amazon EKS

1. autentikasi Helm client Anda ke registrasi Amazon ECR bahwa grafik Helm Anda di-host. Token autentikasi harus diperoleh untuk setiap registrasi yang digunakan, dan token berlaku selama 12 jam. Untuk informasi selengkapnya, lihat Otentikasi registri pribadi di Amazon ECR.

```
aws ecr get-login-password \
     --region us-west-2 | helm registry login \
     --username AWS \
     --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

2. Instal bagan. Ganti *helm-test-chart* dengan repositori Anda dan *0.1.0* dengan tag bagan Helm Anda.

```
helm install ecr-chart-demo oci://aws_account_id.dkr.ecr.region.amazonaws.com/helm-test-chart --version 0.1.0
```

Outputnya akan terlihat serupa dengan ini:

```
NAME: ecr-chart-demo
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

3. Verifikasi instalasi grafik.

```
helm list -n default
```

Contoh output:

```
NAME            NAMESPACE       REVISION        UPDATED
     STATUS          CHART                   APP VERSION
ecr-chart-demo  default         1               2022-06-01 15:56:40.128669157 +0000
 UTC deployed        helm-test-chart-0.1.0   1.16.0
```

4. (Opsional) Lihat bagan `ConfigMap` Helm yang diinstal.

```
kubectl describe configmap helm-test-chart-configmap
```

5. Setelah selesai, Anda dapat menghapus rilis grafik dari klaster Anda.

```
helm uninstall ecr-chart-demo
```

Menandatangani gambar di Amazon ECR

Amazon ECR terintegrasi dengan AWS Signer untuk menyediakan dua cara bagi Anda untuk menandatangani gambar kontainer Anda: penandatanganan terkelola (otomatis, disarankan) dan penandatanganan manual (sisi klien). Anda dapat menyimpan gambar kontainer dan tanda tangan di repositori pribadi Anda.

Pilih metode penandatanganan

Amazon ECR mendukung dua metode untuk menandatangani gambar kontainer:

Penandatanganan terkelola (disarankan)

Penandatanganan terkelola secara otomatis menghasilkan tanda tangan kriptografi saat gambar didorong ke Amazon ECR. Metode ini menyederhanakan pengaturan. Penandatanganan terkelola adalah pendekatan yang disarankan untuk sebagian besar pengguna. Untuk informasi selengkapnya, lihat Penandatanganan terkelola.

Penandatanganan manual

Penandatanganan manual menggunakan CLI Notasi AWS Signer dan plugin untuk menandatangani gambar sebelum mendorongnya ke Amazon ECR. Metode ini memberikan kontrol lebih besar atas proses penandatanganan dan berguna ketika Anda perlu menandatangani gambar di luar alur kerja push atau memerlukan kontrol halus atas operasi penandatanganan. Untuk informasi selengkapnya, lihat Penandatanganan manual.

Pertimbangan

Berikut ini harus dipertimbangkan saat menggunakan penandatanganan gambar Amazon ECR:

- Tanda tangan yang disimpan di repositori Anda dihitung terhadap kuota layanan untuk jumlah maksimum gambar per repositori. Setiap tanda tangan dihitung sebagai 1 artefak terhadap gambar per kuota repositori. Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECR.
- Ketika artefak referensi hadir dalam repositori, kebijakan siklus hidup Amazon ECR akan secara otomatis membersihkan artefak tersebut dalam waktu 24 jam setelah penghapusan gambar subjek.

Penandatanganan terkelola

Penandatanganan terkelola Amazon ECR secara otomatis menandatangani gambar penampung Anda dengan membuat tanda tangan kriptografi menggunakan AWS Penandatangan saat gambar didorong ke Amazon ECR. Ini menghilangkan kebutuhan untuk menginstal dan mengkonfigurasi alat sisi klien dan memungkinkan Anda untuk mengatur penandatanganan secara terpusat sebagai konfigurasi registri.

Prasyarat

Untuk mengonfigurasi penandatanganan terkelola, Anda membuat konfigurasi penandatanganan dengan Amazon ECR yang mereferensikan satu atau beberapa profil penandatanganan Penandatangan dan, secara opsional, filter repositori yang membatasi repositori mana yang harus ditandatangani gambarnya. Setelah dikonfigurasi, Amazon ECR mengelola penandatanganan secara otomatis menandatangani gambar saat didorong menggunakan identitas entitas yang mendorong gambar.

Sebelum Anda dapat mengonfigurasi penandatanganan terkelola, Anda harus memiliki yang berikut:

- Profil penandatanganan Penandatangan — Buat setidaknya satu profil penandatanganan Penandatangan. Profil penandatanganan adalah sumber daya AWS Penandatangan unik yang dapat Anda gunakan untuk melakukan operasi penandatanganan di Amazon ECR. Profil penandatanganan memungkinkan Anda untuk menandatangani dan memverifikasi artefak kode, seperti gambar kontainer dan AWS Lambda bundel penerapan. Setiap profil penandatanganan menunjuk platform penandatanganan untuk ditandatangani, ID platform, dan informasi khusus platform lainnya. Misalnya, profil penandatanganan ARN terlihat seperti ini:. `arn:`*`partition`*`:signer:`*`region`*`:`*`account-id`*`:/signing-profiles/`*`profile-name`*
- Izin IAM — Prinsip IAM yang mendorong gambar harus memiliki izin IAM yang diperlukan untuk mengakses profil penandatanganan Penandatangan yang relevan dan repositori ECR yang relevan. Anda perlu mengubah kebijakan berbasis identitas untuk prinsipal IAM untuk menyertakan izin untuk operasi repositori ECR dan operasi penandatanganan Penandatangan. Contoh kebijakan berikut menunjukkan izin yang diperlukan:

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
```

```
        "Sid":"UploadSignaturePermissions",
        "Effect":"Allow",
        "Action":[
            "ecr:CompleteLayerUpload",
            "ecr:UploadLayerPart",
            "ecr:InitiateLayerUpload",
            "ecr:BatchCheckLayerAvailability",
            "ecr:PutImage"
        ],
        "Resource":"arn:aws:ecr:region:account-id:repository/repository-name"
    },
    {
        "Sid": "SignPermissions",
        "Effect": "Allow",
        "Action": [
            "signer:SignPayload"
        ],
        "Resource": "arn:aws:signer:region:account-id:/signing-profiles/signing-profile-
name"
    }
    ]
}
```

Dengan penandatanganan terkelola Amazon ECR, Anda dapat membuat beberapa aturan penandatanganan (hingga 10 per registri) untuk membuat batas keamanan yang lebih kuat. Misalnya, Anda dapat menjalankan beberapa pipeline build dan ingin membatasi repositori mana yang dapat ditandatangani oleh setiap pipeline. Dalam setiap aturan, Anda mengonfigurasi profil penandatanganan dan menentukan filter nama repositori. Saat gambar baru didorong, Amazon ECR cocok dengan aturan penandatanganan dan profil penandatanganan mana yang dapat menandatangani gambar. Jika ada beberapa kecocokan, Amazon ECR menghasilkan beberapa tanda tangan.

> **Note**
>
> Jika Anda memverifikasi tanda tangan secara manual, Anda masih perlu menginstal CLI Notasi.

> **Note**
>
> Penandatanganan terkelola Amazon ECR tersedia di semua AWS Wilayah tempat penandatanganan gambar kontainer dengan AWS Penandatangan tersedia.

Memulai

Ikuti langkah-langkah ini untuk mengonfigurasi penandatanganan terkelola. Anda memberikan Amazon ECR referensi ke profil penandatanganan Penandatangan dan, secara opsional, filter yang membatasi repositori mana yang harus ditandatangani gambarnya.

Konsol Manajemen AWS

Gunakan langkah-langkah berikut untuk mengonfigurasi penandatanganan terkelola menggunakan Konsol Manajemen AWS.

1. Buka konsol Amazon ECR. Di panel navigasi kiri, pilih Registri pribadi, Fitur & pengaturan, Penandatanganan terkelola.
2. Pada halaman Aturan penandatanganan, pilih Buat aturan.
3. Pada halaman Profil penandatanganan, di bawah Pilih profil AWS penandatangan, pilih Buat profil AWS penandatangan baru, masukkan nama Profil, dan, secara opsional, ubah periode validitas tanda tangan. Kemudian pilih Berikutnya.
4. Pada halaman Filter, di bawah Pilih repositori, masukkan filter nama Repositori. Kemudian pilih Berikutnya.
5. Pada halaman Tinjau dan buat, verifikasi profil AWS Penandatangan dan filter nama Repositori yang telah Anda masukkan. Jika semuanya terlihat benar, pilih Simpan.

AWS CLI

Gunakan AWS CLI perintah berikut untuk mengonfigurasi penandatanganan terkelola.

- Buat aturan penandatanganan

  Buat konfigurasi penandatanganan dengan ARN profil penandatanganan Anda. Buat file JSON dengan konten berikut:

```
{
```

```
    "rules": [
        {
            "signingProfileArn": "arn:aws:signer:region:account-id:/signing-
profiles/profile-name",
            "repositoryFilters": [
                {
                    "filter": "test*",
                    "filterType": "WILDCARD_MATCH"
                }
            ]
        }
    ]
}
```

Kemudian jalankan perintah berikut:

```
aws ecr --region region \
    put-signing-configuration \
    --signing-configuration file://signing-config.json
```

Anda akan melihat respons API yang berisi konfigurasi penandatanganan.

- Lihat konfigurasi penandatanganan Anda

  Ambil konfigurasi penandatanganan Anda:

  ```
  aws ecr --region region \
      get-signing-configuration
  ```

  Anda akan melihat respons API yang berisi konfigurasi penandatanganan.

- Periksa status penandatanganan gambar

  Dorong gambar ke repositori Anda. Contoh:

  ```
  docker pull ubuntu

  IMAGE_NAME="account-id.dkr.ecr.region.amazonaws.com/repository-name"
  IMAGE_TAG="${IMAGE_NAME}:test-1"

  docker tag ubuntu $IMAGE_TAG
  docker push $IMAGE_TAG
  ```

Setelah mendorong, gunakan tag gambar Anda untuk memeriksa status penandatanganan:

```
aws ecr --region region \
    describe-image-signing-status \
    --repository-name repository-name \
    --image-id imageTag=test-1
```

Jika nama repositori cocok dengan filter repositori yang ditentukan dalam konfigurasi penandatanganan, Anda akan melihat status penandatanganan dalam respons API. Jika status berhasil, Anda akan melihat tanda tangan didorong ke repositori Anda.

- Hapus konfigurasi penandatanganan

  Hapus konfigurasi penandatanganan Anda:

  ```
  aws ecr --region region \
      delete-signing-configuration
  ```

  Anda akan melihat respons API yang berisi konfigurasi penandatanganan yang dihapus.

Pertimbangan-pertimbangan

Keterbatasan dan kemampuan berikut berlaku untuk penandatanganan terkelola:

- Penandatanganan lintas wilayah tidak didukung — Profil penandatanganan harus berada di wilayah yang sama dengan registri Amazon ECR Anda. Anda tidak dapat menggunakan profil penandatanganan dari satu wilayah untuk menandatangani gambar di registri yang terletak di wilayah lain.
- Penandatanganan lintas akun didukung — Profil penandatanganan dapat berada di akun yang berbeda dari registri ECR Amazon Anda. Ini memungkinkan organisasi untuk mengelola profil penandatanganan secara terpusat sambil memungkinkan pengembang di akun lain untuk menggunakannya. Untuk informasi selengkapnya, lihat Menyiapkan penandatanganan lintas akun untuk Penandatangan di Panduan AWS Signer Pengembang.
- Tanda tangan tidak dapat ditandatangani — Anda tidak dapat menandatangani tanda tangan sendiri. Hanya gambar kontainer yang dapat ditandatangani.

Verifikasi tanda tangan

Setelah menandatangani gambar kontainer, Anda dapat memverifikasi tanda tangan untuk memastikan bahwa gambar belum dirusak dan berasal dari sumber tepercaya. Amazon ECR mendukung beberapa metode untuk memverifikasi tanda tangan:

Verifikasi terkelola dengan Amazon EKS

Amazon EKS menyediakan integrasi asli untuk verifikasi tanda tangan otomatis. Saat mengonfigurasi verifikasi tanda tangan di kluster Amazon EKS, layanan akan secara otomatis memverifikasi tanda tangan gambar sebelum mengizinkan container dijalankan. Untuk informasi selengkapnya tentang mengonfigurasi verifikasi tanda tangan, lihat Memvalidasi tanda tangan gambar kontainer selama penerapan di Panduan Pengguna Amazon EKS.

Pengontrol penerimaan Lambda untuk Amazon ECS

Amazon ECS menyediakan kait siklus hidup layanan yang memungkinkan Anda menjalankan logika kustom selama penerapan layanan. Kait ini dapat memicu AWS Lambda fungsi pada titik-titik tertentu dalam proses penerapan, memungkinkan Anda untuk memvalidasi tanda tangan gambar kontainer sebelum mengizinkan layanan dimulai. Untuk informasi selengkapnya, lihat Memverifikasi tanda tangan gambar kontainer untuk Amazon ECS di Panduan Pengembang.AWS Signer

Verifikasi manual dengan Notasi CLI

Anda dapat memverifikasi tanda tangan secara manual menggunakan CLI Notasi. Metode ini mengharuskan Anda untuk menginstal dan mengkonfigurasi CLI Notasi pada mesin lokal Anda atau di lingkungan verifikasi Anda. Untuk petunjuk mendetail tentang memverifikasi gambar menggunakan CLI Notasi, lihat Memverifikasi gambar secara lokal setelah masuk AWS Signer ke Panduan Pengembang.

Konfigurasikan otentikasi untuk klien Notasi

Jika Anda menggunakan penandatanganan manual atau memverifikasi tanda tangan secara manual menggunakan CLI Notasi, Anda harus mengonfigurasi klien Notasi sehingga dapat mengautentikasi ke Amazon ECR. Jika Anda menginstal Docker pada host yang sama tempat Anda menginstal klien Notation, maka Notation akan menggunakan kembali metode otentikasi yang sama yang Anda gunakan untuk klien Docker. Docker `login` dan `logout` perintah akan memungkinkan Notasi `sign` dan `verify` perintah untuk menggunakan kredenal yang sama, dan Anda tidak perlu

mengautentikasi Notasi secara terpisah. Untuk informasi selengkapnya tentang mengonfigurasi klien Notasi Anda untuk otentikasi, lihat Mengautentikasi dengan pendaftar yang sesuai dengan OCI di dokumentasi Proyek Notaris.

Jika Anda tidak menggunakan Docker atau alat lain yang menggunakan kredensi Docker, sebaiknya gunakan Amazon ECR Docker Credential Helper sebagai toko kredensi Anda. Untuk informasi selengkapnya tentang cara menginstal dan mengonfigurasi Amazon ECR Credential Helper, lihat Amazon ECR Docker Credential Helper.

Penandatanganan manual

Penandatanganan manual menggunakan CLI Notasi AWS Signer dan plugin untuk menandatangani gambar sebelum mendorongnya ke Amazon ECR. Metode ini memberikan kontrol lebih besar atas proses penandatanganan dan berguna ketika Anda perlu menandatangani gambar di luar alur kerja push atau memerlukan kontrol halus atas operasi penandatanganan.

Untuk petunjuk mendetail tentang penandatanganan gambar kontainer menggunakan CLI Notasi AWS Signer dan, lihat Menandatangani gambar kontainer di Penandatangan dan topik terkait di AWS Signer Panduan Pengembang.

Prasyarat

Sebelum Anda mulai, Prasyarat berikut harus dipenuhi.

- Instal dan konfigurasikan versi terbaru dari file AWS CLI. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui versi terbaru dari Panduan AWS Command Line Interface Pengguna. AWS CLI
- Instal CLI Notasi dan plugin untuk AWS Signer Notasi. Untuk informasi selengkapnya, lihat Prasyarat untuk menandatangani gambar kontainer di Panduan Pengembang.AWS Signer
- Miliki gambar kontainer yang disimpan di repositori pribadi Amazon ECR untuk ditandatangani. Lihat informasi yang lebih lengkap di Mendorong gambar ke repositori pribadi Amazon ECR.

Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR

Pemindaian gambar Amazon ECR membantu mengidentifikasi kerentanan perangkat lunak dalam gambar kontainer Anda. Jenis pemindaian berikut ditawarkan.

Important

Beralih antara pemindaian yang ditingkatkan dan pemindaian Dasar akan menyebabkan pemindaian yang telah dibuat sebelumnya tidak lagi tersedia. Anda harus mengatur pemindaian Anda lagi. Namun, jika Anda beralih kembali ke jenis pemindaian sebelumnya, pemindaian yang telah ditetapkan akan tersedia.

Note

Gambar yang diarsipkan tidak dapat dipindai. Gambar yang diarsipkan harus dipulihkan sebelum dapat dipindai. Untuk informasi selengkapnya tentang pengarsipan dan pemulihan gambar, lihat. Mengarsipkan gambar di Amazon ECR

- Pemindaian yang disempurnakan - Amazon ECR terintegrasi dengan Amazon Inspector untuk menyediakan pemindaian otomatis dan berkelanjutan dari repositori Anda. Gambar kontainer Anda dipindai untuk sistem operasi dan kerentanan paket bahasa pemrograman. Saat kerentanan baru muncul, hasil pemindaian diperbarui dan Amazon Inspector memancarkan acara EventBridge untuk memberi tahu Anda. Pemindaian yang disempurnakan memberikan yang berikut:
  - Kerentanan paket OS dan bahasa pemrograman
  - Dua frekuensi pemindaian: Pindai saat push dan pemindaian terus menerus
- Pemindaian dasar — Amazon ECR menggunakan teknologi AWS asli dengan database Common Vulnerabilities and Exposures (CVEs) untuk memindai kerentanan sistem operasi.

  Dengan pemindaian dasar, Anda mengonfigurasi repositori untuk memindai saat push atau Anda dapat melakukan pemindaian manual dan Amazon ECR menyediakan daftar temuan pemindaian. Pemindaian dasar menyediakan yang berikut:
  - Pemindaian OS

- Dua frekuensi pemindaian: Manual dan pemindaian saat push

> **Important**
>
> Versi baru Amazon ECR Basic Scanning tidak menggunakan `imageScanStatus` atribut `imageScanFindingsSummary` dan dari respons `DescribeImages` API untuk mengembalikan hasil pemindaian. Gunakan `DescribeImageScanFindings` API sebagai gantinya. Untuk informasi selengkapnya, lihat `DescribeImageScanFindings`.

Filter untuk memilih repositori mana yang dipindai di Amazon ECR

Saat Anda mengonfigurasi pemindaian gambar untuk registri pribadi Anda, Anda dapat menggunakan filter untuk memilih repositori mana yang dipindai.

Saat pemindaian dasar digunakan, Anda dapat menentukan pemindaian pada filter push untuk menentukan repositori mana yang diatur untuk melakukan pemindaian gambar saat gambar baru didorong. Setiap repositori yang tidak cocok dengan pemindaian pemindaian dasar pada filter push akan diatur ke frekuensi pemindaian manual yang berarti untuk melakukan pemindaian, Anda harus memicu pemindaian secara manual.

Saat pemindaian yang disempurnakan digunakan, Anda dapat menentukan filter terpisah untuk pemindaian saat push dan pemindaian berkelanjutan. Repositori apa pun yang tidak cocok dengan filter pemindaian yang disempurnakan akan menonaktifkan pemindaian. Jika Anda menggunakan pemindaian yang disempurnakan dan menentukan filter terpisah untuk pemindaian pada push dan pemindaian berkelanjutan di mana beberapa filter cocok dengan repositori yang sama, maka Amazon ECR memberlakukan filter pemindaian berkelanjutan melalui pemindaian pada filter push untuk repositori itu.

Filter wildcard

Ketika filter ditentukan, filter tanpa wildcard akan cocok dengan semua nama repositori yang berisi filter. Filter dengan wildcard (*) cocok dengan nama repositori mana pun di mana wildcard menggantikan nol atau lebih karakter dalam nama repositori.

Tabel berikut memberikan contoh di mana nama repositori dinyatakan pada sumbu horizontal dan contoh filter ditentukan pada sumbu vertikal.

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod	Ya	Ya	Ya	Ya	Ya
*prod	Ya	Ya	Tidak	Tidak	Tidak
prod*	Ya	Tidak	Ya	Tidak	Ya
prod	Ya	Ya	Ya	Ya	Ya
prod*repo	Tidak	Tidak	Ya	Tidak	Ya

Pindai gambar untuk kerentanan paket OS dan bahasa pemrograman di Amazon ECR

Pemindaian Amazon ECR yang disempurnakan adalah integrasi dengan Amazon Inspector yang menyediakan pemindaian kerentanan untuk gambar kontainer Anda. Gambar kontainer Anda dipindai untuk sistem operasi dan kerentanan paket bahasa pemrograman. Anda dapat melihat temuan pemindaian dengan Amazon ECR dan dengan Amazon Inspector secara langsung. Untuk informasi selengkapnya tentang Amazon Inspector, lihat Memindai gambar kontainer dengan Amazon Inspector di Panduan Pengguna Amazon Inspector.

Dengan pemindaian yang disempurnakan, Anda dapat memilih repositori mana yang dikonfigurasi untuk pemindaian otomatis dan berkelanjutan dan mana yang dikonfigurasi untuk pemindaian saat push. Ini dilakukan dengan mengatur filter pemindaian.

Pertimbangan untuk pemindaian yang ditingkatkan

Pertimbangkan hal berikut sebelum mengaktifkan pemindaian Amazon ECR yang disempurnakan.

- Tidak ada biaya tambahan dari Amazon ECR untuk menggunakan fitur ini, namun ada biaya dari Amazon Inspector untuk memindai gambar Anda. Fitur ini tersedia di Wilayah di mana Amazon Inspector didukung. Untuk informasi lebih lanjut, lihat:
  - Harga Amazon Inspector - Harga Amazon Inspector.
  - Wilayah yang didukung Amazon Inspector — Wilayah dan titik akhir.

- Pemindaian Amazon ECR yang disempurnakan menunjukkan bagaimana gambar digunakan di Amazon EKS dan Amazon ECS. Anda dapat melihat kapan gambar terakhir digunakan dan mengidentifikasi berapa banyak cluster yang menggunakan setiap gambar. Informasi ini membantu Anda memprioritaskan remediasi kerentanan untuk gambar yang digunakan secara aktif. Anda dapat dengan cepat menentukan cluster mana yang mungkin terpengaruh oleh kerentanan yang baru ditemukan. Untuk informasi selengkapnya tentang cara meminta informasi ini dan melihat tanggapannya, lihat `DescribeImageScanFindings`.
- Amazon Inspector mendukung pemindaian untuk sistem operasi tertentu. Untuk daftar selengkapnya, lihat Sistem operasi yang didukung - Pemindaian Amazon ECR di Panduan Pengguna Amazon Inspector.
- Amazon Inspector menggunakan peran IAM terkait layanan, yang menyediakan izin yang diperlukan untuk menyediakan pemindaian yang disempurnakan untuk repositori Anda. Peran IAM terkait layanan dibuat secara otomatis oleh Amazon Inspector saat pemindaian yang disempurnakan diaktifkan untuk registri pribadi Anda. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon Inspector di Panduan Pengguna Amazon Inspector.
- Saat Anda awalnya mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda, Amazon Inspector hanya mengenali gambar yang didorong ke Amazon ECR dalam 14 hari terakhir, berdasarkan stempel waktu push gambar. Gambar yang lebih tua akan memiliki status `SCAN_ELIGIBILITY_EXPIRED` pemindaian. Jika Anda ingin gambar-gambar ini dipindai oleh Amazon Inspector, Anda harus mendorongnya lagi ke repositori Anda.
- Saat pemindaian yang disempurnakan diaktifkan untuk registri pribadi Amazon ECR Anda, repositori yang cocok dengan filter pemindaian dipindai hanya menggunakan pemindaian yang disempurnakan. Repositori apa pun yang tidak cocok dengan filter akan memiliki frekuensi `Off` pemindaian dan tidak akan dipindai. Pemindaian manual menggunakan pemindaian yang disempurnakan tidak didukung. Untuk informasi selengkapnya, lihat Filter untuk memilih repositori mana yang dipindai di Amazon ECR.
- Jika Anda menentukan filter terpisah untuk pemindaian pada push dan pemindaian berkelanjutan di mana beberapa filter cocok dengan repositori yang sama, maka Amazon ECR memberlakukan filter pemindaian berkelanjutan melalui pemindaian pada filter push untuk repositori tersebut.
- Saat pemindaian yang ditingkatkan diaktifkan, Amazon ECR mengirimkan peristiwa ke EventBridge saat frekuensi pemindaian untuk repositori diubah. Amazon Inspector memancarkan peristiwa EventBridge saat pemindaian awal selesai dan saat temuan pemindaian gambar dibuat, diperbarui, atau ditutup.

Mengubah durasi pemindaian yang disempurnakan untuk gambar di Amazon Inspector

Setelah mengaktifkan pemindaian yang disempurnakan, Amazon ECR terus memindai gambar yang baru didorong untuk durasi yang dikonfigurasi. Secara default, Amazon Inspector memantau repositori Anda hingga gambar dihapus atau pemindaian yang disempurnakan dinonaktifkan. Anda dapat mengonfigurasi durasi tanggal push (hingga Lifetime) dan durasi pemindaian ulang di konsol Amazon Inspector agar sesuai dengan kebutuhan lingkungan Anda. Ketika durasi pemindaian untuk repositori berlalu, status pemindaian ditampilkan sebagai. `SCAN_ELIGIBILITY_EXPIRED` Untuk informasi selengkapnya tentang mengonfigurasi setelan durasi pemindaian ulang untuk Amazon ECR di Amazon Inspector, lihat Mengonfigurasi durasi pemindaian ulang Amazon ECR di Panduan Pengguna Amazon Inspector.

Izin IAM diperlukan untuk pemindaian yang ditingkatkan di Amazon ECR

Pemindaian Amazon ECR yang disempurnakan memerlukan peran IAM terkait layanan Amazon Inspector dan bahwa prinsipal IAM yang mengaktifkan dan menggunakan pemindaian yang disempurnakan memiliki izin untuk memanggil Amazon Inspector yang diperlukan untuk pemindaian. APIs Peran IAM terkait layanan Amazon Inspector dibuat secara otomatis oleh Amazon Inspector ketika pemindaian yang disempurnakan diaktifkan untuk registri pribadi Anda. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon Inspector di Panduan Pengguna Amazon Inspector.

Kebijakan IAM berikut memberikan izin yang diperlukan untuk mengaktifkan dan menggunakan pemindaian yang disempurnakan. Ini termasuk izin yang diperlukan untuk Amazon Inspector untuk membuat peran IAM terkait layanan serta izin Amazon Inspector API yang diperlukan untuk mengaktifkan dan menonaktifkan pemindaian yang disempurnakan dan mengambil temuan pemindaian.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "inspector2:Enable",
```

```
                "inspector2:Disable",
                "inspector2:ListFindings",
                "inspector2:ListAccountPermissions",
                "inspector2:ListCoverage"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "inspector2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Mengkonfigurasi pemindaian yang disempurnakan untuk gambar di Amazon ECR

Konfigurasikan pemindaian yang ditingkatkan per Wilayah untuk registri pribadi Anda.

Verifikasi bahwa Anda memiliki izin IAM yang tepat untuk mengonfigurasi pemindaian yang disempurnakan. Untuk informasi, lihat Izin IAM diperlukan untuk pemindaian yang ditingkatkan di Amazon ECR.

Konsol Manajemen AWS

Untuk mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah untuk mengatur konfigurasi pemindaian.
3. Di panel navigasi, pilih Registri pribadi, lalu pilih Pengaturan.
4. Pada halaman konfigurasi Scanning, untuk jenis Scan pilih Enhanced scanning.

Secara default, ketika pemindaian yang ditingkatkan dipilih, semua repositori Anda terus dipindai.

5. Untuk memilih repositori tertentu untuk terus memindai, kosongkan kotak Continuous scan all repositories, dan kemudian tentukan filter Anda:

   > **Important**
   >
   > Filter tanpa wildcard akan cocok dengan semua nama repositori yang berisi filter. Filter dengan wildcard (*) cocok dengan nama repositori di mana wildcard menggantikan nol atau lebih karakter dalam nama repositori. Untuk melihat contoh bagaimana filter berperilaku, lihatthe section called “Filter wildcard”.

   a. Masukkan filter berdasarkan nama repositori, lalu pilih Tambahkan filter.

   b. Tentukan repositori mana yang akan dipindai saat gambar didorong:

      - Untuk memindai semua repositori saat push, pilih Pindai saat mendorong semua repositori.
      - Untuk memilih repositori tertentu untuk memindai saat push, masukkan filter berdasarkan nama repositori, lalu pilih Tambahkan filter.

6. Pilih Simpan.
7. Ulangi langkah-langkah ini di setiap Wilayah di mana Anda ingin mengaktifkan pemindaian yang disempurnakan.

AWS CLI

Gunakan AWS CLI perintah berikut untuk mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda menggunakan AWS CLI. Anda dapat menentukan filter pemindaian menggunakan `rules` objek.

- put-registry-scanning-configuration (AWS CLI)

  Contoh berikut mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda. Secara default, ketika tidak `rules` ditentukan, Amazon ECR menyetel konfigurasi pemindaian ke pemindaian berkelanjutan untuk semua repositori.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
     --region us-east-2
```

Contoh berikut mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda dan menentukan filter pemindaian. Filter pemindaian dalam contoh mengaktifkan pemindaian terus menerus untuk semua repositori dengan `prod` namanya.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
     --rules '[{"repositoryFilters" : [{"filter":"prod","filterType" :
 "WILDCARD"}],"scanFrequency" : "CONTINUOUS_SCAN"}]' \
     --region us-east-2
```

Contoh berikut mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda dan menentukan beberapa filter pemindaian. Filter pemindaian dalam contoh mengaktifkan pemindaian berkelanjutan untuk semua repositori dengan `prod` namanya dan mengaktifkan pemindaian pada push hanya untuk semua repositori lainnya.

```
aws ecr put-registry-scanning-configuration \
     --scan-type ENHANCED \
     --rules '[{"repositoryFilters" : [{"filter":"prod","filterType" :
 "WILDCARD"}],"scanFrequency" : "CONTINUOUS_SCAN"},{"repositoryFilters" :
 [{"filter":"*","filterType" : "WILDCARD"}],"scanFrequency" : "SCAN_ON_PUSH"}]' \
     --region us-west-2
```

EventBridge peristiwa dikirim untuk pemindaian yang ditingkatkan di Amazon ECR

Saat pemindaian yang ditingkatkan diaktifkan, Amazon ECR mengirimkan peristiwa ke EventBridge saat frekuensi pemindaian untuk repositori diubah. Amazon Inspector mengirimkan peristiwa ke EventBridge saat pemindaian awal selesai dan saat temuan pemindaian gambar dibuat, diperbarui, atau ditutup.

Acara untuk perubahan frekuensi pemindaian repositori

Ketika pemindaian yang disempurnakan diaktifkan untuk registri Anda, peristiwa berikut dikirim oleh Amazon ECR ketika ada perubahan dengan sumber daya yang telah ditingkatkan pemindaian diaktifkan. Ini termasuk repositori baru yang sedang dibuat, frekuensi pemindaian untuk repositori yang diubah, atau ketika gambar dibuat atau dihapus di repositori dengan pemindaian yang ditingkatkan diaktifkan. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0abEXAMPLE",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
   {
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

Acara untuk pemindaian gambar awal (pemindaian yang disempurnakan)

Saat pemindaian yang disempurnakan diaktifkan untuk registri Anda, peristiwa berikut akan dikirim oleh Amazon Inspector saat pemindaian gambar awal selesai. Parameter `finding-severity-counts` hanya akan mengembalikan nilai untuk suatu tingkat keparahan jika ada. Contohnya, jika citra tidak mengandung temuan di tingkat `CRITICAL`, maka tidak ada hitungan kritis yang dikembalikan. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan paket OS dan bahasa pemrograman di Amazon ECR.

Pola acara:

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Scan"]
}
```

Contoh output:

```
{
    "version": "0",
    "id": "739c0d3c-4f02-85c7-5a88-94a9EXAMPLE",
    "detail-type": "Inspector2 Scan",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2021-12-03T18:03:16Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample"
    ],
    "detail": {
        "scan-status": "INITIAL_SCAN_COMPLETE",
        "repository-name": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/
amazon-ecs-sample",
        "finding-severity-counts": {
            "CRITICAL": 7,
            "HIGH": 61,
            "MEDIUM": 62,
            "TOTAL": 158
        },
        "image-digest":
 "sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
        "image-tags": [
```

```
            "latest"
        ]
    }
}
```

Acara untuk pembaruan pencarian pemindaian gambar (pemindaian yang disempurnakan)

Saat pemindaian yang disempurnakan diaktifkan untuk registri Anda, peristiwa berikut akan dikirim oleh Amazon Inspector saat temuan pemindaian gambar dibuat, diperbarui, atau ditutup. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan paket OS dan bahasa pemrograman di Amazon ECR.

Pola acara:

```
{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Finding"]
}
```

Contoh output:

```
{
    "version": "0",
    "id": "42dbea55-45ad-b2b4-87a8-afaEXAMPLE",
    "detail-type": "Inspector2 Finding",
    "source": "aws.inspector2",
    "account": "123456789012",
    "time": "2021-12-03T18:02:30Z",
    "region": "us-east-2",
    "resources": [
        "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample/
sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77eEXAMPLE"
    ],
    "detail": {
        "awsAccountId": "123456789012",
        "description": "In libssh2 v1.9.0 and earlier versions, the SSH_MSG_DISCONNECT
 logic in packet.c has an integer overflow in a bounds check, enabling an attacker to
 specify an arbitrary (out-of-bounds) offset for a subsequent memory read. A crafted
 SSH server may be able to disclose sensitive information or cause a denial of service
 condition on the client system when a user connects to the server.",
        "findingArn": "arn:aws:inspector2:us-east-2:123456789012:finding/
be674aaddd0f75ac632055EXAMPLE",
        "firstObservedAt": "Dec 3, 2021, 6:02:30 PM",
```

```
        "inspectorScore": 6.5,
        "inspectorScoreDetails": {
            "adjustedCvss": {
                "adjustments": [],
                "cvssSource": "REDHAT_CVE",
                "score": 6.5,
                "scoreSource": "REDHAT_CVE",
                "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
                "version": "3.0"
            }
        },
        "lastObservedAt": "Dec 3, 2021, 6:02:30 PM",
        "packageVulnerabilityDetails": {
            "cvss": [
                {
                    "baseScore": 6.5,
                    "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
                    "source": "REDHAT_CVE",
                    "version": "3.0"
                },
                {
                    "baseScore": 5.8,
                    "scoringVector": "AV:N/AC:M/Au:N/C:P/I:N/A:P",
                    "source": "NVD",
                    "version": "2.0"
                },
                {
                    "baseScore": 8.1,
                    "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:H",
                    "source": "NVD",
                    "version": "3.1"
                }
            ],
            "referenceUrls": [
                "https://access.redhat.com/errata/RHSA-2020:3915"
            ],
            "source": "REDHAT_CVE",
            "sourceUrl": "https://access.redhat.com/security/cve/CVE-2019-17498",
            "vendorCreatedAt": "Oct 16, 2019, 12:00:00 AM",
            "vendorSeverity": "Moderate",
            "vulnerabilityId": "CVE-2019-17498",
            "vulnerablePackages": [
                {
                    "arch": "X86_64",
```

```
                    "epoch": 0,
                    "name": "libssh2",
                    "packageManager": "OS",
                    "release": "12.amzn2.2",
                    "sourceLayerHash":
 "sha256:72d97abdfae3b3c933ff41e39779cc72853d7bd9dc1e4800c5294dEXAMPLE",
                    "version": "1.4.3"
                }
            ]
        },
        "remediation": {
            "recommendation": {
                "text": "Update all packages in the vulnerable packages section to
 their latest versions."
            }
        },
        "resources": [
            {
                "details": {
                    "awsEcrContainerImage": {
                        "architecture": "amd64",
                        "imageHash":
 "sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
                        "imageTags": [
                            "latest"
                        ],
                        "platform": "AMAZON_LINUX_2",
                        "pushedAt": "Dec 3, 2021, 6:02:13 PM",
                        "lastInUseAt": "Dec 3, 2021, 6:02:13 PM",
                        "inUseCount": 1,
                        "registry": "123456789012",
                        "repositoryName": "amazon/amazon-ecs-sample"
                    }
                },
                "id": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-
sample/sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77EXAMPLE",
                "partition": "N/A",
                "region": "N/A",
                "type": "AWS_ECR_CONTAINER_IMAGE"
            }
        ],
        "severity": "MEDIUM",
        "status": "ACTIVE",
        "title": "CVE-2019-17498 - libssh2",
```

```
        "type": "PACKAGE_VULNERABILITY",
        "updatedAt": "Dec 3, 2021, 6:02:30 PM"
    }
}
```

Mengambil temuan untuk pemindaian yang ditingkatkan di Amazon ECR

Anda dapat mengambil temuan pemindaian untuk pemindaian gambar terakhir yang disempurnakan, dan kemudian membuka temuan di Amazon Inspector untuk melihat lebih detail. Kerentanan perangkat lunak yang ditemukan terdaftar berdasarkan tingkat keparahan berdasarkan database Common Vulnerabilities and Exposures ()CVEs.

Untuk detail pemecahan masalah untuk beberapa masalah umum saat memindai citra, lihat Memecahkan masalah pemindaian gambar di Amazon ECR.

Konsol Manajemen AWS

Lakukan langkah-langkah berikut untuk mengambil temuan pemindaian citra menggunakan Konsol Manajemen AWS.

Untuk mengambil temuan pemindaian gambar

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah tempat repositori Anda ada.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih repositori yang berisi citra yang akan diambil hasil pemindaiannya.
5. Pada halaman Gambar, di bawah kolom Tag gambar, pilih tag gambar untuk mengambil temuan pemindaian.
6. Untuk melihat detail selengkapnya di konsol Amazon Inspector, pilih nama kerentanan di kolom Nama.

AWS CLI

Gunakan AWS CLI perintah berikut untuk mengambil temuan pemindaian gambar menggunakan. AWS CLI Anda dapat menentukan citra menggunakan `imageTag` atau `imageDigest`, yang keduanya dapat diperoleh dengan menggunakan perintah CLI list-images.

- describe-image-scan-findings (AWS CLI)

  Contoh berikut menggunakan tanda citra.

  ```
  aws ecr describe-image-scan-findings \
       --repository-name name \
       --image-id imageTag=tag_name \
       --region us-east-2
  ```

  Contoh berikut menggunakan digest citra.

  ```
  aws ecr describe-image-scan-findings \
       --repository-name name \
       --image-id imageDigest=sha256_hash \
       --region us-east-2
  ```

Pindai gambar untuk kerentanan OS di Amazon ECR

Pemindaian dasar Amazon ECR menggunakan teknologi AWS asli untuk memindai gambar kontainer Anda dari kerentanan perangkat lunak. Pemindaian dasar menyediakan deteksi kerentanan di seluruh rangkaian luas sistem operasi populer, sumber lebih dari 50 umpan data untuk menghasilkan temuan untuk kerentanan dan eksposur umum (). CVEs Sumber-sumber ini termasuk penasihat keamanan vendor, umpan data, umpan intelijen ancaman, serta National Vulnerability Database (NVD) dan MITRE.

Pemindaian dasar Amazon ECR didukung di semua wilayah yang tercantum dalam AWS Layanan menurut Wilayah.

Amazon ECR menggunakan tingkat keparahan untuk CVE dari sumber distribusi hulu jika tersedia. Jika tidak, skor Common Vulnerability Scoring System (CVSS) digunakan. Skor CVSS dapat digunakan untuk mengetahui tingkat kepelikan kelemahan NVD. Untuk informasi selengkapnya, lihat Tingkat kepelikan kelemahan NVD.

Pemindaian dasar Amazon ECR mendukung filter untuk menentukan repositori mana yang akan dipindai saat push. Repositori apa pun yang tidak cocok dengan pemindaian pada filter push diatur ke frekuensi pemindaian manual yang berarti Anda harus memulai pemindaian secara manual. Gambar dapat dipindai sekali per 24 jam. 24 jam termasuk pemindaian awal pada push, jika dikonfigurasi, dan pemindaian manual apa pun. Dengan pemindaian dasar, Anda dapat memindai hingga 100.000 gambar per 24 jam dalam registri tertentu.

Temuan pemindaian citra yang telah diselesaikan terakhir dapat diambil untuk setiap citra. Saat pemindaian gambar selesai, Amazon ECR mengirimkan acara ke Amazon EventBridge. Untuk informasi selengkapnya, lihat Acara Amazon ECR dan EventBridge.

Dukungan sistem operasi untuk pemindaian dasar

Sebagai praktik keamanan terbaik dan untuk cakupan lanjutan, kami menyarankan Anda untuk terus menggunakan versi sistem operasi yang didukung. Sesuai dengan kebijakan vendor, sistem operasi yang dihentikan tidak lagi diperbarui dengan tambalan dan, dalam banyak kasus, nasihat keamanan baru tidak lagi dirilis untuk mereka. Selain itu, beberapa vendor menghapus penasihat dan deteksi keamanan yang ada dari feed mereka ketika sistem operasi yang terpengaruh mencapai akhir dukungan standar. Setelah distribusi kehilangan dukungan dari vendornya, Amazon ECR mungkin tidak lagi mendukung pemindaian kerentanan. Setiap temuan yang dihasilkan Amazon ECR untuk sistem operasi yang dihentikan harus digunakan hanya untuk tujuan informasi. Untuk mengetahui daftar lengkap sistem dan versi operasi yang didukung, lihat Sistem operasi yang didukung - Pemindaian Amazon Inspector di Panduan Pengguna Amazon Inspector.

Mengkonfigurasi pemindaian dasar untuk gambar di Amazon ECR

Secara default, Amazon ECR mengaktifkan pemindaian dasar untuk semua pendaftar pribadi. Akibatnya, kecuali Anda telah mengubah pengaturan pemindaian pada registri pribadi Anda, tidak perlu mengaktifkan pemindaian dasar.

Anda dapat menggunakan langkah-langkah berikut untuk menentukan satu atau lebih pemindaian pada filter push.

Untuk mengaktifkan pemindaian dasar untuk registri pribadi Anda

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/ private-registry/repositori
2. Dari bilah navigasi, pilih Wilayah untuk mengatur konfigurasi pemindaian.
3. Di panel navigasi, pilih Registri pribadi, Pemindaian.
4. Pada halaman konfigurasi Pemindaian, Untuk jenis Pindai pilih Pemindaian dasar.
5. Secara default semua repositori Anda diatur untuk pemindaian Manual. Anda dapat secara opsional mengonfigurasi pemindaian saat push dengan menentukan Pindai pada filter push. Anda dapat mengatur pemindaian pada push untuk semua repositori atau repositori individual. Untuk informasi selengkapnya, lihat Filter untuk memilih repositori mana yang dipindai di Amazon ECR.

> **Note**
>
> Jika pemindaian pada push diaktifkan untuk repositori, pemindaian juga dilakukan pada gambar yang dipulihkan setelah diarsipkan. Tidak ada pemindaian lama yang akan tersedia dari gambar yang dipulihkan.

Memindai gambar secara manual untuk kerentanan OS di Amazon ECR

Jika repositori Anda tidak dikonfigurasi untuk memindai saat push, Anda dapat memulai pemindaian gambar secara manual. Gambar dapat dipindai sekali per 24 jam. 24 jam termasuk pemindaian awal pada push, jika dikonfigurasi, dan pemindaian manual apa pun.

Untuk detail pemecahan masalah untuk beberapa masalah umum saat memindai citra, lihat Memecahkan masalah pemindaian gambar di Amazon ECR.

Konsol Manajemen AWS

Gunakan langkah-langkah berikut untuk memulai pemindaian citra manual menggunakan Konsol Manajemen AWS.

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/ private-registry/repositori
2. Dari bilah navigasi, pilih Wilayah untuk membuat repositori Anda.
3. Di panel navigasi, pilih Repositori.
4. Pada halaman Repositori, pilih repositori yang berisi citra yang akan dipindai.
5. Pada halaman citra, pilih citra yang akan dipindai dan kemudian pilih Pindai.

AWS CLI

- start-image-scan (AWS CLI)

  Contoh berikut menggunakan tanda citra.

  ```
  aws ecr start-image-scan --repository-name name --image-id imageTag=tag_name --
  region us-east-2
  ```

  Contoh berikut menggunakan digest citra.

```
aws ecr start-image-scan --repository-name name --image-id imageDigest=sha256_hash
 --region us-east-2
```

AWS Tools for Windows PowerShell

- Dapatkan- ECRImage ScanFinding (AWS Tools for Windows PowerShell)

  Contoh berikut menggunakan tanda citra.

  ```
  Start-ECRImageScan -RepositoryName name -ImageId_ImageTag tag_name -Region us-
  east-2 -Force
  ```

  Contoh berikut menggunakan digest citra.

  ```
  Start-ECRImageScan -RepositoryName name -ImageId_ImageDigest sha256_hash -
  Region us-east-2 -Force
  ```

Mengambil temuan untuk pemindaian dasar di Amazon ECR

Anda dapat mengambil temuan pemindaian untuk pemindaian gambar dasar yang terakhir selesai. Kerentanan perangkat lunak yang ditemukan terdaftar berdasarkan tingkat keparahan berdasarkan database Common Vulnerabilities and Exposures ()CVEs.

Untuk detail pemecahan masalah untuk beberapa masalah umum saat memindai citra, lihat Memecahkan masalah pemindaian gambar di Amazon ECR.

Konsol Manajemen AWS

Lakukan langkah-langkah berikut untuk mengambil temuan pemindaian citra menggunakan Konsol Manajemen AWS.

Untuk mengambil temuan pemindaian gambar

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/ private-registry/repositori
2. Dari bilah navigasi, pilih Wilayah untuk membuat repositori Anda.
3. Di panel navigasi, pilih Repositori.

4. Pada halaman Repositori, pilih repositori yang berisi citra yang akan diambil hasil pemindaiannya.
5. Pada halaman Gambar, di bawah kolom Tag gambar, pilih tag gambar untuk mengambil temuan pemindaian.

AWS CLI

Gunakan AWS CLI perintah berikut untuk mengambil temuan pemindaian gambar menggunakan. AWS CLI Anda dapat menentukan citra menggunakan `imageTag` atau `imageDigest`, yang keduanya dapat diperoleh dengan menggunakan perintah CLI list-images.

- describe-image-scan-findings (AWS CLI)

  Contoh berikut menggunakan tanda citra.

```
aws ecr describe-image-scan-findings --repository-name name --image-id
 imageTag=tag_name --region us-east-2
```

  Contoh berikut menggunakan digest citra.

```
aws ecr describe-image-scan-findings --repository-name name --image-id
 imageDigest=sha256_hash --region us-east-2
```

AWS Tools for Windows PowerShell

- Dapatkan- ECRImage ScanFinding (AWS Tools for Windows PowerShell)

  Contoh berikut menggunakan tanda citra.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageTag tag_name -
Region us-east-2
```

  Contoh berikut menggunakan digest citra.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2
```

Memecahkan masalah pemindaian gambar di Amazon ECR

Berikut ini adalah kegagalan pemindaian citra yang umum. Anda dapat melihat kesalahan seperti ini di konsol Amazon ECR dengan menampilkan detail gambar atau melalui API atau AWS CLI dengan menggunakan API. `DescribeImageScanFindings`

UnsupportedImageError

Anda mungkin mendapatkan `UnsupportedImageError` kesalahan saat mencoba melakukan pemindaian dasar pada gambar yang dibuat menggunakan sistem operasi yang Amazon ECR tidak mendukung pemindaian gambar dasar. Amazon ECR mendukung pemindaian kelemahan paket untuk sebagian besar versi distribusi Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine, dan RHEL Linux. Setelah distribusi kehilangan dukungan dari vendor, Amazon ECR mungkin tidak lagi mendukung pemindaian kelemahan. Amazon ECR tidak mendukung pemindaian citra yang dibangun dari citra Scratch Docker.

> **Important**
>
> Saat menggunakan pemindaian yang disempurnakan, Amazon Inspector mendukung pemindaian untuk sistem operasi dan jenis media tertentu. Untuk daftar selengkapnya, lihat Sistem operasi dan tipe media yang didukung di Panduan Pengguna Amazon Inspector.

Tingkat kepelikan `UNDEFINED` dikembalikan

Anda mungkin menerima temuan pindaian yang memiliki tingkat kepelikan `UNDEFINED`. Berikut ini adalah penyebab umum untuk ini:

- Kelemahan tidak ditetapkan sebagai prioritas oleh sumber CVE.
- Kelemahan diberikan prioritas yang tidak dikenali Amazon ECR.

Untuk menentukan tingkat kepelikan dan deskripsi kelemahan, Anda dapat melihat CVE langsung dari sumber.

Memahami status pemindaian **SCAN_ELIGIBILITY_EXPIRED**

Saat pemindaian yang disempurnakan menggunakan Amazon Inspector diaktifkan untuk registri pribadi Anda dan Anda melihat kerentanan pemindaian, Anda mungkin melihat status pemindaian. `SCAN_ELIGIBILITY_EXPIRED` Berikut ini adalah penyebab paling umum dari hal ini.

- Saat Anda awalnya mengaktifkan pemindaian yang disempurnakan untuk registri pribadi Anda, Amazon Inspector hanya mengenali gambar yang didorong ke Amazon ECR dalam 30 hari terakhir, berdasarkan stempel waktu push gambar. Gambar yang lebih tua akan memiliki status `SCAN_ELIGIBILITY_EXPIRED` pemindaian. Jika Anda ingin gambar-gambar ini dipindai oleh Amazon Inspector, Anda harus mendorongnya lagi ke repositori Anda.
- Jika durasi pemindaian ulang ECR diubah di konsol Amazon Inspector dan waktu tersebut berlalu, status pemindaian gambar diubah menjadi `inactive` kode `expired` alasan, dan semua temuan terkait untuk gambar dijadwalkan ditutup. Ini menghasilkan konsol Amazon ECR yang mencantumkan status pemindaian sebagai `SCAN_ELIGIBILITY_EXPIRED`.

Sinkronkan registri hulu dengan registri pribadi Amazon ECR

Menggunakan aturan pull through cache, Anda dapat menyinkronkan konten registri upstream dengan registri pribadi Amazon ECR Anda.

Amazon ECR saat ini mendukung pembuatan aturan cache tarik untuk pendaftar hulu berikut:

- Amazon ECR Public, Kubernetes container image registry, dan Quay (tidak memerlukan otentikasi)
- Docker Hub, Microsoft Azure Container Registry, Container Registry, GitHub GitLab Container Registry dan Chainguard Registry (memerlukan otentikasi dengan rahasia) AWS Secrets Manager
- Amazon ECR (memerlukan otentikasi dengan peran AWS IAM)

Untuk GitLab Container Registry, Amazon ECR mendukung pull through cache hanya dengan GitLab penawaran Software as a Service (SaaS). Untuk informasi lebih lanjut tentang menggunakan penawaran GitLab SaaS, lihat GitLab .com.

Untuk registrasi upstream yang memerlukan otentikasi dengan rahasia (seperti Docker Hub), Anda harus menyimpan kredensialnya secara rahasia. AWS Secrets Manager Anda dapat menggunakan konsol Amazon ECR untuk membuat rahasia Secrets Manager untuk setiap registri upstream yang diautentikasi. Untuk informasi selengkapnya tentang membuat rahasia Secrets Manager menggunakan konsol Secrets Manager, lihatMenyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager.

Untuk Amazon ECR, Anda harus membuat peran IAM jika pendaftar ECR Amazon hulu dan hilir milik akun yang berbeda. AWS Untuk informasi selengkapnya tentang cara membuat sebuah IAM role, lihat Kebijakan IAM diperlukan untuk ECR lintas akun ke ECR menarik cache.

Setelah Anda membuat aturan pull through cache untuk registri upstream, tarik gambar dari registri upstream menggunakan URI registri pribadi Amazon ECR Anda. Amazon ECR kemudian membuat repositori dan menyimpan gambar itu di registri pribadi Anda. Untuk permintaan tarik berikutnya dari gambar yang di-cache dengan tag yang diberikan, Amazon ECR memeriksa registri hulu untuk versi baru gambar dengan tag spesifik tersebut dan mencoba memperbarui gambar di registri pribadi Anda setidaknya sekali setiap 24 jam.

Templat pembuatan repositori

Amazon ECR telah menambahkan dukungan untuk template pembuatan repositori, yang memberi Anda kontrol untuk menentukan konfigurasi awal untuk repositori baru yang dibuat oleh Amazon ECR atas nama Anda menggunakan aturan pull through cache. Setiap template berisi awalan namespace repositori yang digunakan untuk mencocokkan repositori baru dengan template tertentu. Template dapat menentukan konfigurasi untuk semua pengaturan repositori termasuk kebijakan akses berbasis sumber daya, kekekalan tag, enkripsi, dan kebijakan siklus hidup. Pengaturan dalam template pembuatan repositori hanya diterapkan selama pembuatan repositori dan tidak berpengaruh pada repositori atau repositori yang ada yang dibuat menggunakan metode lain. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.

Pertimbangan untuk menggunakan aturan pull through cache

Pertimbangkan hal berikut saat menggunakan Amazon ECR tarik melalui aturan cache.

- Membuat aturan pull through cache tidak didukung di Wilayah berikut.
  - China (Beijing) (`cn-north-1`)
  - China (Ningxia) (`cn-northwest-1`)
  - AWS GovCloud (AS-Timur) (`us-gov-east-1`)
  - AWS GovCloud (AS-Barat) (`us-gov-west-1`)
- AWS Lambda tidak mendukung penarikan gambar kontainer dari Amazon ECR menggunakan aturan cache tarik.
- Saat menarik gambar menggunakan cache tarik, titik akhir layanan Amazon ECR FIPS tidak didukung saat pertama kali gambar ditarik. Menggunakan titik akhir layanan Amazon ECR FIPS berfungsi pada tarikan berikutnya.
- Untuk repositori upstream yang memerlukan otentikasi, ketika gambar ditarik melalui URI registri pribadi Amazon ECR untuk pertama kalinya atau untuk memperbarui cache, penarikan gambar dimulai oleh pengguna yang terkait dengan kredensyal yang dikonfigurasi dalam aturan pull through cache. Penarikan berikutnya akan mengembalikan gambar langsung dari cache di registri pribadi pelanggan.
- Untuk repositori upstream yang tidak memerlukan otentikasi, ketika gambar ditarik melalui URI registri pribadi Amazon ECR, penarikan gambar dimulai oleh alamat IP. AWS

- Saat pelanggan menarik gambar yang di-cache melalui URI registri pribadi Amazon ECR, Amazon ECR memeriksa apakah gambar tersebut telah memvalidasi gambar terhadap registri hulu dalam 24 jam terakhir. Jika jendela 24 jam telah kedaluwarsa, Amazon ECR mengirimkan permintaan upstream untuk memeriksa versi yang lebih baru dan memperbarui cache jika ada. Jika jendela belum kedaluwarsa, Amazon ECR menyajikan gambar yang di-cache tanpa menghubungi hulu.
- Memanggil `ListImageReferrers` API ke repositori pull through cache yang dibuat mengembalikan artefak perujuk yang sesuai dengan OCI ke cache pribadi.
- Amazon ECR memeriksa apakah artefak perujuk telah diperbarui dalam 6 jam terakhir. Jika jendela 6 jam telah kedaluwarsa, Amazon ECR mengirimkan permintaan upstream untuk memeriksa versi yang lebih baru dan memperbarui cache jika ada.
- Jika Amazon ECR tidak dapat memperbarui gambar dari registri hulu karena alasan apa pun dan gambar ditarik, gambar cache terakhir akan tetap ditarik.
- Saat membuat rahasia Secrets Manager yang berisi kredensyal registri hulu, nama rahasia harus menggunakan awalan. `ecr-pullthroughcache/` Rahasianya juga harus berada di akun dan Wilayah yang sama tempat aturan pull through cache dibuat.
- Saat gambar multi-arsitektur ditarik menggunakan aturan cache pull through, daftar manifes dan setiap gambar yang direferensikan dalam daftar manifes ditarik ke repositori Amazon ECR. Jika Anda hanya ingin menarik arsitektur tertentu, Anda dapat menarik gambar menggunakan intisari gambar atau tag yang terkait dengan arsitektur daripada tag yang terkait dengan daftar manifes.
- Amazon ECR menggunakan peran IAM terkait layanan, yang menyediakan izin yang diperlukan Amazon ECR untuk membuat repositori, mengambil nilai rahasia Secrets Manager untuk otentikasi, dan mendorong gambar yang di-cache atas nama Anda. Peran IAM terkait layanan dibuat secara otomatis saat aturan pull through cache dibuat. Untuk informasi selengkapnya, lihat Peran terkait layanan Amazon ECR untuk menarik cache.
- Secara default, prinsipal IAM yang menarik gambar yang di-cache memiliki izin yang diberikan kepada mereka melalui kebijakan IAM mereka. Anda dapat menggunakan kebijakan izin registri pribadi Amazon ECR untuk cakupan lebih lanjut izin entitas IAM. Untuk informasi selengkapnya, lihat Menggunakan izin registri.
- Repositori Amazon ECR yang dibuat menggunakan alur kerja pull through cache diperlakukan seperti repositori ECR Amazon lainnya. Semua fitur repositori, seperti replikasi dan pemindaian gambar didukung.
- Saat Amazon ECR membuat repositori baru atas nama Anda menggunakan tindakan pull through cache, pengaturan default berikut diterapkan ke repositori kecuali ada template pembuatan repositori yang cocok. Anda dapat menggunakan template pembuatan repositori untuk menentukan pengaturan yang diterapkan ke repositori yang dibuat oleh Amazon ECR atas nama

Anda. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.

- Kekekalan tag - Kekekalan tag menentukan apakah tag gambar dapat ditimpa. Secara default, tag gambar dapat berubah (dapat ditimpa). Anda dapat mengubah perilaku tag dengan mengonfigurasi filter pengecualian tag di kotak teks pengecualian tag Mutable saat Mutable dipilih, atau kotak teks pengecualian tag Immutable saat Immutable dipilih.
- Enkripsi — `AES256` Enkripsi default digunakan.
- Izin repositori - Dihilangkan, tidak ada kebijakan izin repositori yang diterapkan.
- Kebijakan siklus hidup - Dihilangkan, tidak ada kebijakan siklus hidup yang diterapkan.
- Tag sumber daya - Dihilangkan, tidak ada tag sumber daya yang diterapkan.

- Mengaktifkan kekekalan tag gambar untuk repositori menggunakan aturan cache tarik melalui akan mencegah Amazon ECR memperbarui gambar menggunakan tag yang sama.
- Ketika gambar ditarik menggunakan aturan pull through cache untuk pertama kalinya rute ke internet mungkin diperlukan. Ada keadaan tertentu di mana rute ke internet diperlukan sehingga yang terbaik adalah mengatur rute untuk menghindari kegagalan. Jadi, jika Anda telah mengonfigurasi Amazon ECR untuk AWS PrivateLink menggunakan titik akhir VPC antarmuka, maka Anda perlu memastikan tarikan pertama memiliki rute ke internet. Salah satu cara untuk melakukannya adalah dengan membuat subnet publik di VPC yang sama, dengan gateway internet, dan kemudian merutekan semua lalu lintas keluar ke internet dari subnet pribadi mereka ke subnet publik. Penarikan gambar berikutnya menggunakan aturan pull through cache tidak memerlukan ini. Untuk informasi selengkapnya, lihat Contoh opsi perutean di Panduan Pengguna Amazon Virtual Private Cloud.

Izin IAM diperlukan untuk menyinkronkan registri upstream dengan registri pribadi Amazon ECR

Selain izin Amazon ECR API yang diperlukan untuk mengautentikasi ke registri pribadi dan untuk mendorong dan menarik gambar, izin tambahan berikut diperlukan untuk menggunakan aturan cache tarik melalui secara efektif.

- `ecr:CreatePullThroughCacheRule`— Memberikan izin untuk membuat aturan cache pull through. Izin ini harus diberikan melalui kebijakan IAM berbasis identitas.
- `ecr:BatchImportUpstreamImage`— Memberikan izin untuk mengambil gambar eksternal dan mengimpornya ke registri pribadi Anda. Izin ini dapat diberikan dengan menggunakan kebijakan

izin registri pribadi, kebijakan IAM berbasis identitas, atau dengan menggunakan kebijakan izin repositori berbasis sumber daya. Untuk informasi selengkapnya tentang menggunakan izin repositori, lihat. Kebijakan repositori pribadi di Amazon ECR

- `ecr:CreateRepository`— Memberikan izin untuk membuat repositori di registri pribadi. Izin ini diperlukan jika repositori yang menyimpan gambar yang di-cache belum ada. Izin ini dapat diberikan oleh kebijakan IAM berbasis identitas atau kebijakan izin registri pribadi.

Menggunakan izin registri

Izin registri pribadi Amazon ECR dapat digunakan untuk mencakup izin entitas IAM individu untuk menggunakan cache pull through. Jika entitas IAM memiliki lebih banyak izin yang diberikan oleh kebijakan IAM daripada yang diberikan oleh kebijakan izin registri, kebijakan IAM akan diutamakan. Misalnya, jika pengguna memiliki `ecr:*` izin yang diberikan, tidak ada izin tambahan yang diperlukan di tingkat registri.

Untuk membuat kebijakan izin registri pribadi ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pernyataan izin registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Izin registri.
4. Pada halaman Izin registri, pilih Hasilkan pernyataan.
5. Untuk setiap pull through pernyataan kebijakan izin cache yang ingin Anda buat, lakukan hal berikut.

    a. Untuk jenis Policy, pilih Pull through cache policy.

    b. Untuk id Pernyataan, berikan nama untuk kebijakan pernyataan cache tarik melalui.

    c. Untuk entitas IAM, tentukan pengguna, grup, atau peran yang akan disertakan dalam kebijakan.

    d. Untuk namespace Repositori, pilih aturan pull through cache untuk mengaitkan kebijakan dengan.

    e. Untuk nama Repositori, tentukan nama dasar repositori untuk menerapkan aturan. Misalnya, jika Anda ingin menentukan repositori Amazon Linux di Amazon ECR Public, nama repositori akan menjadi. `amazonlinux`

Untuk membuat kebijakan izin registri pribadi ()AWS CLI

Gunakan AWS CLI perintah berikut untuk menentukan izin registri pribadi menggunakan. AWS CLI

1. Buat file lokal bernama `ptc-registry-policy.json` dengan isi kebijakan registri Anda. Contoh berikut memberikan `ecr-pull-through-cache-user` izin untuk membuat repositori dan menarik gambar dari Amazon ECR Public, yang merupakan sumber upstream yang terkait dengan aturan cache pull through yang dibuat sebelumnya.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PullThroughCacheFromReadOnlyRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/ecr-pull-through-cache-user"
      },
      "Action": [
        "ecr:CreateRepository",
        "ecr:BatchImportUpstreamImage"
      ],
      "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/ecr-public/
*"
    }
  ]
}
```

   > **Important**
   >
   > `ecr-CreateRepository`Izin hanya diperlukan jika repositori yang menyimpan gambar yang di-cache belum ada. Misalnya, jika tindakan pembuatan repositori dan tindakan tarik gambar sedang dilakukan oleh prinsipal IAM terpisah seperti administrator dan pengembang.

2. Gunakan put-registry-policyperintah untuk mengatur kebijakan registri.

```
aws ecr put-registry-policy \
     --policy-text file://ptc-registry.policy.json
```

Langkah selanjutnya

Setelah Anda siap untuk mulai menggunakan aturan pull through cache, berikut ini adalah langkah selanjutnya.

- Buat aturan cache pull through. Untuk informasi selengkapnya, lihat Membuat aturan cache tarik melalui di Amazon ECR.
- Buat template pembuatan repositori. Template pembuatan repositori memberi Anda kontrol untuk menentukan pengaturan yang akan digunakan untuk repositori baru yang dibuat oleh Amazon ECR atas nama Anda selama tindakan pull through cache. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.

Menyiapkan izin untuk ECR lintas akun ke ECR PTC

Fitur cache Amazon ECR ke Amazon ECR (ECR ke ECR) memungkinkan sinkronisasi otomatis gambar antara Wilayah, akun, AWS atau keduanya. Dengan ECR ke ECR PTC, Anda dapat mendorong gambar ke registri ECR Amazon utama Anda dan mengonfigurasi aturan pull through cache untuk menyimpan gambar di registri Amazon ECR hilir.

Kebijakan IAM diperlukan untuk ECR lintas akun ke ECR menarik cache

Untuk menyimpan gambar di antara pendaftar Amazon ECR di berbagai AWS akun, buat peran IAM di akun hilir dan konfigurasikan kebijakan di bagian ini untuk memberikan izin berikut:

- Amazon ECR memerlukan izin untuk menarik gambar dari registri ECR Amazon hulu atas nama Anda. Anda dapat memberikan izin ini dengan membuat peran IAM dan kemudian menentukannya dalam aturan cache tarik melalui Anda.
- Pemilik registri hulu juga harus memberikan pemilik registri cache dengan izin yang diperlukan untuk menarik gambar ke kebijakan sumber daya.

Kebijakan

- Membuat peran IAM untuk menentukan izin cache tarik melalui
- Membuat kebijakan Trust untuk peran IAM
- Membuat kebijakan sumber daya di registri ECR Amazon hulu

Membuat peran IAM untuk menentukan izin cache tarik melalui

Contoh berikut menunjukkan kebijakan izin yang memberikan izin peran IAM untuk menarik gambar dari registri ECR Amazon hulu atas nama Anda. Saat Amazon ECR mengambil peran tersebut, Amazon akan menerima izin yang ditentukan dalam kebijakan ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken",
                "ecr:BatchImportUpstreamImage",
                "ecr:BatchGetImage",
                "ecr:GetImageCopyStatus",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

Membuat kebijakan Trust untuk peran IAM

Contoh berikut menunjukkan kebijakan kepercayaan yang mengidentifikasi cache penarikan ECR Amazon sebagai prinsip AWS layanan yang dapat mengambil peran tersebut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "pullthroughcache.ecr.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Membuat kebijakan sumber daya di registri ECR Amazon hulu

Pemilik registri Amazon ECR hulu juga harus menambahkan kebijakan registri atau kebijakan repositori untuk memberikan pemilik registri hilir izin yang diperlukan untuk melakukan tindakan berikut.

> **Note**
>
> Kebijakan sumber daya berikut hanya diperlukan untuk penarikan ECR lintas akun ke ECR melalui konfigurasi cache. Untuk cache penarikan lintas wilayah dengan akun yang sama, Anda hanya memerlukan kebijakan peran IAM dan kebijakan kepercayaan yang ditampilkan di bagian sebelumnya. Izin utama akun root tidak diperlukan ketika pendaftar hulu dan hilir berada di akun yang sama. AWS

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::444455556666:root"
    },
    "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchImportUpstreamImage",
```

```
            "ecr:GetImageCopyStatus"
        ],
        "Resource": "arn:aws:ecr:region:111122223333:repository/*"
}
```

Membuat aturan cache tarik melalui di Amazon ECR

Untuk setiap registri upstream yang berisi gambar yang ingin Anda cache di registri pribadi Amazon ECR Anda, Anda harus membuat aturan cache pull through.

Untuk registrasi upstream yang memerlukan otentikasi dengan rahasia, Anda harus menyimpan kredensialnya dalam rahasia Secrets Manager. Anda dapat menggunakan rahasia yang ada atau membuat rahasia bersama baru. Anda dapat membuat rahasia Secrets Manager di konsol Amazon ECR atau konsol Secrets Manager. Untuk membuat rahasia Secrets Manager menggunakan konsol Secrets Manager alih-alih konsol Amazon ECR, lihatMenyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager.

Prasyarat

- Verifikasi bahwa Anda memiliki izin IAM yang tepat untuk membuat aturan pull through cache. Untuk informasi, lihat Izin IAM diperlukan untuk menyinkronkan registri upstream dengan registri pribadi Amazon ECR.
- Untuk pendaftar upstream yang memerlukan otentikasi dengan rahasia: Jika Anda ingin menggunakan rahasia yang ada, verifikasi bahwa rahasia Secrets Manager memenuhi persyaratan berikut:
  - Nama rahasia dimulai dengan`ecr-pullthroughcache/`. Konsol Manajemen AWS Satu-satunya menampilkan rahasia Secrets Manager dengan `ecr-pullthroughcache/` awalan.
  - Akun dan Wilayah tempat rahasianya berada harus sesuai dengan akun dan Wilayah tempat aturan pull through cache berada.

Untuk membuat aturan pull through cache (Konsol Manajemen AWS)

Langkah-langkah berikut menunjukkan cara membuat aturan cache pull through dan rahasia Secrets Manager menggunakan konsol Amazon ECR. Untuk membuat rahasia menggunakan konsol Secrets Manager, lihatMenyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager.

Untuk Amazon ECR Public, registri kontainer Kubernetes, atau Quay

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.
5. Pada Langkah 1: Tentukan halaman sumber, untuk Registry, pilih Amazon ECR Public, Kubernetes, atau Quay dari daftar registri upstream dan kemudian pilih Next.
6. Pada Langkah 2: Tentukan halaman tujuan, untuk awalan repositori Amazon ECR, tentukan awalan namespace repositori yang akan digunakan saat menyimpan gambar yang ditarik dari registri publik sumber dan kemudian pilih Berikutnya. Secara default, namespace diisi tetapi namespace khusus dapat ditentukan juga.
7. Pada Langkah 3: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.
8. Ulangi langkah sebelumnya untuk setiap pull through cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk Docker Hub

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.
5. Pada Langkah 1: Tentukan halaman sumber, untuk Registry, pilih Docker Hub, Next.
6. Pada Langkah 2: Konfigurasikan halaman otentikasi, untuk kredensyal Upstream, Anda harus menyimpan kredensyal otentikasi Anda untuk Docker Hub secara rahasia. AWS Secrets Manager Anda dapat menentukan rahasia yang ada atau menggunakan konsol Amazon ECR untuk membuat rahasia baru.

    a. Untuk menggunakan rahasia yang ada, pilih Gunakan AWS rahasia yang ada. Untuk nama Rahasia gunakan drop-down untuk memilih rahasia yang ada, lalu pilih Berikutnya.

> **Note**
>
> Konsol Manajemen AWS Satu-satunya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan. Rahasianya juga harus berada di akun dan Wilayah yang sama tempat aturan pull through cache dibuat.

b. Untuk membuat rahasia baru, pilih Buat AWS rahasia, lakukan hal berikut, lalu pilih Berikutnya.

   i. Untuk nama Rahasia, tentukan nama deskriptif untuk rahasia tersebut. Nama rahasia harus berisi 1-512 karakter Unicode.

   ii. Untuk email Docker Hub, tentukan email Docker Hub Anda.

   iii. Untuk token akses Docker Hub, tentukan token akses Docker Hub Anda. Untuk informasi selengkapnya tentang membuat token akses Docker Hub, lihat Membuat dan mengelola token akses dalam dokumentasi Docker.

7. Pada Langkah 3: Tentukan halaman tujuan, untuk awalan repositori Amazon ECR, tentukan namespace repositori yang akan digunakan saat menyimpan gambar yang ditarik dari registri publik sumber dan kemudian pilih Berikutnya.

   Secara default, namespace diisi tetapi namespace khusus dapat ditentukan juga.

8. Pada Langkah 4: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.

9. Ulangi langkah sebelumnya untuk setiap pull through cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk Registri GitHub Kontainer

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.
5. Pada Langkah 1: Tentukan halaman sumber, untuk Registry, pilih Registry GitHub Container, Next.

6. Pada Langkah 2: Konfigurasikan halaman otentikasi, untuk kredensyal Upstream, Anda harus menyimpan kredensyal otentikasi Anda untuk Container Registry secara rahasia. GitHub AWS Secrets Manager Anda dapat menentukan rahasia yang ada atau menggunakan konsol Amazon ECR untuk membuat rahasia baru.

   a. Untuk menggunakan rahasia yang ada, pilih Gunakan AWS rahasia yang ada. Untuk nama Rahasia gunakan drop-down untuk memilih rahasia yang ada, lalu pilih Berikutnya.

      > **Note**
      >
      > Konsol Manajemen AWS Satu-satunya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan. Rahasianya juga harus berada di akun dan Wilayah yang sama tempat aturan pull through cache dibuat.

   b. Untuk membuat rahasia baru, pilih Buat AWS rahasia, lakukan hal berikut, lalu pilih Berikutnya.

      i. Untuk nama Rahasia, tentukan nama deskriptif untuk rahasia tersebut. Nama rahasia harus berisi 1-512 karakter Unicode.

      ii. Untuk nama pengguna GitHub Container Registry, tentukan nama pengguna GitHub Container Registry Anda.

      iii. Untuk token akses GitHub Container Registry, tentukan token akses GitHub Container Registry Anda. Untuk informasi selengkapnya tentang membuat token GitHub akses, lihat Mengelola token akses pribadi Anda dalam GitHub dokumentasi.

7. Pada Langkah 3: Tentukan halaman tujuan, untuk awalan repositori Amazon ECR, tentukan namespace repositori yang akan digunakan saat menyimpan gambar yang ditarik dari registri publik sumber dan kemudian pilih Berikutnya.

   Secara default, namespace diisi tetapi namespace khusus dapat ditentukan juga.

8. Pada Langkah 4: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.

9. Ulangi langkah sebelumnya untuk setiap pull through cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk Microsoft Azure Container Registry

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.
5. Pada Langkah 1: Tentukan halaman sumber, lakukan hal berikut.
   a. Untuk Registry, pilih Microsoft Azure Container Registry
   b. Untuk URL registri Sumber, tentukan nama registri kontainer Microsoft Azure Anda, lalu pilih Berikutnya.

   > **Important**
   >
   > Anda hanya perlu menentukan awalan, karena `.azurecr.io` akhiran diisi atas nama Anda.

6. Pada Langkah 2: Konfigurasikan halaman otentikasi, untuk kredensyal Upstream, Anda harus menyimpan kredensyal otentikasi Anda untuk Microsoft Azure Container Registry secara rahasia. AWS Secrets Manager Anda dapat menentukan rahasia yang ada atau menggunakan konsol Amazon ECR untuk membuat rahasia baru.
   a. Untuk menggunakan rahasia yang ada, pilih Gunakan AWS rahasia yang ada. Untuk nama Rahasia gunakan drop-down untuk memilih rahasia yang ada, lalu pilih Berikutnya.

   > **Note**
   >
   > Konsol Manajemen AWS Satu-satunya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan. Rahasianya juga harus berada di akun dan Wilayah yang sama tempat aturan pull through cache dibuat.

   b. Untuk membuat rahasia baru, pilih Buat AWS rahasia, lakukan hal berikut, lalu pilih Berikutnya.
      i. Untuk nama Rahasia, tentukan nama deskriptif untuk rahasia tersebut. Nama rahasia harus berisi 1-512 karakter Unicode.

ii. Untuk nama pengguna Microsoft Azure Container Registry, tentukan nama pengguna Microsoft Azure Container Registry Anda.

iii. Untuk token akses Microsoft Azure Container Registry, tentukan token akses Microsoft Azure Container Registry Anda. Untuk informasi selengkapnya tentang cara membuat token akses Microsoft Azure Container Registry, lihat Membuat token - portal di dokumentasi Microsoft Azure.

7. Pada Langkah 3: Tentukan halaman tujuan, untuk awalan repositori Amazon ECR, tentukan namespace repositori yang akan digunakan saat menyimpan gambar yang ditarik dari registri publik sumber dan kemudian pilih Berikutnya.

   Secara default, namespace diisi tetapi namespace khusus dapat ditentukan juga.

8. Pada Langkah 4: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.

9. Ulangi langkah sebelumnya untuk setiap pull through cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk Registri GitLab Kontainer

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.

2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.

3. Di panel navigasi, pilih Registri pribadi, Tarik cache.

4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.

5. Pada Langkah 1: Tentukan halaman sumber, untuk Registry, pilih Registry GitLab Container, Next.

6. Pada Langkah 2: Konfigurasikan halaman otentikasi, untuk kredensyal Upstream, Anda harus menyimpan kredensyal otentikasi Anda untuk Container Registry secara rahasia. GitLab AWS Secrets Manager Anda dapat menentukan rahasia yang ada atau menggunakan konsol Amazon ECR untuk membuat rahasia baru.

   a. Untuk menggunakan rahasia yang ada, pilih Gunakan AWS rahasia yang ada. Untuk nama Rahasia gunakan drop-down untuk memilih rahasia yang ada, lalu pilih Berikutnya. Untuk informasi selengkapnya tentang membuat rahasia Secrets Manager menggunakan konsol Secrets Manager, lihatMenyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager.

> **Note**
>
> Konsol Manajemen AWS Satu-satunya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan. Rahasianya juga harus berada di akun dan Wilayah yang sama tempat aturan pull through cache dibuat.

b. Untuk membuat rahasia baru, pilih Buat AWS rahasia, lakukan hal berikut, lalu pilih Berikutnya.

   i. Untuk nama Rahasia, tentukan nama deskriptif untuk rahasia tersebut. Nama rahasia harus berisi 1-512 karakter Unicode.

   ii. Untuk nama pengguna GitLab Container Registry, tentukan nama pengguna GitLab Container Registry Anda.

   iii. Untuk token akses GitLab Container Registry, tentukan token akses GitLab Container Registry Anda. Untuk informasi selengkapnya tentang membuat token akses GitLab Container Registry, lihat Token akses pribadi, Token akses grup, atau Token akses Proyek, dalam GitLab dokumentasi.

7. Pada Langkah 3: Tentukan halaman tujuan, untuk awalan repositori Amazon ECR, tentukan namespace repositori yang akan digunakan saat menyimpan gambar yang ditarik dari registri publik sumber dan kemudian pilih Berikutnya.

   Secara default, namespace diisi tetapi namespace khusus dapat ditentukan juga.

8. Pada Langkah 4: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.

9. Ulangi langkah sebelumnya untuk setiap pull through cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk registri pribadi Amazon ECR dalam akun Anda AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah tempat Anda ingin mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.

5. Pada Langkah 1: Tentukan halaman hulu, untuk Registry, pilih Amazon ECR Private and This account. Untuk Wilayah, pilih Wilayah untuk registri ECR Amazon hulu, lalu pilih Berikutnya.
6. Pada Langkah 2: Tentukan halaman ruang nama, untuk Cache namespace, pilih apakah akan membuat tarik melalui repositori cache dengan awalan tertentu atau tanpa awalan. Jika Anda memilih awalan tertentu, Anda harus menentukan nama awalan yang akan digunakan sebagai bagian dari namespace untuk menyimpan gambar dari registri hulu.
7. Untuk namespace Upstream, pilih apakah akan menarik dari awalan tertentu yang ada di registri upstream. Jika Anda memilih tidak ada awalan, Anda dapat menarik dari repositori apa pun di registri hulu. Tentukan awalan repositori upstream jika diminta, lalu pilih Berikutnya.

   > **Note**
   >
   > Untuk mempelajari lebih lanjut tentang menyesuaikan cache dan ruang nama upstream, lihat. Menyesuaikan awalan repositori untuk ECR ke ECR menarik cache
8. Pada Langkah 3: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.
9. Ulangi langkah-langkah ini untuk setiap penarikan cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk registri pribadi Amazon ECR dari akun lain AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.
5. Pada Langkah 1: Tentukan halaman hulu, untuk Registry, pilih akun Amazon ECR Private dan Cross. Untuk Wilayah, pilih Wilayah untuk registri ECR Amazon hulu. Untuk Akun, tentukan ID AWS akun untuk registri ECR Amazon hulu, lalu pilih Berikutnya.
6. Pada Langkah 2: Tentukan halaman izin, untuk peran IAM, pilih peran yang akan digunakan untuk tarik lintas akun melalui akses cache dan kemudian pilih Buat.

> **Note**
>
> Pastikan Anda memilih peran IAM yang menggunakan izin yang dibuat. Kebijakan IAM diperlukan untuk ECR lintas akun ke ECR menarik cache

7. Pada Langkah 3: Tentukan halaman ruang nama, untuk Cache namespace, pilih apakah akan membuat tarik melalui repositori cache dengan awalan tertentu atau tanpa awalan. Jika Anda memilih awalan tertentu, Anda harus menentukan nama awalan yang akan digunakan sebagai bagian dari namespace untuk menyimpan gambar dari registri hulu.
8. Untuk namespace Upstream, pilih apakah akan menarik dari awalan tertentu yang ada di registri upstream. Jika Anda memilih tidak ada awalan, Anda dapat menarik dari repositori apa pun di registri hulu. Tentukan awalan repositori upstream jika diminta, lalu pilih Berikutnya.

> **Note**
>
> Untuk mempelajari lebih lanjut tentang menyesuaikan cache dan ruang nama upstream, lihat. Menyesuaikan awalan repositori untuk ECR ke ECR menarik cache

9. Pada Langkah 4: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.
10. Ulangi langkah-langkah ini untuk setiap penarikan cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk Registri Chainguard

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan registri pribadi Anda.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Tarik melalui cache, pilih Tambahkan aturan.
5. Pada Langkah 1: Tentukan halaman sumber, untuk Registry, pilih Chainguard Registry, Next.
6. Pada Langkah 2: Konfigurasikan halaman otentikasi, untuk kredensyal Upstream, Anda harus menyimpan kredensyal otentikasi Anda untuk Chainguard Registry secara rahasia. AWS Secrets Manager Anda dapat menentukan rahasia yang ada atau menggunakan konsol Amazon ECR untuk membuat rahasia baru.

a. Untuk menggunakan rahasia yang ada, pilih Gunakan AWS rahasia yang ada. Untuk nama Rahasia gunakan drop-down untuk memilih rahasia yang ada, lalu pilih Berikutnya. Untuk informasi selengkapnya tentang membuat rahasia Secrets Manager menggunakan konsol Secrets Manager, lihatMenyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager.

> **Note**
>
> Konsol Manajemen AWS Satu-satunya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan. Rahasianya juga harus berada di akun dan Wilayah yang sama tempat aturan pull through cache dibuat.

b. Untuk membuat rahasia baru, pilih Buat AWS rahasia, lakukan hal berikut, lalu pilih Berikutnya.

   i. Untuk nama Rahasia, tentukan nama deskriptif untuk rahasia tersebut. Nama rahasia harus berisi 1-512 karakter Unicode.

   ii. Untuk nama pengguna Chainguard Registry, tentukan nama pengguna Registry Chainguard Anda.

   iii. Untuk token tarik Chainguard Registry, tentukan token tarik Chainguard Registry Anda. Untuk informasi selengkapnya tentang membuat token tarik Chainguard Registry, lihat Mengautentikasi dengan Token Tarik di dokumentasi Chainguard.

7. Pada Langkah 3: Tentukan halaman tujuan, untuk awalan repositori Amazon ECR, tentukan namespace repositori yang akan digunakan saat menyimpan gambar yang ditarik dari registri sumber dan kemudian pilih Berikutnya.

   Secara default, namespace diisi tetapi namespace khusus dapat ditentukan juga.

8. Pada Langkah 4: Tinjau dan buat halaman, tinjau konfigurasi aturan pull through cache dan kemudian pilih Create.

9. Ulangi langkah sebelumnya untuk setiap pull through cache yang ingin Anda buat. Aturan pull through cache dibuat secara terpisah untuk setiap Wilayah.

Untuk membuat aturan pull through cache (AWS CLI)

Gunakan AWS CLI perintah create-pull-through-cache-rule untuk membuat aturan cache pull through untuk registri pribadi Amazon ECR. Untuk registrasi upstream yang memerlukan otentikasi dengan rahasia, Anda harus menyimpan kredensialnya dalam rahasia Secrets Manager. Untuk membuat rahasia menggunakan konsol Secrets Manager, lihatMenyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager.

Contoh berikut disediakan untuk setiap registri upstream yang didukung.

Untuk Amazon ECR Publik

Contoh berikut membuat aturan cache pull through untuk registri Publik Amazon ECR. Ini menentukan awalan repositori`ecr-public`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `ecr-public/`*`upstream-repository-name`*

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --upstream-registry-url public.ecr.aws \
     --region us-east-2
```

Untuk Registri Kontainer Kubernetes

Contoh berikut membuat aturan pull through cache untuk registri publik Kubernetes. Ini menentukan awalan repositori`kubernetes`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `kubernetes/`*`upstream-repository-name`*

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix kubernetes \
     --upstream-registry-url registry.k8s.io \
     --region us-east-2
```

Untuk Quay

Contoh berikut membuat aturan pull through cache untuk registri publik Quay. Ini menentukan awalan repositori`quay`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `quay/`*`upstream-repository-name`*

```
aws ecr create-pull-through-cache-rule \
```

```
    --ecr-repository-prefix quay \
    --upstream-registry-url quay.io \
    --region us-east-2
```

Untuk Docker Hub

Contoh berikut membuat aturan pull through cache untuk registri Docker Hub. Ini menentukan awalan repositori`docker-hub`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `docker-hub/`*`upstream-repository-name`* Anda harus menentukan Nama Sumber Daya Amazon (ARN) lengkap dari rahasia yang berisi kredensyal Docker Hub Anda.

```
aws ecr create-pull-through-cache-rule \
    --ecr-repository-prefix docker-hub \
    --upstream-registry-url registry-1.docker.io \
    --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
    --region us-east-2
```

Untuk Registri GitHub Kontainer

Contoh berikut membuat aturan pull through cache untuk GitHub Container Registry. Ini menentukan awalan repositori`github`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `github/`*`upstream-repository-name`* Anda harus menentukan Nama Sumber Daya Amazon (ARN) lengkap dari rahasia yang berisi kredenal Registri GitHub Penampung Anda.

```
aws ecr create-pull-through-cache-rule \
    --ecr-repository-prefix github \
    --upstream-registry-url ghcr.io \
    --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
    --region us-east-2
```

Untuk Microsoft Azure Container Registry

Contoh berikut membuat aturan pull through cache untuk Microsoft Azure Container Registry. Ini menentukan awalan repositori`azure`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `azure/`*`upstream-repository-name`* Anda harus menentukan Nama Sumber Daya Amazon (ARN) lengkap dari rahasia yang berisi kredensyal Microsoft Azure Container Registry Anda.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix azure \
     --upstream-registry-url myregistry.azurecr.io \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Untuk Registri GitLab Kontainer

Contoh berikut membuat aturan pull through cache untuk GitLab Container Registry. Ini menentukan awalan repositori`gitlab`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `gitlab/`*`upstream-repository-name`* Anda harus menentukan Nama Sumber Daya Amazon (ARN) lengkap dari rahasia yang berisi kredenal Registri GitLab Penampung Anda.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix gitlab \
     --upstream-registry-url registry.gitlab.com \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Untuk registri pribadi Amazon ECR dalam akun Anda AWS

Contoh berikut membuat aturan pull through cache untuk registri pribadi Amazon ECR untuk Lintas wilayah dalam akun yang sama AWS . Ini menentukan awalan repositori`ecr`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `ecr/`*`upstream-repository-name`*

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr \
     --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
     --region us-east-2
```

Untuk registri pribadi Amazon ECR dari akun lain AWS

Contoh berikut membuat aturan pull through cache untuk registri pribadi Amazon ECR untuk Lintas wilayah dalam akun yang sama AWS . Ini menentukan awalan repositori`ecr`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema

penamaan. `ecr/`*`upstream-repository-name`* Anda harus menentukan Nama Sumber Daya Amazon (ARN) lengkap dari peran IAM dengan izin yang dibuat. Membuat aturan cache tarik melalui di Amazon ECR

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix ecr \
 --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
 --custom-role-arn arn:aws:iam::aws_account_id:role/example-role \
 --region us-east-2
```

Untuk Registri Chainguard

Contoh berikut membuat aturan pull through cache untuk Registry Chainguard. Ini menentukan awalan repositori`chainguard`, yang menghasilkan setiap repositori yang dibuat menggunakan aturan pull through cache untuk memiliki skema penamaan. `chainguard/`*`upstream-repository-name`* Anda harus menentukan Nama Sumber Daya Amazon (ARN) lengkap dari rahasia yang berisi kredenal Registri Chainguard Anda.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix chainguard \
     --upstream-registry-url cgr.dev \
     --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-2
```

Langkah selanjutnya

Setelah Anda membuat aturan cache pull through, berikut ini adalah langkah-langkah selanjutnya:

- Buat template pembuatan repositori. Template pembuatan repositori memberi Anda kontrol untuk menentukan pengaturan yang akan digunakan untuk repositori baru yang dibuat oleh Amazon ECR atas nama Anda selama tindakan pull through cache. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.
- Validasi tarik Anda melalui aturan cache. Saat memvalidasi aturan pull through cache, Amazon ECR membuat koneksi jaringan dengan registri upstream, memverifikasi bahwa ia dapat mengakses rahasia Secrets Manager yang berisi kredensyal untuk registri upstream, dan otentikasi itu berhasil. Untuk informasi selengkapnya, lihat Memvalidasi aturan pull through cache di Amazon ECR.

- Mulai gunakan aturan cache pull through Anda. Untuk informasi selengkapnya, lihat Menarik gambar dengan aturan cache pull through di Amazon ECR.

Memvalidasi aturan pull through cache di Amazon ECR

Setelah Anda membuat aturan cache pull through, untuk registri upstream yang memerlukan otentikasi, Anda dapat memvalidasi bahwa aturan tersebut berfungsi dengan baik. Saat memvalidasi aturan pull through cache, Amazon ECR membuat koneksi jaringan dengan registri upstream, memverifikasi bahwa ia dapat mengakses rahasia Secrets Manager yang berisi kredensyal untuk registri upstream, dan memverifikasi bahwa otentikasi berhasil.

Sebelum Anda mulai bekerja dengan aturan cache pull through Anda, verifikasi bahwa Anda memiliki izin IAM yang tepat. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk menyinkronkan registri upstream dengan registri pribadi Amazon ECR.

Untuk memvalidasi aturan pull through cache ()Konsol Manajemen AWS

Langkah-langkah berikut menunjukkan cara memvalidasi aturan cache tarik melalui menggunakan konsol Amazon ECR.

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah yang berisi aturan pull through cache untuk memvalidasi.
3. Di panel navigasi, pilih Registri pribadi, Tarik cache.
4. Pada halaman konfigurasi Pull through cache, pilih aturan pull through cache untuk memvalidasi. Kemudian, gunakan menu tarik-turun Tindakan dan pilih Lihat detail.
5. Pada halaman detail aturan cache pull through, gunakan menu tarik-turun Tindakan dan pilih Verifikasi otentikasi. Amazon ECR akan menampilkan spanduk dengan hasilnya.
6. Ulangi langkah-langkah ini untuk setiap pull through aturan cache yang ingin Anda validasi.

Untuk memvalidasi aturan pull through cache ()AWS CLI

AWS CLI Perintah validate-pull-through-cache-rule digunakan untuk memvalidasi aturan cache pull through untuk registri pribadi Amazon ECR. Contoh berikut menggunakan awalan `ecr-public` namespace. Ganti nilai itu dengan nilai awalan untuk aturan pull through cache untuk memvalidasi.

```
aws ecr validate-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
```

```
    --region us-east-2
```

Dalam tanggapan, `isValid` parameter menunjukkan apakah validasi berhasil atau tidak. Jika`true`, Amazon ECR dapat mencapai registri hulu dan otentikasi berhasil. Jika`false`, ada masalah dan validasi gagal. `failure`Parameter menunjukkan penyebabnya.

Menarik gambar dengan aturan cache pull through di Amazon ECR

Contoh berikut menunjukkan sintaks perintah untuk digunakan saat menarik gambar menggunakan aturan pull through cache. Jika Anda menerima kesalahan saat menarik gambar upstream menggunakan aturan cache pull through, lihat Memecahkan masalah penarikan melalui cache di Amazon ECR kesalahan yang paling umum dan cara mengatasinya.

Sebelum Anda mulai bekerja dengan aturan cache pull through Anda, verifikasi bahwa Anda memiliki izin IAM yang tepat. Untuk informasi selengkapnya, lihat Izin IAM diperlukan untuk menyinkronkan registri upstream dengan registri pribadi Amazon ECR.

> **Note**
>
> Contoh berikut menggunakan nilai namespace repositori Amazon ECR default yang digunakan. Konsol Manajemen AWS Pastikan Anda menggunakan URI repositori pribadi Amazon ECR yang telah dikonfigurasi.

Untuk Amazon ECR Publik

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/
image_name:tag
```

Registri kontainer Kubernetes

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/kubernetes/repository_name/
image_name:tag
```

dermaga

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/quay/repository_name/
image_name:tag
```

Hub Docker

Untuk gambar resmi Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

> **Note**
>
> Untuk gambar resmi Docker Hub, `/library` awalan harus disertakan. Untuk semua repositori Docker Hub lainnya, Anda harus menghilangkan awalan. `/library`

Untuk semua gambar Docker Hub lainnya:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

GitHub Registri Kontainer

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/
image_name:tag
```

Registri Kontainer Microsoft Azure

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/
image_name:tag
```

GitLab Registri Kontainer

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

Registri Chainguard

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/chainguard/repository_name/
image_name:tag
```

Menyimpan kredensi repositori upstream Anda secara rahasia AWS Secrets Manager

Saat membuat aturan cache pull through untuk repositori upstream yang memerlukan otentikasi, Anda harus menyimpan kredensialnya dalam rahasia Secrets Manager. Mungkin ada biaya untuk menggunakan rahasia Secrets Manager. Untuk informasi selengkapnya, lihat harga AWS Secrets Manager.

Prosedur berikut memandu Anda melalui cara membuat rahasia Secrets Manager untuk setiap repositori upstream yang didukung. Anda dapat secara opsional menggunakan alur kerja aturan create pull through cache di konsol Amazon ECR untuk membuat rahasia alih-alih membuat rahasia menggunakan konsol Secrets Manager. Untuk informasi selengkapnya, lihat Membuat aturan cache tarik melalui di Amazon ECR.

Docker Hub

Untuk membuat rahasia Secrets Manager untuk kredensial Docker Hub Anda ()Konsol Manajemen AWS

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Pilih Simpan rahasia baru.
3. Pada halaman Pilih jenis rahasia, lakukan hal berikut.
    a. Untuk Tipe rahasia, pilih Tipe rahasia lainnya.
    b. Pada pasangan kunci/nilai, buat dua baris untuk kredensyal Docker Hub Anda. Anda dapat menyimpan hingga 65536 byte secara rahasia.
        i. Untuk key/value pasangan pertama, tentukan `username` sebagai kunci dan nama pengguna Docker Hub Anda sebagai nilainya.
        ii. Untuk key/value pasangan kedua, tentukan `accessToken` sebagai kunci dan token akses Docker Hub Anda sebagai nilainya. Untuk informasi selengkapnya tentang membuat token akses Docker Hub, lihat Membuat dan mengelola token akses dalam dokumentasi Docker.
    c. Untuk kunci Enkripsi, pertahankan AWS KMS key nilai aws/secretsmanager default lalu pilih Berikutnya. Tidak ada biaya untuk menggunakan kunci ini. Untuk informasi selengkapnya, lihat Enkripsi dan dekripsi rahasia di Secrets Manager di AWS Secrets Manager Panduan Pengguna.

> **Important**
>
> Anda harus menggunakan kunci `aws/secretsmanager` enkripsi default untuk mengenkripsi rahasia Anda. Amazon ECR tidak mendukung penggunaan kunci terkelola pelanggan (CMK) untuk ini.

4. Pada halaman Konfigurasi rahasia, lakukan hal berikut.

   a. Masukkan nama Rahasia deskriptif dan Deskripsi. Nama rahasia harus berisi 1-512 karakter Unicode dan diawali dengan. `ecr-pullthroughcache/`

   > **Important**
   >
   > Amazon ECR Konsol Manajemen AWS hanya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan.

   b. (Opsional) Di bagian Tag, tambahkan tag ke rahasia Anda. Untuk menandai strategi, lihat Menandai rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Jangan menyimpan informasi sensitif dalam tag karena tidak dienkripsi.

   c. (Opsional) Di Izin sumber daya, untuk menambahkan kebijakan sumber daya ke rahasia Anda, pilih Edit izin. Untuk informasi selengkapnya, lihat Melampirkan kebijakan izin ke rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna.

   d. (Opsional) Dalam rahasia Replikasi, untuk mereplikasi rahasia Anda ke yang lain Wilayah AWS, pilih Replikasi rahasia. Anda dapat mereplikasi rahasia Anda sekarang atau kembali dan mereplikasi nanti. Untuk informasi selengkapnya, lihat Mereplikasi rahasia ke Wilayah lain di Panduan AWS Secrets Manager Pengguna.

   e. Pilih Berikutnya.

5. (Opsional) Pada halaman Konfigurasi rotasi, Anda dapat mengaktifkan rotasi otomatis. Anda juga dapat mematikan rotasi untuk saat ini dan kemudian menyalakannya nanti. Untuk informasi selengkapnya, lihat Memutar rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Pilih Berikutnya.

6. Pada halaman Ulasan, tinjau detail rahasia Anda, lalu pilih Store.

   Secrets Manager kembali ke daftar rahasia. Jika rahasia baru Anda tidak muncul, pilih tombol refresh.

GitHub Container Registry

Untuk membuat rahasia Secrets Manager untuk kredensi Registry GitHub Container Anda ()Konsol Manajemen AWS

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Pilih Simpan rahasia baru.
3. Pada halaman Pilih jenis rahasia, lakukan hal berikut.
   a. Untuk Tipe rahasia, pilih Tipe rahasia lainnya.
   b. Dalam pasangan kunci/nilai, buat dua baris untuk kredensyal Anda GitHub . Anda dapat menyimpan hingga 65536 byte secara rahasia.
      i. Untuk key/value pasangan pertama, tentukan `username` sebagai kunci dan GitHub nama pengguna Anda sebagai nilainya.
      ii. Untuk key/value pasangan kedua, tentukan `accessToken` sebagai kunci dan token GitHub akses Anda sebagai nilainya. Untuk informasi selengkapnya tentang membuat token GitHub akses, lihat Mengelola token akses pribadi Anda dalam GitHub dokumentasi.
   c. Untuk kunci Enkripsi, pertahankan AWS KMS key nilai aws/secretsmanager default lalu pilih Berikutnya. Tidak ada biaya untuk menggunakan kunci ini. Untuk informasi selengkapnya, lihat Enkripsi dan dekripsi rahasia di Secrets Manager di AWS Secrets Manager Panduan Pengguna.

   > **Important**
   >
   > Anda harus menggunakan kunci `aws/secretsmanager` enkripsi default untuk mengenkripsi rahasia Anda. Amazon ECR tidak mendukung penggunaan kunci terkelola pelanggan (CMK) untuk ini.

4. Pada halaman Konfigurasi rahasia, lakukan hal berikut:
   a. Masukkan nama Rahasia deskriptif dan Deskripsi. Nama rahasia harus berisi 1-512 karakter Unicode dan diawali dengan. `ecr-pullthroughcache/`

> **Important**
>
> Amazon ECR Konsol Manajemen AWS hanya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan.

b. (Opsional) Di bagian Tag, tambahkan tag ke rahasia Anda. Untuk menandai strategi, lihat Menandai rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Jangan menyimpan informasi sensitif dalam tag karena tidak dienkripsi.

c. (Opsional) Di Izin sumber daya, untuk menambahkan kebijakan sumber daya ke rahasia Anda, pilih Edit izin. Untuk informasi selengkapnya, lihat Melampirkan kebijakan izin ke rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna.

d. (Opsional) Dalam rahasia Replikasi, untuk mereplikasi rahasia Anda ke yang lain Wilayah AWS, pilih Replikasi rahasia. Anda dapat mereplikasi rahasia Anda sekarang atau kembali dan mereplikasi nanti. Untuk informasi selengkapnya, lihat Mereplikasi rahasia ke Wilayah lain di Panduan AWS Secrets Manager Pengguna.

e. Pilih Berikutnya.

5. (Opsional) Pada halaman Konfigurasi rotasi, Anda dapat mengaktifkan rotasi otomatis. Anda juga dapat mematikan rotasi untuk saat ini dan kemudian menyalakannya nanti. Untuk informasi selengkapnya, lihat Memutar rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Pilih Berikutnya.

6. Pada halaman Ulasan, tinjau detail rahasia Anda, lalu pilih Store.

   Secrets Manager kembali ke daftar rahasia. Jika rahasia baru Anda tidak muncul, pilih tombol refresh.

Microsoft Azure Container Registry

Untuk membuat rahasia Secrets Manager untuk kredensyal Microsoft Azure Container Registry Anda ()Konsol Manajemen AWS

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Pilih Simpan rahasia baru.
3. Pada halaman Pilih jenis rahasia, lakukan hal berikut.

   a. Untuk Tipe rahasia, pilih Tipe rahasia lainnya.

b. Dalam pasangan kunci/nilai, buat dua baris untuk kredensyal Microsoft Azure Anda. Anda dapat menyimpan hingga 65536 byte secara rahasia.

   i. Untuk key/value pasangan pertama, tentukan `username` sebagai kunci dan nama pengguna Microsoft Azure Container Registry Anda sebagai nilainya.

   ii. Untuk key/value pasangan kedua, tentukan `accessToken` sebagai kunci dan token akses Microsoft Azure Container Registry Anda sebagai nilainya. Untuk informasi selengkapnya tentang cara membuat token akses Microsoft Azure, lihat Membuat token - portal di dokumentasi Microsoft Azure.

c. Untuk kunci Enkripsi, pertahankan AWS KMS key nilai aws/secretsmanager default lalu pilih Berikutnya. Tidak ada biaya untuk menggunakan kunci ini. Untuk informasi selengkapnya, lihat Enkripsi dan dekripsi rahasia di Secrets Manager di AWS Secrets Manager Panduan Pengguna.

> **Important**
>
> Anda harus menggunakan kunci `aws/secretsmanager` enkripsi default untuk mengenkripsi rahasia Anda. Amazon ECR tidak mendukung penggunaan kunci terkelola pelanggan (CMK) untuk ini.

4. Pada halaman Konfigurasi rahasia, lakukan hal berikut:

   a. Masukkan nama Rahasia deskriptif dan Deskripsi. Nama rahasia harus berisi 1-512 karakter Unicode dan diawali dengan. `ecr-pullthroughcache/`

   > **Important**
   >
   > Amazon ECR Konsol Manajemen AWS hanya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan.

   b. (Opsional) Di bagian Tag, tambahkan tag ke rahasia Anda. Untuk menandai strategi, lihat Menandai rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Jangan menyimpan informasi sensitif dalam tag karena tidak dienkripsi.

   c. (Opsional) Di Izin sumber daya, untuk menambahkan kebijakan sumber daya ke rahasia Anda, pilih Edit izin. Untuk informasi selengkapnya, lihat Melampirkan kebijakan izin ke rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna.

d. (Opsional) Dalam rahasia Replikasi, untuk mereplikasi rahasia Anda ke yang lain Wilayah AWS, pilih Replikasi rahasia. Anda dapat mereplikasi rahasia Anda sekarang atau kembali dan mereplikasi nanti. Untuk informasi selengkapnya, lihat Mereplikasi rahasia ke Wilayah lain di Panduan AWS Secrets Manager Pengguna.

e. Pilih Berikutnya.

5. (Opsional) Pada halaman Konfigurasi rotasi, Anda dapat mengaktifkan rotasi otomatis. Anda juga dapat mematikan rotasi untuk saat ini dan kemudian menyalakannya nanti. Untuk informasi selengkapnya, lihat Memutar rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Pilih Berikutnya.

6. Pada halaman Ulasan, tinjau detail rahasia Anda, lalu pilih Store.

   Secrets Manager kembali ke daftar rahasia. Jika rahasia baru Anda tidak muncul, pilih tombol refresh.

GitLab Container Registry

Untuk membuat rahasia Secrets Manager untuk kredensi Registry GitLab Container Anda ()Konsol Manajemen AWS

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Pilih Simpan rahasia baru.
3. Pada halaman Pilih jenis rahasia, lakukan hal berikut.

   a. Untuk Tipe rahasia, pilih Tipe rahasia lainnya.

   b. Dalam pasangan kunci/nilai, buat dua baris untuk kredensyal Anda GitLab . Anda dapat menyimpan hingga 65536 byte secara rahasia.

      i. Untuk key/value pasangan pertama, tentukan `username` sebagai kunci dan nama pengguna GitLab Container Registry Anda sebagai nilainya.

      ii. Untuk key/value pasangan kedua, tentukan `accessToken` sebagai kunci dan token akses GitLab Container Registry Anda sebagai nilainya. Untuk informasi selengkapnya tentang membuat token akses GitLab Container Registry, lihat Token akses pribadi, Token akses grup, atau Token akses Proyek, dalam GitLab dokumentasi.

   c. Untuk kunci Enkripsi, pertahankan AWS KMS key nilai aws/secretsmanager default lalu pilih Berikutnya. Tidak ada biaya untuk menggunakan kunci ini. Untuk informasi

selengkapnya, lihat Enkripsi dan dekripsi rahasia di Secrets Manager di AWS Secrets Manager Panduan Pengguna.

> **Important**
>
> Anda harus menggunakan kunci `aws/secretsmanager` enkripsi default untuk mengenkripsi rahasia Anda. Amazon ECR tidak mendukung penggunaan kunci terkelola pelanggan (CMK) untuk ini.

4. Pada halaman Konfigurasi rahasia, lakukan hal berikut:

   a. Masukkan nama Rahasia deskriptif dan Deskripsi. Nama rahasia harus berisi 1-512 karakter Unicode dan diawali dengan. `ecr-pullthroughcache/`

   > **Important**
   >
   > Amazon ECR Konsol Manajemen AWS hanya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan.

   b. (Opsional) Di bagian Tag, tambahkan tag ke rahasia Anda. Untuk menandai strategi, lihat Menandai rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Jangan menyimpan informasi sensitif dalam tag karena tidak dienkripsi.

   c. (Opsional) Di Izin sumber daya, untuk menambahkan kebijakan sumber daya ke rahasia Anda, pilih Edit izin. Untuk informasi selengkapnya, lihat Melampirkan kebijakan izin ke rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna.

   d. (Opsional) Dalam rahasia Replikasi, untuk mereplikasi rahasia Anda ke yang lain Wilayah AWS, pilih Replikasi rahasia. Anda dapat mereplikasi rahasia Anda sekarang atau kembali dan mereplikasi nanti. Untuk informasi selengkapnya, lihat Mereplikasi rahasia ke Wilayah lain di Panduan AWS Secrets Manager Pengguna.

   e. Pilih Berikutnya.

5. (Opsional) Pada halaman Konfigurasi rotasi, Anda dapat mengaktifkan rotasi otomatis. Anda juga dapat mematikan rotasi untuk saat ini dan kemudian menyalakannya nanti. Untuk informasi selengkapnya, lihat Memutar rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Pilih Berikutnya.

6. Pada halaman Ulasan, tinjau detail rahasia Anda, lalu pilih Store.

Secrets Manager kembali ke daftar rahasia. Jika rahasia baru Anda tidak muncul, pilih tombol refresh.

Chainguard Registry

Untuk membuat rahasia Secrets Manager untuk kredensi Chainguard Anda ()Konsol Manajemen AWS

1. Buka konsol Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Pilih Simpan rahasia baru.
3. Pada halaman Pilih jenis rahasia, lakukan hal berikut.
   a. Untuk Tipe rahasia, pilih Tipe rahasia lainnya.
   b. Dalam pasangan kunci/nilai, buat dua baris untuk kredensyal Chainguard Anda. Anda dapat menyimpan hingga 65536 byte secara rahasia.
      i. Untuk key/value pasangan pertama, tentukan `username` sebagai kunci dan nama pengguna Registry Chainguard Anda sebagai nilainya.
      ii. Untuk key/value pasangan kedua, tentukan `accessToken` sebagai kunci dan token akses Chainguard Registry Anda sebagai nilainya. Untuk informasi selengkapnya tentang membuat token tarik Chainguard Registry, lihat Mengautentikasi dengan Token Tarik di dokumentasi Chainguard.
   c. Untuk kunci Enkripsi, pertahankan AWS KMS key nilai aws/secretsmanager default lalu pilih Berikutnya. Tidak ada biaya untuk menggunakan kunci ini. Untuk informasi selengkapnya, lihat Enkripsi dan dekripsi rahasia di Secrets Manager di AWS Secrets Manager Panduan Pengguna.

   > **Important**
   >
   > Anda harus menggunakan kunci `aws/secretsmanager` enkripsi default untuk mengenkripsi rahasia Anda. Amazon ECR tidak mendukung penggunaan kunci terkelola pelanggan (CMK) untuk ini.

4. Pada halaman Konfigurasi rahasia, lakukan hal berikut:
   a. Masukkan nama Rahasia deskriptif dan Deskripsi. Nama rahasia harus berisi 1-512 karakter Unicode dan diawali dengan. `ecr-pullthroughcache/`

> **Important**
>
> Amazon ECR Konsol Manajemen AWS hanya menampilkan rahasia Secrets Manager dengan nama menggunakan `ecr-pullthroughcache/` awalan.

b. (Opsional) Di bagian Tag, tambahkan tag ke rahasia Anda. Untuk menandai strategi, lihat Menandai rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Jangan menyimpan informasi sensitif dalam tag karena tidak dienkripsi.

c. (Opsional) Di Izin sumber daya, untuk menambahkan kebijakan sumber daya ke rahasia Anda, pilih Edit izin. Untuk informasi selengkapnya, lihat Melampirkan kebijakan izin ke rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna.

d. (Opsional) Dalam rahasia Replikasi, untuk mereplikasi rahasia Anda ke yang lain Wilayah AWS, pilih Replikasi rahasia. Anda dapat mereplikasi rahasia Anda sekarang atau kembali dan mereplikasi nanti. Untuk informasi selengkapnya, lihat Mereplikasi rahasia ke Wilayah lain di Panduan AWS Secrets Manager Pengguna.

e. Pilih Berikutnya.

5. (Opsional) Pada halaman Konfigurasi rotasi, Anda dapat mengaktifkan rotasi otomatis. Anda juga dapat mematikan rotasi untuk saat ini dan kemudian menyalakannya nanti. Untuk informasi selengkapnya, lihat Memutar rahasia Secrets Manager di Panduan AWS Secrets Manager Pengguna. Pilih Berikutnya.

6. Pada halaman Ulasan, tinjau detail rahasia Anda, lalu pilih Store.

   Secrets Manager kembali ke daftar rahasia. Jika rahasia baru Anda tidak muncul, pilih tombol refresh.

Menyesuaikan awalan repositori untuk ECR ke ECR menarik cache

Tarik aturan cache mendukung awalan repositori ecr dan awalan repositori hulu. Awalan repositori ecr adalah awalan namespace repositori di registri cache Amazon ECR yang terkait dengan aturan. Semua repositori yang menggunakan awalan ini menjadi tarik melalui repositori berkemampuan cache untuk registri hulu yang ditentukan dalam aturan. Misalnya, awalan `prod` berlaku untuk semua repositori yang dimulai dengan. `prod/` Untuk menerapkan template ke semua repositori di registri Anda yang tidak memiliki aturan pull through cache terkait, gunakan `ROOT` sebagai awalan.

> **Important**
>
> Selalu ada asumsi `/` yang diterapkan pada akhir prefiks. Jika Anda menentukan `ecr-public` sebagai awalan, Amazon ECR memperlakukannya sebagai. `ecr-public/`

Awalan repositori upstream cocok dengan nama repositori upstream. Secara default, ini diatur keROOT, yang memungkinkan pencocokan dengan repositori upstream apa pun. Anda dapat menyetel awalan repositori upstream hanya ketika awalan repositori Amazon ECR memiliki nilai non. `ROOT`

Tabel berikut menunjukkan pemetaan antara nama repositori cache dan nama repositori upstream berdasarkan konfigurasi awalan mereka dalam aturan pull through cache.

Ruang nama cache	Namespace hulu	Hubungan pemetaan (repositori cache → repositori hulu)

ecr-publik	ROOT (default)	`ecr-public/my-app/image1` → `my-app/image1`
		
		`ecr-public/my-app/image2` → `my-app/image2`

AKAR	AKAR	`my-app/image1` → `my-app/image1`
tim-a	tim-a	`team-a/myapp/image1` → `team-a/myapp/image1`
aplikasi saya	aplikasi hulu	`my-app/image1` → `upstream-app/image1`

Memecahkan masalah penarikan melalui cache di Amazon ECR

Saat menarik gambar upstream menggunakan aturan pull through cache, berikut ini adalah kesalahan paling umum yang mungkin Anda terima.

Repositori tidak ada

Kesalahan yang menunjukkan bahwa repositori tidak ada paling sering disebabkan oleh repositori yang tidak ada di registri pribadi Amazon ECR Anda atau `ecr:CreateRepository` izin yang tidak diberikan kepada prinsipal IAM yang menarik gambar hulu. Untuk mengatasi kesalahan ini, Anda harus memverifikasi bahwa URI repositori dalam perintah tarik Anda sudah benar, izin IAM yang diperlukan diberikan kepada prinsipal IAM yang menarik gambar upstream, atau bahwa repositori untuk gambar upstream yang akan didorong dibuat di registri pribadi Amazon ECR Anda sebelum melakukan penarikan gambar upstream. Untuk informasi selengkapnya tentang izin IAM yang diperlukan, lihat Izin IAM diperlukan untuk menyinkronkan registri upstream dengan registri pribadi Amazon ECR

Berikut ini adalah contoh kesalahan ini.

```
Error response from daemon: repository 111122223333.dkr.ecr.us-east-1.amazonaws.com/
ecr-public/amazonlinux/amazonlinux not found: name unknown: The repository with
 name 'ecr-public/amazonlinux/amazonlinux' does not exist in the registry with id
 '111122223333'
```

Gambar yang diminta tidak ditemukan

Kesalahan yang menunjukkan bahwa gambar tidak dapat ditemukan paling sering disebabkan oleh gambar yang tidak ada di registri hulu atau `ecr:BatchImportUpstreamImage` izin yang tidak diberikan kepada kepala sekolah IAM yang menarik gambar hulu tetapi repositori sudah dibuat di registri pribadi Amazon ECR Anda. Untuk mengatasi kesalahan ini, Anda harus memverifikasi gambar hulu dan nama tag gambar sudah benar dan bahwa itu ada dan izin IAM yang diperlukan diberikan kepada prinsipal IAM yang menarik gambar hulu. Untuk informasi selengkapnya tentang izin IAM yang diperlukan, lihat. Izin IAM diperlukan untuk menyinkronkan registri upstream dengan registri pribadi Amazon ECR

Berikut ini adalah contoh kesalahan ini.

```
Error response from daemon: manifest for 111122223333.dkr.ecr.us-
east-1.amazonaws.com/ecr-public/amazonlinux/amazonlinux:latest not found: manifest
 unknown: Requested image not found
```

403 Terlarang saat menarik dari repositori Docker Hub

Saat menarik dari repositori Docker Hub yang ditandai sebagai Gambar Resmi Docker, Anda harus menyertakan URI yang `/library/` Anda gunakan. Misalnya,

aws_account_id.dkr.ecr.*region*.amazonaws.com/docker-hub/**library/***image_name:tag*. Jika Anda menghilangkan gambar Resmi Docker Hub /library/ untuk, 403 Forbidden kesalahan akan dikembalikan saat Anda mencoba menarik gambar menggunakan aturan cache tarik. Untuk informasi selengkapnya, lihat Menarik gambar dengan aturan cache pull through di Amazon ECR.

Berikut ini adalah contoh kesalahan ini.

```
Error response from daemon: failed to resolve reference "111122223333.dkr.ecr.us-
west-2.amazonaws.com/docker-hub/amazonlinux:2023": pulling from host
 111122223333.dkr.ecr.us-west-2.amazonaws.com failed with status code [manifests
 2023]: 403 Forbidden
```

Replikasi image privat di Amazon ECR

Anda dapat mengonfigurasi registri pribadi Amazon ECR Anda untuk mendukung replikasi repositori Anda. Amazon ECR mendukung replikasi lintas wilayah dan lintas akun. Agar replikasi lintas akun terjadi, akun tujuan harus mengonfigurasi kebijakan izin registri agar replikasi dari registri sumber terjadi. Untuk informasi selengkapnya, lihat Izin registri pribadi di Amazon ECR.

Topik

Persyaratan kebijakan replikasi lintas akun

Agar replikasi ECR lintas akun berfungsi dengan baik, Anda harus memahami akun mana yang memerlukan kebijakan mana yang dikonfigurasi. Bagian ini menjelaskan persyaratan kebijakan untuk akun sumber dan tujuan.

Ikhtisar konfigurasi kebijakan

Replikasi ECR lintas akun hanya memerlukan konfigurasi kebijakan pada akun tujuan. Akun sumber tidak memerlukan repositori khusus atau kebijakan registri.

- Akun Sumber: Konfigurasikan aturan replikasi dalam pengaturan registri. Tidak ada kebijakan tambahan yang diperlukan pada repositori sumber.
- Akun Tujuan: Konfigurasikan kebijakan izin registri untuk memungkinkan akun sumber mereplikasi gambar.

Persyaratan kebijakan registri tujuan

Akun tujuan harus mengonfigurasi kebijakan izin registri yang memberikan izin akun sumber untuk melakukan tindakan berikut:

- `ecr:ReplicateImage`- Memungkinkan akun sumber untuk mereplikasi gambar ke registri tujuan

- `ecr:CreateRepository`- Memungkinkan ECR untuk secara otomatis membuat repositori di registri tujuan jika belum ada

> **Important**
>
> Jika Anda tidak memberikan `ecr:CreateRepository` izin, Anda harus secara manual membuat repositori dengan nama yang sama di akun tujuan sebelum replikasi dapat berhasil.

Contoh kebijakan registri tujuan:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountReplication",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "ecr:ReplicateImage",
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Persyaratan akun sumber

Akun sumber hanya perlu:

- Konfigurasikan aturan replikasi dalam pengaturan registri untuk menentukan akun tujuan dan wilayah
- Pastikan replikasi konfigurasi utama IAM memiliki izin ECR yang diperlukan

Tidak ada kebijakan tambahan yang diperlukan pada repositori sumber. Repositori sumber tidak memerlukan kebijakan repositori yang memberikan izin replikasi.

Kesalahpahaman umum

Berikut ini adalah kesalahpahaman umum tentang kebijakan replikasi lintas akun ECR:

- Kesalahpahaman: Repositori sumber memerlukan kebijakan yang memungkinkan akun tujuan untuk mereplikasi gambar.

  Realitas: Repositori sumber tidak memerlukan kebijakan khusus untuk replikasi.
- Kesalahpahaman: Akun sumber dan tujuan memerlukan kebijakan registri.

  Realitas: Hanya akun tujuan yang memerlukan kebijakan izin registri.
- Kesalahpahaman: Kebijakan repositori dan kebijakan registri adalah hal yang sama.

  Realitas: Kebijakan repositori mengontrol akses ke repositori individual, sementara kebijakan registri mengontrol operasi tingkat registri seperti replikasi.

Memecahkan masalah kegagalan replikasi

Jika replikasi lintas akun gagal, periksa hal berikut:

- Verifikasi bahwa akun tujuan memiliki kebijakan izin registri yang dikonfigurasi
- Pastikan kebijakan registri mencakup keduanya `ecr:ReplicateImage` dan `ecr:CreateRepository` tindakan
- Konfirmasikan ID akun sumber ditentukan dengan benar dalam kebijakan registri tujuan
- Periksa apakah repositori tujuan ada (jika tidak `ecr:CreateRepository` diberikan)
- Meninjau CloudTrail log untuk panggilan gagal `CreateRepository` atau `ReplicateImage` API

Pertimbangan untuk replikasi citra pribadi

Hal-hal berikut ini harus dipertimbangkan ketika menggunakan replikasi citra pribadi.

- Hanya konten repositori yang didorong atau dikembalikan ke repositori setelah replikasi dikonfigurasi direplikasi. Konten apa pun yang sudah ada sebelumnya dalam repositori tidak

direplikasi. Jika gambar dipulihkan setelah replikasi dihidupkan, itu akan direplikasi. Jika dipulihkan sebelum replikasi dihidupkan, itu tidak akan direplikasi.

- Nama repositori akan tetap sama di seluruh Wilayah dan akun saat replikasi telah terjadi. Amazon ECR tidak mendukung perubahan nama repositori selama replikasi.
- Pertama kali Anda mengonfigurasi registri pribadi Anda untuk replikasi, Amazon ECR membuat peran IAM terkait layanan atas nama Anda. Peran IAM terkait layanan memberikan layanan replikasi Amazon ECR izin yang diperlukan untuk membuat repositori dan mereplikasi gambar di registri Anda. Untuk informasi selengkapnya, lihat Menggunakan Peran Terkait Layanan untuk Amazon ECR.
- Agar replikasi lintas akun terjadi, tujuan registri pribadi harus memberikan izin untuk mengizinkan registri sumber mereplikasi gambarnya. Ini dilakukan dengan menetapkan kebijakan izin registri pribadi. Untuk informasi selengkapnya, lihat Izin registri pribadi di Amazon ECR.
- Jika kebijakan izin untuk registri pribadi diubah untuk menghapus izin, setiap replikasi yang sedang berlangsung yang sebelumnya diberikan dapat diselesaikan.
- Agar replikasi lintas wilayah terjadi, akun sumber dan tujuan harus ikut serta dalam Wilayah sebelum tindakan replikasi apa pun yang terjadi di dalam atau ke Wilayah tersebut. Untuk informasi selengkapnya, lihat Mengelola Wilayah AWS di Referensi Umum Amazon Web Services.
- Replikasi Lintas Wilayah tidak didukung antar AWS partisi. Misalnya, repositori di tidak `us-west-2` dapat direplikasi ke. `cn-north-1` Untuk informasi selengkapnya tentang AWS partisi, lihat format ARN di Referensi Umum AWS .
- Konfigurasi replikasi untuk registri pribadi dapat berisi hingga 25 tujuan unik di semua aturan, dengan maksimal 10 aturan total. Setiap aturan dapat berisi hingga 100 filter. Ini memungkinkan untuk menentukan aturan terpisah untuk repositori yang berisi gambar yang digunakan untuk produksi dan pengujian, misalnya.
- Konfigurasi replikasi mendukung pemfilteran repositori mana dalam registri pribadi direplikasi dengan menentukan awalan repositori. Sebagai contoh, lihat Contoh: Mengonfigurasi replikasi lintas wilayah menggunakan filter repositori.
- Tindakan replikasi hanya terjadi sekali per push gambar atau pemulihan gambar. Sebagai contoh, jika Anda mengonfigurasi replikasi lintas wilayah dari `us-west-2` ke `us-east-1` dan dari `us-east-1` ke `us-east-2`, citra yang didorong ke `us-west-2` bereplikasi hanya ke `us-east-1`, ia tidak mereplikasi lagi ke `us-east-2`. Perilaku ini berlaku untuk replikasi lintas wilayah maupun lintas akun.
- Mayoritas gambar mereplikasi dalam waktu kurang dari 30 menit, tetapi dalam kasus yang jarang terjadi replikasi mungkin memakan waktu lebih lama.

- Replikasi registri tidak melakukan tindakan penghapusan atau tindakan arsip apa pun. Gambar dan repositori yang direplikasi dapat dihapus atau diarsipkan ketika tidak lagi digunakan.
- Jika gambar yang akan direplikasi diarsipkan di tujuan, maka itu akan dipulihkan di tujuan.
- Ketika gambar diarsipkan di wilayah sumber, itu tidak akan diarsipkan di wilayah tujuan yang ditentukan oleh konfigurasi replikasi.
- Kebijakan repositori, termasuk kebijakan IAM, dan kebijakan siklus hidup tidak direplikasi dan tidak memiliki efek apa pun selain pada repositori yang ditetapkan untuknya.
- Pengaturan repositori tidak direplikasi secara default, Anda dapat mereplikasi pengaturan repositori menggunakan templat pembuatan repositori. Pengaturan ini mencakup mutabilitas tag, enkripsi, izin repositori, dan kebijakan siklus hidup. Untuk informasi selengkapnya tentang template pembuatan repositori, lihat. Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi
- Jika ketetapan tanda diaktifkan pada repositori dan sebuah citra direplikasi yang menggunakan tanda yang sama sebagai citra yang ada, citra direplikasi tetapi tidak akan berisi duplikasi tandanya. Hal ini dapat mengakibatkan citra yang tidak ditandai.
- Saat mereplikasi gambar, jika pemasangan gumpalan telah dikonfigurasi, ECR akan memeriksa untuk memastikan lapisan apa pun dari repositori sumber sudah ada di registri tujuan. Jika ada lapisan yang sudah ada di registri tujuan, ECR akan memasang lapisan tersebut.

> **Note**
>
> Jika registri sumber berbeda dari registri tujuannya, pemasangan gumpalan perlu diaktifkan untuk kedua pendaftar untuk ECR untuk memasang lapisan yang direplikasi.

Contoh replikasi gambar pribadi untuk Amazon ECR

Contoh berikut menunjukkan kasus penggunaan umum untuk replikasi gambar pribadi. Jika Anda mengonfigurasi replikasi dengan menggunakan AWS CLI, Anda dapat menggunakan contoh JSON sebagai titik awal saat Anda membuat file JSON Anda. Jika Anda mengonfigurasi replikasi dengan menggunakan Konsol Manajemen AWS, Anda akan melihat JSON serupa ketika Anda meninjau aturan replikasi Anda di halaman Tinjauan dan kirim.

Contoh: mengonfigurasi replikasi lintas wilayah untuk satu Wilayah tujuan

Berikut ini adalah contoh untuk mengonfigurasi replikasi lintas wilayah dalam satu registri. Contoh ini mengasumsikan bahwa ID akun Anda adalah `111122223333` dan bahwa Anda menentukan konfigurasi replikasi ini di wilayah selain `us-west-2`.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Contoh: Mengonfigurasi replikasi lintas wilayah menggunakan filter repositori

Berikut ini menunjukkan contoh untuk mengkonfigurasi replikasi lintas wilayah untuk repositori yang cocok dengan nilai nama awalan. Contoh ini mengasumsikan ID akun Anda `111122223333` dan bahwa Anda menentukan konfigurasi replikasi ini di Wilayah selain `us-west-1` dan memiliki repositori dengan awalan. `prod`

```
{
 "rules": [{
  "destinations": [{
   "region": "us-west-1",
   "registryId": "111122223333"
  }],
  "repositoryFilters": [{
   "filter": "prod",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

Contoh: mengonfigurasi replikasi lintas wilayah untuk beberapa Wilayah tujuan

Berikut ini adalah contoh untuk mengonfigurasi replikasi lintas wilayah dalam satu registri. Contoh ini mengasumsikan ID akun Anda `111122223333` dan Anda menentukan konfigurasi replikasi ini di Wilayah selain atau. `us-west-1` `us-west-2`

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-1",
                    "registryId": "111122223333"
                },
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Contoh: mengonfigurasi replikasi lintas akun

Berikut ini adalah contoh untuk mengonfigurasi replikasi lintas akun untuk registri Anda. Contoh ini mengonfigurasi replikasi ke akun `444455556666` dan ke Wilayah `us-west-2`.

> **Important**
>
> Agar replikasi lintas akun terjadi, akun tujuan harus mengonfigurasi kebijakan izin registri untuk memungkinkan replikasi terjadi. Untuk informasi selengkapnya, lihat Izin registri pribadi di Amazon ECR.

```
{
    "rules": [
        {
```

```
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "444455556666"
                }
            ]
        }
    ]
}
```

Contoh: Menentukan beberapa aturan dalam konfigurasi

Berikut ini menunjukkan contoh untuk mengkonfigurasi beberapa aturan replikasi untuk registri Anda. Contoh ini mengonfigurasi replikasi untuk *111122223333* akun dengan satu aturan yang mereplikasi repositori dengan awalan `prod` ke `us-west-2` Wilayah dan repositori dengan awalan ke Wilayah. `test` `us-east-2` Konfigurasi replikasi dapat berisi hingga 10 aturan, dengan setiap aturan menentukan hingga 25 tujuan.

```
{
 "rules": [{
   "destinations": [{
    "region": "us-west-2",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "prod",
    "filterType": "PREFIX_MATCH"
   }]
  },
  {
   "destinations": [{
    "region": "us-east-2",
    "registryId": "111122223333"
   }],
   "repositoryFilters": [{
    "filter": "test",
    "filterType": "PREFIX_MATCH"
   }]
  }
 ]
}
```

Contoh: Menghapus semua pengaturan replikasi

Berikut ini menunjukkan contoh untuk menghapus semua pengaturan replikasi dari registri Anda. Untuk menghapus pengaturan replikasi, Anda harus mengkonfigurasi array aturan kosong.

```
{
    "rules": []
}
```

> **Important**
>
> Menghapus pengaturan replikasi tidak menghapus repositori atau gambar yang direplikasi sebelumnya. Anda harus menghapus konten yang direplikasi secara manual jika tidak lagi diperlukan.

Mengkonfigurasi replikasi gambar pribadi di Amazon ECR

Konfigurasikan replikasi per Wilayah untuk registri pribadi Anda. Anda dapat mengonfigurasi replikasi lintas wilayah atau replikasi lintas akun.

Untuk contoh bagaimana replikasi umumnya digunakan, lihatContoh replikasi gambar pribadi untuk Amazon ECR.

Untuk mengonfigurasi pengaturan replikasi registri (Konsol Manajemen AWS)

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah untuk mengonfigurasi pengaturan replikasi registri.
3. Di panel navigasi, pilih Registri pribadi.
4. Pada halaman registri pribadi, pilih Pengaturan dan kemudian pilih Edit di bawah konfigurasi Replikasi.
5. Pada halaman Replikasi, pilih Tambahkan aturan replikasi.
6. Pada halaman Jenis tujuan, pilih apakah akan mengaktifkan replikasi lintas wilayah, replikasi lintas akun, atau keduanya, lalu pilih Berikutnya.
7. Jika replikasi lintas wilayah diaktifkan, maka untuk Konfigurasi wilayah tujuan, pilih satu atau beberapa wilayah Tujuan, lalu pilih Berikutnya.

8. Jika replikasi lintas akun diaktifkan, maka untuk replikasi Cross-account, pilih pengaturan replikasi lintas akun untuk registri. Untuk akun Tujuan, masukkan ID akun untuk akun tujuan dan satu atau beberapa wilayah Tujuan untuk direplikasi. Pilih Akun tujuan+untuk mengonfigurasi akun tambahan sebagai tujuan replikasi.

   > **Important**
   >
   > Agar replikasi lintas akun terjadi, akun tujuan harus mengonfigurasi kebijakan izin registri untuk memungkinkan replikasi terjadi. Untuk informasi selengkapnya, lihat Izin registri pribadi di Amazon ECR.

9. (Opsional) Pada halaman Tambahkan filter, tentukan satu atau beberapa filter untuk aturan replikasi lalu pilih Tambah. Ulangi langkah ini untuk setiap filter yang ingin Anda kaitkan dengan tindakan replikasi. Filter harus ditentukan sebagai awalan nama repositori. Jika tidak ada filter yang ditambahkan, isi semua repositori direplikasi. Pilih Berikutnya setelah semua filter telah ditambahkan.
10. Pada halaman Tinjau dan kirim, tinjau konfigurasi aturan replikasi lalu pilih Aturan Kirim.

Untuk mengonfigurasi pengaturan replikasi registri (AWS CLI)

1. Buat file JSON yang berisi aturan replikasi untuk menentukan registri Anda. Konfigurasi replikasi dapat berisi hingga 10 aturan, dengan hingga 25 tujuan unik di semua aturan dan 100 filter per setiap aturan. Untuk mengonfigurasi replikasi lintas wilayah dalam akun Anda sendiri, Anda menentukan ID akun Anda sendiri. Untuk contoh lainnya, lihat Contoh replikasi gambar pribadi untuk Amazon ECR.

```
{
 "rules": [{
  "destinations": [{
   "region": "destination_region",
   "registryId": "destination_accountId"
  }],
  "repositoryFilters": [{
   "filter": "repository_prefix_name",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

2. Membuat konfigurasi replikasi untuk registri Anda.

```
aws ecr put-replication-configuration \
     --replication-configuration file://replication-settings.json \
     --region us-west-2
```

3. Konfirmasikan pengaturan registri Anda.

```
aws ecr describe-registry \
     --region us-west-2
```

Menghapus pengaturan replikasi gambar pribadi di Amazon ECR

Untuk menghapus atau menonaktifkan pengaturan replikasi untuk registri pribadi Anda, Anda perlu mengkonfigurasi konfigurasi replikasi kosong. Tidak ada perintah penghapusan khusus di AWS CLI.

Untuk menghapus pengaturan replikasi registri ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah untuk menghapus pengaturan replikasi registri Anda.
3. Di panel navigasi, pilih Registri pribadi.
4. Pada halaman registri pribadi, pilih Pengaturan dan kemudian pilih Edit di bawah konfigurasi Replikasi.
5. Hapus semua aturan replikasi yang ada dengan memilih opsi hapus untuk setiap aturan.
6. Pilih Simpan untuk menerapkan konfigurasi replikasi kosong.

Untuk menghapus pengaturan replikasi registri ()AWS CLI

1. Buat file JSON dengan array aturan kosong untuk menghapus semua pengaturan replikasi.

```
{
    "rules": []
}
```

2. Terapkan konfigurasi replikasi kosong ke registri Anda.

```
aws ecr put-replication-configuration \
```

```
    --replication-configuration file://empty-replication-settings.json \
    --region us-west-2
```

3. Konfirmasikan bahwa pengaturan replikasi telah dihapus.

```
aws ecr describe-registry \
    --region us-west-2
```

Output harus menunjukkan kosong `replicationConfiguration` tanpa aturan.

> **Important**
>
> Menghapus pengaturan replikasi tidak menghapus repositori atau gambar yang direplikasi sebelumnya. Anda harus menghapus konten yang direplikasi secara manual jika tidak lagi diperlukan.

Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi

Gunakan templat pembuatan repositori Amazon ECR untuk menentukan pengaturan untuk repositori yang dibuat oleh Amazon ECR atas nama Anda. Pengaturan dalam template pembuatan repositori hanya diterapkan selama pembuatan repositori dan tidak berpengaruh pada repositori atau repositori yang ada yang dibuat menggunakan metode lain. Saat ini, template pembuatan repositori dapat digunakan untuk menerapkan pengaturan selama pembuatan repositori untuk fitur-fitur ini:

- Tarik melalui cache
- Buat saat push
- Replikasi

Cara kerja template pembuatan repositori

Ada kalanya Amazon ECR perlu membuat repositori pribadi baru atas nama Anda. Contoh:

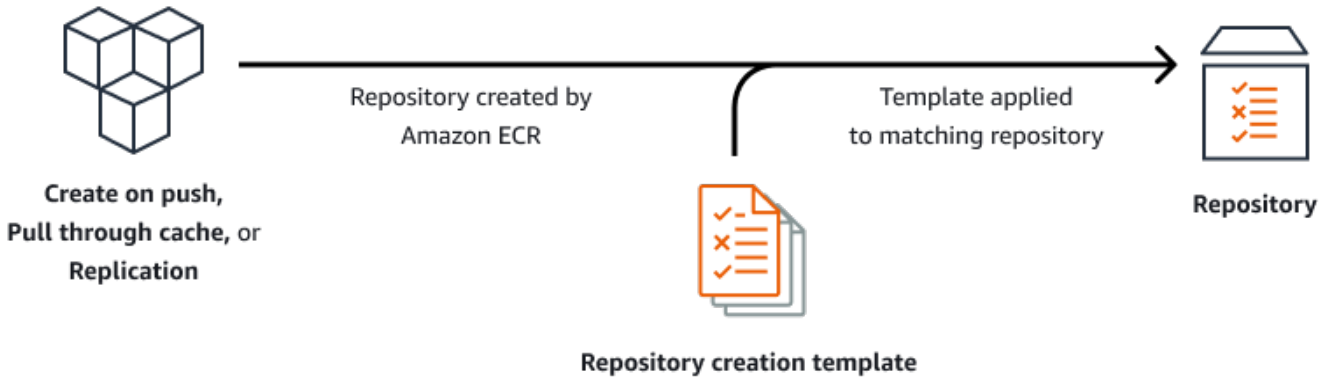
- Pertama kali Anda menggunakan aturan pull through cache untuk mengambil konten repositori upstream dan menyimpannya di registri pribadi Amazon ECR Anda.
- Saat Anda mendorong gambar ke repositori yang belum ada.
- Saat Anda ingin Amazon ECR mereplikasi repositori ke wilayah atau akun lain.

Jika tidak ada template pembuatan repositori yang cocok dengan aturan cache pull through atau repositori yang direplikasi, Amazon ECR menggunakan pengaturan default untuk repositori baru. Pengaturan default ini termasuk mematikan kekekalan tag, menggunakan `AES-256` enkripsi, dan tidak menerapkan kebijakan repositori atau siklus hidup apa pun.

Jika tidak ada template pembuatan repositori yang cocok dengan repositori target untuk push gambar, Amazon ECR tidak akan membuat repositori dengan pengaturan default.

Menggunakan template pembuatan repositori memberi Anda kemampuan untuk menentukan pengaturan Amazon ECR berlaku untuk repositori baru yang dibuat melalui pull through cache, create on push, dan tindakan replikasi. Anda dapat menentukan kekekalan tag, konfigurasi enkripsi, izin repositori, kebijakan siklus hidup, dan tag sumber daya untuk repositori baru.

Diagram berikut menunjukkan alur kerja yang Amazon ECR gunakan saat template pembuatan repositori digunakan.



Berikut ini menjelaskan setiap parameter dalam template pembuatan repositori secara rinci.

Awalan

Awalan adalah awalan namespace repositori untuk dikaitkan dengan template. Semua repositori yang dibuat menggunakan awalan ini akan memiliki pengaturan yang diterapkan yang ditentukan dalam template ini. Misalnya, awalan `prod` akan berlaku untuk semua repositori yang dimulai dengan. `prod/` Demikian pula, awalan `prod/team` akan berlaku untuk semua repositori yang dimulai dengan. `prod/team/` Dalam registri yang berisi dua template, jika satu template memiliki awalan “prod” dan yang lainnya memiliki awalan "prod/team", the template with the prefix "prod/team" will be applied to all repositories whose names start with "prod/team/”.

Untuk menerapkan template ke semua repositori di registri Anda yang tidak memiliki template pembuatan terkait, Anda dapat menggunakan `ROOT` sebagai awalan.

> **Important**
>
> Selalu ada asumsi `/` diterapkan pada akhir awalan. Jika Anda menentukan `ecr-public` sebagai awalan, Amazon ECR memperlakukannya sebagai. `ecr-public/` Saat menggunakan aturan pull through cache, awalan repositori yang Anda tentukan selama pembuatan aturan adalah apa yang harus Anda tentukan sebagai awalan template pembuatan repositori Anda juga.

Deskripsi

Deskripsi template ini opsional dan digunakan untuk menggambarkan tujuan template pembuatan repositori.

Diterapkan Untuk

Pengaturan yang diterapkan untuk menentukan repositori Amazon ECR-dibuat mana yang akan dibuat dengan template ini. Nilai yang valid adalah `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH`, dan `REPLICATION`. Misalnya, pertama kali Anda menggunakan aturan pull through cache untuk mengambil konten repositori upstream dan menyimpannya di registri pribadi Amazon ECR Anda. Jika tidak ada template pembuatan repositori yang cocok dengan aturan cache pull through Anda, Amazon ECR menggunakan pengaturan default untuk repositori baru.

Peran pembuatan repositori

Peran pembuatan repositori adalah ARN peran IAM yang akan diasumsikan oleh Amazon ECR saat membuat dan mengonfigurasi repositori melalui templat pembuatan repositori. Peran ini harus disediakan saat menggunakan tag repositori and/or KMS di template, jika tidak, pembuatan repositori akan gagal.

Tanda ketetapan citra

Pengaturan mutabilitas tag yang akan digunakan untuk repositori yang dibuat menggunakan template. Jika parameter ini dihilangkan, pengaturan default MUTABLE akan digunakan yang akan memungkinkan tag gambar untuk ditimpa. Ini adalah pengaturan yang disarankan untuk digunakan untuk template yang digunakan untuk repositori yang dibuat dengan tindakan pull through cache. Ini memastikan bahwa Amazon ECR dapat memperbarui gambar yang di-cache saat tag sama.

Jika IMMUTABLE ditentukan, semua tag gambar dalam repositori akan kekal yang akan mencegahnya ditimpa.

Konfigurasi enkripsi

> **Important**
>
> Enkripsi sisi server dua lapis dengan AWS KMS (DSSE-KMS) hanya tersedia di Wilayah. AWS GovCloud (US)

Konfigurasi enkripsi yang digunakan untuk repositori yang dibuat menggunakan template.

Jika Anda menggunakan jenis enkripsi KMS, isi repositori akan dienkripsi menggunakan enkripsi sisi server dengan kunci yang disimpan di. AWS Key Management Service AWS KMS Saat Anda menggunakan AWS KMS untuk mengenkripsi data, Anda dapat menggunakan AWS KMS kunci AWS terkelola default untuk Amazon ECR, atau menentukan AWS KMS kunci Anda sendiri, yang sudah Anda buat. Anda selanjutnya dapat memilih untuk menggunakan enkripsi Single-layer atau Dual-layer dengan. AWS KMS Untuk informasi selengkapnya, lihat Enkripsi saat diam. Jika Anda menggunakan jenis enkripsi KMS dan menggunakannya dengan replikasi lintas wilayah, Anda mungkin memerlukan izin tambahan. Untuk informasi selengkapnya, lihat Membuat kebijakan kunci KMS untuk replikasi.

Jika Anda menggunakan jenis enkripsi AES256, Amazon ECR menggunakan enkripsi sisi server dengan kunci enkripsi terkelola Amazon S3 yang mengenkripsi citra di repositori menggunakan algoritme enkripsi AES-256. Untuk informasi selengkapnya, lihat Melindungi data menggunakan enkripsi sisi server dengan kunci enkripsi terkelola Amazon S3 (SSE-S3) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Izin repositori

Kebijakan repositori untuk diterapkan ke repositori yang dibuat menggunakan template. Kebijakan repositori menggunakan izin berbasis sumber daya untuk mengontrol akses ke repositori. Izin berbasis sumber daya memungkinkan Anda menentukan pengguna atau peran IAM mana yang memiliki akses ke repositori dan tindakan apa yang dapat mereka lakukan di situ. Secara default, hanya AWS akun yang membuat repositori yang memiliki akses ke repositori. Anda dapat menerapkan dokumen kebijakan untuk memberikan atau menolak izin tambahan ke repositori Anda. Untuk informasi selengkapnya, lihat Kebijakan repositori pribadi di Amazon ECR.

Kebijakan siklus hidup repositori

Kebijakan siklus hidup yang akan digunakan untuk repositori yang dibuat menggunakan template. Kebijakan siklus hidup memberikan kontrol lebih besar atas pengelolaan siklus hidup gambar dalam repositori pribadi. Sebuah kebijakan siklus hidup berisi satu atau lebih aturan, di mana setiap aturan mendefinisikan sebuah tindakan untuk Amazon ECR. Ini menyediakan cara untuk mengotomatiskan pembersihan gambar kontainer Anda dengan kedaluwarsa gambar berdasarkan usia atau hitungan. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.

Tag sumber daya

Tag sumber daya adalah metadata untuk diterapkan ke repositori untuk membantu Anda mengkategorikan dan mengaturnya. Setiap tag terdiri atas sebuah kunci dan sebuah nilai

opsional, dan Anda menentukan sendiri keduanya. Izin ini perlu diterapkan pada kebijakan registri tujuan jika Anda menggunakan templat pembuatan repositori dengan replikasi lintas wilayah.

Membuat template pembuatan repositori di Amazon ECR

Anda dapat membuat template pembuatan repositori untuk menentukan pengaturan yang akan digunakan untuk repositori yang dibuat oleh Amazon ECR atas nama Anda selama pull through cache, create on push, atau tindakan replikasi. Setelah template pembuatan repositori dibuat, semua repositori baru yang dibuat akan memiliki pengaturan yang diterapkan. Ini tidak berpengaruh pada repositori yang dibuat sebelumnya.

Saat menyiapkan repositori dengan templat, Anda memiliki opsi untuk menentukan kunci KMS dan tag sumber daya. Jika Anda bermaksud menggunakan kunci KMS, tag sumber daya, atau kombinasi keduanya dalam satu atau beberapa templat, Anda perlu:

- Buat kebijakan khusus untuk templat pembuatan repositori.
- Buat peran IAM untuk template pembuatan repositori.

Setelah dikonfigurasi, Anda dapat melampirkan peran khusus ke templat tertentu di registri Anda.

Izin IAM untuk membuat template pembuatan repositori

Izin berikut diperlukan untuk prinsipal IAM untuk mengelola template pembuatan repositori. Izin ini harus diberikan menggunakan kebijakan IAM berbasis identitas.

- `ecr:CreateRepositoryCreationTemplate`— Memberikan izin untuk membuat template pembuatan repositori.
- `ecr:UpdateRepositoryCreationTemplate`— Memberikan izin untuk memperbarui template pembuatan repositori.
- `ecr:DescribeRepositoryCreationTemplates`— Memberikan izin untuk membuat daftar template pembuatan repositori dalam registri.
- `ecr:DeleteRepositoryCreationTemplate`— Memberikan izin untuk menghapus template pembuatan repositori.
- `ecr:CreateRepository`— Memberikan izin untuk membuat repositori Amazon ECR.

- `ecr:PutLifecyclePolicy`— Memberikan izin untuk membuat kebijakan siklus hidup dan menerapkannya ke repositori. Izin ini hanya diperlukan jika template pembuatan repositori menyertakan kebijakan siklus hidup.
- `ecr:SetRepositoryPolicy`— Memberikan izin untuk membuat kebijakan izin untuk repositori. Izin ini hanya diperlukan jika template pembuatan repositori menyertakan kebijakan repositori.
- `iam:PassRole`Memberikan izin untuk mengizinkan entitas untuk meneruskan peran ke layanan atau aplikasi. Izin ini diperlukan untuk layanan dan aplikasi yang perlu mengambil peran untuk melakukan tindakan atas nama Anda.

Buat kebijakan khusus untuk templat pembuatan repositori

Anda dapat menggunakan Konsol Manajemen AWS untuk menentukan kebijakan yang selanjutnya akan dikaitkan dengan peran IAM. Peran IAM ini kemudian dapat digunakan sebagai peran pembuatan repositori saat mengonfigurasi template pembuatan repositori.

Konsol Manajemen AWS

Untuk menggunakan editor kebijakan JSON untuk membuat kebijakan kustom untuk template pembuatan repositori.

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi sebelah kiri, pilih Kebijakan.
3. Pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan kebijakan berikut di bidang JSON.

   JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "ecr:CreateRepository",
                    "ecr:ReplicateImage",
```

```
                    "ecr:TagResource"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "kms:CreateGrant",
                    "kms:RetireGrant",
                    "kms:DescribeKey"
                ],
                "Resource": "*"
            }
        ]
}
```

6. Selesaikan peringatan keamanan, kesalahan, atau peringatan umum yang dihasilkan selama validasi kebijakan, lalu pilih Berikutnya.
7. Setelah selesai menambahkan izin ke kebijakan, pilih Berikutnya.
8. Pada halaman Tinjau dan buat, ketik Nama Kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
9. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.
10. Buat peran untuk menetapkan kebijakan ini untuk templat pembuatan, lihatBuat peran IAM untuk template pembuatan repositori.

Buat peran IAM untuk template pembuatan repositori

Anda dapat menggunakan Konsol Manajemen AWS untuk membuat peran yang dapat digunakan oleh Amazon ECR saat Anda menentukan peran pembuatan repositori dalam templat pembuatan repositori yang menggunakan tag repositori atau KMS dalam templat.

Konsol Manajemen AWS

Untuk membuat peran.

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol, pilih Peran dan kemudian pilih Buat peran.

3. Pilih Jenis peran kebijakan kepercayaan khusus.
4. Di bagian Kebijakan kepercayaan khusus, tempel kebijakan kepercayaan khusus yang tercantum di bawah ini:

   JSON

```
{
"Version":"2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "ecr.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
    ]
}
```

5. Pilih Berikutnya.
6. Dari halaman Tambahkan izin, pilih kotak centang di samping kebijakan khusus yang Anda buat sebelumnya dari daftar kebijakan Izin, lalu pilih Berikutnya.
7. Untuk Nama peran, masukkan nama peran Anda. Nama peran harus unik di dalam diri Anda Akun AWS. Bila nama peran digunakan dalam kebijakan atau sebagai bagian dari ARN, nama peran tersebut peka huruf besar/kecil. Saat nama peran muncul ke pelanggan di konsol, seperti selama proses masuk, nama peran tidak peka huruf besar/kecil. Karena berbagai entitas mungkin mereferensikan peran, Anda tidak dapat mengedit nama peran setelah dibuat.
8. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk peran baru ini.
9. Tinjau peran, lalu pilih Buat peran.

Buat template pembuatan repositori

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk template Anda, Anda dapat melanjutkan untuk membuat template pembuatan repositori.

Konsol Manajemen AWS

Untuk membuat template pembuatan repositori ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah untuk membuat template pembuatan repositori di.
3. Di panel navigasi, pilih Registri pribadi, Template pembuatan repositori.
4. Pada halaman template pembuatan repositori, pilih Buat template.
5. Pada Langkah 1: Tentukan halaman template, untuk detail Template, pilih awalan khusus untuk menerapkan template ke awalan namespace repositori tertentu atau pilih Awalan apa pun di registri ECR Anda untuk menerapkan templat ke semua repositori yang tidak cocok dengan templat lain di Wilayah.

   a. Jika Anda memilih awalan tertentu, untuk Awalan tentukan awalan namespace repositori untuk menerapkan template ke. Selalu ada asumsi `/` diterapkan pada akhir awalan. Misalnya, awalan `prod` akan berlaku untuk semua repositori yang dimulai dengan. `prod/` Demikian pula, awalan `prod/team` akan berlaku untuk semua repositori yang dimulai dengan. `prod/team/`

   b. Jika Anda memilih awalan apa pun di registri ECR Anda, Awalan akan diatur ke. `ROOT`

6. Untuk Diterapkan, tentukan alur kerja Amazon ECR mana yang akan diterapkan template ini. Pilihannya adalah`PULL_THROUGH_CACHE`,`CREATE_ON_PUSH`, dan`REPLICATION`.
7. Untuk deskripsi Template, tentukan deskripsi opsional untuk template dan kemudian pilih Berikutnya.
8. Pada Langkah 2: Tambahkan halaman konfigurasi pembuatan repositori, tentukan konfigurasi pengaturan repositori untuk diterapkan ke repositori yang dibuat menggunakan templat.

   a. Untuk mutabilitas tag Gambar, pilih pengaturan mutabilitas tag yang akan digunakan. Untuk informasi selengkapnya, lihat Mencegah tag gambar ditimpa di Amazon ECR.

      - Mutable - Pilih opsi ini jika Anda ingin tag gambar ditimpa. Direkomendasikan untuk repositori yang menggunakan tindakan pull through cache untuk memastikan Amazon ECR dapat memperbarui gambar yang di-cache. Selain itu, untuk menonaktifkan pembaruan tag untuk beberapa tag yang dapat berubah, masukkan nama tag atau gunakan wildcard (*) untuk mencocokkan beberapa tag serupa di kotak teks pengecualian tag Mutable.

- Immutable - Pilih opsi ini jika Anda ingin mencegah tag gambar ditimpa, dan itu berlaku untuk semua tag dan pengecualian di repositori saat mendorong gambar dengan tag yang ada. Amazon ECR mengembalikan `ImageTagAlreadyExistsException` jika Anda mencoba mendorong gambar dengan tag yang ada. Selain itu, untuk mengaktifkan pembaruan tag untuk beberapa tag yang tidak dapat diubah, masukkan nama tag atau gunakan wildcard (*) untuk mencocokkan beberapa tag serupa di kotak teks pengecualian tag Immutable.

b. Untuk konfigurasi Enkripsi, pilih pengaturan enkripsi yang akan digunakan. Untuk informasi selengkapnya, lihat Enkripsi saat diam.

   Saat AES-256 dipilih, Amazon ECR menggunakan enkripsi sisi server dengan kunci enkripsi yang dikelola Amazon Simple Storage Service yang mengenkripsi data Anda saat istirahat menggunakan algoritme enkripsi AES-256 standar industri. Ini ditawarkan tanpa biaya tambahan.

   Saat AWS KMS dipilih, Amazon ECR menggunakan enkripsi sisi server dengan kunci yang disimpan di (). AWS Key Management Service AWS KMS Saat Anda menggunakan AWS KMS untuk mengenkripsi data, Anda dapat menggunakan kunci AWS terkelola default, yang dikelola oleh Amazon ECR, atau menentukan AWS KMS kunci Anda sendiri, yang disebut sebagai kunci yang dikelola pelanggan.

   > **Note**
   >
   > Pengaturan enkripsi untuk repositori tidak dapat diubah setelah repositori dibuat.

c. Untuk izin Repositori, tentukan kebijakan izin repositori yang akan diterapkan ke repositori yang dibuat menggunakan templat ini. Anda dapat menggunakan drop-down secara opsional untuk memilih salah satu sampel JSON untuk kasus penggunaan yang paling umum. Untuk informasi selengkapnya, lihat Kebijakan repositori pribadi di Amazon ECR.

d. Untuk kebijakan siklus hidup Repositori, tentukan kebijakan siklus hidup repositori yang akan diterapkan ke repositori yang dibuat menggunakan templat ini. Anda dapat menggunakan drop-down secara opsional untuk memilih salah satu sampel JSON untuk kasus penggunaan yang paling umum. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.

e. Untuk AWS tag Repositori, tentukan metadata, dalam bentuk pasangan kunci-nilai, untuk dikaitkan dengan repositori yang dibuat menggunakan templat ini dan kemudian pilih Berikutnya. Untuk informasi selengkapnya, lihat Menandai repositori pribadi di Amazon ECR.

f. Untuk peran pembuatan Repositori, pilih peran IAM khusus dari menu drop-down yang akan digunakan untuk templat pembuatan repositori saat menggunakan tag repositori atau KMS di templat (lihat detailnya). Kemudian pilih Berikutnya. Buat peran IAM untuk template pembuatan repositori

9. Pada Langkah 3: Tinjau dan buat halaman, tinjau pengaturan yang Anda tentukan untuk template pembuatan repositori. Pilih opsi Edit untuk membuat perubahan. Pilih Buat setelah Anda selesai.

AWS CLI

create-repository-creation-template AWS CLI Perintah ini digunakan untuk membuat template pembuatan repositori untuk registri pribadi Anda.

Untuk membuat template pembuatan repositori ()AWS CLI

1. Gunakan AWS CLI untuk menghasilkan kerangka untuk create-repository-creation-templateperintah.

```
aws ecr create-repository-creation-template \
    --generate-cli-skeleton
```

Output dari perintah menampilkan sintaks lengkap dari template pembuatan repositori.

```
{
"appliedFor":[""], // string array, but valid are PULL_THROUGH_CACHE,
 CREATE_ON_PUSH, and REPLICATION
"prefix": "string",
    "description": "string",
    "imageTagMutability":
 "MUTABLE"|"IMMUTABLE"|"IMMUTABLE_WITH_EXCLUSION"|"MUTABLE_WITH_EXCLUSION",
    "imageTagMutabilityExclusionFilters": [
        "filterType": "WILDCARD",
        "filter": "string"
    ],
    "repositoryPolicy": "string",
```

```
    "lifecyclePolicy": "string"
"encryptionConfiguration": {
"encryptionType": "AES256"|"KMS",
        "kmsKey": "string"
    },
    "resourceTags": [
        {
"Key": "string",
            "Value": "string"
        }
    ],
    "customRoleArn": "string", // must be a valid IAM Role ARN
}
```

2. Buat file bernama `repository-creation-template.json` dengan output dari langkah sebelumnya. Template ini menetapkan kunci enkripsi KMS untuk repositori apa pun yang dibuat di bawah `prod/*` dengan kebijakan repositori yang memungkinkan mendorong dan menarik gambar ke repositori future, menetapkan kebijakan siklus hidup yang akan kedaluwarsa gambar yang lebih lama dari dua minggu dan menetapkan peran khusus yang memungkinkan ECR mengakses kunci KMS dan menetapkan tag sumber daya ke repositori future. `examplekey`

```
{
"prefix": "prod",
    "description": "For repositories cached from my PTC rule and in my
 replication configuration that start with 'prod/'",
    "appliedFor": ["PULL_THROUGH_CACHE", "CREATE_ON_PUSH", "REPLICATION"],
    "encryptionConfiguration": {
"encryptionType": "KMS",
        "kmsKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-
cdef-example11111"
    },
    "resourceTags": [
        {
"Key": "examplekey",
            "Value": "examplevalue"
        }
    ],
    "imageTagMutability": "IMMUTABLE_WITH_EXCLUSION",
    "imageTagMutabilityExclusionFilters": [
      {
      "filterType": "WILDCARD",
```

```
      "filter": "latest"
      },
      {
      "filterType": "WILDCARD",
      "filter": "beta*"
      }
    ]
    "repositoryPolicy": "{\"Version\":\"2012-10-17\",       \"Statement\":
[{\"Sid\":\"AllowPushPullIAMRole\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":
\"arn:aws:iam::111122223333:user\/IAMusername\"},\"Action\":[\"ecr:BatchGetImage
\",\"ecr:BatchCheckLayerAvailability\",\"ecr:CompleteLayerUpload\",
\"ecr:GetDownloadUrlForLayer\",\"ecr:InitiateLayerUpload\",\"ecr:PutImage\",
\"ecr:UploadLayerPart\"]}]}",
    "lifecyclePolicy": "{\"rules\":[{\"rulePriority\":1,\"description\":\"Expire
 images older than 14 days\",\"selection\":{\"tagStatus\":\"any\",\"countType
\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},\"action\":
{\"type\":\"expire\"}}]}",
    "customRoleArn": "arn:aws:iam::111122223333:role/myRole"
}
```

3. Gunakan perintah berikut untuk membuat template pembuatan repositori. Pastikan Anda menentukan nama file konfigurasi yang dibuat pada langkah sebelumnya sebagai pengganti `repository-creation-template.json` dalam contoh berikut.

```
aws ecr create-repository-creation-template \
    --cli-input-json file://repository-creation-template.json
```

Memperbarui template pembuatan repositori

Anda dapat mengedit template pembuatan repositori jika Anda perlu mengubah konfigurasinya. Setelah template pembuatan repositori diedit, konfigurasi baru akan berlaku untuk template yang ada.

> **Important**
>
> Ini tidak berpengaruh pada repositori yang dibuat sebelumnya.

Konsol Manajemen AWS

Untuk mengedit template pembuatan repositori ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah tempat templat pembuatan repositori untuk diedit.
3. Di panel navigasi, pilih Registri pribadi, lalu pilih Pengaturan.
4. Dari bilah navigasi, pilih template pembuatan Repositori.
5. Pada halaman template pembuatan Repositori, pilih template pembuatan repositori untuk diedit.
6. Dari menu tarik-turun Tindakan, pilih Edit.
7. Tinjau dan perbarui pengaturan konfigurasi.
8. Pilih pembaruan untuk menerapkan konfigurasi template pembuatan baru.

AWS CLI

Untuk mengedit template pembuatan repositori ()AWS CLI

- Gunakan update-repository-creation-templateperintah untuk memperbarui template pembuatan repositori yang ada. Anda harus menentukan `prefix` nilai template. Contoh berikut memperbarui template pembuatan repositori dengan awalan. `prod`

```
aws ecr update-repository-creation-template \
     --prefix prod \
     --image-tag-mutability="IMMUTABLE_WITH_EXCLUSION" \
     --image-tag-mutability-exclusion-filters filterType=WILDCARD, filter=latest
```

Output dari perintah menampilkan detail template pembuatan repositori yang diperbarui.

Menghapus template pembuatan repositori di Amazon ECR

Anda dapat menghapus template pembuatan repositori jika Anda selesai menggunakannya. Setelah template pembuatan repositori dihapus, setiap repositori yang baru dibuat di bawah awalan terkait selama penarikan cache atau tindakan replikasi akan mewarisi pengaturan default, kecuali template lain yang cocok ditemukan, lihat. Cara kerja template pembuatan repositori

> **Important**
>
> Ini tidak berpengaruh pada repositori yang dibuat sebelumnya.

Konsol Manajemen AWS

Untuk menghapus template pembuatan repositori ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah tempat template pembuatan repositori untuk dihapus.
3. Di panel navigasi, pilih Registri pribadi, Template pembuatan repositori.
4. Pada halaman template pembuatan repositori, pilih template pembuatan repositori untuk dihapus.
5. Dari menu tarik-turun Tindakan, pilih Hapus.

AWS CLI

Untuk menghapus template pembuatan repositori ()AWS CLI

- Gunakan delete-repository-creation-templateperintah.html untuk menghapus template pembuatan repositori yang ada. Anda harus menentukan `prefix` nilai template. Contoh berikut menghapus template pembuatan repositori dengan awalan. `prod`

  ```
  aws ecr delete-repository-creation-template \
       --prefix prod
  ```

  Output dari perintah menampilkan detail template pembuatan repositori yang dihapus.

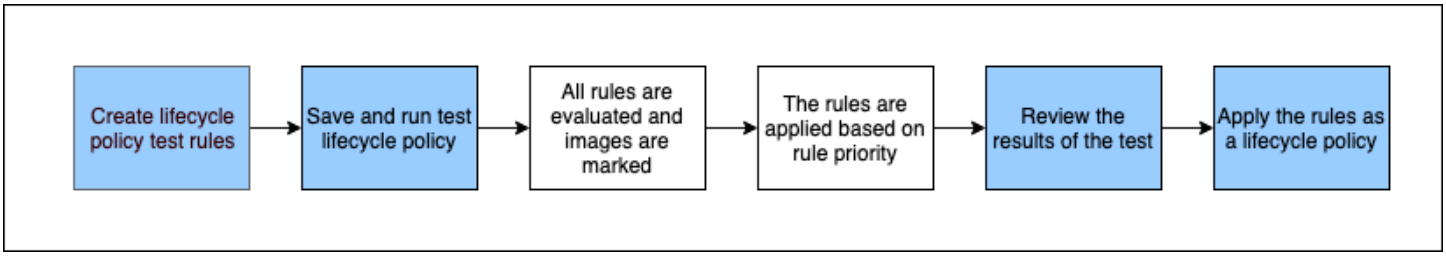
Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR

Kebijakan siklus hidup Amazon ECR memberikan kontrol lebih atas manajemen siklus hidup citra dalam repositori pribadi. Kebijakan siklus hidup berisi satu atau beberapa aturan, dan setiap aturan menentukan tindakan untuk Amazon ECR. Berdasarkan kriteria kedaluwarsa dalam kebijakan siklus hidup, gambar dapat diarsipkan atau kedaluwarsa berdasarkan kriteria yang ditentukan dalam kebijakan siklus hidup dalam waktu 24 jam. Saat Amazon ECR melakukan tindakan berdasarkan kebijakan siklus hidup, tindakan ini diambil sebagai peristiwa di. AWS CloudTrail Untuk informasi selengkapnya, lihat Mencatat tindakan Amazon ECR dengan AWS CloudTrail.

Cara kerja kebijakan siklus hidup

Kebijakan siklus hidup terdiri dari satu atau lebih aturan yang menentukan citra mana dalam repositori yang akan kedaluwarsa. Saat mempertimbangkan penggunaan kebijakan siklus hidup, penting untuk menggunakan pratinjau kebijakan siklus hidup untuk mengonfirmasi citra mana yang kebijakan siklus hidupnya kedaluwarsa sebelum menerapkannya ke repositori. Setelah kebijakan siklus hidup diterapkan ke repositori, Anda akan mengharapkan bahwa gambar menjadi kedaluwarsa dalam waktu 24 jam setelah memenuhi kriteria kedaluwarsa. Saat Amazon ECR melakukan tindakan berdasarkan kebijakan siklus hidup, tindakan ini akan diambil sebagai peristiwa di. AWS CloudTrail Untuk informasi selengkapnya, lihat Mencatat tindakan Amazon ECR dengan AWS CloudTrail.

Diagram berikut menunjukkan alur kerja kebijakan siklus hidup.



1. Buat satu atau lebih aturan pengujian.
2. Simpan aturan pengujian dan jalankan pratinjau.
3. Evaluator kebijakan siklus hidup membaca semua aturan dan menandai citra yang terpengaruh oleh setiap aturan.
4. Evaluator kebijakan siklus hidup kemudian menerapkan aturan, berdasarkan prioritas aturan, dan menampilkan gambar mana dalam repositori yang ditetapkan untuk kedaluwarsa atau

diarsipkan. Nomor prioritas aturan yang lebih rendah berarti prioritas yang lebih tinggi. Misalnya, aturan dengan prioritas 1 lebih diutamakan daripada aturan dengan prioritas 2.

5. Tinjau hasil tes, memastikan bahwa gambar yang ditandai kedaluwarsa atau diarsipkan adalah apa yang Anda inginkan.
6. Terapkan aturan pengujian sebagai kebijakan siklus hidup untuk repositori.
7. Setelah kebijakan siklus hidup dibuat, Anda akan mengharapkan bahwa gambar kedaluwarsa atau diarsipkan dalam waktu 24 jam setelah memenuhi kriteria kedaluwarsa.

Aturan evaluasi kebijakan siklus hidup

Evaluator kebijakan siklus hidup bertanggung jawab menguraikan JSON plaintext dari kebijakan siklus hidup, mengevaluasi semua aturan, kemudian menerapkan aturan tersebut berdasarkan prioritas aturan pada citra dalam repositori. Berikut ini penjelasan logika evaluator kebijakan siklus hidup secara lebih rinci. Sebagai contoh, lihat Contoh kebijakan siklus hidup di Amazon ECR.

- Ketika artefak referensi hadir dalam repositori, kebijakan siklus hidup Amazon ECR secara otomatis kedaluwarsa atau mengarsipkan artefak tersebut dalam waktu 24 jam setelah penghapusan atau pengarsipan gambar subjek.
- Semua aturan dievaluasi pada saat yang sama tanpa memperhatikan prioritas aturan. Setelah dievaluasi, semua aturan kemudian diterapkan berdasarkan prioritas aturan.
- Gambar kedaluwarsa atau diarsipkan dengan tepat satu atau nol aturan.
- Gambar yang cocok dengan persyaratan penandaan aturan tidak dapat kedaluwarsa atau diarsipkan oleh aturan dengan prioritas lebih rendah.
- Aturan tidak pernah dapat menandai gambar yang ditandai dengan aturan prioritas yang lebih tinggi, tetapi masih dapat mengidentifikasi mereka seolah-olah mereka belum kedaluwarsa atau diarsipkan.
- Himpunan semua aturan yang memilih kelas penyimpanan tertentu harus berisi satu set awalan yang unik.
- Hanya satu aturan memilih kelas penyimpanan tertentu yang diizinkan untuk memilih gambar yang tidak ditandai.
- Jika gambar direferensikan oleh daftar manifes, gambar tidak dapat kedaluwarsa atau diarsipkan tanpa daftar manifes dihapus atau diarsipkan terlebih dahulu.
- Kedaluwarsa selalu dipesan oleh `pushed_at_time` atau `transitioned_at_time` dan selalu kedaluwarsa gambar lama sebelum yang lebih baru. Jika gambar diarsipkan dan kemudian

dipulihkan pada titik mana pun di masa lalu, gambar `last_activated_at` digunakan sebagai gantinya. `pushed_at_time`

- Aturan kebijakan siklus hidup dapat menentukan salah satu `tagPatternList` atau`tagPrefixList`, tetapi tidak keduanya. Namun, kebijakan siklus hidup mungkin berisi beberapa aturan di mana aturan yang berbeda menggunakan daftar pola dan awalan. Gambar berhasil dicocokkan jika semua tag dalam `tagPrefixList` nilai `tagPatternList` atau dicocokkan dengan salah satu tag gambar.
- `tagPrefixList`Parameter `tagPatternList` atau hanya dapat digunakan jika `tagStatus` adatagged.
- Saat menggunakan`tagPatternList`, gambar berhasil dicocokkan jika cocok dengan filter wildcard. Misalnya, jika filter diterapkan, itu akan cocok dengan tag gambar yang namanya dimulai dengan `prod` sepertiprod,`prod1`, atau. `prod* production-team1` Demikian pula, jika filter `*prod*` diterapkan, itu akan cocok dengan tag gambar yang namanya berisi `prod` seperti `repo-production` atau. `prod-team`

  > **Important**
  >
  > Ada batas maksimum empat wildcard (*) per string. Misalnya, `["*test*1*2*3", "test*1*2*3*"]` valid tetapi `["test*1*2*3*4*5*6"]` tidak valid.

- Saat menggunakan`tagPrefixList`, gambar berhasil dicocokkan jika semua filter wildcard dalam `tagPrefixList` nilai dicocokkan dengan tag gambar mana pun.
- `countUnit`Parameter hanya digunakan jika `countType` adalah`sinceImagePushed`,`sinceImagePulled`, atau`sinceImageTransitioned`.
- Dengan`countType = imageCountMoreThan`, gambar diurutkan dari yang termuda ke terlama berdasarkan `pushed_at_time` dan kemudian semua gambar yang lebih besar dari jumlah yang ditentukan kedaluwarsa atau diarsipkan.
- Dengan`countType = sinceImagePushed`, semua gambar yang `pushed_at_time` lebih tua dari jumlah hari yang ditentukan berdasarkan `countNumber` kedaluwarsa atau diarsipkan.
- Dengan`countType = sinceImagePulled`, semua gambar yang `last_recorded_pulltime` lebih tua dari jumlah hari yang ditentukan berdasarkan `countNumber` diarsipkan. Jika gambar tidak pernah ditarik, gambar `pushed_at_time` digunakan sebagai pengganti gambar`last_recorded_pulltime`. Jika gambar diarsipkan dan kemudian dipulihkan pada titik mana pun di masa lalu, tetapi tidak pernah ditarik sejak gambar dipulihkan, gambar `last_activated_at` digunakan sebagai pengganti gambar. `last_recorded_pulltime`

- Dengan`countType = sinceImageTransitioned`, semua gambar yang diarsipkan yang `last_archived_at` lebih tua dari jumlah hari yang ditentukan berdasarkan `countNumber` kedaluwarsa.
- Kedaluwarsa selalu dipesan oleh `pushed_at_time` dan selalu kedaluwarsa gambar lama sebelum yang lebih baru.

Membuat pratinjau kebijakan siklus hidup di Amazon ECR

Anda dapat menggunakan pratinjau kebijakan siklus hidup untuk melihat dampak kebijakan siklus hidup pada repositori gambar sebelum Anda menerapkannya. Hal ini dianggap sebagai praktik terbaik untuk melakukan pratinjau sebelum menerapkan kebijakan siklus hidup pada repositori.

> **Note**
>
> Jika Anda menggunakan replikasi Amazon ECR untuk membuat salinan repositori di berbagai Wilayah atau akun, perhatikan bahwa kebijakan siklus hidup hanya dapat mengambil tindakan pada repositori di Wilayah tempat ia dibuat. Oleh karena itu, jika replikasi diaktifkan, Anda mungkin ingin membuat kebijakan siklus hidup di setiap Wilayah dan akun tempat Anda mereplikasi repositori.

Untuk membuat pratinjau kebijakan siklus hidup (Konsol Manajemen AWS)

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori untuk melakukan pratinjau kebijakan siklus hidup.
3. Di panel navigasi, di bawah Registri pribadi, pilih Repositori.
4. Pada halaman Private repositories, pilih repositori dan yang menggunakan drop down Actions untuk memilih kebijakan Siklus Hidup.
5. Pada halaman aturan kebijakan siklus hidup untuk repositori, pilih Edit aturan pengujian, Buat aturan.
6. Tentukan detail berikut untuk setiap aturan kebijakan siklus hidup pengujian.

    a. Untuk Prioritas aturan, ketikkan nomor untuk prioritas aturan. Prioritas aturan menentukan urutan aturan kebijakan siklus hidup yang diterapkan. Angka yang lebih rendah berarti

prioritas yang lebih tinggi. Misalnya, aturan dengan prioritas 1 lebih diutamakan daripada aturan dengan prioritas 2.

b. Untuk Deskripsi aturan, ketikkan deskripsi untuk aturan kebijakan siklus hidup.

c. Untuk status Gambar, pilih Tagged (pencocokan wildcard), Tagged (pencocokan awalan), Untagged, atau Any.

> **Important**
>
> Jika Anda menentukan beberapa tanda, hanya citra dengan semua tanda yang ditentukan yang dipilih.

d. Jika Anda memilih Tagged (pencocokan wildcard) untuk status Gambar, lalu untuk Menentukan tag untuk pencocokan wildcard, Anda dapat menentukan daftar tag gambar dengan wildcard (*) untuk mengambil tindakan dengan kebijakan siklus hidup Anda. Misalnya, jika gambar Anda ditandai sebagai`prod`,, `prod1``prod2`, dan seterusnya, Anda akan menentukan `prod*` untuk mengambil tindakan pada semuanya. Jika Anda menentukan beberapa tanda, hanya citra dengan semua tanda yang ditentukan yang dipilih.

> **Important**
>
> Ada batas maksimum empat wildcard (`*`) per string. Misalnya, `["*test*1*2*3", "test*1*2*3*"]` valid tetapi `["test*1*2*3*4*5*6"]` tidak valid.

e. Jika Anda memilih Tagged (pencocokan awalan) untuk status Gambar, lalu untuk Menentukan tag untuk pencocokan awalan, Anda dapat menentukan daftar tag gambar yang akan diambil tindakan dengan kebijakan siklus hidup Anda.

f. Untuk kriteria Pencocokan, pilih Hari sejak gambar dibuat, Hari sejak waktu tarik terakhir yang direkam, Hari sejak gambar diarsipkan, atau Jumlah gambar, lalu tentukan nilainya.

g. Untuk tindakan Aturan, pilih Kedaluwarsa atau Arsip.

h. Pilih Simpan.

7. Buat aturan kebijakan siklus hidup pengujian tambahan dengan mengulangi langkah 5—7.
8. Untuk menjalankan pratinjau kebijakan siklus hidup, pilih Simpan dan jalankan tes.
9. Di bawah citra sesuai untuk aturan pengujian siklus hidup, tinjau dampak pratinjau kebijakan siklus hidup Anda.
10. Jika Anda puas dengan hasil pratinjau, pilih Terapkan sebagai kebijakan siklus hidup untuk membuat kebijakan siklus hidup dengan aturan yang ditentukan. Anda harus mengharapkan

bahwa setelah menerapkan kebijakan siklus hidup, gambar yang terpengaruh akan kedaluwarsa atau diarsipkan dalam waktu 24 jam.

11. Jika Anda tidak puas dengan hasil pratinjau, Anda dapat menghapus satu atau beberapa aturan siklus hidup pengujian dan membuat satu atau beberapa aturan untuk menggantinya, lalu ulangi pengujian.

Membuat kebijakan siklus hidup untuk repositori di Amazon ECR

Gunakan kebijakan siklus hidup untuk membuat seperangkat aturan yang kedaluwarsa atau mengarsipkan gambar repositori yang tidak digunakan. Setelah membuat kebijakan siklus hidup, gambar yang terpengaruh akan kedaluwarsa atau diarsipkan dalam waktu 24 jam.

> **Note**
>
> Jika Anda menggunakan replikasi Amazon ECR untuk membuat salinan repositori di berbagai Wilayah atau akun, perhatikan bahwa kebijakan siklus hidup hanya dapat mengambil tindakan pada repositori di Wilayah tempat ia dibuat. Oleh karena itu, jika replikasi diaktifkan, Anda mungkin ingin membuat kebijakan siklus hidup di setiap Wilayah dan akun tempat Anda mereplikasi repositori.

Prasyarat

Praktik terbaik: Membuat pratinjau kebijakan siklus hidup untuk memverifikasi bahwa gambar kedaluwarsa atau diarsipkan oleh aturan kebijakan siklus hidup Anda adalah yang Anda inginkan. Untuk petunjuk, lihat Membuat pratinjau kebijakan siklus hidup di Amazon ECR.

Untuk membuat kebijakan siklus hidup (Konsol Manajemen AWS)

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/repositori.
2. Dari bilah navigasi, pilih Wilayah yang berisi repositori untuk membuat kebijakan siklus hidup.
3. Di panel navigasi, di bawah Registri pribadi, pilih Repositori.
4. Pada halaman Private repositories, pilih repositori dan yang menggunakan drop down Actions untuk memilih kebijakan Siklus Hidup.
5. Pada halaman aturan kebijakan siklus hidup untuk repositori, pilih Buat aturan.
6. Masukkan detail berikut untuk aturan kebijakan siklus hidup Anda.

a. Untuk Prioritas aturan, ketikkan nomor untuk prioritas aturan. Prioritas aturan menentukan urutan aturan kebijakan siklus hidup yang diterapkan. Nomor prioritas aturan yang lebih rendah berarti prioritas yang lebih tinggi. Misalnya, aturan dengan prioritas 1 lebih diutamakan daripada aturan dengan prioritas 2.

b. Untuk Deskripsi aturan, ketikkan deskripsi untuk aturan kebijakan siklus hidup.

c. Untuk status Gambar, pilih Tagged (pencocokan wildcard), Tagged (pencocokan awalan), Untagged, atau Any.

> **Important**
>
> Jika Anda menentukan beberapa tanda, hanya citra dengan semua tanda yang ditentukan yang dipilih.

d. Jika Anda memilih Tagged (pencocokan wildcard) untuk status Gambar, lalu untuk Menentukan tag untuk pencocokan wildcard, Anda dapat menentukan daftar tag gambar dengan wildcard (*) untuk mengambil tindakan dengan kebijakan siklus hidup Anda. Misalnya, jika gambar Anda ditandai sebagai`prod`,, `prod1``prod2`, dan seterusnya, Anda akan menentukan `prod*` untuk mengambil tindakan pada semuanya. Jika Anda menentukan beberapa tanda, hanya citra dengan semua tanda yang ditentukan yang dipilih.

> **Important**
>
> Ada batas maksimum empat wildcard (`*`) per string. Misalnya, `["*test*1*2*3", "test*1*2*3*"]` valid tetapi `["test*1*2*3*4*5*6"]` tidak valid.

e. Jika Anda memilih Tagged (pencocokan awalan) untuk status Gambar, lalu untuk Menentukan tag untuk pencocokan awalan, Anda dapat menentukan daftar tag gambar yang akan diambil tindakan dengan kebijakan siklus hidup Anda.

f. Untuk kriteria Pencocokan, pilih Hari sejak gambar dibuat, Hari sejak waktu tarik terakhir yang direkam, Hari sejak gambar diarsipkan, atau Jumlah gambar, lalu tentukan nilainya.

g. Untuk tindakan Aturan, pilih Kedaluwarsa atau Arsip.

h. Pilih Simpan.

7. Buat aturan kebijakan siklus hidup tambahan dengan mengulangi langkah 5-7.

Untuk membuat kebijakan siklus hidup (AWS CLI)

1. Dapatkan nama repositori untuk membuat kebijakan siklus hidup.

```
aws ecr describe-repositories
```

2. Buat file lokal bernama `policy.json` dengan isi kebijakan siklus hidup. Untuk contoh kebijakan siklus hidup, lihat Contoh kebijakan siklus hidup di Amazon ECR.
3. Buat kebijakan siklus hidup dengan menentukan nama repositori dan referensi file JSON kebijakan siklus hidup yang Anda buat.

```
aws ecr put-lifecycle-policy \
      --repository-name repository-name \
      --lifecycle-policy-text file://policy.json
```

Contoh kebijakan siklus hidup di Amazon ECR

Berikut ini adalah contoh kebijakan siklus hidup yang menunjukkan sintaks.

Untuk melihat informasi selengkapnya tentang properti kebijakan, lihatProperti kebijakan siklus hidup di Amazon ECR. Untuk petunjuk tentang membuat kebijakan siklus hidup menggunakan AWS CLI, lihat. Untuk membuat kebijakan siklus hidup (AWS CLI)

Templat kebijakan siklus hidup

Isi kebijakan siklus hidup Anda dievaluasi sebelum dikaitkan dengan repositori. Berikut ini adalah templat sintaks JSON untuk kebijakan siklus hidup.

```
{
        "rules": [
            {
                "rulePriority": integer,
                "description": "string",
                "selection": {
                    "tagStatus": "tagged"|"untagged"|"any",
                    "tagPatternList": list<string>,
                    "tagPrefixList": list<string>,
                    "storageClass": "standard"|"archive",
                    "countType":
 "imageCountMoreThan"|"sinceImagePushed"|"sinceImagePulled"|"sinceImageTransitioned",
```

```
                    "countUnit": "string",
                    "countNumber": integer
                },
                "action": {
                    "type": "expire"|"transition",
                    "targetStorageClass": "archive"
                }
            }
        ]
    }
```

Memfilter usia citra

Contoh berikut menunjukkan sintaks kebijakan siklus hidup untuk kebijakan yang kedaluwarsa gambar dengan tag yang dimulai `prod` dengan menggunakan `tagPatternList` dari `prod*` yang juga lebih lama dari hari. `14`

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Pemfilteran pada waktu terakhir yang ditarik

Contoh berikut menunjukkan sintaks kebijakan siklus hidup untuk kebijakan yang mentransisikan gambar ke penyimpanan arsip yang belum ditarik dalam beberapa hari. `90`

```
{
```

```
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive images not pulled in 90 days",
            "selection": {
                "tagStatus": "any",
                "countType": "sinceImagePulled",
                "countUnit": "days",
                "countNumber": 90
            },
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        }
    ]
}
```

> **Important**
>
> Jenis `sinceImagePulled` hitungan harus digunakan dengan `transition` tindakan. Itu tidak dapat digunakan dengan `expire` tindakan. Untuk menghapus gambar berdasarkan aktivitas tarik, pertama-tama transisi ke penyimpanan arsip menggunakan`sinceImagePulled`, lalu gunakan `sinceImageTransitioned` dengan `expire` tindakan untuk menghapusnya. Gambar harus disimpan dalam penyimpanan arsip minimal 90 hari sebelum penghapusan.

Pemfilteran pada waktu transisi arsip

Contoh berikut menunjukkan sintaks kebijakan siklus hidup untuk kebijakan yang kedaluwarsa gambar yang diarsipkan yang telah berada di penyimpanan arsip selama lebih dari beberapa hari. `365`

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images archived for more than 365 days",
            "selection": {
                "tagStatus": "any",
```

```
                "storageClass": "archive",
                "countType": "sinceImageTransitioned",
                "countUnit": "days",
                "countNumber": 365
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

> **Important**
>
> Jenis `sinceImageTransitioned` hitungan harus digunakan dengan `expire` tindakan dan kelas `archive` penyimpanan. Gambar harus disimpan dalam penyimpanan arsip minimal 90 hari sebelum penghapusan.

Memfilter jumlah citra

Contoh berikut menunjukkan sintaks kebijakan siklus hidup untuk kebijakan yang hanya menyimpan satu gambar yang tidak diberi tag dan kedaluwarsa semua gambar lainnya.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Memfilter beberapa aturan

Contoh berikut menggunakan beberapa aturan dalam kebijakan siklus hidup. Contoh repositori dan kebijakan siklus hidup diberikan bersama dengan penjelasan dari hasilnya.

Contoh A

Isi repositori:

- citra A, Taglist: ["beta-1", "prod-1"], Didorong: 10 hari yang lalu
- citra B, Taglist: ["beta-2", "prod-2"], Didorong: 9 hari yang lalu
- citra C, Taglist: ["beta-3"], Didorong: 8 hari yang lalu

Teks kebijakan siklus hidup:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
```

```
            }
        }
    ]
}
```

Logika kebijakan siklus hidup ini adalah:

- Aturan 1 mengidentifikasi citra yang ditandai dengan prefiks `prod`. Ini harus menandai citra, dimulai dengan yang tertua, sampai ada satu atau lebih sedikit citra yang tersisa yang sesuai. Ini menandai citra A untuk kedaluwarsa.
- Aturan 2 mengidentifikasi citra yang ditandai dengan prefiks `beta`. Ini harus menandai citra, dimulai dengan yang tertua, sampai ada satu atau lebih sedikit citra yang tersisa yang sesuai. Ini menandai citra A dan citra B untuk kedaluwarsa. Namun, citra A sudah terlihat oleh Aturan 1 dan jika citra B kedaluwarsa, maka akan melanggar Aturan 1 dan dengan demikian dilewati.
- Hasil: citra A kedaluwarsa.

Contoh B

Ini adalah repositori yang sama seperti contoh sebelumnya tetapi urutan prioritas aturan diubah untuk mengcitrakan hasilnya.

Isi repositori:

- citra A, Taglist: ["beta-1", "prod-1"], Didorong: 10 hari yang lalu
- citra B, Taglist: ["beta-2", "prod-2"], Didorong: 9 hari yang lalu
- citra C, Taglist: ["beta-3"], Didorong: 8 hari yang lalu

Teks kebijakan siklus hidup:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["beta*"],
                "countType": "imageCountMoreThan",
```

```
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Logika kebijakan siklus hidup ini adalah:

- Aturan 1 mengidentifikasi citra yang ditandai dengan prefiks `beta`. Ini harus menandai citra, dimulai dengan yang tertua, sampai ada satu atau lebih sedikit citra yang tersisa yang sesuai. Ini melihat ketiga citra dan akan menandai citra A dan citra B untuk kedaluwarsa.
- Aturan 2 mengidentifikasi citra yang ditandai dengan prefiks `prod`. Ini harus menandai citra, dimulai dengan yang tertua, sampai ada satu atau lebih sedikit citra yang tersisa yang sesuai. Ini tidak akan melihat citra karena semua citra yang tersedia sudah dilihat oleh Aturan 1 dan dengan demikian akan menandai tidak ada citra tambahan.
- Hasil: citra A dan B kedaluwarsa.

Memfilter beberapa tanda dalam satu aturan

Contoh berikut menentukan sintaks kebijakan siklus hidup untuk beberapa pola tag dalam satu aturan. Contoh repositori dan kebijakan siklus hidup diberikan bersama dengan penjelasan dari hasilnya.

Contoh A

Ketika beberapa pola tag ditentukan pada satu aturan, gambar harus cocok dengan semua pola tag yang terdaftar.

Isi repositori:

- citra A, Taglist: ["alpha-1"], Didorong: 12 hari yang lalu
- citra B, Taglist: ["beta-1"], Didorong: 11 hari yang lalu
- citra C, Taglist: ["alpha-2", "beta-2"], Didorong: 10 hari yang lalu
- citra D, Taglist: ["alpha-3"], Didorong: 4 hari yang lalu
- citra E, Taglist: ["beta-3"], Didorong: 3 hari yang lalu
- citra F, Taglist: ["alpha-4", "beta-4"], Didorong: 2 hari yang lalu

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Logika kebijakan siklus hidup ini adalah:

- Aturan 1 mengidentifikasi gambar yang ditandai dengan awalan `alpha` dan. `beta` Ini melihat citra C dan F. Ini harus menandai citra yang lebih tua dari lima hari, yang akan menjadi citra C.
- Hasil: citra C kedaluwarsa.

Contoh B

Contoh berikut mengcitrakan bahwa tanda tidak bersifat eksklusif.

Isi repositori:

- citra A, Taglist: ["alpha-1", "beta-1", "gamma-1"], Didorong: 10 hari yang lalu
- citra B, Taglist: ["alpha-2", "beta-2"], Didorong: 9 hari yang lalu
- citra C, Taglist: ["alpha-3", "beta-3", "gamma-2"], Didorong: 8 hari yang lalu

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Logika kebijakan siklus hidup ini adalah:

- Aturan 1 mengidentifikasi gambar yang ditandai dengan awalan `alpha` dan. `beta` Ini melihat semua citra. Ini harus menandai citra, dimulai dengan yang tertua, sampai ada satu atau lebih sedikit citra yang tersisa yang sesuai. Ini menandai citra A dan B untuk kedaluwarsa.
- Hasil: citra A dan B kedaluwarsa.

Memfilter semua citra

Contoh kebijakan siklus hidup berikut ini menentukan semua citra dengan filter yang berbeda. Contoh repositori dan kebijakan siklus hidup diberikan bersama dengan penjelasan dari hasilnya.

Contoh A

Berikut ini adalah sintaks kebijakan siklus hidup untuk kebijakan yang berlaku untuk semua aturan tetapi hanya mempertahankan satu citra dan membuat semua lainnya kedaluwarsa.

Isi repositori:

- citra A, Taglist: ["alpha-1"], Didorong: 4 hari yang lalu
- citra B, Taglist: ["beta-1"], Didorong: 3 hari yang lalu
- citra C, Taglist: [], Didorong: 2 hari yang lalu
- citra D, Taglist: ["alpha-2"], Didorong: 1 hari yang lalu

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Logika kebijakan siklus hidup ini adalah:

- Aturan 1 mengidentifikasi semua citra. Ini melihat citra A, B, C, dan D. Ini membuat semua citra selain yang terbaru kedaluwarsa. Ini menandai citra A, B, dan C untuk kedaluwarsa.
- Hasil: citra A, B, dan C kedaluwarsa.

Contoh B

Contoh berikut ini mengcitrakan kebijakan siklus hidup yang menggabungkan semua jenis aturan dalam satu kebijakan.

Isi repositori:

- Citra A, Taglist: ["alpha-1", "beta-1"], Didorong: 4 hari yang lalu
- Citra B, Taglist: [], Didorong: 3 hari yang lalu
- Citra C, Taglist: ["alpha-2"], Didorong: 2 hari yang lalu
- Citra D, Taglist: ["git hash"], Didorong: 1 hari yang lalu
- Citra E, Taglist: [], Didorong: 1 hari yang lalu

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 3,
            "description": "Rule 3",
            "selection": {
                "tagStatus": "any",
```

```
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Logika kebijakan siklus hidup ini adalah:

- Aturan 1 mengidentifikasi citra yang ditandai dengan prefiks `alpha`. Ini mengidentifikasi citra A dan C. Ini mempertahankan citra terbaru dan menandai sisanya untuk kedaluwarsa. Ini menandai citra A untuk kedaluwarsa.
- Aturan 2 mengidentifikasi citra yang tidak ditandai. Ini mengidentifikasi citra B dan E. Ini menandai semua citra yang lebih tua dari satu hari untuk kedaluwarsa. Ini menandai citra B untuk kedaluwarsa.
- Aturan 3 mengidentifikasi semua citra. Ini mengidentifikasi citra A, B, C, D, dan E. Ini mempertahankan citra terbaru dan menandai sisanya untuk kedaluwarsa. Namun, ini tidak dapat menandai citra A, B, C, atau E karena citra tersebut diidentifikasi oleh aturan prioritas yang lebih tinggi. Ini menandai citra D untuk kedaluwarsa.
- Hasil: Citra A, B, dan D kedaluwarsa.

Contoh arsip

Contoh berikut menunjukkan kebijakan siklus hidup yang mengarsipkan gambar alih-alih menghapusnya.

Mengarsipkan gambar yang lebih tua dari jumlah hari tertentu

Contoh berikut menunjukkan kebijakan siklus hidup yang mengarsipkan gambar dengan tag yang dimulai dengan `prod` yang lebih tua dari 30 hari:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive production images older than 30 days",
```

```
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 30
            },
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        }
    ]
}
```

Pengarsipan gambar tidak ditarik dalam jumlah hari tertentu

Contoh berikut menunjukkan kebijakan siklus hidup yang mengarsipkan gambar yang belum ditarik dalam 90 hari:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive images not pulled in 90 days",
            "selection": {
                "tagStatus": "any",
                "countType": "sinceImagePulled",
                "countUnit": "days",
                "countNumber": 90
            },
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        }
    ]
}
```

Menggabungkan arsip dan aturan kedaluwarsa

Contoh berikut menunjukkan kebijakan siklus hidup yang mengarsipkan gambar yang lebih tua dari 30 hari dan kemudian secara permanen kedaluwarsa gambar yang telah diarsipkan selama lebih dari 365 hari:

> **Note**
>
> Gambar yang diarsipkan memiliki durasi penyimpanan minimum 90 hari. Anda tidak dapat mengonfigurasi kebijakan siklus hidup yang menghapus gambar yang telah berada di arsip kurang dari 90 hari. Jika Anda harus menghapus gambar yang telah diarsipkan kurang dari 90 hari, Anda harus menggunakan batch-delete-image API, tetapi Anda akan dikenakan biaya untuk durasi penyimpanan minimum 90 hari.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Archive images older than 30 days",
            "selection": {
                "tagStatus": "any",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 30
            },
            "action": {
                "type": "transition",
                "targetStorageClass": "archive"
            }
        },
        {
            "rulePriority": 2,
            "description": "Expire images archived for more than 365 days",
            "selection": {
                "tagStatus": "any",
                "storageClass": "archive",
                "countType": "sinceImageTransitioned",
                "countUnit": "days",
                "countNumber": 365
            },
```

```
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Properti kebijakan siklus hidup di Amazon ECR

Kebijakan siklus hidup memiliki properti berikut.

Untuk melihat contoh kebijakan siklus hidup, lihat. Contoh kebijakan siklus hidup di Amazon ECR Untuk petunjuk tentang membuat kebijakan siklus hidup menggunakan AWS CLI, lihat. Untuk membuat kebijakan siklus hidup (AWS CLI)

Prioritas peraturan

`rulePriority`

Jenis: integer

Wajib: ya

Menetapkan urutan aturan yang diterapkan, terendah ke tertinggi. Aturan kebijakan siklus hidup dengan prioritas 1 diterapkan terlebih dahulu, aturan dengan prioritas berikutnya, dan seterusnya. 2 Ketika Anda menambahkan aturan ke kebijakan siklus hidup, Anda harus memberi nilai unik pada masing-masing aturan untuk `rulePriority`. Nilai tidak perlu berurutan di seluruh aturan dalam kebijakan. Aturan dengan nilai `tagStatus` `any` harus memiliki nilai tertinggi untuk `rulePriority` dan dievaluasi terakhir.

Deskripsi

`description`

Jenis: string

Wajib: tidak

(Opsional) Menjelaskan tujuan sebuah aturan dalam kebijakan siklus hidup.

Status tanda

`tagStatus`

Jenis: string

Wajib: ya

Menentukan apakah aturan kebijakan siklus hidup yang Anda tambahkan menentukan tanda untuk sebuah citra. Pilihan yang dapat diterima adalah `tagged`, `untagged`, atau `any`. Jika Anda menentukan aturan `any`, maka semua citra dievaluasi oleh aturan tersebut. Jika Anda menentukan`tagged`, maka Anda juga harus menentukan `tagPrefixList` nilai atau `tagPatternList` nilai. Jika Anda menentukan`untagged`, maka Anda harus menghilangkan keduanya `tagPrefixList` dan`tagPatternList`.

Daftar pola tag

`tagPatternList`

Jenis: daftar [string]

Wajib: ya, jika `tagStatus` disetel ke tag dan `tagPrefixList` tidak ditentukan

Saat membuat kebijakan siklus hidup untuk gambar yang diberi tag, sebaiknya gunakan a untuk menentukan tag yang akan `tagPatternList` kedaluwarsa. Anda menentukan daftar pola tag gambar yang dipisahkan koma yang mungkin berisi wildcard (*) untuk mengambil tindakan dengan kebijakan siklus hidup Anda. Misalnya, jika gambar Anda ditandai sebagai`prod`,, `prod1``prod2`, dan seterusnya, Anda akan menggunakan daftar pola tag `prod*` untuk menentukan semuanya. Jika Anda menentukan beberapa tanda, hanya citra dengan semua tanda yang ditentukan yang dipilih.

> **Important**
>
> Ada batas maksimum empat wildcard (*) per string. Misalnya, `["*test*1*2*3", "test*1*2*3*"]` valid tetapi `["test*1*2*3*4*5*6"]` tidak valid.

Daftar prefiks tanda

`tagPrefixList`

Jenis: daftar [string]

Wajib: ya, jika `tagStatus` disetel ke tag dan `tagPatternList` tidak ditentukan

Hanya digunakan jika Anda menentukan `"tagStatus": "tagged"` dan Anda tidak menentukan. `tagPatternList` Anda harus menentukan daftar prefiks tanda citra yang dipisahkan koma untuk mengambil tindakan dengan kebijakan siklus hidup Anda. Misalnya, jika citra Anda ditandai sebagai `prod`, `prod1`, `prod2`, dan seterusnya, Anda akan menggunakan prefiks `prod` untuk menentukan semuanya. Jika Anda menentukan beberapa tanda, hanya citra dengan semua tanda yang ditentukan yang dipilih.

Kelas penyimpanan

`storageClass`

Jenis: string

Diperlukan: ya, `countType` jika `sinceImageTransitioned`

Aturan hanya akan memilih gambar dari kelas penyimpanan ini. Saat menggunakan `countType` dari`imageCountMoreThan`,`sinceImagePushed`, atau`sinceImagePulled`, satu-satunya nilai yang didukung adalah`standard`. Saat menggunakan jenis hitungan`sinceImageTransitioned`, ini diperlukan, dan satu-satunya nilai yang didukung adalah`archive`. Jika Anda menghilangkan ini, nilai `standard` akan digunakan.

Jenis hitungan

`countType`

Jenis: string

Wajib: ya

Tentukan jenis jumlah untuk diterapkan pada citra.

Jika `countType` diatur ke `imageCountMoreThan`, Anda juga menentukan `countNumber` untuk membuat aturan yang menetapkan batas pada jumlah citra yang ada di repositori Anda. Jika `countType` diatur ke`sinceImagePushed`,`sinceImagePulled`, atau`sinceImageTransitioned`, Anda juga menentukan `countUnit` dan `countNumber` menentukan batas waktu pada gambar yang ada di repositori Anda.

Unit hitungan

`countUnit`

Jenis: string

Diperlukan: ya, hanya jika `countType` diatur ke`sinceImagePushed`,`sinceImagePulled`, atau `sinceImageTransitioned`

Tentukan unit hitungan `days` untuk menunjukkan bahwa sebagai unit waktu, di samping `countNumber`, yang merupakan jumlah hari.

Ini seharusnya hanya `countType` ditentukan kapan`sinceImagePushed`,`sinceImagePulled`, atau`sinceImageTransitioned`; kesalahan akan terjadi jika Anda menentukan satuan hitungan kapan `countType` ada nilai lainnya.

Jumlah hitungan

`countNumber`

Jenis: integer

Wajib: ya

Tentukan jumlah hitungan. Nilai yang dapat diterima adalah integer positif (nilai `0` tidak diterima).

Jika `countType` yang digunakan adalah `imageCountMoreThan`, maka nilainya adalah jumlah maksimum citra yang ingin Anda pertahankan di repositori Anda. Jika `countType` yang digunakan adalah `sinceImagePushed`, maka nilainya adalah batas usia maksimum untuk citra Anda. Jika yang `countType` digunakan adalah`sinceImagePulled`, maka nilainya adalah jumlah hari maksimum sejak gambar terakhir ditarik. Jika yang `countType` digunakan adalah`sinceImageTransitioned`, maka nilainya adalah jumlah hari maksimum sejak gambar diarsipkan.

Tindakan

`type`

Jenis: string

Wajib: ya

Tentukan jenis tindakan. Nilai yang didukung adalah `expire` (untuk menghapus gambar) dan `transition` (untuk memindahkan gambar ke penyimpanan arsip).

`targetStorageClass`

Jenis: string

Diperlukan: ya, `type` jika `transition`

Kelas penyimpanan yang Anda inginkan dari kebijakan siklus hidup untuk mentransisikan gambar. `archive`adalah satu-satunya nilai yang didukung.

Pengecualian pembaruan waktu tarik

Amazon ECR memperbarui `LastRecordedPullTime` stempel waktu pada setiap tarikan kecuali penarikan oleh Inspector. AWS Pengecualian pembaruan waktu tarik memungkinkan Anda menentukan peran IAM ARNs yang seharusnya tidak memperbarui waktu tarik gambar saat mereka menarik gambar, seperti penarikan oleh pemindai pihak ketiga (seperti Crowdstrike, Snyk, dan Trivy). Ini berguna untuk gambar yang digunakan untuk pengujian atau CI/CD tujuan di mana Anda tidak ingin waktu tarik memengaruhi keputusan kebijakan siklus hidup.

Ketika peran dalam daftar pengecualian menarik gambar, waktu tarik tetap tidak berubah. Peran lain apa pun terus memperbarui waktu tarik (perilaku saat ini). Anda dapat mengonfigurasi hingga 100 pengecualian per akun.

Mengelola pengecualian pembaruan waktu tarik

Untuk mengelola pengecualian pembaruan waktu tarik, Anda memerlukan izin IAM berikut:

- `ecr:CreatePullTimeUpdateExclusion`— Memberikan izin untuk menambahkan peran ARN ke daftar pengecualian.
- `ecr:DeletePullTimeUpdateExclusion`— Memberikan izin untuk menghapus peran ARN dari daftar pengecualian.
- `ecr:ListPullTimeUpdateExclusions`— Memberikan izin untuk mencantumkan semua peran ARNs dalam daftar pengecualian.

> **Note**
>
> Anda tidak perlu `iam:PassRole` izin. Amazon ECR tidak mengambil peran untuk melakukan tindakan; itu hanya menggunakan konfigurasi pengecualian ARNs untuk menentukan apakah waktu tarik gambar harus diperbarui.

Anda dapat mengelola pengecualian pembaruan waktu tarik menggunakan konsol Amazon ECR atau CLI. AWS

Konsol Manajemen AWS

Untuk mengelola pengecualian pembaruan waktu tarik ()Konsol Manajemen AWS

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/ private-registry/repositori
2. Dari bilah navigasi, pilih Wilayah.
3. Di panel navigasi, pilih Registri pribadi, Fitur & Pengaturan, lalu pilih Pengecualian pembaruan waktu tarik.
4. Untuk menambahkan pengecualian, pilih Tambahkan pengecualian, masukkan peran ARN, lalu pilih Tambah.
5. Untuk menghapus pengecualian, pilih peran ARN dari daftar dan pilih Hapus.
6. Untuk melihat semua pengecualian, daftar menampilkan semua peran ARNs yang dikonfigurasi.

AWS CLI

Untuk membuat pengecualian pembaruan waktu tarik

- Gunakan create-pull-time-update-exclusion perintah untuk menambahkan peran ARN ke daftar pengecualian:

```
aws ecr create-pull-time-update-exclusion \
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

Perintah mengembalikan peran ARN dan stempel waktu pembuatan:

```
{
   "roleArn": "arn:aws:iam::123456789012:role/scanner-role",
   "createdAt": 1745531331.0
}
```

Untuk menghapus pengecualian pembaruan waktu tarik

- Gunakan delete-pull-time-update-exclusion perintah untuk menghapus peran ARN dari daftar pengecualian:

```
aws ecr delete-pull-time-update-exclusion \
```

```
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

Perintah mengembalikan peran ARN yang telah dihapus:

```
{
  "roleArn": "arn:aws:iam::123456789012:role/scanner-role"
}
```

Untuk mencantumkan pengecualian pembaruan waktu tarik

1. Gunakan list-pull-time-update-exclusions perintah untuk mencantumkan semua peran ARNs dalam daftar pengecualian:

```
aws ecr list-pull-time-update-exclusions
```

Jika tidak ada pengecualian yang dikonfigurasi, perintah mengembalikan daftar kosong:

```
{
   "pullTimeUpdateExclusions": []
}
```

Jika pengecualian dikonfigurasi, perintah mengembalikan daftar peran ARNs:

```
{
   "pullTimeUpdateExclusions": [
        "arn:aws:iam::123456789012:role/security-role"
    ]
}
```

2. Untuk melakukan paginasi hasil, gunakan `--next-token` parameter `--max-results` dan:

```
aws ecr list-pull-time-update-exclusions \
    --max-results 4
```

Perintah mengembalikan hingga jumlah hasil yang ditentukan dan `nextToken` jika lebih banyak hasil tersedia:

```
{
   "pullTimeUpdateExclusions": [
```

```
        "arn:aws:iam::123456789012:role/security-role1",
        "arn:aws:iam::123456789012:role/security-role2",
        "arn:aws:iam::123456789012:role/security-role3",
        "arn:aws:iam::123456789012:role/security-role4"
    ],
    "nextToken": "ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79..."
}
```

Untuk mengambil halaman hasil berikutnya, gunakan `nextToken` dari respons sebelumnya:

```
aws ecr list-pull-time-update-exclusions \
    --max-results 4 \
    --next-token ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79...
```

Pertimbangan untuk pengecualian pembaruan waktu tarik

Pertimbangkan hal berikut saat menggunakan pengecualian pembaruan waktu tarik:

- Ukuran halaman default untuk pengecualian daftar adalah 100. Anda dapat menggunakan pagination dengan `maxResults` dan `nextToken` parameter.
- Hanya peran IAM yang valid ARNs dalam format ARN yang benar yang diterima.
- Jika Anda mencoba membuat pengecualian yang sudah ada, Anda akan menerima `ExclusionAlreadyExistsException` kesalahan. Jika Anda mencoba menghapus pengecualian yang tidak ada, Anda akan menerima `ExclusionNotFoundException` kesalahan.

Amazon Elastic Container Registry

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Third-partyAuditor secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari program AWS kepatuhan. Untuk mempelajari tentang program kepatuhan yang berlaku di Amazon ECR, lihat Layanan AWS dalam Cakupan melalui Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon ECR. Topik berikut menunjukkan kepada Anda cara mengonfigurasi Amazon ECR untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Amazon ECR Anda.

Topik

Identity and Access Management untuk Amazon Elastic Container Registry

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (mendapatkan izin) untuk menggunakan sumber daya Amazon ECR. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Pemecahan Masalah Identitas dan Akses Amazon Elastic Container Registry)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana Amazon Elastic Container Registry bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensyal dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensyal. Google/ Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensyal sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan

memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Identity-based kebijakan

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Identity-based kebijakan dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada beberapa identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Resource-based kebijakan

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3.

Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Resource-based kebijakan adalah kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – Menentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCP) – Menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCP) dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Amazon Elastic Container Registry bekerja dengan IAM

Sebelum menggunakan IAM untuk mengelola akses ke Amazon ECR, Anda harus memahami fitur IAM mana yang tersedia untuk digunakan dengan Amazon ECR. Untuk mendapatkan tampilan tingkat tinggi tentang cara Amazon ECR dan AWS layanan lainnya bekerja dengan IAM, lihat AWS Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM.

Topik

Kebijakan Amazon ECR Identity-based

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Amazon ECR support tindakan, sumber daya, dan kunci syarat tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat Referensi Elemen kebijakan IAM JSON dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Tindakan kebijakan di Amazon ECR menggunakan prefiks berikut sebelum tindakan: `ecr:` Misalnya, untuk memberikan izin kepada seseorang untuk menggambarkan instans DB dengan operasi `CreateRepository` API Amazon ECR, Anda menyertakan tindakan `ecr:CreateRepository` dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. Amazon ECR menentukan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Misalnya, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "ecr:Describe*"
```

Untuk melihat daftar tindakan Amazon ECR, lihatTindakan, Sumber Daya, dan kunci syarat untuk Amazon Elastic Elastic di Panduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Sumber daya repositori Amazon ECR memiliki ARN berikut:

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

Untuk informasi selengkapnya tentang format ARN, lihat Nama Sumber Daya Amazon (ARN) dan Ruang Nama AWS Layanan.

Misalnya, untuk menentukan wilayah `us-east-1` `my-repo` repositori dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

Untuk menentukan semua repositori milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

Untuk menentukan beberapa sumber daya dalam pernyataan tunggal, pisahkan ARN dengan koma.

```
"Resource": [
      "resource1",
      "resource2"
```

Untuk melihat daftar jenis sumber daya Amazon ECR dan ARN-nya, lihatSumber Daya yang Ditetapkan oleh Amazon Elastic Container Registry dalam Panduan Pengguna IAM. Untuk mempelajari tindakan mana yang dapat menentukan ARN setiap sumber daya, lihat Tindakan yang Ditentukan oleh Amazon Elastic Container Registry.

Kunci syarat

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Amazon ECR menentukan set kunci syaratnya sendiri dan juga support penggunaan beberapa kunci syarat global. Untuk melihat semua kunci kondisi AWS global, lihat Kunci Konteks Kondisi AWS Global di Panduan Pengguna IAM.

Semua tindakan Amazon ECR support kunci syarat `aws:ResourceTag` dan `ecr:ResourceTag`. Untuk informasi selengkapnya, lihat Menggunakan Tag-Based Access Control.

Untuk melihat daftar kunci syarat Amazon ECR, lihat Kondisi Kunci yang Ditetapkan oleh Amazon Elastic Container Registry dalam Panduan Pengguna IAM. Untuk mempelajari tindakan dan sumber daya yang mana Anda diperbolehkan menggunakan kunci syarat, lihat Tindakan yang Ditentukan oleh Amazon Elastic Container Registry.

Contoh

Untuk melihat contoh kebijakan berbasis identitas Amazon ECR, lihat Contoh Identity-based kebijakan Amazon Elastic Container Registry.

Kebijakan berbasis sumber daya Amazon ECR

Resource-based kebijakan adalah dokumen kebijakan JSON yang menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya Amazon ECR dan dalam kondisi apa. Amazon ECR mendukung kebijakan izin berbasis sumber daya untuk repositori Amazon ECR. Resource-based kebijakan memungkinkan Anda memberikan izin penggunaan ke akun lain berdasarkan per sumber daya. Anda juga dapat menggunakan kebijakan berbasis sumber daya untuk mengizinkan layanan AWS mengakses repositori Amazon ECR Anda.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan principal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berada di AWS akun yang berbeda, Anda juga harus memberikan izin entitas utama untuk mengakses sumber daya. Berikan izin dengan melampirkan kebijakan berbasis identitas ke entitas tersebut. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal di akun yang sama, Anda tidak memerlukan izin repositori Amazon ECR tambahan dalam kebijakan berbasis identitas. Untuk informasi selengkapnya, lihat Perbedaan Peran IAM dari Resource-based Kebijakan di Panduan Pengguna IAM.

Layanan Amazon ECR hanya support satu jenis kebijakan berbasis sumber daya yang disebut kebijakan repositori, yang terlampir pada repositori. Kebijakan ini menentukan entitas prinsipal (akun, pengguna, peran, dan pengguna gabungan) yang dapat melakukan tindakan pada repositori. Untuk mempelajari cara melampirkan kebijakan berbasis sumber daya ke repositori, lihat Kebijakan repositori pribadi di Amazon ECR.

> **Note**
>
> Dalam kebijakan repositori Amazon ECR, elemen kebijakan `Sid` mendukung karakter tambahan dan spasi yang tidak didukung dalam kebijakan IAM.

Contoh

Untuk melihat contoh kebijakan berbasis sumber daya Amazon ECR, lihat Contoh kebijakan repositori pribadi di Amazon ECR,

Otorisasi berdasarkan tag Amazon ECR

Anda dapat melampirkan tanda di sumber daya Amazon ECR atau tanda yang lolos dalam permintaan untuk Amazon ECR. Untuk mengontrol akses berdasarkan tandanya, Anda memberikan informasi tanda di elemen syarat kebijakan dengan menggunakan kunci syarat `ecr:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi lebih lanjut tentang penandaan Amazon ECR sumber daya, lihat Menandai repositori pribadi di Amazon ECR.

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tanda pada sumber daya tersebut, lihat Menggunakan Tag-Based Access Control.

Peran Amazon ECR IAM

Peran IAM adalah entitas dalam AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensial sementara dengan Amazon ECR

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensil keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRoleatau. GetFederationToken

Amazon ECR support penggunaan kredensial sementara.

Service-linked peran

Service-linked peran memungkinkan AWS layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Service-linked peran muncul di akun IAM Anda dan dimiliki oleh layanan. Administrator IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Amazon ECR mendukung peran terkait layanan. Untuk informasi selengkapnya, lihat Menggunakan Peran Terkait Layanan untuk Amazon ECR.

Contoh Identity-based kebijakan Amazon Elastic Container Registry

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon ECR. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM dengan menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon ECR, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Elastic Container Registry di Referensi Otorisasi Layanan.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat Kebijakan pada Tab JSON dalam Panduan Pengguna IAM.

Topik

- Praktik Terbaik Kebijakan
- Menggunakan konsol Amazon ECR
- Izinkan Pengguna untuk Melihat Izin Mereka Sendiri
- Mengakses Satu Repositori Amazon ECR

Praktik Terbaik Kebijakan

Identity-based kebijakan menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon ECR di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua

permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.

- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan konsol Amazon ECR

Untuk mengakses konsol Amazon Elastic Container Registry, Anda harus memiliki rangkaian izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Amazon ECR di akun Anda AWS . Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Untuk memastikan bahwa entitas tersebut masih dapat menggunakan konsol Amazon ECR, tambahkan kebijakan `AmazonEC2ContainerRegistryReadOnly` AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan Izin ke Pengguna dalam Panduan Pengguna IAM.

Untuk melihat izin kebijakan ini, lihat AmazonElasticContainerRegistryPublicReadOnlydi Referensi Kebijakan AWS Terkelola.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai alternatif, hanya izinkan akses ke tindakan yang cocok dengan operasi API yang sedang Anda coba lakukan.

Izinkan Pengguna untuk Melihat Izin Mereka Sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengakses Satu Repositori Amazon ECR

Dalam contoh ini, Anda ingin memberikan pengguna di AWS akun Anda akses ke salah satu repositori Amazon ECR Anda,. `my-repo` Anda juga ingin mengizinkan pengguna untuk mendorong, menarik, dan memasukkan citra.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"GetAuthorizationToken",
         "Effect":"Allow",
         "Action":[
            "ecr:GetAuthorizationToken"
         ],
         "Resource":"*"
      },
      {
         "Sid":"ManageRepositoryContents",
         "Effect":"Allow",
         "Action":[
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
         ],
         "Resource":"arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
      }
   ]
}
```

Menggunakan Tag-Based Access Control

Tindakan Amazon ECR `CreateRepository` API memungkinkan Anda menentukan tag saat membuat repositori. Untuk informasi selengkapnya, lihat Menandai repositori pribadi di Amazon ECR.

Untuk mengizinkan pengguna menandai repositori pada pembuatan, mereka harus memiliki izin untuk menggunakan tindakan yang membuat sumber daya (misalnya, `ecr:CreateRepository`). Jika tanda ditentukan dalam aksi pembuatan sumber daya, Amazon melakukan otorisasi tambahan pada tindakan `ecr:CreateRepository` untuk memverifikasi apakah pengguna memiliki izin untuk membuat tanda.

Anda dapat menggunakan kontrol akses berbasis tag melalui kebijakan IAM. Berikut ini adalah beberapa contohnya.

Kebijakan berikut hanya akan mengizinkan pengguna untuk membuat atau menandai repositori sebagai. `key=environment,value=dev`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateTaggedRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        },
        {
            "Sid": "AllowTagRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:TagResource"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        }
    ]
}
```

Kebijakan berikut akan memungkinkan pengguna untuk menarik gambar dari semua repositori kecuali mereka ditandai sebagai. `key=environment,value=prod`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/environment": "prod"
                }
            }
        }
    ]
}
```

AWS kebijakan terkelola untuk Amazon Elastic Container Registry

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

Amazon ECR menyediakan beberapa kebijakan terkelola yang dapat Anda lampirkan ke identitas IAM atau ke instans Amazon EC2. Kebijakan terkelola ini memungkinkan tingkat kontrol yang berbeda atas akses ke sumber daya Amazon ECR dan operasi API. Untuk informasi lebih lanjut tentang setiap operasi API yang disebutkan dalam kebijakan ini, lihat Tindakan di Referensi Amazon Elastic Container Registry.

Topik

- AmazonEC2ContainerRegistryFullAccess
- AmazonEC2ContainerRegistryPowerUser
- AmazonEC2ContainerRegistryPullOnly
- AmazonEC2ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- ECRTemplateServiceRolePolicy
- Pembaruan Amazon ECR ke AWS kebijakan terkelola

AmazonEC2ContainerRegistryFullAccess

Anda dapat melampirkan kebijakan `AmazonEC2ContainerRegistryFullAccess` ke identitas IAM Anda. Kebijakan ini memberikan akses administratif ke sumber daya Amazon ECR dan memberikan identitas IAM (seperti pengguna, grup, atau peran) akses ke layanan AWS yang terintegrasi dengan Amazon ECR untuk menggunakan semua fitur Amazon ECR. Menggunakan kebijakan ini memungkinkan akses ke semua fitur Amazon ECR yang tersedia di. Konsol Manajemen AWS

Untuk melihat izin kebijakan ini, lihat AmazonEC2ContainerRegistryFullAccessdi Referensi Kebijakan AWS Terkelola.

AmazonEC2ContainerRegistryPowerUser

Anda dapat melampirkan kebijakan `AmazonEC2ContainerRegistryPowerUser` ke identitas IAM. Kebijakan ini memberikan izin administratif yang mengizinkan pengguna IAM membaca dan menulis ke repositori, namun tidak mengizinkan mereka menghapus repositori atau mengubah dokumen kebijakan yang diterapkan di dalamnya.

Untuk melihat izin kebijakan ini, lihat AmazonEC2ContainerRegistryPowerUserdi Referensi Kebijakan AWS Terkelola.

AmazonEC2ContainerRegistryPullOnly

Anda dapat melampirkan kebijakan `AmazonEC2ContainerRegistryPullOnly` ke identitas IAM Anda. Kebijakan ini memberikan izin untuk menarik gambar kontainer dari Amazon ECR. Jika registri diaktifkan untuk cache pull-through, itu juga akan memungkinkan penarikan untuk mengimpor gambar dari registri hulu.

Untuk melihat izin kebijakan ini, lihat AmazonEC2ContainerRegistryPullOnlydi Referensi Kebijakan AWS Terkelola.

AmazonEC2ContainerRegistryReadOnly

Anda dapat melampirkan kebijakan `AmazonEC2ContainerRegistryReadOnly` ke identitas IAM. Kebijakan ini memberikan izin baca saja untuk Amazon ECR. Ini termasuk kemampuan untuk membuat daftar repositori dan citra dalam repositori. Ini juga mencakup kemampuan untuk menarik citra dari Amazon ECR dengan Docker CLI.

Untuk melihat izin kebijakan ini, lihat AmazonEC2ContainerRegistryReadOnlydi Referensi Kebijakan AWS Terkelola.

AWSECRPullThroughCache_ServiceRolePolicy

Anda tidak dapat melampirkan kebijakan IAM `AWSECRPullThroughCache_ServiceRolePolicy` terkelola ke entitas IAM Anda. Kebijakan ini dilampirkan ke peran terkait layanan yang memungkinkan Amazon ECR untuk mendorong gambar ke repositori Anda melalui alur kerja pull through cache. Untuk informasi selengkapnya, lihat Peran terkait layanan Amazon ECR untuk menarik cache.

ECRReplicationServiceRolePolicy

Anda tidak dapat melampirkan kebijakan IAM `ECRReplicationServiceRolePolicy` terkelola ke entitas IAM Anda. Kebijakan ini dilampirkan ke peran terkait layanan yang mengizinkan Amazon ECR melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Menggunakan Peran Terkait Layanan untuk Amazon ECR.

ECRTemplateServiceRolePolicy

Anda tidak dapat melampirkan kebijakan IAM `ECRTemplateServiceRolePolicy` terkelola ke entitas IAM Anda. Kebijakan ini dilampirkan ke peran terkait layanan yang mengizinkan Amazon ECR melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Menggunakan Peran Terkait Layanan untuk Amazon ECR.

Pembaruan Amazon ECR ke AWS kebijakan terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Amazon ECR sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, silakan berlangganan RSS feed pada halaman riwayat Dokumen Amazon ECR.

Perubahan	Deskripsi	Tanggal
Peran terkait layanan Amazon ECR untuk menarik cache — Permbaruan ke kebijakan yang sudah ada	Amazon ECR menambahkan izin baru ke kebijakan `AWSECRPullThroughCache_ServiceRolePolicy` . Izin ini memungkinkan Amazon ECR untuk menarik gambar dari registri pribadi ECR. Ini diperlukan	Maret 12, 2025

Perubahan	Deskripsi	Tanggal
	saat menggunakan aturan pull through cache untuk menyimpan gambar dari registri pribadi Amazon ECR lainnya.	
AmazonEC2ContainerRegistryPullOnly – Kebijakan baru	Amazon ECR menambahkan kebijakan baru yang memberikan izin khusus tarik ke Amazon ECR.	Oktober 10, 2024
ECRTemplateServiceRolePolicy – Kebijakan baru	Amazon ECR menambahkan kebijakan baru. Kebijakan ini dikaitkan dengan peran `ECRTemplateServiceRolePolicy` terkait layanan untuk fitur template pembuatan repositori.	Juni 20, 2024
AWSECRPullThroughCache_ServiceRolePolicy — Perbarui ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AWSECRPullThroughCache_ServiceRolePolicy` . Izin ini memungkinkan Amazon ECR untuk mengambil konten terenkripsi dari rahasia Secrets Manager. Ini diperlukan saat menggunakan aturan pull through cache untuk menyimpan gambar dari registri hulu yang memerlukan otentikasi.	15 November 2023

Perubahan	Deskripsi	Tanggal
AWSECRPullThroughCache_ ServiceRolePolicy — Kebijakan baru	Amazon ECR menambahkan kebijakan baru. Kebijakan ini dikaitkan dengan peran `AWSServiceRoleForECRPullThroughCache` terkait layanan untuk fitur pull through cache.	29 November 2021
ECRReplicationServiceRolePolicy – Kebijakan baru	Amazon ECR menambahkan kebijakan baru. Kebijakan ini dikaitkan dengan peran `AWSServiceRoleForECRReplication` terkait layanan untuk fitur replikasi.	4 Desember 2020
AmazonEC2ContainerRegistryFullAccess – Pembaruan ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AmazonEC2ContainerRegistryFullAccess`. Izin ini mengizinkan prinsipal untuk membuat peran terkait layanan Amazon ECR.	4 Desember 2020
AmazonEC2ContainerRegistryReadOnly – Pembaruan ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AmazonEC2ContainerRegistryReadOnly` yang mengizinkan prinsipal untuk membaca kebijakan siklus hidup, daftar tanda, dan menjelaskan temuan pemindaian untuk citra.	10 Desember 2019

Perubahan	Deskripsi	Tanggal
AmazonEC2ContainerRegistryPowerUser – Pembaruan ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AmazonEC2ContainerRegistryPowerUser`. Mereka mengizinkan prinsipal untuk membaca kebijakan siklus hidup, daftar tanda, dan menjelaskan temuan pemindaian untuk citra.	10 Desember 2019
AmazonEC2ContainerRegistryFullAccess – Pembaruan ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AmazonEC2ContainerRegistryFullAccess`. Mereka memungkinkan kepala sekolah untuk mencari acara manajemen atau peristiwa AWS CloudTrail Wawasan yang ditangkap oleh. CloudTrail	10 November 2017
AmazonEC2ContainerRegistryReadOnly – Pembaruan ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AmazonEC2ContainerRegistryReadOnly`. Mereka mengizinkan prinsipal untuk menggambarkan citra Amazon ECR.	11 Oktober 2016

Perubahan	Deskripsi	Tanggal
AmazonEC2ContainerRegistryPowerUser – Pembaruan ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AmazonEC2ContainerRegistryPowerUser` . Mereka mengizinkan prinsipal untuk menggambarkan citra Amazon ECR.	11 Oktober 2016
AmazonEC2ContainerRegistryReadOnly – Kebijakan baru	Amazon ECR menambahkan kebijakan baru yang memberikan izin hanya-baca ke Amazon ECR. Izin ini mencakup kemampuan untuk membuat daftar repositori dan citra dalam repositori. Izin ini juga termasuk kemampuan untuk menarik citra dari Amazon ECR dengan Docker CLI.	21 Desember 2015
AmazonEC2ContainerRegistryPowerUser – Kebijakan baru	Amazon ECR menambahkan kebijakan baru yang memberikan izin administratif yang memungkinkan pengguna membaca dan menulis ke repositori tetapi tidak mengizinkan mereka untuk menghapus repositori atau mengubah dokumen kebijakan yang diterapkan padanya.	21 Desember 2015

Perubahan	Deskripsi	Tanggal
AmazonEC2Container RegistryFullAccess – Kebijakan baru	Amazon ECR menambahkan kebijakan baru. Kebijakan ini juga memberikan akses penuh ke Amazon ECR.	21 Desember 2015
Amazon ECR mulai melacak perubahan	Amazon ECR mulai melacak perubahan untuk kebijakan AWS terkelola.	24 Juni 2021

Menggunakan Peran Terkait Layanan untuk Amazon ECR

Amazon Elastic Container Registry (Amazon ECR) AWS Identity and Access Management menggunakan peran terkait layanan (IAM) untuk memberikan izin yang diperlukan untuk menggunakan replikasi dan menarik fitur cache. Peran terkait layanan adalah jenis IAM role unik yang terhubung langsung ke Amazon ECR. Peran terkait layanan yang telah ditetapkan oleh Amazon ECR. Ini mencakup semua izin yang diperlukan layanan untuk mendukung replikasi dan menarik fitur cache untuk registri pribadi Anda. Setelah Anda mengonfigurasi replikasi atau menarik cache untuk registri Anda, peran terkait layanan dibuat secara otomatis atas nama Anda. Untuk informasi selengkapnya, lihat Pengaturan registri pribadi di Amazon ECR.

Peran terkait layanan membuat pengaturan replikasi dan menarik cache dengan Amazon ECR lebih mudah. Hal ini karena, dengan menggunakannya, Anda tidak perlu menambahkan semua izin yang diperlukan secara manual. Amazon ECR menentukan izin peran terkait layanan, dan kecuali ditentukan lain, hanya Amazon ECR yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin. Kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran terkait layanan yang sesuai hanya setelah menonaktifkan replikasi atau menarik cache pada registri Anda. Ini memastikan bahwa Anda tidak secara tidak sengaja menghapus izin yang diperlukan Amazon ECR untuk fitur-fitur ini.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat Layanan AWS yang bekerja dengan IAM. Pada halaman yang ditautkan ke ini, cari layanan yang memiliki Ya di kolom peran. Service-linked Pilih Ya dengan tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Topik

Wilayah yang Didukung untuk Peran Terkait Layanan Amazon ECR

Amazon ECR mendukung penggunaan peran terkait layanan di semua Wilayah tempat layanan ECR Amazon tersedia. Untuk informasi selengkapnya tentang ketersediaan Wilayah ECR Amazon, lihat AWS Wilayah dan Titik Akhir.

Peran terkait layanan Amazon ECR untuk replikasi

Amazon ECR menggunakan peran terkait layanan bernama yang AWSServiceRoleForECRReplicationmemungkinkan Amazon ECR mereplikasi gambar di beberapa akun.

Service-linked izin peran untuk Amazon ECR

Peran AWSServiceRoleForECRReplication terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `replication.ecr.amazonaws.com`

Berikut ini kebijakan izin peran `ECRReplicationServiceRolePolicy` yang mengizinkan Amazon ECR untuk menggunakan tindakan berikut pada sumber daya:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> `ReplicateImage` adalah API internal yang digunakan oleh Amazon ECR untuk mereplikasi dan tidak dapat disebut secara langsung.

Anda harus mengonfigurasikan izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin Service-Linked Peran di Panduan Pengguna IAM.

Membuat Peran Terkait Layanan untuk Amazon ECR

Anda tidak perlu membuat peran terkait layanan untuk Amazon ECR secara manual. Saat Anda mengonfigurasi pengaturan replikasi untuk registri Anda di Konsol Manajemen AWS, API AWS CLI, atau AWS API, Amazon ECR akan membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, lalu ingin membuatnya lagi, Anda dapat menggunakan proses yang sama untuk membuat ulang peran tersebut di akun Anda. Saat Anda mengkonfigurasi pengaturan replikasi untuk registrasi Anda, Amazon ECR membuat peran terkait layanan untuk Anda kembali.

Mengedit Peran Terkait Layanan untuk Amazon ECR

Amazon ECR tidak mengizinkan pengeditan peran AWSServiceRoleForECRReplication terkait layanan secara manual. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi lebih lanjut, lihat Mengedit peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus Peran Terkait Layanan untuk Amazon ECR

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami menyarankan Anda menghapus peran tersebut. Dengan begitu Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Namun, Anda harus menghapus

konfigurasi replikasi untuk registrasi Anda di setiap Wilayah sebelum Anda dapat menghapus peran terkait layanan secara manual.

> **Note**
>
> Jika Anda mencoba untuk menghapus sumber daya ketika layanan Amazon ECR masih menggunakan peran, tindakan penghapusan yang Anda lakukan mungkin gagal. Jika hal tersebut terjadi, tunggu beberapa menit dan coba lagi.

Untuk menghapus sumber daya Amazon ECR yang digunakan oleh AWSServiceRoleForECRReplication

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah konfigurasi di mana replikasi Anda diatur.
3. Di panel navigasi, pilih Registri pribadi.
4. Pada halaman registri pribadi, pada bagian konfigurasi replikasi, pilih Edit.
5. Untuk menghapus semua aturan replikasi Anda, pilih Hapus semua. Langkah ini membutuhkan konfirmasi.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForECRReplicationterkait layanan. Untuk informasi selengkapnya, lihat Menghapus Service-Linked Peran di Panduan Pengguna IAM.

Peran terkait layanan Amazon ECR untuk menarik cache

Amazon ECR menggunakan peran terkait layanan bernama AWSServiceRoleForECRPullThroughCacheyang memberikan izin kepada Amazon ECR untuk melakukan tindakan atas nama Anda guna menyelesaikan tindakan pull through cache. Untuk informasi selengkapnya tentang pull through cache, lihatTemplate untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.

Service-linked izin peran untuk Amazon ECR

Peran AWSServiceRoleForECRPullThroughCacheterkait layanan mempercayai layanan berikut untuk mengambil peran tersebut.

- `pullthroughcache.ecr.amazonaws.com`

Detail izin

Kebijakan `AWSECRPullThroughCache_ServiceRolePolicy` izin dilampirkan ke peran terkait layanan. Kebijakan terkelola ini memberikan izin Amazon ECR untuk melakukan tindakan berikut. Untuk informasi selengkapnya, lihat [AWSECRPullThroughCache_ServiceRolePolicy].

- `ecr`— Memungkinkan layanan Amazon ECR untuk menarik dan mendorong gambar ke repositori pribadi.
- `secretsmanager:GetSecretValue`— Memungkinkan layanan Amazon ECR untuk mengambil konten rahasia yang dienkripsi. AWS Secrets Manager Ini diperlukan saat menggunakan aturan pull through cache untuk menyimpan gambar dari registri upstream yang memerlukan otentikasi di registri pribadi Anda. Izin ini hanya berlaku untuk rahasia dengan awalan `ecr-pullthroughcache/` nama.

`AWSECRPullThroughCache_ServiceRolePolicy`Kebijakan ini berisi JSON berikut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ECR",
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage",
                "ecr:BatchGetImage",
                "ecr:BatchImportUpstreamImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetImageCopyStatus"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "SecretsManager",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecr-pullthroughcache/
*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
}
```

Anda harus mengonfigurasikan izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat izin Service-linked peran di Panduan Pengguna IAM.

Membuat Peran Terkait Layanan untuk Amazon ECR

Anda tidak perlu membuat peran terkait layanan Amazon ECR secara manual untuk menarik cache. Saat Anda membuat aturan cache pull through untuk registri pribadi Anda di Konsol Manajemen AWS, API AWS CLI, atau AWS API, Amazon ECR membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, lalu ingin membuatnya lagi, Anda dapat menggunakan proses yang sama untuk membuat ulang peran tersebut di akun Anda. Saat Anda membuat aturan cache pull through untuk registri pribadi Anda, Amazon ECR akan membuat peran terkait layanan untuk Anda lagi jika belum ada.

Mengedit Peran Terkait Layanan untuk Amazon ECR

Amazon ECR tidak mengizinkan pengeditan peran AWSServiceRoleForECRPullThroughCacheterkait layanan secara manual. Setelah peran terkait layanan dibuat, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi lebih lanjut, lihat Mengedit peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus Peran Terkait Layanan untuk Amazon ECR

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami menyarankan Anda menghapus peran tersebut. Dengan begitu Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Namun, Anda harus menghapus aturan pull through cache untuk registri Anda di setiap Wilayah sebelum Anda dapat menghapus peran terkait layanan secara manual.

> **Note**
>
> Jika Anda mencoba menghapus sumber daya saat layanan Amazon ECR masih menggunakan peran tersebut, tindakan penghapusan Anda mungkin gagal. Jika hal tersebut terjadi, tunggu beberapa menit dan coba lagi.

Untuk menghapus sumber daya Amazon ECR yang digunakan oleh AWSServiceRoleForECRPullThroughCache peran yang terhubung dengan layanan

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah tempat aturan cache tarik Anda dibuat.
3. Di panel navigasi, pilih Registri pribadi.
4. Pada halaman registri pribadi, pada bagian konfigurasi Tarik melalui cache, pilih Edit.
5. Untuk setiap aturan pull through cache yang telah Anda buat, pilih aturan dan kemudian pilih Hapus aturan.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForECRPullThroughCacheterkait layanan. Untuk informasi selengkapnya, lihat Menghapus Service-Linked Peran di Panduan Pengguna IAM.

Peran terkait layanan Amazon ECR untuk templat pembuatan repositori

Amazon ECR menggunakan nama peran terkait layanan AWSServiceRoleForECRTemplateyang memberikan izin kepada Amazon ECR untuk melakukan tindakan atas nama Anda guna menyelesaikan tindakan template pembuatan repositori.

Service-linked izin peran untuk Amazon ECR

Peran AWSServiceRoleForECRTemplateterkait layanan mempercayai layanan berikut untuk mengambil peran tersebut.

- `ecr.amazonaws.com`

Detail izin

Kebijakan `ECRTemplateServiceRolePolicy` izin dilampirkan ke peran terkait layanan. Kebijakan terkelola ini memberikan izin Amazon ECR untuk melakukan tindakan pembuatan repositori atas nama Anda.

`ECRTemplateServiceRolePolicy`Kebijakan ini berisi JSON berikut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
     "Sid": "CreateRepositoryWithTemplate",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Anda harus mengonfigurasikan izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat izin Service-linked peran di Panduan Pengguna IAM.

Membuat Peran Terkait Layanan untuk Amazon ECR

Anda tidak perlu membuat peran terkait layanan Amazon ECR secara manual untuk template pembuatan repositori. Saat Anda membuat aturan template pembuatan repositori untuk registri

pribadi Anda di, API Konsol Manajemen AWS, atau AWS API AWS CLI, Amazon ECR membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, lalu ingin membuatnya lagi, Anda dapat menggunakan proses yang sama untuk membuat ulang peran tersebut di akun Anda. Saat Anda membuat aturan pembuatan repositori untuk registri pribadi Anda, Amazon ECR akan membuat peran terkait layanan untuk Anda lagi jika belum ada.

Mengedit Peran Terkait Layanan untuk Amazon ECR

Amazon ECR tidak mengizinkan pengeditan peran AWSServiceRoleForECRTemplateterkait layanan secara manual. Setelah peran terkait layanan dibuat, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi lebih lanjut, lihat Mengedit peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus Peran Terkait Layanan untuk Amazon ECR

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami menyarankan Anda menghapus peran tersebut. Dengan begitu Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Namun, Anda harus menghapus aturan pembuatan repositori untuk registri Anda di setiap Wilayah sebelum Anda dapat menghapus peran terkait layanan secara manual.

> **Note**
>
> Jika Anda mencoba menghapus sumber daya saat layanan Amazon ECR masih menggunakan peran tersebut, tindakan penghapusan Anda mungkin gagal. Jika hal tersebut terjadi, tunggu beberapa menit dan coba lagi.

Untuk menghapus sumber daya Amazon ECR yang digunakan oleh AWSServiceRoleForECRTemplate peran yang terhubung dengan layanan

1. Buka konsol Amazon ECR di https://console.aws.amazon.com/ecr/.
2. Dari bilah navigasi, pilih Wilayah tempat aturan pembuatan repositori Anda dibuat.
3. Di panel navigasi, pilih Registri pribadi.
4. Pada halaman registri pribadi, pada bagian Template pembuatan repositori, pilih Edit.

5. Untuk setiap aturan pembuatan repositori yang telah Anda buat, pilih aturan dan kemudian pilih Hapus aturan.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForECRTemplateterkait layanan. Untuk informasi selengkapnya, lihat Menghapus Service-Linked Peran di Panduan Pengguna IAM.

Pemecahan Masalah Identitas dan Akses Amazon Elastic Container Registry

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan mengatasi masalah umum yang mungkin Anda temui saat bekerja menggunakan Amazon ECR dan IAM.

Topik

- Saya tidak Berwenang untuk Melakukan Tindakan di Amazon ECR
- Saya Tidak Berwenang untuk Melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon ECR saya

Saya tidak Berwenang untuk Melakukan Tindakan di Amazon ECR

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `ecr:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `ecr:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya Tidak Berwenang untuk Melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon ECR.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi saat pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon ECR. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon ECR saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, lihat hal berikut:

- Untuk mempelajari apakah Amazon ECR support fitur ini, lihat Bagaimana Amazon Elastic Container Registry bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.

- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Perlindungan data dalam Amazon ECR

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon Elastic Container Service. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data AWS . Untuk informasi tentang perlindungan data di Eropa, lihat Pusat Peraturan Umum Perlindungan Data (GDPR).

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.

- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon ECS atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Topik

Enkripsi saat diam

> **Important**
>
> Dual-layer enkripsi sisi server dengan AWS KMS (DSSE-KMS) hanya tersedia di Wilayah. AWS GovCloud (US)

Amazon ECR menyimpan citra di bucket Amazon S3 yang dikelola Amazon ECR. Secara default, Amazon ECR menggunakan enkripsi sisi server dengan kunci enkripsi Amazon yang S3-managed mengenkripsi data Anda saat istirahat menggunakan algoritma enkripsi. AES-256 Ini tidak memerlukan tindakan apa pun dari bagian Anda dan ditawarkan tanpa biaya tambahan. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Server-Side S3-Managed Enkripsi dengan Kunci Enkripsi Amazon (SSE-S3) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk kontrol lebih lanjut atas enkripsi untuk repositori Amazon ECR Anda, Anda dapat menggunakan enkripsi sisi server dengan kunci KMS yang disimpan di (). AWS Key Management Service AWS KMS Saat Anda menggunakan AWS KMS untuk mengenkripsi data, Anda dapat menggunakan default Kunci yang dikelola AWS, yang dikelola oleh Amazon ECR, atau menentukan kunci KMS Anda sendiri (disebut sebagai kunci yang dikelola pelanggan). Untuk informasi

selengkapnya, lihat Melindungi Data Menggunakan Server-Side Enkripsi dengan kunci KMS yang Disimpan di AWS KMS (SSE-KMS) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Anda dapat memilih untuk menerapkan dua lapisan enkripsi ke gambar Amazon ECR Anda menggunakan enkripsi sisi server dual-layer dengan (). AWS KMS DSSE-KMS DSSE-KMSopsi mirip denganSSE-KMS, tetapi menerapkan dua lapisan enkripsi individu, bukan satu lapisan. Untuk informasi selengkapnya, lihat Menggunakan enkripsi sisi server dua lapis dengan kunci () . AWS KMS DSSE-KMS

Setiap repositori Amazon ECR memiliki konfigurasi enkripsi, yang diatur saat repositori tersebut dibuat. Anda dapat menggunakan konfigurasi enkripsi yang berbeda pada setiap repositori. Untuk informasi selengkapnya, lihat Membuat repositori pribadi Amazon ECR untuk menyimpan gambar.

Ketika repositori dibuat dengan AWS KMS enkripsi diaktifkan, kunci KMS digunakan untuk mengenkripsi isi repositori. Selain itu, Amazon ECR menambahkan AWS KMS hibah ke kunci KMS dengan repositori Amazon ECR sebagai pokok penerima hibah.

Berikut ini memberikan pemahaman tingkat tinggi tentang bagaimana Amazon ECR terintegrasi dengan AWS KMS untuk mengenkripsi dan mendekripsi repositori Anda:

1. Saat membuat repositori, Amazon ECR mengirimkan DescribeKeypanggilan AWS KMS untuk memvalidasi dan mengambil Nama Sumber Daya Amazon (ARN) dari kunci KMS yang ditentukan dalam konfigurasi enkripsi.
2. Amazon ECR mengirimkan dua CreateGrantpermintaan AWS KMS untuk membuat hibah pada kunci KMS untuk memungkinkan Amazon ECR mengenkripsi dan mendekripsi data menggunakan kunci data.
3. Saat mendorong gambar, GenerateDataKeypermintaan dibuat untuk AWS KMS yang menentukan kunci KMS yang akan digunakan untuk mengenkripsi lapisan gambar dan manifes.
4. AWS KMS menghasilkan kunci data baru, mengenkripsi di bawah kunci KMS yang ditentukan, dan mengirimkan kunci data terenkripsi untuk disimpan dengan metadata lapisan gambar dan manifes gambar.
5. Saat menarik gambar, permintaan Dekripsi dibuat untuk AWS KMS, menentukan kunci data terenkripsi.
6. AWS KMS mendekripsi kunci data terenkripsi dan mengirimkan kunci data yang didekripsi ke Amazon S3.
7. Kunci data digunakan untuk mendekripsi layer gambar sebelum layer gambar ditarik.

8. Ketika repositori dihapus, Amazon ECR mengirimkan dua RetireGrantpermintaan untuk menghentikan hibah yang AWS KMS dibuat untuk repositori.

Pertimbangan-pertimbangan

Poin-poin berikut harus dipertimbangkan saat menggunakan enkripsi AWS KMS berbasis (SSE-KMSatauDSSE-KMS) dengan Amazon ECR.

- Jika Anda membuat repositori Amazon ECR Anda dengan enkripsi KMS dan Anda tidak menentukan kunci KMS, Amazon ECR menggunakan Kunci yang dikelola AWS dengan alias secara default. `aws/ecr` Kunci KMS ini dibuat di akun Anda saat pertama kali Anda membuat repositori dengan enkripsi KMS yang diaktifkan.
- Konfigurasi Enkripsi Repositori tidak dapat diubah setelah repositori dibuat.
- Saat Anda menggunakan enkripsi KMS dengan kunci KMS Anda sendiri, kunci harus ada di wilayah yang sama sebagai repositori Anda.
- Bantuan yang dibuat Amazon ECR atas nama Anda tidak harus dicabut. Jika Anda mencabut hibah yang memberikan izin Amazon ECR untuk menggunakan AWS KMS kunci di akun Anda, Amazon ECR tidak dapat mengakses data ini, mengenkripsi gambar baru yang didorong ke repositori, atau mendekripsi ketika ditarik. Ketika Anda mencabut bantuan untuk Amazon ECR, perubahan terjadi segera. Untuk mencabut hak akses, Anda harus menghapus repositori bukan mencabut bantuan. Ketika repositori dihapus, Amazon ECR mempensiunkan (retire) bantuan atas nama Anda.
- Ada biaya yang terkait dengan penggunaan AWS KMS kunci. Untuk informasi selengkapnya, lihat harga AWS Key Management Service.
- Ada biaya yang terkait dengan penggunaan enkripsi sisi server dual-layer. Untuk informasi selengkapnya, lihat harga Amazon ECR

Izin IAM yang Diperlukan

Saat membuat atau menghapus repositori Amazon ECR dengan enkripsi server-side menggunakan AWS KMS, izin yang diperlukan tergantung pada kunci KMS tertentu yang Anda gunakan.

Izin IAM yang diperlukan saat menggunakan Kunci yang dikelola AWS untuk Amazon ECR

Secara default, ketika AWS KMS enkripsi diaktifkan untuk repositori Amazon ECR tetapi tidak ada kunci KMS yang ditentukan, ECR untuk Kunci yang dikelola AWS Amazon digunakan. Ketika kunci KMS AWS-managed untuk Amazon ECR digunakan untuk mengenkripsi repositori, setiap prinsipal

yang memiliki izin untuk membuat repositori juga dapat mengaktifkan enkripsi pada repositori. AWS KMS Namun, IAM principal yang menghapus repositori harus memiliki izin `kms:RetireGrant`. Hal ini memungkinkan pensiun dari hibah yang ditambahkan ke AWS KMS kunci ketika repositori dibuat.

Contoh kebijakan IAM berikut dapat ditambahkan sebagai kebijakan inline untuk pengguna guna memastikan bahwa mereka memiliki izin minimum yang diperlukan untuk menghapus repositori yang memiliki enkripsi yang diaktifkan. Kunci KMS yang digunakan untuk mengenkripsi repositori dapat ditentukan menggunakan parameter sumber daya.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-kms-permissions",
    "Statement": [
        {
            "Sid": "AllowAccessToRetireTheGrantsAssociatedWithTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:RetireGrant"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Izin IAM yang diperlukan saat menggunakan kunci terkelola pelanggan

Saat membuat repositori dengan AWS KMS enkripsi diaktifkan menggunakan kunci yang dikelola pelanggan, ada izin yang diperlukan untuk kebijakan kunci KMS dan kebijakan IAM untuk pengguna atau peran yang membuat repositori.

Saat membuat kunci KMS Anda sendiri, Anda dapat menggunakan kebijakan kunci default AWS KMS membuatnya atau Anda dapat menentukan sendiri. Untuk memastikan bahwa kunci yang dikelola pelanggan tetap dapat dikelola oleh pemilik akun, kebijakan kunci untuk kunci KMS harus memungkinkan semua AWS KMS tindakan untuk pengguna root akun. Izin cakupan tambahan dapat ditambahkan ke kebijakan kunci tetapi pengguna root paling tidak harus diberikan izin untuk mengelola kunci KMS. Untuk mengizinkan kunci KMS digunakan hanya untuk permintaan yang

berasal dari Amazon ECR, Anda dapat menggunakan kunci ViaService kondisi kms: dengan nilainya. `ecr.<region>.amazonaws.com`

Contoh kebijakan kunci berikut memberikan AWS akun (pengguna root) yang memiliki kunci KMS akses penuh ke kunci KMS. Untuk informasi selengkapnya tentang kebijakan kunci contoh ini, lihat Mengizinkan akses ke AWS akun dan mengaktifkan kebijakan IAM di Panduan AWS Key Management Service Pengembang.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

Pengguna IAM, peran IAM, atau AWS akun yang membuat repositori Anda harus memilikikms:CreateGrant,kms:RetireGrant, dan kms:DescribeKey izin selain izin Amazon ECR yang diperlukan.

> **Note**
>
> Izin kms:RetireGrant harus ditambahkan ke kebijakan IAM untuk pengguna atau peran yang membuat repositori. Izin kms:CreateGrant dan kms:DescribeKey dapat ditambahkan ke kebijakan kunci untuk kunci KMS atau kebijakan IAM untuk pengguna atau peran yang membuat repositori. Untuk informasi selengkapnya tentang cara kerja AWS KMS izin, lihat Izin AWS KMS API: Referensi tindakan dan sumber daya di Panduan AWS Key Management Service Pengembang.

Contoh kebijakan IAM berikut dapat ditambahkan sebagai kebijakan inline untuk pengguna guna memastikan bahwa mereka memiliki izin minimum yang diperlukan untuk membuat repositori dengan enkripsi yang diaktifkan dan menghapus repositori ketika mereka selesai. Yang AWS KMS key digunakan untuk mengenkripsi repositori dapat ditentukan menggunakan parameter sumber daya.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-kms-permissions",
    "Statement": [
        {
            "Sid":
 "AllowAccessToCreateAndRetireTheGrantsAssociatedWithTheKeyAsWellAsDescribeTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:RetireGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Izinkan pengguna mencantumkan kunci KMS di konsol saat membuat repositori

Bila menggunakan konsol Amazon ECR untuk membuat repositori, Anda dapat memberikan izin untuk mengizinkan pengguna memasukkan kunci KMS yang dikelola pelanggan di Wilayah saat mengaktifkan enkripsi untuk repositori. Contoh kebijakan IAM berikut menunjukkan izin yang diperlukan untuk memasukkan kunci KMS dan alias saat menggunakan konsol.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
```

```
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
}
```

Memantau interaksi Amazon ECR dengan AWS KMS

Anda dapat menggunakan AWS CloudTrail untuk melacak permintaan yang dikirimkan Amazon ECR atas nama Anda. AWS KMS Entri log di CloudTrail log berisi kunci konteks enkripsi untuk membuatnya lebih mudah diidentifikasi.

Konteks enkripsi Amazon ECR

Konteks enkripsi adalah seperangkat pasangan nilai kunci yang berisi data non-rahasia yang berubah-ubah. Ketika Anda menyertakan konteks enkripsi dalam permintaan untuk mengenkripsi data, secara AWS KMS kriptografis mengikat konteks enkripsi ke data terenkripsi. Untuk mendekripsi data, Anda harus meneruskan konteks enkripsi yang sama.

Dalam permintaannya GenerateDataKeydan Dekripsi ke, AWS KMS Amazon ECR menggunakan konteks enkripsi dengan dua pasangan nama-nilai yang mengidentifikasi repositori dan bucket Amazon S3 yang digunakan. Ini seperti yang ditunjukkan dalam contoh berikut. Nama-nama tidak bervariasi, tetapi nilai-nilai konteks enkripsi gabungan akan berbeda untuk setiap nilai.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/
sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df",
    "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
}
```

Anda dapat menggunakan konteks enkripsi untuk mengidentifikasi operasi kriptografi ini dalam catatan audit dan log, seperti AWS CloudTraildan Amazon CloudWatch Logs, dan sebagai syarat untuk otorisasi dalam kebijakan dan hibah.

Enkripsi konteks Amazon ECR terdiri dari dua pasangan nama-nilai.

- aws:s3:arn – Pasangan nama-nilai pertama mengidentifikasi bucket. Kuncinya adalah `aws:s3:arn`. Nilai tersebut adalah Amazon Resource Name (ARN) dari bucket Amazon S3.

  ```
  "aws:s3:arn": "ARN of an Amazon S3 bucket"
  ```

  Sebagai contoh, jika ARN bucket adalah `arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`, mkaa konteks enkripsi akan mencakup pasangan berikut.

  ```
  "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-
  ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
  ```

- aws:ecr:arn – Pasangan nama-nilai kedua mengidentifikasi Amazon Resource Name (ARN) dari repositori. Kuncinya adalah `aws:ecr:arn`. Nilai tersebut merupakan ARN dari repositori.

  ```
  "aws:ecr:arn": "ARN of an Amazon ECR repository"
  ```

  Misalnya, jika ARN repositori adalah `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`, maka konteks enkripsi akan mencakup pasangan berikut.

  ```
  "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
  ```

Pemecahan masalah

Ketika menghapus repositori Amazon ECR dengan konsol, jika repositori berhasil dihapus tetapi Amazon ECR tidak dapat mempensiunkan (retire) bantuan yang ditambahkan ke kunci KMS Anda untuk repositori Anda, maka Anda akan menerima pesan kesalahan berikut.

```
The repository [{repository-name}] has been deleted successfully but the grants created
 by the kmsKey [{kms_key}] failed to be retired
```

Ketika ini terjadi, Anda dapat mempensiunkan AWS KMS hibah untuk repositori sendiri.

Untuk pensiun AWS KMS hibah untuk repositori secara manual

1. Buat daftar hibah untuk AWS KMS kunci yang digunakan untuk repositori. Nilai `key-id` termasuk dalam pesan kesalahan yang Anda terima dari konsol. Anda juga dapat menggunakan `list-keys` perintah untuk mencantumkan kunci KMS Kunci yang dikelola AWS dan yang dikelola pelanggan di Wilayah tertentu di akun Anda.

   ```
   aws kms list-grants \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
        --region us-west-2
   ```

   Outputnya termasuk `EncryptionContextSubset` dengan Amazon Resource Name (ARN) dari repositori Anda. Ini dapat digunakan untuk menentukan bantuan yang ditambahkan ke kunci mana yang ingin Anda pensiunkan (retire). Nilai `GrantId` akan digunakan saat mempensiunkan (retire) bantuan pada langkah berikutnya.

2. Pensiun setiap hibah untuk AWS KMS kunci yang ditambahkan untuk repositori. Ganti nilai untuk *`GrantId`* dengan ID hibah dari output dari langkah sebelumnya.

   ```
   aws kms retire-grant \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
        --grant-id GrantId \
        --region us-west-2
   ```

Validasi kepatuhan Amazon Elastic Container Registry

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Keamanan Infrastruktur di Amazon Elastic Container Registry

Sebagai layanan terkelola, Amazon Elastic Container Registry dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Amazon ECR melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Anda dapat menghubungi operasi API ini dari lokasi jaringan mana pun, tetapi Amazon ECR support kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan kebijakan Amazon ECR untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) atau VPC tertentu. Secara efektif, ini mengisolasi akses jaringan ke sumber daya Amazon ECR tertentu hanya dari VPC tertentu dalam jaringan. AWS Untuk informasi selengkapnya, lihat Titik akhir VPC antarmuka Amazon ECR (AWS PrivateLink).

Titik akhir VPC antarmuka Amazon ECR (AWS PrivateLink)

Anda dapat meningkatkan postur keamanan VPC Anda dengan mengonfigurasi Amazon ECR untuk menggunakan antarmuka VPC endpoint. Endpoint VPC didukung oleh AWS PrivateLink, sebuah teknologi yang memungkinkan Anda mengakses Amazon ECR API secara pribadi melalui alamat IP pribadi (IPv4 dan IPv6). AWS PrivateLink membatasi semua lalu lintas jaringan antara VPC Anda dan Amazon ECR ke jaringan Amazon. Anda tidak memerlukan sebuah gateway internet, perangkat NAT, atau gateway privat virtual.

Untuk informasi selengkapnya tentang AWS PrivateLink dan titik akhir VPC, lihat Titik Akhir VPC di Panduan Pengguna Amazon VPC.

Pertimbangan untuk VPC endpoint Amazon ECR

Sebelum Anda mengkonfigurasi VPC endpoint untuk Amazon ECR, perhatikan pertimbangan berikut.

- Untuk mengizinkan tugas Amazon ECS Anda yang dihosting di instans Amazon EC2 menarik gambar pribadi dari Amazon ECR, buat titik akhir VPC antarmuka untuk Amazon ECS. Untuk informasi selengkapnya, lihat Titik Akhir VPC Antarmuka (AWS PrivateLink) di Panduan Pengembang Layanan Amazon Elastic Container.
- Tugas Amazon ECS yang dihosting di Fargate yang menarik gambar kontainer dari Amazon ECR dapat membatasi akses ke VPC tertentu yang digunakan tugas mereka dan ke titik akhir VPC yang digunakan layanan dengan menambahkan kunci kondisi ke peran IAM eksekusi tugas untuk tugas tersebut. Untuk informasi selengkapnya, lihat Izin IAM Opsional untuk Tugas Fargate dalam Menarik Citra Amazon ECR melalui Antarmuka Titik Akhir dalam Panduan Developer Amazon Elastic Container Service.
- Grup keamanan yang dilampirkan ke VPC endpoint harus mengizinkan koneksi masuk pada port 443 dari subnet privat VPC.
- Titik akhir Amazon ECR VPC mendukung konektivitas dual-stack (IPv4 dan IPv6). Saat Anda membuat titik akhir VPC dual-stack, itu dapat menangani lalu lintas melalui alamat IP pribadi IPv4 dan IPv6.
- Titik akhir VPC mendukung repositori Publik Amazon ECR melalui titik akhir SDK AWS API di US East (Virginia N.).
- Titik akhir VPC hanya mendukung DNS yang AWS disediakan melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat Pengaturan DHCP dalam Panduan Pengguna Amazon VPC.
- Jika kontainer Anda memiliki koneksi yang tersedia ke Amazon S3, koneksi mereka mungkin akan terganggu sebentar ketika Anda menambahkan titik akhir gateway Amazon S3. Jika Anda ingin menghindari gangguan ini, buatlah VPC baru yang menggunakan titik akhir gateway Amazon S3 dan kemudian migrasikan klaster Amazon ECS dan kontainer ke VPC baru.
- Saat gambar ditarik menggunakan aturan cache tarik untuk pertama kalinya, jika Anda telah mengonfigurasi Amazon ECR untuk AWS PrivateLink menggunakan titik akhir VPC antarmuka, maka Anda perlu membuat subnet publik di VPC yang sama, dengan gateway NAT, dan kemudian merutekan semua lalu lintas keluar ke internet dari subnet pribadinya ke gateway NAT agar tarikan berfungsi. Penarikan gambar berikutnya tidak memerlukan ini. Untuk informasi selengkapnya, lihat Skenario: Mengakses internet dari subnet pribadi di Panduan Pengguna Amazon Virtual Private Cloud.
- Untuk beban kerja yang membutuhkan modul kriptografi tervalidasi FIPS 140-3, Amazon ECR mendukung titik akhir FIPS untuk titik akhir VPC.

Pertimbangan untuk citra Windows

Citra yang didasarkan pada sistem operasi Windows termasuk artefak yang distribusinya dibatasi oleh lisensi. Secara default, ketika Anda mendorong citra Windows ke repositori Amazon ECR, lapisan yang menyertakan artefak ini tidak didorong karena mereka dianggap sebagai lapisan asing. Ketika artefak disediakan oleh Microsoft, lapisan asing diambil dari infrastruktur Microsoft Azure. Dengan demikian, untuk mengaktifkan kontainer Anda untuk menarik lapisan asing ini dari Azure, langkah-langkah tambahan selain membuat VPC endpoint diperlukan.

Hal ini dimungkinkan untuk menimpa perilaku ini ketika mendorong citra Windows ke Amazon ECR dengan menggunakan tanda `--allow-nondistributable-artifacts` di Docker daemon. Bila diaktifkan, tanda ini akan mendorong lapisan berlisensi untuk Amazon ECR yang mengizinkan citra ini ditarik dari Amazon ECR melalui VPC endpoint tanpa memerlukan akses tambahan ke Azure.

> **Important**
>
> Menggunakan tanda `--allow-nondistributable-artifacts` tidak menghalangi kewajiban Anda untuk mematuhi persyaratan lisensi citra berbasis kontainer Windows; Anda tidak dapat mengirim konten Windows untuk redistribusi publik atau pihak ke tiga. Penggunaan dalam lingkungan Anda sendiri diperbolehkan.

Untuk mengaktifkan penggunaan tanda ini untuk instalasi Docker, Anda harus memodifikasi file konfigurasi Docker daemon yang, tergantung pada instalasi Docker Anda, biasanya dapat dikonfigurasi di pengaturan atau menu preferensi pada bagian Mesin Docker atau dengan mengedit file `C:\ProgramData\docker\config\daemon.json` secara langsung.

Berikut ini adalah contoh konfigurasi yang diperlukan. Ganti nilai dengan repositori URI yang Anda gunakan untuk mendorong citra.

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
    ]
}
```

Setelah memodifikasi file konfigurasi Docker daemon, Anda harus memulai ulang Docker daemon sebelum mencoba mendorong citra Anda. Konfirmasikan dorongan yang dikerjakan dengan memverifikasi bahwa lapisan dasar didorong ke repositori Anda.

> **Note**
>
> Lapisan dasar untuk citra Windows berukuran besar. Ukuran lapisan akan menyebabkan waktu mendorong menjadi lebih lama dan biaya penyimpanan tambahan di Amazon ECR untuk citra ini akan dibebankan. Dengan demikian, sebaiknya hanya menggunakan opsi ini bila diperlukan jika benar-benar diperlukan untuk mengurangi waktu pembuatan dan biaya penyimpanan yang sedang berlangsung. Misalnya, citra `mcr.microsoft.com/windows/servercore` berukuran sekitar 1,7 GiB ketika dikompresi di Amazon ECR.

Buat VPC endpoint untuk Amazon ECR

Untuk membuat VPC endpoint untuk layanan Amazon ECR, gunakan prosedurMembuat Antarmuka Titik Akhir dalam Panduan Pengguna Amazon VPC.

Titik akhir Amazon ECR VPC mendukung konektivitas dual-stack (IPv4 dan IPv6). Saat Anda membuat titik akhir VPC dual-stack, secara otomatis menangani lalu lintas melalui alamat IP pribadi IPv4 dan IPv6. Titik akhir akan merutekan lalu lintas menggunakan versi IP yang sesuai berdasarkan konfigurasi jaringan klien Anda dan kemampuan titik akhir.

Jika Anda memiliki titik akhir IPv4-only VPC yang ada dan ingin bermigrasi ke titik akhir dual-stack, Anda dapat memodifikasi titik akhir yang ada untuk mendukung konektivitas dual-stack, atau membuat titik akhir dual-stack baru. Saat membuat atau memodifikasi titik akhir, pastikan VPC dan subnet Anda mendukung versi IP yang ingin Anda gunakan. Setelah membuat endpoint dual-stack, endpoint akan mendukung IPv4 dan IPv6.

Tugas Amazon ECS yang dihosting di instans Amazon EC2 memerlukan titik akhir Amazon ECR dan titik akhir gateway Amazon S3.

Tugas Amazon ECS yang dihosting di Fargate menggunakan `1.4.0` versi platform atau yang lebih baru memerlukan titik akhir Amazon ECR VPC dan titik akhir gateway Amazon S3.

Tugas Amazon ECS yang dihosting di Fargate yang menggunakan **`1.3.0`** versi platform atau sebelumnya hanya memerlukan com.amazonaws. ***`region`***.ecr.dkr Titik akhir Amazon ECR VPC dan titik akhir gateway Amazon S3.

> **Note**
>
> Urutan titik akhir boleh dibuat di dalam.

com.amazonaws. ***region***.ecr.dkr

titik akhir ini digunakan untuk Docker Registry API. Perintah klien Docker seperti `push` dan `pull` mengunakan titik akhir ini.

Saat Anda membuat titik akhir ini, Anda harus mengaktifkan nama host DNS pribadi. Untuk melakukan ini, pastikan opsi Aktifkan Nama DNS Pribadi dipilih di konsol VPC Amazon saat Anda membuat titik akhir VPC.

Untuk koneksi yang sesuai dengan FIPS 140-3, gunakan nama titik akhir FIPS com.amazonaws. ***region***.ecr-fips.dkr

com.amazonaws. ***region***.ecr.api

> **Note**
>
> Yang ditentukan *region* mewakili pengenal Wilayah untuk AWS Wilayah yang didukung oleh Amazon ECR, seperti `us-east-2` untuk Wilayah AS Timur (Ohio).

Untuk koneksi yang sesuai dengan FIPS 140-3, gunakan nama titik akhir FIPS: com.amazonaws. *region*.ecr-fips.dkr dan com.amazonaws. ***region***.ecr-fips.api.

titik akhir ini digunakan untuk panggilan ke API Amazon ECR. Tindakan API seperti `DescribeImages` dan `CreateRepository` masuk ke titik akhir ini.

Ketika titik akhir ini dibuat, Anda memiliki opsi untuk mengaktifkan nama host DNS pribadi. Aktifkan pengaturan ini dengan memilih Aktifkan Nama DNS privat di konsol VPC saat Anda membuat VPC endpoint. Jika Anda mengaktifkan nama host DNS pribadi untuk titik akhir VPC, perbarui SDK Anda atau AWS CLI ke versi terbaru sehingga menentukan URL titik akhir saat menggunakan SDK atau tidak diperlukan. AWS CLI

Untuk koneksi yang sesuai dengan FIPS 140-3, gunakan nama titik akhir FIPS com.amazonaws. ***region***.ecr-fips.api.

Jika Anda mengaktifkan nama host DNS pribadi dan menggunakan SDK atau AWS CLI versi yang dirilis sebelum 24 Januari 2019, Anda harus menggunakan `--endpoint-url` parameter untuk menentukan titik akhir antarmuka. Contoh berikut menunjukkan format untuk titik akhir URL.

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

Jika Anda tidak mengaktifkan nama host DNS privat untuk VPC endpoint, Anda harus menggunakan parameter `--endpoint-url` yang menentukan ID VPC endpoint untuk titik akhir antarmuka. Contoh berikut menunjukkan format untuk titik akhir URL.

```
aws ecr create-repository --repository-name name --endpoint-url
 https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

Untuk koneksi yang sesuai dengan FIPS 140-3, gunakan URL titik akhir FIPS:

```
aws ecr create-repository --repository-name name --endpoint-url https://api.ecr-
fips.region.amazonaws.com
```

Buat titik akhir gateway Amazon S3

Untuk tugas-tugas Amazon ECS Anda yang berfungsi untuk menarik citra privat dari Amazon ECR, Anda harus membuat titik akhir gateway untuk Amazon S3. titik akhir gateway diperlukan karena Amazon ECR menggunakan Amazon S3 untuk menyimpan lapisan citra Anda. Ketika kontainer Anda mengunduh citra dari Amazon ECR, mereka harus mengakses Amazon ECR untuk mendapatkan manifest citra dan kemudian Amazon S3 untuk mengunduh lapisan citra yang sebenarnya. Berikut adalah Amazon Resource Name (ARN) dari bucket Amazon S3 yang berisi lapisan untuk setiap citra Docker.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

> **Note**
>
> Jika membuat titik akhir VPC dual-stack untuk Amazon ECR, Anda juga perlu membuat titik akhir Amazon S3 Gateway atau Interface dual-stack. Lihat dokumentasi S3 untuk detailnya.

Menggunakan prosedur Membuat titik akhir gateway dalam Panduan Pengguna Amazon VPC untuk membuat titik akhir gateway Amazon S3 berikut untuk Amazon ECR. Saat membuat titik akhir, pastikan untuk memilih tabel rute untuk VPC Anda.

com.amazonaws. ***region***.s3

titik akhir gateway Amazon S3 menggunakan dokumen kebijakan IAM untuk membatasi akses ke layanan. Kebijakan Akses penuh dapat digunakan karena pembatasan yang telah Anda

masukkan ke dalam tugas IAM role Anda atau kebijakan pengguna IAM lainnya yang masih berlaku di atas kebijakan ini. Jika Anda ingin membatasi akses bucket Amazon S3 ke izin minimum yang diperlukan untuk menggunakan Amazon ECR, lihat Izin Bucket Amazon S3 Minimum untuk Amazon ECR.

Izin Bucket Amazon S3 Minimum untuk Amazon ECR

titik akhir gateway Amazon S3 menggunakan dokumen kebijakan IAM untuk membatasi akses ke layanan. Untuk mengizinkan hanya izin minimum bucket Amazon S3 untuk Amazon ECR, maka batasi akses ke bucket Amazon S3 yang menggunakan Amazon ECR ketika Anda membuat dokumen kebijakan IAM untuk titik akhir.

Tabel berikut menjelaskan izin kebijakan bucket Amazon S3 yang dibutuhkan oleh Amazon ECR.

Izin	Deskripsi
`arn:aws:s3:::prod-`*`region`*`-starport-layer-bucket/*`	Menyediakan akses ke bucket Amazon S3 yang berisi lapisan untuk setiap citra Docker. Merepresentasikan identifier Wilayah untuk Wilayah AWS yang didukung oleh Amazon ECR, seperti `us-east-2` untuk (US East (Ohio).

Contoh

Contoh berikut menggambarkan bagaimana memberikan akses ke bucket Amazon S3 yang diperlukan untuk operasi Amazon ECR.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
```

```
  ]
}
```

Buat titik akhir CloudWatch Log

Tugas Amazon ECS menggunakan tipe peluncuran Fargate yang menggunakan VPC tanpa gateway internet yang juga menggunakan **`awslogs`** driver log untuk mengirim informasi log ke Log mengharuskan Anda membuat CloudWatch com.amazonaws. ***`region`***.logs antarmuka VPC CloudWatch endpoint untuk Log. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan titik akhir VPC antarmuka di Panduan Pengguna Amazon CloudWatch Logs.

Buat kebijakan titik akhir untuk VPC endpoint Amazon ECR

Kebijakan VPC endpoint adalah kebijakan sumber daya IAM yang Anda lampirkan ke titik akhir ketika membuat atau memodifikasi titik akhir. Jika Anda tidak melampirkan kebijakan saat membuat titik akhir, AWS lampirkan kebijakan default untuk Anda yang memungkinkan akses penuh ke layanan. Kebijakan titik akhir tidak membatalkan atau mengganti kebijakan pengguna IAM atau kebijakan khusus layanan. Ini adalah kebijakan terpisah untuk mengendalikan akses dari titik akhir ke layanan tertentu. Kebijakan titik akhir harus ditulis dalam format JSON. Untuk informasi selengkapnya, lihat Pengontrolan Akses ke Layanan dengan VPC endpoint dalam Panduan Pengguna Amazon VPC.

Kami merekomendasikan agar Anda membuat kebijakan sumber daya IAM tunggal dan melampirkannya ke kedua VPC endpoint Amazon ECR.

Berikut adalah contoh kebijakan titik akhir untuk Amazon ECR. Kebijakan ini mengizinkan IAM role tertentu untuk menarik citra dari Amazon ECR.

```
{
 "Statement": [{
  "Sid": "AllowPull",
  "Principal": {
   "AWS": "arn:aws:iam::1234567890:role/role_name"
  },
  "Action": [
   "ecr:BatchGetImage",
   "ecr:GetDownloadUrlForLayer",
                  "ecr:GetAuthorizationToken"
  ],
  "Effect": "Allow",
  "Resource": "*"
 }]
```

```
}
```

Contoh kebijakan titik akhir berikut mencegah penghapusan repositori tertentu.

```
{
 "Statement": [{
   "Sid": "AllowAll",
   "Principal": "*",
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*"
  },
  {
   "Sid": "PreventDelete",
   "Principal": "*",
   "Action": "ecr:DeleteRepository",
   "Effect": "Deny",
   "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  }
 ]
}
```

Contoh kebijakan titik akhir berikut menggabungkan dua contoh sebelumnya ke kebijakan tunggal.

```
{
 "Statement": [{
   "Sid": "AllowAll",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "*",
   "Resource": "*"
  },
  {
   "Sid": "PreventDelete",
   "Effect": "Deny",
   "Principal": "*",
   "Action": "ecr:DeleteRepository",
   "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  },
  {
   "Sid": "AllowPull",
   "Effect": "Allow",
   "Principal": {
```

```
    "AWS": "arn:aws:iam::1234567890:role/role_name"
   },
   "Action": [
    "ecr:BatchGetImage",
    "ecr:GetDownloadUrlForLayer",
                         "ecr:GetAuthorizationToken"
   ],
   "Resource": "*"
  }
 ]
}
```

Untuk mengubah kebijakan VPC endpoint untuk Amazon ECR

1. Buka konsol Amazon VPC di. https://console.aws.amazon.com/vpc/
2. Di panel navigasi, pilih Titik akhir.
3. Jika Anda belum membuat VPC endpoint untuk Amazon ECR, lihat Buat VPC endpoint untuk Amazon ECR.
4. Pilih VPC endpoint Amazon ECR untuk menambahkan kebijakan, dan pilih tab Kebijakan di bagian bawah layar.
5. Pilih Edit Kebijakan dan buat perubahan pada kebijakan.
6. Pilih Terapkan untuk menyimpan kebijakan tersebut.

Subnet bersama

Anda tidak dapat membuat, mendeskripsikan, memodifikasi, atau menghapus titik akhir VPC di subnet yang dibagikan dengan Anda. Namun, Anda dapat menggunakan titik akhir VPC di subnet yang dibagikan dengan Anda.

Cross-service pencegahan wakil bingung

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Cross-service peniruan identitas dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (layanan yang disebut). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara

yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data untuk semua layanan dengan principal layanan yang telah diberi akses ke sumber daya di akun Anda.

Sebaiknya gunakan `aws:SourceArn`atau kunci konteks kondisi `aws:SourceAccount`global dalam kebijakan sumber daya untuk membatasi izin yang diberikan Amazon ECR layanan lain ke sumber daya. Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Cara paling efektif untuk melindungi dari masalah "confused deputy" adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks global `aws:SourceArn` dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*`.

Jika nilai `aws:SourceArn` tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci konteks kondisi global tersebut untuk membatasi izin.

Nilai `aws:SourceArn` harus ResourceDescription.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan global dalam kebijakan repositori Amazon ECR untuk mengizinkan AWS CodeBuild akses ke tindakan Amazon ECR API yang diperlukan untuk integrasi dengan layanan tersebut sekaligus mencegah masalah deputi yang membingungkan.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
```

```
            ],
            "Resource": "*",
            "Condition":{
               "ArnLike":{
                  "aws:SourceArn":"arn:aws:codebuild:us-
east-1:123456789012:project/project-name"
               },
               "StringEquals":{
                  "aws:SourceAccount":"123456789012"
               }
            }
         }
      ]
}
```

Pemantauan Amazon ECR

Anda dapat memantau penggunaan API Amazon ECR Anda dengan Amazon CloudWatch, yang mengumpulkan dan memproses data mentah dari Amazon ECR menjadi metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini dicatat untuk jangka waktu dua minggu sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif tentang penggunaan API Anda. Data metrik Amazon ECR secara otomatis dikirim ke CloudWatch dalam periode satu menit. Untuk informasi selengkapnya CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

Amazon ECR menyediakan metrik berdasarkan penggunaan API Anda untuk tindakan otorisasi, dorongan citra, dan tarikan citra.

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon ECR dan AWS solusi Anda. Kami menyarankan Anda mengumpulkan data pemantauan dari sumber daya yang membentuk AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Namun sebelum Anda mulai memantau Amazon ECR, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan berikut:

- Apa sasaran pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah berikutnya adalah menetapkan baseline untuk kinerja Amazon ECR normal di lingkungan Anda, dengan mengukur kinerja di berbagai waktu dan dengan kondisi beban yang berbeda. Saat Anda memantau Amazon ECR, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja baru, mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Topik

- Metrik repositori Amazon ECR
- Acara Amazon ECR dan EventBridge
- Mencatat tindakan Amazon ECR dengan AWS CloudTrail

Memvisualisasikan kuota layanan Anda dan mengatur alarm

Anda dapat menggunakan CloudWatch konsol untuk memvisualisasikan kuota layanan Anda dan melihat bagaimana penggunaan Anda saat ini dibandingkan dengan kuota layanan. Anda juga dapat mengatur alarm agar diberi notifikasi ketika mendekati kuota.

Untuk memvisualisasikan kuota layanan dan secara opsional mengatur alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pada tab Semua metrik, pilih Penggunaan, lalu pilih Oleh Sumber Daya AWS .

   Daftar metrik penggunaan kuota layanan muncul.
4. Pilih kotak centang di samping salah satu metrik.

   Grafik menampilkan penggunaan AWS sumber daya Anda saat ini.
5. Untuk menambahkan kuota layanan Anda ke grafik, lakukan hal berikut:

   a. Pilih tab Metrik bergrafik.

   b. Pilih Pernyataan matematika, Mulai dengan pernyataan kosong. Lalu, di baris baru, di bawah Rincian, masukkan **`SERVICE_QUOTA(m1)`**.

      Baris baru ditambahkan ke grafik, menampilkan kuota layanan untuk sumber daya yang diwakili dalam metrik.
6. Untuk melihat penggunaan Anda saat ini sebagai persentase dari kuota, tambahkan pernyataan baru atau ubah pernyataan SERVICE_QUOTA saat ini. Untuk pernyataan baru, gunakan **`m1/60/SERVICE_QUOTA(m1)*100`**.
7. (Opsional) Untuk mengatur alarm yang memberi tahu Anda jika mendekati kuota layanan, lakukan hal berikut:

   a. Di **`m1/60/SERVICE_QUOTA(m1)*100`** baris, di bawah Tindakan, pilih ikon alarm. Terlihat seperti lonceng.

Halaman pembuatan alarm muncul.

b. Di bawah Ketentuan, pastikan bahwa Jenis ambang batas bersifat Statis dan Setiap kali Expression1 diatur menjadi Lebih Besar. Di bawah dari, masukkan **80**. Tindakan ini akan membuat alarm masuk ke status ALARM ketika penggunaan Anda melebihi 80 persen dari kuota.

c. Pilih Selanjutnya.

d. Di halaman berikutnya, pilih topik Amazon SNS atau buat topik baru. Topik ini akan diberi tahu ketika alarm masuk ke status ALARM. Lalu, pilih Selanjutnya.

e. Di halaman berikutnya, masukkan nama dan penjelasan untuk alarm, lalu pilih Selanjutnya.

f. Pilih Buat alarm.

Metrik penggunaan Amazon ECR

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

Metrik penggunaan Amazon ECR sesuai dengan kuota AWS layanan. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi lebih lanjut tentang kuota layanan Amazon ECR, lihat Kuota layanan Amazon ECR.

Amazon ECR menerbitkan metrik berikut di namespace `AWS/Usage`.

Metrik	Deskripsi
`CallCount`	Jumlah tindakan panggilan API dari akun Anda. Sumber daya ditentukan oleh dimensi yang terkait dengan metrik. Statistik yang paling berguna untuk metrik ini adalah `SUM`, yang mewakili jumlah nilai dari semua kontributor selama periode yang ditentukan.
`ResourceCount`	Jumlah sumber daya yang ditentukan di akun Anda. Sumber daya tersebut ditentukan oleh dimensi yang dikaitkan dengan metrik.

Metrik	Deskripsi
	Statistik yang paling berguna untuk metrik ini adalah`MAXIMUM`, yang mewakili jumlah maksimum sumber daya yang digunakan selama periode 5 menit.

Dimensi berikut digunakan untuk menyempurnakan metrik penggunaan API yang diterbitkan oleh Amazon ECR.

Dimensi	Deskripsi
`Service`	Nama AWS layanan yang berisi sumber daya. Untuk metrik penggunaan Amazon ECR, nilai untuk dimensi ini adalah `ECR`.
`Type`	Tipe entitas yang dilaporkan. Saat ini, satu-satunya nilai yang valid untuk metrik penggunaan Amazon ECR API adalah. `API`
`Resource`	Tipe sumber daya yang sedang berjalan. Saat ini, Amazon ECR mengembalikan informasi tentang penggunaan API Anda untuk tindakan API berikut. • `GetAuthorizationToken` • `BatchCheckLayerAvailability` • `InitiateLayerUpload` • `UploadLayerPart` • `CompleteLayerUpload` • `PutImage` • `BatchGetImage` • `GetDownloadUrlForLayer`
`Class`	Kelas sumber daya yang akan dilacak. Saat ini, Amazon ECR tidak menggunakan dimensi kelas.

Dimensi berikut digunakan untuk menyempurnakan metrik penggunaan Sumber Daya yang diterbitkan oleh Amazon ECR.

Dimensi	Deskripsi
`Service`	Nama AWS layanan yang berisi sumber daya. Untuk metrik penggunaan Amazon ECR, nilai untuk dimensi ini adalah `ECR`.
`Type`	Tipe entitas yang dilaporkan. Saat ini, satu-satunya nilai yang valid untuk metrik penggunaan sumber daya Amazon ECR adalah. `RESOURCE`
`Resource`	Tipe sumber daya yang sedang berjalan. Saat ini, Amazon ECR mengembalikan informasi tentang penggunaan sumber daya Anda untuk metrik berikut. • `RepositoryCount` • `ImagesPerRepositoryCount`
`ResourceId`	Identifier untuk sumber daya yang dikeluarkan penggunaa n. Saat ResourceId ini, hanya relevan dengan `ImagesPer RepositoryCount` dan nilainya diformat sebagai repositor y/your_repository_name. Misalnya: "repository/my-repo” mengembalikan jumlah gambar dalam repositori dengan nama “my-repo”.

Laporan penggunaan Amazon ECR

AWS menyediakan alat pelaporan gratis yang disebut Cost Explorer yang memungkinkan Anda menganalisis biaya dan penggunaan sumber daya Amazon ECR Anda.

Gunakan Cost Explorer untuk melihat bagan penggunaan dan biaya Anda. Anda dapat melihat data hingga 13 bulan terakhir, dan memperkirakan seberapa besar kemungkinan pengeluaran Anda untuk tiga bulan ke depan. Anda dapat menggunakan Cost Explorer untuk melihat pola pengeluaran sumber daya AWS Anda dari waktu ke waktu, mengidentifikasi area-area yang memerlukan penyelidikan lebih lanjut, dan melihat tren yang dapat Anda gunakan untuk memahami biaya Anda. Anda juga dapat menentukan rentang waktu untuk data, dan melihat data waktu berdasarkan hari atau bulan.

Data pengukuran dalam Laporan Biaya dan Penggunaan menunjukkan penggunaan di semua repositori Amazon ECR Anda. Untuk informasi selengkapnya, lihat Penandaan sumber daya Anda untuk penagihan.

Untuk informasi selengkapnya tentang membuat Laporan AWS Biaya dan Penggunaan, lihat Laporan AWS Biaya dan Penggunaan di Panduan AWS Billing Pengguna.

Metrik repositori Amazon ECR

Amazon ECR mengirimkan metrik jumlah tarik repositori ke Amazon. CloudWatch Data metrik Amazon ECR secara otomatis dikirim ke CloudWatch dalam periode 1 menit. Untuk informasi selengkapnya CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

Topik

Mengaktifkan metrik CloudWatch

Amazon ECR mengirimkan metrik repositori secara otomatis untuk semua repositori. Tidak perlu mengambil langkah manual apa pun.

Metrik dan dimensi yang tersedia

Bagian berikut mencantumkan metrik dan dimensi yang dikirimkan Amazon ECR ke Amazon. CloudWatch

Metrik Amazon ECR

Amazon ECR menyediakan metrik bagi Anda untuk memantau repositori Anda. Anda dapat mengukur jumlah tarik.

Namespace `AWS/ECR` mencakup metrik berikut.

`RepositoryPullCount`

Jumlah total tarikan untuk gambar di repositori.

Dimensi yang valid: `RepositoryName`.

Statistik yang valid: Rata-rata, Minimum, Maksimum, Jumlah, Jumlah Sampel. Statistik yang paling berguna adalah Sum.

Satuan: Bilangan bulat.

Dimensi untuk metrik Amazon ECR

Metrik Amazon ECR menggunakan `AWS/ECR` namespace dan menyediakan metrik untuk dimensi berikut.

`RepositoryName`

Dimensi ini menyaring data yang Anda minta untuk semua gambar kontainer dalam repositori tertentu.

Melihat metrik Amazon ECR menggunakan konsol CloudWatch

Anda dapat melihat metrik repositori Amazon ECR di konsol. CloudWatch CloudWatch Konsol menyediakan tampilan sumber daya Anda yang berbutir halus dan dapat disesuaikan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Acara Amazon ECR dan EventBridge

Amazon EventBridge memungkinkan Anda untuk mengotomatiskan AWS layanan Anda dan merespons secara otomatis peristiwa sistem seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke EventBridge dalam waktu dekat. Anda dapat menulis aturan sederhana untuk menunjukkan kejadian mana yang sesuai kepentingan Anda, dan memasukkan tindakan otomatis apa yang diambil ketika suatu kejadian sesuai dengan suatu aturan. Tindakan yang dapat dipicu secara otomatis meliputi hal-hal berikut:

- Menambahkan peristiwa ke grup log di CloudWatch Log
- Memanggil fungsi AWS Lambda
- Meminta Perintah Amazon EC2 Run
- Mengirim peristiwa ke Amazon Kinesis Data Streams

- Mengaktifkan mesin AWS Step Functions negara
- Memberi tahu topik Amazon SNS atau antrean Amazon SQS

Untuk informasi selengkapnya, lihat Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Contoh kejadian dari Amazon ECR

Berikut adalah contoh kejadian dari Amazon ECR. Kejadian dipancarkan atas dasar upaya terbaik.

Acara untuk push gambar yang lengkap

Kejadian berikut dikirim ketika setiap dorongan citra selesai. Untuk informasi selengkapnya, lihat Mendorong gambar Docker ke repositori pribadi Amazon ECR.

```
{
    "version": "0",
    "id": "13cde686-328b-6117-af20-0e5566167482",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-11-16T01:54:34Z",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "result": "SUCCESS",
        "repository-name": "my-repository-name",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "PUSH",
        "image-tag": "latest"
    }
}
```

Acara untuk aksi pull through cache

Peristiwa berikut dikirim ketika tindakan pull through cache dicoba. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR.

```
{
```

```
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Pull Through Cache Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2023-02-29T02:36:48Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecr:us-west-2:123456789012:repository/docker-hub/alpine"
    ],
    "detail": {
        "rule-version": "1",
        "sync-status": "SUCCESS",
        "ecr-repository-prefix": "docker-hub",
        "repository-name": "docker-hub/alpine",
        "upstream-registry-url": "public.ecr.aws",
        "image-tag": "3.17.2",
        "image-digest":
 "sha256:4aa08ef415aecc80814cb42fa41b658480779d80c77ab15EXAMPLE",
    }
}
```

Acara untuk pemindaian gambar selesai (pemindaian dasar)

Ketika pemindaian dasar diaktifkan untuk registri Anda, peristiwa berikut dikirim ketika setiap pemindaian gambar selesai. Parameter `finding-severity-counts` hanya akan mengembalikan nilai untuk suatu tingkat keparahan jika ada. Contohnya, jika citra tidak mengandung temuan di tingkat `CRITICAL`, maka tidak ada hitungan kritis yang dikembalikan. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan OS di Amazon ECR.

> **Note**
>
> Untuk detail tentang peristiwa yang dipancarkan Amazon Inspector saat pemindaian yang disempurnakan diaktifkan, lihat. EventBridge peristiwa dikirim untuk pemindaian yang ditingkatkan di Amazon ECR

```
{
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Image Scan",
```

```
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-10-29T02:36:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecr:us-east-1:123456789012:repository/my-repository-name"
    ],
    "detail": {
        "scan-status": "COMPLETE",
        "repository-name": "my-repository-name",
        "finding-severity-counts": {
        "CRITICAL": 10,
        "MEDIUM": 9
      },
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "image-tags": []
    }
}
```

Acara untuk pemberitahuan perubahan pada sumber daya dengan pemindaian yang disempurnakan diaktifkan (pemindaian yang disempurnakan)

Ketika pemindaian yang disempurnakan diaktifkan untuk registri Anda, peristiwa berikut dikirim oleh Amazon ECR ketika ada perubahan dengan sumber daya yang telah ditingkatkan pemindaian diaktifkan. Ini termasuk repositori baru yang sedang dibuat, frekuensi pemindaian untuk repositori yang diubah, atau ketika gambar dibuat atau dihapus di repositori dengan pemindaian yang ditingkatkan diaktifkan. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0ab8638973bf",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
```

```
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
   {
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

Acara untuk penghapusan gambar

Kejadian berikut dikirim ketika suatu citra dihapus. Untuk informasi selengkapnya, lihat Menghapus gambar di Amazon ECR.

```
{
    "version": "0",
    "id": "dd3b46cb-2c74-f49e-393b-28286b67279d",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-11-16T02:01:05Z",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "result": "SUCCESS",
        "repository-name": "my-repository-name",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "DELETE",
```

```
        "image-tag": "latest"
    }
}
```

Acara untuk aksi arsip gambar

Peristiwa berikut dikirim saat gambar diarsipkan. `target-storage-class`Bidang akan diatur ke`ARCHIVE`. Acara ini mencakup jenis media manifes dan artefak untuk mengidentifikasi jenis konten yang diarsipkan.

```
{
    "version": "0",
    "id": "4f5ec4d5-4de4-7aad-a046-EXAMPLE",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-08-06T00:58:09Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "action-type": "UPDATE_STORAGE_CLASS",
        "target-storage-class": "ARCHIVE",
        "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
        "repository-name": "ubuntu",
        "result": "SUCCESS",
        "manifest-media-type": "application/vnd.oci.image.manifest.v1+json",
        "artifact-media-type": "application/vnd.oci.image.config.v1+json"
    }
}
```

Acara untuk tindakan pemulihan gambar

Acara berikut dikirim ketika gambar yang diarsipkan dipulihkan. `target-storage-class`Bidang akan diatur ke`STANDARD`. Acara ini mencakup `last-activated-at` bidang yang menunjukkan kapan gambar terakhir dipulihkan.

```
{
    "version": "0",
    "id": "7b8fc5e6-5ef5-8bbe-b157-EXAMPLE",
    "detail-type": "ECR Image Action",
    "source": "aws.ecr",
```

```
    "account": "123456789012",
    "time": "2019-08-06T01:15:22Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "action-type": "UPDATE_STORAGE_CLASS",
        "target-storage-class": "STANDARD",
        "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
        "repository-name": "ubuntu",
        "result": "SUCCESS",
        "manifest-media-type": "application/vnd.oci.image.manifest.v1+json",
        "artifact-media-type": "application/vnd.oci.image.config.v1+json",
        "last-activated-at": "2025-10-10T19:13:02.74Z"
    }
}
```

Acara untuk tindakan pemulihan perujuk

Peristiwa berikut dikirim ketika perujuk yang diarsipkan (artefak referensi seperti SBOM, tanda tangan, atau pengesahan) dipulihkan. Perhatikan bahwa `detail-type` adalah `ECR Referrer Action` untuk membedakannya dari tindakan gambar biasa. `artifact-media-type`Bidang `manifest-media-type` dan mengidentifikasi jenis perujuk tertentu yang dipulihkan. Dalam contoh ini, artefak SBOM sedang dipulihkan.

```
{
    "version": "0",
    "id": "8c9gd6f7-6fg6-9ccf-c268-EXAMPLE",
    "detail-type": "ECR Referrer Action",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-08-06T01:20:45Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "action-type": "UPDATE_STORAGE_CLASS",
        "target-storage-class": "STANDARD",
        "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
        "repository-name": "sbom",
        "result": "SUCCESS",
        "manifest-media-type": "application/vnd.cncf.oras.artifact.manifest.v1+json",
        "artifact-media-type": "text/sbom+json",
```

```
        "last-activated-at": "2025-10-10T19:13:02.74Z"
    }
}
```

Acara untuk replikasi gambar yang lengkap

Peristiwa berikut dikirim ketika setiap replikasi gambar selesai. Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR.

```
{
  "version": "0",
  "id": "c8b133b1-6029-ee73-e2a1-4f466b8ba999",
  "detail-type": "ECR Replication Action",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2024-05-08T20:44:54Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecr:us-east-1:123456789012:repository/docker-hub/alpine"
  ],
  "detail": {
    "result": "SUCCESS",
    "repository-name": "docker-hub/alpine",
    "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
    "source-account": "123456789012",
    "action-type": "REPLICATE",
    "source-region": "us-west-2",
    "image-tag": "3.17.2"
  }
}
```

Mencatat tindakan Amazon ECR dengan AWS CloudTrail

Amazon ECR terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon ECR. CloudTrail menangkap tindakan Amazon ECR berikut sebagai peristiwa:

- Semua panggilan API, termasuk panggilan dari konsol Amazon ECR
- Semua tindakan yang diambil karena pengaturan enkripsi pada repositori Anda

- Semua tindakan yang diambil karena aturan kebijakan siklus hidup, termasuk tindakan yang berhasil dan tidak berhasil

> **Important**
>
> Karena keterbatasan ukuran CloudTrail peristiwa individual, untuk tindakan kebijakan siklus hidup di mana 10 gambar atau lebih kedaluwarsa Amazon ECR mengirimkan beberapa peristiwa ke. CloudTrail Selain itu, Amazon ECR menyertakan maksimum 100 tag per gambar.

Saat jejak dibuat, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk Amazon ECR. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi ini, Anda dapat menentukan permintaan yang dibuat ke Amazon ECR, alamat IP asal permintaan tersebut dibuat, siapa yang membuat permintaan, kapan permintaan dibuat, dan detail lainnya.

Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.

Informasi Amazon ECR di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di Amazon ECR, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Amazon ECR, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Bila Anda membuat jejak di konsol, Anda dapat menerapkan jejak ke satu Wilayah atau semua Wilayah. Trail mencatat peristiwa di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi lebih lanjut, lihat:

- Membuat jejak untuk AWS akun Anda
- AWS integrasi layanan dengan log CloudTrail

- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa Wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua tindakan Amazon ECR API dicatat oleh CloudTrail dan didokumentasikan dalam Referensi API Amazon Elastic Container Registry. Saat Anda melakukan tugas umum, bagian dibuat dalam file CloudTrail log untuk setiap tindakan API yang merupakan bagian dari tugas tersebut. Misalnya, ketika Anda membuat repositori,`GetAuthorizationToken`, `CreateRepository` dan `SetRepositoryPolicy` bagian yang dihasilkan dalam file CloudTrail log. Saat Anda mendorong gambar ke repositori,,,`InitiateLayerUpload`,`UploadLayerPart`, `CompleteLayerUploadPutImage`, dan, jika pemasangan gumpalan diaktifkan, `MountLayer` bagian dihasilkan. Ketika Anda menarik suatu citra, bagian `GetDownloadUrlForLayer` dan `BatchGetImage` dihasilkan. Saat Anda mengarsipkan atau memulihkan `UpdateImageStorageClass` bagian gambar dihasilkan. Ketika OCI klien yang mendukung OCI 1.1 spesifikasi mengambil daftar perujuk, atau artefak referensi, untuk gambar yang menggunakan Referrers API, sebuah peristiwa akan dipancarkan. `ListImageReferrers` CloudTrail Untuk contoh tugas umum ini, lihat CloudTrail contoh entri log.

Setiap entri kejadian atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan dibuat dengan root atau kredensyal pengguna
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan
- Apakah permintaan itu dibuat oleh AWS layanan lain

Untuk informasi selengkapnya, lihat CloudTrail `userIdentity` Elemen.

Memahami entri file log Amazon ECR

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan informasi lainnya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

CloudTrail contoh entri log

Berikut ini adalah contoh entri CloudTrail log untuk beberapa tugas ECR Amazon yang umum.

Contoh-contoh ini telah diformat untuk meningkatkan keterbacaan. Dalam file CloudTrail log, semua entri dan peristiwa digabungkan menjadi satu baris. Selain itu, contoh ini telah terbatas pada entri tunggal Amazon ECR. Dalam file CloudTrail log nyata, Anda melihat entri dan peristiwa dari beberapa AWS layanan.

> **Important**
>
> SourceIPaddress adalah alamat IP tempat permintaan dibuat. Untuk tindakan yang berasal dari konsol layanan, alamat yang dilaporkan adalah untuk sumber daya dasar Anda, bukan server web konsol. Untuk layanan di AWS, hanya nama DNS yang ditampilkan. Kami masih mengevaluasi autentikasi dengan IP sumber klien bahkan jika itu disunting ke nama AWS DNS layanan.

Topik

Contoh: Buat tindakan repositori

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateRepository` tindakan.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
```

```
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-11T21:54:07Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            }
        }
    },
    "eventTime": "2018-07-11T22:17:43Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "CreateRepository",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "repositoryName": "testrepo"
    },
    "responseElements": {
        "repository": {
            "repositoryArn": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "repositoryName": "testrepo",
            "repositoryUri": "123456789012.dkr.ecr.us-east-2.amazonaws.com/testrepo",
            "createdAt": "Jul 11, 2018 10:17:44 PM",
            "registryId": "123456789012"
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "accountId": "123456789012"
        }
    ],
    "eventType": "AwsApiCall",
```

```
    "recipientAccountId": "123456789012"
}
```

Contoh: AWS KMS **`CreateGrant`**Tindakan API saat membuat repositori Amazon ECR

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan AWS KMS `CreateGrant` tindakan saat membuat repositori Amazon ECR dengan enkripsi KMS diaktifkan. Untuk setiap repositori yang dibuat dengan enkripsi KMS diaktifkan, Anda akan melihat dua entri `CreateGrant` log di. CloudTrail

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIEP6W46J43IG7LXAQ",
        "arn": "arn:aws:iam::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {

            },
            "webIdFederationData": {

            },
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-06-10T19:22:10Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-06-10T19:22:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "keyId": "4b55e5bf-39c8-41ad-b589-18464af7758a",
        "granteePrincipal": "ecr.us-west-2.amazonaws.com",
        "operations": [
```

```
            "GenerateDataKey",
            "Decrypt"
        ],
        "retiringPrincipal": "ecr.us-west-2.amazonaws.com",
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecr:arn": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo"
            }
        }
    },
    "responseElements": {
        "grantId": "3636af9adfee1accb67b83941087dcd45e7fadc4e74ff0103bb338422b5055f3"
    },
    "requestID": "047b7dea-b56b-4013-87e9-a089f0f6602b",
    "eventID": "af4c9573-c56a-4886-baca-a77526544469",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:123456789012:key/4b55e5bf-39c8-41ad-
b589-18464af7758a"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Contoh: Tindakan dorongan citra

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan dorongan gambar yang menggunakan `PutImage` tindakan.

> **Note**
>
> Saat mendorong gambar, Anda juga akan melihat`InitiateLayerUpload`,`UploadLayerPart`, dan `CompleteLayerUpload` referensi di CloudTrail log.

```
{
    "eventVersion": "1.04",
```

```
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "PutImage",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageTag": "latest",
  "registryId": "123456789012",
  "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest\":
 \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n      },
\n      {\n         \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
```

```
\",\n         \"size\": 37720774,\n         \"digest\":
 \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
   ]\n}"
 },
 "responseElements": {
  "image": {
   "repositoryName": "testrepo",
   "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
```

```
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest\":
 \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n      },
\n      {\n         \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",\n         \"size\": 37720774,\n         \"digest\":
 \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
   ]\n}",
   "registryId": "123456789012",
   "imageId": {
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
    "imageTag": "latest"
   }
  }
 },
 "requestID": "cf044b7d-5f9d-11e9-9b2a-95983139cc57",
 "eventID": "2bfd4ee2-2178-4a82-a27d-b12939923f0f",
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
```

```
}
```

Contoh: Tindakan tarikan citra

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan tarikan gambar yang menggunakan `BatchGetImage` tindakan.

> **Note**
>
> Saat menarik gambar, jika Anda belum memiliki gambar secara lokal, Anda juga akan melihat `GetDownloadUrlForLayer` referensi di CloudTrail log.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T17:23:20Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "BatchGetImage",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "ecr.amazonaws.com",
 "userAgent": "ecr.amazonaws.com",
 "requestParameters": {
  "imageIds": [{
   "imageTag": "latest"
  }],
  "acceptedMediaTypes": [
   "application/json",
   "application/vnd.oci.image.manifest.v1+json",
```

```
    "application/vnd.oci.image.index.v1+json",
    "application/vnd.docker.distribution.manifest.v2+json",
    "application/vnd.docker.distribution.manifest.list.v2+json",
    "application/vnd.docker.distribution.manifest.v1+prettyjws"
   ],
   "repositoryName": "testrepo",
   "registryId": "123456789012"
  },
  "responseElements": null,
  "requestID": "2a1b97ee-5fa3-11e9-a8cd-cd2391aeda93",
  "eventID": "c84f5880-c2f9-4585-9757-28fa5c1065df",
  "resources": [{
   "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
   "accountId": "123456789012"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Contoh: Tindakan kebijakan siklus hidup citra

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan kapan gambar kedaluwarsa karena aturan kebijakan siklus hidup. Jenis kejadian ini dapat ditemukan dengan mem-filter `PolicyExecutionEvent` untuk bidang nama kejadian.

Saat Anda menguji pratinjau kebijakan siklus hidup, Amazon ECR menghasilkan entri CloudTrail log dengan bidang nama peristiwa`DryRunEvent`, dengan struktur yang sama persis dengan. `PolicyExecutionEvent` Dengan mengubah nama acara menjadi`DryRunEvent`, Anda dapat memfilter pada acara dry run sebagai gantinya.

> **Important**
>
> Karena keterbatasan ukuran CloudTrail peristiwa individual, untuk tindakan kebijakan siklus hidup di mana 10 gambar atau lebih kedaluwarsa Amazon ECR mengirimkan beberapa peristiwa ke. CloudTrail Selain itu, Amazon ECR menyertakan maksimum 100 tag per gambar.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
```

```
        "accountId": "123456789012",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-03-12T20:22:12Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "PolicyExecutionEvent",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "9354dd7f-9aac-4e9d-956d-12561a4923aa",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
            "accountId": "123456789012",
            "type": "AWS::ECR::Repository"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "repositoryName": "testrepo",
        "lifecycleEventPolicy": {
            "lifecycleEventRules": [
                {
                    "rulePriority": 1,
                    "description": "remove all images > 2",
                    "lifecycleEventSelection": {
                        "tagStatus": "Any",
                        "tagPrefixList": [],
                        "countType": "Image count more than",
                        "countNumber": 2
                    },
                    "action": "expire"
                }
            ],
            "lastEvaluatedAt": 0,
            "policyVersion": 1,
            "policyId": "ceb86829-58e7-9498-920c-aa042e33037b"
        },
        "lifecycleEventImageActions": [
            {
```

```
                "lifecycleEventImage": {
                    "digest":
 "sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45",
                    "tagStatus": "Tagged",
                    "tagList": [
                        "alpine"
                    ],
                    "pushedAt": 1584042813000
                },
                "rulePriority": 1
            },
            {
                "lifecycleEventImage": {
                    "digest":
 "sha256:6ab380c5a5acf71c1b6660d645d2cd79cc8ce91b38e0352cbf9561e050427baf",
                    "tagStatus": "Tagged",
                    "tagList": [
                        "centos"
                    ],
                    "pushedAt": 1584042842000
                },
                "rulePriority": 1
            }
        ],
        "lifecycleEventFailureDetails": [
            {
                "lifecycleEventImage": {
                    "digest":
 "sha256:9117e1bc28cd20751e584b4ccd19b1178d14cf02d134b04ce6be0cc51bff762a",
                    "tagStatus": "Untagged",
                    "tagList": [],
                    "pushedAt": 1584042844000
                },
                "rulePriority": 1,
                "failureCode": "ImageReferencedByManifestList",
                "failureReason": "Requested image referenced by manifest list:
 [sha256:4b27c83d44a18c31543039d9e8b2786043ec6c8d00804d5800c5148d6b6f65bc]"
            }
        ]
    }
}
```

Contoh: Tindakan arsip gambar

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan gambar yang diarsipkan menggunakan `UpdateImageStorageClass` tindakan dengan `targetStorageClass` set ke. `ARCHIVE`


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "UpdateImageStorageClass",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageId": {
   "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
  },
  "targetStorageClass": "ARCHIVE",
  "registryId": "123456789012"
 },
 "responseElements": {
  "image": {
   "registryId": "123456789012",
   "repositoryName": "testrepo",
   "imageId": {
```

```
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
   },
   "imageStatus": "ARCHIVED"
  }
 },
 "requestID": "cf044b7d-EXAMPLE",
 "eventID": "2bfd4ee2-EXAMPLE",
 "readOnly": false,
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "123456789012",
 "eventCategory": "Management"
}
```

Contoh: Tindakan pemulihan gambar

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan gambar yang dipulihkan. Saat Anda memulihkan gambar yang diarsipkan, dua peristiwa dihasilkan:

1. Peristiwa panggilan API saat pemulihan dimulai
2. Peristiwa layanan saat operasi pemulihan asinkron selesai

Acara panggilan API (memulihkan inisiasi)

Contoh berikut menunjukkan panggilan API awal untuk memulihkan gambar menggunakan `UpdateImageStorageClass` tindakan dengan `targetStorageClass` disetel ke`STANDARD`. Respons menunjukkan status gambar sebagai`ACTIVATING`.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
```

```
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "UpdateImageStorageClass",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageId": {
   "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
  },
  "targetStorageClass": "STANDARD",
  "registryId": "123456789012"
 },
 "responseElements": {
  "image": {
   "registryId": "123456789012",
   "repositoryName": "testrepo",
   "imageId": {
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
   },
   "imageStatus": "ACTIVATING"
  }
 },
 "requestID": "cf044b7d-EXAMPLE",
 "eventID": "2bfd4ee2-EXAMPLE",
 "readOnly": false,
 "resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "123456789012",
 "eventCategory": "Management"
```

```
}
```

Acara layanan (pemulihan penyelesaian)

Contoh berikut menunjukkan peristiwa layanan yang dihasilkan ketika operasi pemulihan asinkron selesai. Jenis kejadian ini dapat ditemukan dengan mem-filter `ImageActivationEvent` untuk bidang nama kejadian. `serviceEventDetails`Bagian ini berisi hasil pemulihan dan status gambar akhir.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-03-12T20:22:12Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "ImageActivationEvent",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "9354dd7f-EXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
            "accountId": "123456789012",
            "type": "AWS::ECR::Repository"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "repositoryName": "testrepo",
        "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
        "targetStorageClass": "STANDARD",
        "result": "SUCCESS",
        "imageStatus": "ACTIVE"
    },
```

```
    "eventCategory": "Management"
}
```

Contoh: Tindakan perujuk gambar

Contoh berikut menunjukkan entri AWS CloudTrail log yang menunjukkan kapan klien yang OCI 1.1 sesuai mengambil daftar perujuk, atau artefak referensi, untuk gambar yang menggunakan API. `Referrers`

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-10-08T16:38:39Z",
                "mfaAuthenticated": "false"
            },
            "ec2RoleDelivery": "2.0"
        },
        "invokedBy": "ecr.amazonaws.com"
    },
    "eventTime": "2024-10-08T17:22:51Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "ListImageReferrers",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "ecr.amazonaws.com",
    "userAgent": "ecr.amazonaws.com",
    "requestParameters": {
        "registryId": "123456789012",
        "repositoryName": "testrepo",
```

```
        "subjectId": {
            "imageDigest":
 "sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a"
        },
        "nextToken": "urD72mdD/mC8b5-EXAMPLE"
    },
    "responseElements": null,
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123456789012",
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

Menggunakan Amazon ECR dengan AWS SDK

AWS kit pengembangan perangkat lunak (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK untuk C++	AWS SDK untuk C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK untuk Go	AWS SDK untuk Go contoh kode
AWS SDK untuk Java	AWS SDK untuk Java contoh kode
AWS SDK untuk JavaScript	AWS SDK untuk JavaScript contoh kode
AWS SDK untuk Kotlin	AWS SDK untuk Kotlin contoh kode
AWS SDK untuk .NET	AWS SDK untuk .NET contoh kode
AWS SDK untuk PHP	AWS SDK untuk PHP contoh kode
Alat AWS untuk PowerShell	Alat AWS untuk PowerShell contoh kode
AWS SDK untuk Python (Boto3)	AWS SDK untuk Python (Boto3) contoh kode
AWS SDK untuk Ruby	AWS SDK untuk Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

Contoh kode untuk Amazon ECR menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon ECR dengan kit pengembangan AWS perangkat lunak (SDK).

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode

- Contoh dasar untuk Amazon ECR menggunakan AWS SDK
  - Halo Amazon ECR
  - Pelajari dasar-dasar Amazon ECR dengan SDK AWS
  - Tindakan untuk Amazon ECR menggunakan AWS SDK
    - Gunakan CreateRepository dengan AWS SDK atau CLI
    - Gunakan DeleteRepository dengan AWS SDK atau CLI
    - Gunakan DescribeImages dengan AWS SDK atau CLI
    - Gunakan DescribeRepositories dengan AWS SDK atau CLI
    - Gunakan GetAuthorizationToken dengan AWS SDK atau CLI
    - Gunakan GetRepositoryPolicy dengan AWS SDK atau CLI
    - Gunakan ListImages dengan AWS SDK atau CLI
    - Gunakan PushImageCmd dengan AWS SDK
    - Gunakan PutLifeCyclePolicy dengan AWS SDK atau CLI
    - Gunakan SetRepositoryPolicy dengan AWS SDK atau CLI

Contoh dasar untuk Amazon ECR menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar Amazon Elastic Container Registry dengan AWS SDK.

Contoh

Halo Amazon ECR

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon ECR.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();
```

```
        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Untuk detail API, lihat listImages di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
```

```
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Untuk detail API, lihat listImages di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.
```

```
    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Untuk detail API, lihat listImages di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Pelajari dasar-dasar Amazon ECR dengan SDK AWS

Contoh-contoh kode berikut menunjukkan cara:

- Buat repositori Amazon ECR.
- Tetapkan kebijakan repositori.
- Ambil URI repositori.
- Dapatkan token otorisasi Amazon ECR.
- Tetapkan kebijakan siklus hidup untuk repositori Amazon ECR.
- Dorong gambar Docker ke repositori Amazon ECR.
- Verifikasi keberadaan gambar di repositori Amazon ECR.
- Buat daftar repositori Amazon ECR untuk akun Anda dan dapatkan detailnya.
- Hapus repositori Amazon ECR.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario interaktif yang menunjukkan fitur Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact
 with the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text.
 Without a local image,
 * this Java program will not successfully run. For more information including
 how to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions
 to access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
```

```
            The Amazon Elastic Container Registry (ECR) is a fully-managed
Docker container registry
            service provided by AWS. It allows developers and organizations to
securely
            store, manage, and deploy Docker container images.
            ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
            from building and testing to production deployment.\s

            The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides
a set of methods to
            programmatically interact with the Amazon ECR service. This allows
developers to
            automate the storage, retrieval, and management of container images
as part of their application
            deployment pipelines. With ECR, teams can focus on building and
deploying their
            applications without having to worry about the underlying
infrastructure required to
            host and manage a container registry.

           This scenario walks you through how to perform key operations for
this service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else if (input.trim().equalsIgnoreCase("2")) {
               String repoName = "echo-text";
               ecrActions.deleteECRRepository(repoName);
               return;
           } else {
               // Handle invalid input.
```

```
                System.out.println("Invalid input. Please try again.");
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("""
           1. Create an ECR repository.

           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it
 easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
 repository: " + e.getMessage());
            e.printStackTrace();
            return;
        }
```

```
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
```

```
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the
authorization token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
```

```
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker
images stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images,
ensuring that the
            storage is optimized and the registry remains up-to-date.
            """);
```

```
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker
image to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the
 image in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS
 CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=
%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
```

```
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
repository
        or use the force option (used in this scenario) to delete the repository
and have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " +
e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

  private static void waitForInputToContinue(Scanner scanner) {
      while (true) {
          System.out.println("");
          System.out.println("Enter 'c' followed by <ENTER> to continue:");
```

```
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Kelas pembungkus untuk metode Amazon ECR SDK.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
```

```
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();
```

```
        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
```

```
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " +
repoName + " repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
               }
           }
       });

       // Wait for the CompletableFuture to complete
       response.join();
   }



   private static DockerClient getDockerClient() {
       String osName = System.getProperty("os.name");
       if (osName.startsWith("Windows")) {
           // Make sure Docker Desktop is running.
           String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
           DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
```

```
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
         and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
         It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
```

```
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
```

```
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
```

```
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request =
 DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
 getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
```

```
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the
console.
     * If an exception occurs, the method handles the exception and prints the
error message.
     *
```

```
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
           }
       });
       response.join();
   }

   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }
```

```
        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
     * @throws EcrException                      if there is an unexpected error
setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
          This example policy document grants the specified AWS principal the
permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
```

```
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }
             """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           }
       });
       response.join();
   }

   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
```

```
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
repoData.repositoryUri() + ":latest", imageName).exec();
                try {

getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
ECR");
```

```
                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .

- CreateRepository
- DeleteRepository
- DescribeImages
- DescribeRepositories
- GetAuthorizationToken
- GetRepositoryPolicy
- SetRepositoryPolicy
- StartLifecyclePolicyPreview

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario interaktif yang mendemonstrasikan fitur Amazon ECR.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
```

```
 * This code example requires a local docker image named echo-text. Without a
 local image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
        service provided by AWS. It allows developers and organizations to
 securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images
 throughout their lifecycle,
```

```
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as
part of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
```

```
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-
text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
 is crucial for maintaining
        the security and integrity of your container images. The repository
 policy allows you to
        define specific rules and restrictions for accessing and managing the
 images stored within your ECR
        repository.
```

```
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)
```

```
   println(DASHES)
   println(
       """
       5. Get the ECR Repository URI.

       The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
       a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
       or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
       which includes the ECR repository URI. This allows the container runtime
to pull the
       correct container image from the ECR repository.

       """.trimIndent(),
   )
   waitForInputToContinue(scanner)
   val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
   println("The repository URI is $repositoryURI")
   waitForInputToContinue(scanner)

   println(DASHES)
   println(
       """
       6. Set an ECR Lifecycle Policy.

       An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
       These policies allow you to automatically remove old or unused Docker
images from your repositories,
       freeing up storage space and reducing costs.

       """.trimIndent(),
   )
   waitForInputToContinue(scanner)
   val pol = ecrActions.setLifeCyclePolicy(repoName)
   println(pol)
   waitForInputToContinue(scanner)

   println(DASHES)
   println(
       """
```

```
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
 repository.
        It sets up the Docker client by connecting to the local Docker host using
 the default port.
        It then retrieves the authorization token for the ECR repository by
 making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
 which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
 Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
 repository using the Docker client.
        If the push operation is successful, the method prints a message
 indicating that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon
 ECR by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com
```

```
        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids
 imageTag=$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
```

```
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Kelas pembungkus untuk metode Amazon ECR SDK.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
```

```
            val dockerCmdExecFactory: DockerCmdExecFactory =

NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
             ]
             }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
```

```
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
```

```
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
```

```
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
```

```
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeResponse =
ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
```

```
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")

           val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
           if (tagImageCmd != null) {
               tagImageCmd.exec()
           }
           val pushImageCmd =
               repoData.repositoryUri?.let {
                   dockerClient?.pushImageCmd(it)
                       // ?.withTag("latest")
                       ?.withAuthConfig(authConfig)
               }

           try {
               if (pushImageCmd != null) {
                   pushImageCmd.start().awaitCompletion()
               }
               println("The $imageName was pushed to Amazon ECR")
           } catch (e: IOException) {
               throw RuntimeException(e)
```

```
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
```

```
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
            val registryURL: String =
repoData?.repositoryUri?.split("/")?.get(0) ?: ""

            return AuthConfig()
```

```
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations
 using
    Amazon ECR.
    """

    def __init__(
```

```
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None

    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
ECR provides a simple and scalable way to manage container images throughout
 their lifecycle,
from building and testing to production deployment.

The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of
 their application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required
 to
host and manage a container registry.

This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
```

```
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
        print(f"Creating a repository named {self.repository_name}")
        self.repository =
 self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for
 this repository.
            """
            )
        else:
            print(
                """
```

```
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
the security and integrity of your container images. The repository policy allows
 you to
define specific rules and restrictions for accessing and managing the images
 stored within your ECR
repository.
        """
            )

            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon
 ECR registry.
The `get_authorization_token` method of the `ecr` client is responsible for
 securely accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.
```

```
Without a valid authorization token, you would not be able to perform any
 operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )

        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image
 URI,
which includes the ECR repository URI. This allows the container runtime to pull
 the
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
        press_enter_to_continue()
        print_dashes()

        print(
            """
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored
 in your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.
```

```
This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")

        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
        for line in resp:
            print(line)

        print_dashes()

        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
```

```
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR
 by using the CLI."
        )
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you
 need to authenticate with the registry. You can do this using the AWS CLI:

    aws ecr get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:

   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )

        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
        :param ask: If True, prompts the user to confirm before deleting the
 resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
 '{self.repository_name}? (y/n) "
```

```
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the local Docker image
'{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
```

```
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
 images from a repository.",
        required=False,
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
```

```
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup(False)
```

Kelas ECRWrapper yang membungkus tindakan Amazon ECR.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)
```

```
    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
```

```
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
```

```
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Tindakan untuk Amazon ECR menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan tindakan ECR Amazon individual dengan AWS SDK. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Kutipan ini memanggil Amazon ECR API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Anda dapat melihat tindakan dalam konteks diSkenario untuk Amazon ECR menggunakan AWS SDK.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi API Amazon Elastic Container Registry.

Contoh

- Gunakan CreateRepository dengan AWS SDK atau CLI
- Gunakan DeleteRepository dengan AWS SDK atau CLI
- Gunakan DescribeImages dengan AWS SDK atau CLI
- Gunakan DescribeRepositories dengan AWS SDK atau CLI
- Gunakan GetAuthorizationToken dengan AWS SDK atau CLI
- Gunakan GetRepositoryPolicy dengan AWS SDK atau CLI
- Gunakan ListImages dengan AWS SDK atau CLI
- Gunakan PushImageCmd dengan AWS SDK
- Gunakan PutLifeCyclePolicy dengan AWS SDK atau CLI
- Gunakan SetRepositoryPolicy dengan AWS SDK atau CLI
- Gunakan StartLifecyclePolicyPreview dengan AWS SDK atau CLI

Gunakan **`CreateRepository`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`CreateRepository`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya
- Memulai dengan Amazon ECR

CLI

AWS CLI

Contoh 1: Untuk membuat repositori

`create-repository`Contoh berikut membuat repositori di dalam namespace yang ditentukan dalam registri default untuk akun.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Untuk informasi selengkapnya, lihat Membuat Repositori di Panduan Pengguna Amazon ECR.

Contoh 2: Untuk membuat repositori yang dikonfigurasi dengan immutabilitas tag gambar

`create-repository`Contoh berikut membuat repositori dikonfigurasi untuk kekekalan tag dalam registri default untuk akun.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
```

```
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Untuk informasi selengkapnya, lihat Mutabilitas Tag Gambar di Panduan Pengguna Amazon ECR.

Contoh 3: Untuk membuat repositori yang dikonfigurasi dengan konfigurasi pemindaian

`create-repository`Contoh berikut membuat repositori yang dikonfigurasi untuk melakukan pemindaian kerentanan pada push gambar di registri default untuk akun.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Untuk informasi selengkapnya, lihat Pemindaian Gambar di Panduan Pengguna Amazon ECR.

- Untuk detail API, lihat CreateRepositorydi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
```

```
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Untuk detail API, lihat CreateRepositorydi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
```

```
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
 repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- Untuk detail API, lihat CreateRepositorydi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Untuk detail API, lihat CreateRepositorydi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->createrepository(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_repository_uri) = oo_result->get_repository( )-
>get_repositoryuri( ).
        MESSAGE |Repository created with URI: { lv_repository_uri }| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryalrexex.
        " If repository already exists, retrieve it
        DATA lt_repo_names TYPE /aws1/
cl_ecrrepositorynamels00=>tt_repositorynamelist.
        APPEND NEW /aws1/cl_ecrrepositorynamels00( iv_value =
 iv_repository_name ) TO lt_repo_names.
        DATA(lo_describe_result) = lo_ecr-
>describerepositories( it_repositorynames = lt_repo_names ).
        DATA(lt_repos) = lo_describe_result->get_repositories( ).
        IF lines( lt_repos ) > 0.
          READ TABLE lt_repos INDEX 1 INTO DATA(lo_repo).
          oo_result = NEW /aws1/cl_ecrcrerepositoryrsp( io_repository =
 lo_repo ).
          MESSAGE |Repository { iv_repository_name } already exists.| TYPE 'I'.
        ENDIF.
    ENDTRY.
```

- Untuk detail API, lihat CreateRepositorydi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DeleteRepository`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DeleteRepository`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya
- Memulai dengan Amazon ECR

CLI

AWS CLI

Untuk menghapus repositori

`delete-repository`Contoh perintah force berikut menghapus repositori yang ditentukan dalam registri default untuk akun. `--force`Bendera diperlukan jika repositori berisi gambar.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Output:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
    }
}
```

Untuk informasi selengkapnya, lihat Menghapus Repositori di Panduan Pengguna Amazon ECR.

- Untuk detail API, lihat DeleteRepositorydi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Deletes an ECR (Elastic Container Registry) repository.
    *
    * @param repoName the name of the repository to delete.
    * @throws IllegalArgumentException if the repository name is null or empty.
    * @throws EcrException if there is an error deleting the repository.
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " +
repoName + " repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
```

```
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- Untuk detail API, lihat DeleteRepositorydi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Untuk detail API, lihat DeleteRepositorydi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
```

```
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

- Untuk detail API, lihat DeleteRepositorydi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
TRY.
    " iv_repository_name = 'my-repository'
    lo_ecr->deleterepository(
      iv_repositoryname = iv_repository_name
      iv_force = abap_true ).
    MESSAGE |Repository { iv_repository_name } deleted.| TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- Untuk detail API, lihat DeleteRepositorydi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DescribeImages`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DescribeImages`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya

CLI

AWS CLI

Untuk menggambarkan gambar dalam repositori

`describe-images`Contoh berikut menampilkan rincian tentang gambar dalam `cluster-autoscaler` repositori dengan tag. `v1.13.6`

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Output:

```
{
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
```

```
}
```

- Untuk detail API, lihat DescribeImagesdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
```

```
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Untuk detail API, lihat DescribeImagesdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
```

```
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Untuk detail API, lihat DescribeImagesdi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Untuk detail API, lihat DescribeImagesdi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   TRY.
       " iv_repository_name = 'my-repository'
       " it_image_ids = VALUE #( ( NEW /aws1/cl_ecrimageidentifier( iv_imagetag
= 'latest' ) ) )
       IF it_image_ids IS NOT INITIAL.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name
           it_imageids = it_image_ids ).
       ELSE.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name ).
       ENDIF.
       DATA(lt_image_details) = oo_result->get_imagedetails( ).
       MESSAGE |Found { lines( lt_image_details ) } images in repository.| TYPE
'I'.
     CATCH /aws1/cx_ecrrepositorynotfndex.
       MESSAGE 'Repository not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrimagenotfoundex.
       MESSAGE 'Image not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrinvalidparameterex.
       MESSAGE 'Invalid parameter provided.' TYPE 'I'.
   ENDTRY.
```

- Untuk detail API, lihat DescribeImagesdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`DescribeRepositories`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DescribeRepositories`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya

CLI

AWS CLI

Untuk menggambarkan repositori dalam registri

Contoh ini menjelaskan repositori dalam registri default untuk akun.

Perintah:

```
aws ecr describe-repositories
```

Output:

```
{
    "repositories": [
        {
            "registryId": "012345678910",
            "repositoryName": "ubuntu",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
        },
        {
            "registryId": "012345678910",
            "repositoryName": "test",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
        }
    ]
```

```
}
```

- Untuk detail API, lihat DescribeRepositoriesdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
```

```
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- Untuk detail API, lihat DescribeRepositoriesdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/**
 * Retrieves the repository URI for the specified repository name.
 *
```

```
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
 ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
 describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Untuk detail API, lihat DescribeRepositoriesdi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
```

```
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Untuk detail API, lihat DescribeRepositoriesdi AWS SDK for Python (Boto3) Referensi API.

Rust

SDK for Rust

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
 aws_sdk_ecr::Error> {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- Untuk detail API, lihat DescribeRepositoriesreferensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " it_repository_names = VALUE #( ( NEW /aws1/
cl_ecrrepositorynamels00( iv_value = 'my-repository' ) ) )
        oo_result = lo_ecr->describerepositories(
          it_repositorynames = it_repository_names ).
        DATA(lt_repositories) = oo_result->get_repositories( ).
        MESSAGE |Found { lines( lt_repositories ) } repositories.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- Untuk detail API, lihat DescribeRepositoriesdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`GetAuthorizationToken`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`GetAuthorizationToken`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya

CLI

AWS CLI

Untuk mendapatkan token otorisasi untuk registri default Anda

`get-authorization-token`Contoh perintah berikut mendapatkan token otorisasi untuk registri default Anda.

```
aws ecr get-authorization-token
```

Output:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Untuk detail API, lihat GetAuthorizationTokendi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
```

```
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
```

- Untuk detail API, lihat GetAuthorizationTokendi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- Untuk detail API, lihat GetAuthorizationTokendi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Untuk detail API, lihat GetAuthorizationTokendi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
TRY.
    oo_result = lo_ecr->getauthorizationtoken( ).
    DATA(lt_auth_data) = oo_result->get_authorizationdata( ).
    IF lines( lt_auth_data ) > 0.
      READ TABLE lt_auth_data INDEX 1 INTO DATA(lo_auth_data).
      DATA(lv_token) = lo_auth_data->get_authorizationtoken( ).
      MESSAGE 'Authorization token retrieved.' TYPE 'I'.
    ENDIF.
  CATCH /aws1/cx_ecrserverexception.
    MESSAGE 'Server exception occurred.' TYPE 'I'.
ENDTRY.
```

- Untuk detail API, lihat GetAuthorizationTokendi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`GetRepositoryPolicy`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`GetRepositoryPolicy`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya

CLI

AWS CLI

Untuk mengambil kebijakan repositori untuk repositori

`get-repository-policy`Contoh berikut menampilkan rincian tentang kebijakan repositori untuk repositori. `cluster-autoscaler`

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Output:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Untuk detail API, lihat GetRepositoryPolicydi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Gets the repository policy for the specified repository.
```

```
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
       });

       GetRepositoryPolicyResponse result = response.join();
       return result != null ? result.policyText() : null;
   }
```

- Untuk detail API, lihat GetRepositoryPolicydi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- Untuk detail API, lihat GetRepositoryPolicydi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
```

```
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Untuk detail API, lihat GetRepositoryPolicydi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
TRY.
    " iv_repository_name = 'my-repository'
    oo_result = lo_ecr->getrepositorypolicy(
      iv_repositoryname = iv_repository_name ).
    DATA(lv_policy_text) = oo_result->get_policytext( ).
    MESSAGE 'Repository policy retrieved.' TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
  CATCH /aws1/cx_ecrrepositoryplynot00.
    MESSAGE 'Repository policy not found.' TYPE 'I'.
ENDTRY.
```

- Untuk detail API, lihat GetRepositoryPolicydi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`ListImages`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListImages`.

CLI

AWS CLI

Untuk daftar gambar dalam repositori

`list-images`Contoh berikut menampilkan daftar gambar dalam `cluster-autoscaler` repositori.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Output:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
        }
    ]
}
```

- Untuk detail API, lihat ListImagesdi Referensi AWS CLI Perintah.

Rust

SDK for Rust

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();

    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- Untuk detail API, lihat ListImagesreferensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PushImageCmd** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan`PushImageCmd`.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName  the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   public void pushDockerImage(String repoName, String imageName) {
       System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
       CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
           .thenApply(response -> {
               String token =
response.authorizationData().get(0).authorizationToken();
               String decodedToken = new
String(Base64.getDecoder().decode(token));
               String password = decodedToken.substring(4);

               DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
               Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
               assert repoData != null;
               String registryURL = repoData.repositoryUri().split("/")[0];

               AuthConfig authConfig = new AuthConfig()
```

```
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Untuk detail API, lihat PushImageCmddi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")

           val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
           if (tagImageCmd != null) {
               tagImageCmd.exec()
           }
           val pushImageCmd =
               repoData.repositoryUri?.let {
                   dockerClient?.pushImageCmd(it)
                       // ?.withTag("latest")
                       ?.withAuthConfig(authConfig)
               }

           try {
               if (pushImageCmd != null) {
                   pushImageCmd.start().awaitCompletion()
               }
```

```
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- Untuk detail API, lihat PushImageCmddi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`PutLifeCyclePolicy`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`PutLifeCyclePolicy`.

CLI

AWS CLI

Untuk membuat kebijakan siklus hidup

`put-lifecycle-policy`Contoh berikut membuat kebijakan siklus hidup untuk repositori yang ditentukan dalam registri default untuk akun.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Isi dari `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
```

```
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Output:

```
{
   "registryId": "<aws_account_id>",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\"rules\":[{\"rulePriority\":1,\"description\":
\"Expire images older than 14 days\",\"selection\":{\"tagStatus\":\"untagged\",
\"countType\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},
\"action\":{\"type\":\"expire\"}}]}"
}
```

Untuk informasi selengkapnya, lihat Kebijakan Siklus Hidup di Panduan Pengguna Amazon ECR.

- Untuk detail API, lihat PutLifeCyclePolicydi Referensi AWS CLI Perintah.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
```

```
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Contoh yang menempatkan kebijakan tanggal kedaluwarsa.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
```

```
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Untuk detail API, lihat PutLifeCyclePolicydi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_lifecycle_policy_text = '{"rules":
[{"rulePriority":1,"description":"Expire images older than 14 days",...}]}'
        lo_ecr->putlifecyclepolicy(
          iv_repositoryname = iv_repository_name
          iv_lifecyclepolicytext = iv_lifecycle_policy_text ).
        MESSAGE |Lifecycle policy set for repository { iv_repository_name }.|
 TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
```

```
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrvalidationex.
        MESSAGE 'Invalid lifecycle policy format.' TYPE 'I'.
    ENDTRY.
```

- Untuk detail API, lihat PutLifeCyclePolicydi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`SetRepositoryPolicy`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`SetRepositoryPolicy`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya

CLI

AWS CLI

Untuk menetapkan kebijakan repositori untuk repositori

`set-repository-policy`Contoh berikut melampirkan kebijakan repositori yang terkandung dalam file ke repositori. `cluster-autoscaler`

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Isi dari `my-policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
```

```
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Output:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Untuk detail API, lihat SetRepositoryPolicydi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does
 not exist.
```

```
    * @throws EcrException                        if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
           {
            "Version":"2012-10-17",
            "Statement" : [ {
              "Sid" : "new statement",
              "Effect" : "Allow",
              "Principal" : {
                "AWS" : "%s"
              },
              "Action" : "ecr:BatchGetImage"
            } ]
          }
           """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
```

```
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }
```

- Untuk detail API, lihat SetRepositoryPolicydi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
         {
          "Version":"2012-10-17",
          "Statement" : [ {
            "Sid" : "new statement",
            "Effect" : "Allow",
            "Principal" : {
              "AWS" : "$iamRole"
```

```
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Untuk detail API, lihat SetRepositoryPolicydi AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.
```

```
        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Contoh yang memberikan akses unduhan peran IAM.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.
```

```
        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Untuk detail API, lihat SetRepositoryPolicydi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
TRY.
    " iv_repository_name = 'my-repository'
    " iv_policy_text = '{"Version":"2012-10-17",       "Statement":[...]}'
    lo_ecr->setrepositorypolicy(
      iv_repositoryname = iv_repository_name
      iv_policytext = iv_policy_text ).
    MESSAGE |Policy set for repository { iv_repository_name }.| TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- Untuk detail API, lihat SetRepositoryPolicydi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`StartLifecyclePolicyPreview`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`StartLifecyclePolicyPreview`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Pelajari dasar-dasarnya

CLI

AWS CLI

Untuk membuat pratinjau kebijakan siklus hidup

`start-lifecycle-policy-preview`Contoh berikut membuat pratinjau kebijakan siklus hidup yang ditentukan oleh file JSON untuk repositori tertentu.

```
aws ecr start-lifecycle-policy-preview \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Isi dari `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
```

```
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Output:

```
{
    "registryId": "012345678910",
    "repositoryName": "project-a/amazon-ecs-sample",
    "lifecyclePolicyText": "{\n    \"rules\": [\n        {\n
 \"rulePriority\": 1,\n           \"description\": \"Expire images older than 14
 days\",\n           \"selection\": {\n               \"tagStatus\": \"untagged
\",\n               \"countType\": \"sinceImagePushed\",\n
 \"countUnit\": \"days\",\n               \"countNumber\": 14\n           },\n
          \"action\": {\n               \"type\": \"expire\"\n           }\n
      }\n    ]\n}\n",
    "status": "IN_PROGRESS"
}
```

- Untuk detail API, lihat StartLifecyclePolicyPreviewdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
```

```
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null &&
!
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Untuk detail API, lihat StartLifecyclePolicyPreviewdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
```

```
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Untuk detail API, lihat StartLifecyclePolicyPreviewdi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Skenario untuk Amazon ECR menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Amazon ECR dengan AWS SDK. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Amazon ECR atau digabungkan dengan yang lain. Layanan AWS Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- Memulai dengan Amazon ECR

Memulai dengan Amazon ECR

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat citra Docker
- Buat repositori Amazon ECR
- Hapus sumber daya

Bash

AWS CLI dengan skrip Bash

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori tutorial pengembang Sample.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "================================================="

# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
cliv2.html for installation instructions."
    exit 1
fi
```

```
# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
    echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"

    echo "================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "================================================="
```

```
    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "=================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force ||
 echo "Failed to delete repository, it may not exist or may have already been
 deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
 image, it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
 tagged image, it may not exist or may have already been deleted."
    fi

    echo "Cleanup completed."
    echo "=================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
```

```
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

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

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

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"
```

```
# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
 AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
 handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
```

```
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
 to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
 image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "================================================="

# Ask user if they want to clean up resources
echo ""
echo "=========================================="
echo "CLEANUP CONFIRMATION"
echo "=========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]];
 then
```

```
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "================================================="
```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository
  - GetCallerIdentity

- GetLoginPassword

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon ECR dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Kuota layanan Amazon ECR

Tabel berikut menyediakan kuota layanan default untuk Amazon Elastic Container Registry (Amazon ECR).

Nama	Default	Dapat disesuaikan	Deskripsi
Pemindaian gambar dasar per 24 jam	Setiap Wilayah yang didukung: 100.000	Tidak	Jumlah maksimum gambar yang dapat dipindai dalam periode 24 jam di akun saat ini dan wilayah menggunakan pemindaian dasar. Batas ini mencakup pemindaian pada push dan pemindaian manual.
Filter per aturan dalam konfigurasi replikasi	Setiap Wilayah yang didukung: 100	Tidak	Jumlah maksimum filter per aturan dalam konfigurasi replikasi.
Gambar per repositori	Setiap Wilayah yang didukung: 100.000	Ya	Jumlah citra maksimum per repositori.
Bagian lapisan	Setiap Wilayah yang didukung: 4.200	Tidak	Jumlah maksimum bagian lapisan. Ini hanya berlaku jika Anda menggunakan tindakan API Amazon ECR secara langsung untuk memulai unggahan multipart untuk operasi dorongan citra.

Nama	Default	Dapat disesu an	Deskripsi
Panjang kebijakan siklus hidup	Setiap Wilayah yang didukung: 30.720	Tidak	Jumlah maksimum karakter dalam kebijakan siklus hidup.
Ukuran bagian lapisan maksimum	Setiap Wilayah yang didukung: 10	Tidak	Ukuran maksimum (MiB) dari sebuah bagian lapisan. Ini hanya berlaku jika Anda menggunak an tindakan API Amazon ECR secara langsung untuk memulai unggahan multipart untuk operasi dorongan citra.
Ukuran lapisan maksimum	Setiap Wilayah yang didukung: 52.000	Tidak	Ukuran maksimum (MiB) dari sebuah layer.
Ukuran bagian lapisan minimum	Setiap Wilayah yang didukung: 5	Tidak	Ukuran minimum (MiB) dari bagian lapisan. Ini hanya berlaku jika Anda menggunakan tindakan API Amazon ECR secara langsung untuk memulai unggahan multipart untuk operasi dorongan citra.
Tarik aturan cache per registri	Setiap Wilayah yang didukung: 50	Tidak	Jumlah maksimum aturan cache pull-through.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif BatchCheckLayerAvailability permintaan	Setiap Wilayah yang didukung: 1.000 per detik	Ya	Jumlah maksimum BatchCheckLayerAvailability permintaan yang dapat Anda buat per detik di Wilayah saat ini. Ketika citra didorong ke repositori, setiap lapisan citra diperiksa untuk memverifikasi apakah itu telah diunggah sebelumnya. Jika sudah diunggah, maka lapisan citra dilewati.
Tarif BatchGetImage permintaan	Setiap Wilayah yang didukung: 2.000 per detik	Ya	Jumlah maksimum BatchGetImage permintaan yang dapat Anda buat per detik di Wilayah saat ini. Saat gambar ditarik, BatchGetImage API dipanggil sekali untuk mengambil manifes gambar. Jika Anda meminta peningkatan kuota untuk API ini, tinjau juga GetDownloadUrlForLayer penggunaan Anda.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif CompleteLayerUpload permintaan	Setiap Wilayah yang didukung: 100 per detik	Ya	Jumlah maksimum CompleteLayerUpload permintaan yang dapat Anda buat per detik di Wilayah saat ini. Saat gambar didorong, CompleteLayerUpload API dipanggil sekali per setiap layer gambar baru untuk memverifikasi bahwa unggahan telah selesai.
Tarif GetAuthorizationToken permintaan	Setiap Wilayah yang didukung: 500 per detik	Ya	Jumlah maksimum GetAuthorizationToken permintaan yang dapat Anda buat per detik di Wilayah saat ini.
Tarif GetDownloadUrlForLayer permintaan	Setiap Wilayah yang didukung: 3.000 per detik	Ya	Jumlah maksimum GetDownloadUrlForLayer permintaan yang dapat Anda buat per detik di Wilayah saat ini. Saat gambar ditarik, GetDownloadUrlForLayer API dipanggil sekali per layer gambar yang belum di-cache. Jika Anda meminta peningkatan kuota untuk API ini, tinjau juga BatchGetImage penggunaan Anda.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif InitiateLayerUpload permintaan	Setiap Wilayah yang didukung: 100 per detik	Ya	Jumlah maksimum InitiateLayerUpload permintaan yang dapat Anda buat per detik di Wilayah saat ini. Saat gambar didorong, InitiateLayerUpload API dipanggil sekali per layer gambar yang belum diunggah. Apakah lapisan gambar telah diunggah atau tidak ditentukan oleh tindakan BatchCheckLayerAvailability API.
Tarif PutImage permintaan	Setiap Wilayah yang didukung: 10 per detik	Ya	Jumlah maksimum PutImage permintaan yang dapat Anda buat per detik di Wilayah saat ini. Ketika gambar didorong dan semua lapisan gambar baru telah diunggah, PutImage API dipanggil sekali untuk membuat atau memperbarui manifes gambar dan tag yang terkait dengan gambar.

Nama	Default	Dapat disesuaikan	Deskripsi
Tarif UploadLayerPart permintaan	Setiap Wilayah yang didukung: 500 per detik	Ya	Jumlah maksimum UploadLayerPart permintaan yang dapat Anda buat per detik di Wilayah saat ini. Ketika gambar didorong, setiap layer gambar baru diunggah dalam beberapa bagian dan UploadLayerPart API dipanggil sekali per setiap bagian layer gambar baru.
Tingkat pemindaian citra	Setiap Wilayah yang didukung: 1	Tidak	Jumlah maksimum pemindaian citra per citra, per 24 jam.
Repositori terdaftar	Setiap Wilayah yang didukung: 100.000	Ya	Jumlah maksimum repositori yang dapat Anda buat di akun ini di Wilayah saat ini.
Aturan per kebijakan siklus hidup	Setiap Wilayah yang didukung: 50	Tidak	Jumlah maksimum aturan dalam kebijakan siklus hidup
Aturan per konfigurasi replikasi	Setiap Wilayah yang didukung: 10	Tidak	Jumlah maksimum aturan dalam konfigurasi replikasi.
Tag per citra	Setiap Wilayah yang didukung: 1.000	Tidak	Jumlah maksimum tanda per citra.

Nama	Default	Dapat disesuaikan	Deskripsi
Tujuan unik di semua aturan dalam konfigurasi replikasi	Setiap Wilayah yang didukung: 25	Tidak	Jumlah maksimum tujuan unik di semua aturan dalam konfigurasi replikasi.

Mengelola kuota layanan Amazon ECR Anda di Konsol Manajemen AWS

Amazon ECR telah terintegrasi dengan Service Quotas, AWS sebuah layanan yang memungkinkan Anda untuk melihat dan mengelola kuota Anda dari lokasi pusat. Untuk informasi selengkapnya, lihat Apa Itu Service Quotas? dalam Panduan Pengguna Service Quotas.

Service Quotas mempermudah Anda mencari nilai semua kuota layanan Amazon ECR.

Untuk melihat kuota layanan Amazon ECR (Konsol Manajemen AWS)

1. Buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas/.
2. Di panel navigasi, pilih Layanan AWS .
3. Dari daftar AWS layanan, cari dan pilih Amazon Elastic Container Registry (Amazon ECR).

   Dalam daftar kuota Layanan, Anda dapat melihat nama kuota layanan, nilai yang diterapkan (jika tersedia), kuota AWS default, dan apakah nilai kuota dapat disesuaikan.
4. Untuk melihat informasi tambahan tentang kuota layanan, seperti deskripsi, pilih nama kuota.

Untuk meminta kenaikan kuota, lihat Meminta kenaikan kuota di Panduan Pengguna Service Quotas.

Membuat CloudWatch alarm untuk memantau metrik penggunaan API

Amazon ECR menyediakan metrik CloudWatch penggunaan yang sesuai dengan kuota AWS layanan untuk masing-masing yang APIs terlibat dengan otentikasi registri, push gambar, dan tindakan tarik gambar. Di konsol Service Quotas, Anda dapat memvisualisasikan penggunaan Anda

pada grafik dan mengonfigurasi alarm yang memperingatkan Anda ketika penggunaan mendekati kuota layanan. Untuk informasi selengkapnya, lihat Metrik penggunaan Amazon ECR.

Gunakan langkah-langkah berikut untuk membuat CloudWatch alarm berdasarkan salah satu metrik penggunaan Amazon ECR API.

Untuk membuat alarm berdasarkan kuota penggunaan Amazon ECR Anda (Konsol Manajemen AWS)

1. Buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas/.
2. Di panel navigasi, pilih Layanan AWS .
3. Dari daftar AWS layanan, cari dan pilih Amazon Elastic Container Registry (Amazon ECR).
4. Di daftar Kuota layanan, pilih kuota penggunaan Amazon ECR yang ingin Anda buat alarm.
5. Di bagian alarm CloudWatch Acara Amazon, pilih Buat.
6. Untuk Ambang batas Alarm, pilih persentase nilai kuota yang ingin Anda tetapkan sebagai nilai alarm.
7. Untuk Nama alarm, masukkan nama untuk alarm, lalu pilih Buat.

Pemecahan masalah Amazon ECR

Bab ini membantu Anda menemukan informasi diagnostik untuk Amazon ECR, dan menyediakan langkah-langkah pemecahan masalah untuk masalah umum dan pesan kesalahan.

Topik

Memecahkan masalah perintah dan masalah Docker saat menggunakan Amazon ECR

Dalam beberapa kasus, menjalankan perintah Docker terhadap Amazon ECR dapat mengakibatkan pesan kesalahan. Beberapa pesan kesalahan umum dan kemungkinan solusi dijelaskan di bawah ini.

Topik

Log Docker tidak berisi pesan kesalahan yang diharapkan

Untuk mulai men-debug masalah terkait Docker, mulailah dengan mengaktifkan output debugging Docker pada daemon Docker yang berjalan pada instance host Anda. Jika Anda menggunakan gambar yang diambil dari Amazon ECR pada instance container Amazon ECS, lihat Mengonfigurasi keluaran verbose dari daemon Docker di Panduan Pengembang Layanan Amazon Elastic Container.

Kesalahan: "Verifikasi Sistem Berkas Gagal" atau "404: Citra Tidak Ditemukan" saat menarik citra dari repositori Amazon ECR

Anda mungkin menerima kesalahan `Filesystem verification failed` ketika menggunakan perintah docker pull untuk menarik citra dari repositori Amazon ECR dengan Docker versi 1.9 atau di

atasnya. Anda mungkin menerima kesalahan `404: Image not found` saat Anda menggunakan Docker versi sebelum 1.9.

Beberapa kemungkinan alasan dan penjelasannya diberikan di bawah ini.

Disk lokal penuh

Jika disk lokal yang Anda jalankan docker pull penuh, maka hash SHA-1 yang dihitung pada file lokal mungkin berbeda dari yang dihitung oleh Amazon ECR. Periksa apakah disk lokal Anda memiliki cukup ruang kosong untuk menyimpan citra Docker yang Anda tarik. Anda juga dapat menghapus citra lama untuk memberi ruang bagi yang baru. Gunakan perintah docker images untuk melihat daftar semua citra Docker yang diunduh secara lokal, bersama dengan ukurannya.

Client tidak dapat terhubung ke repositori jarak jauh karena kesalahan jaringan

Panggilan ke repositori Amazon ECR memerlukan koneksi internet yang berfungsi. Verifikasi pengaturan jaringan Anda, dan verifikasi bahwa alat dan aplikasi lain dapat mengakses sumber daya di internet. Jika Anda menjalankan docker pull pada instans Amazon EC2 di subnet pribadi, verifikasi bahwa subnet memiliki rute ke internet. Gunakan server penerjemahan alamat jaringan (NAT) atau gateway NAT terkelola.

Saat ini, panggilan ke repositori Amazon ECR juga memerlukan akses jaringan melalui firewall perusahaan Anda ke Amazon Simple Storage Service (Amazon S3). Jika organisasi Anda menggunakan perangkat lunak firewall atau perangkat NAT yang memungkinkan titik akhir layanan, pastikan bahwa titik akhir layanan Amazon S3 untuk Wilayah Anda saat ini diperbolehkan.

Jika Anda menggunakan Docker di belakang proksi HTTP, Anda dapat mengonfigurasi Docker dengan pengaturan proksi yang sesuai. Untuk informasi selengkapnya, lihat Proksi HTTP dalam dokumentasi Docker.

Kesalahan: "Verifikasi Lapisan Sistem Berkas Gagal" saat menarik citra dari Amazon ECR

Anda mungkin menerima kesalahan `image image-name not found` saat menarik citra menggunakan perintah docker pull. Jika Anda memeriksa log Docker, Anda mungkin melihat kesalahan seperti berikut ini:

```
filesystem layer verification failed for digest sha256:2b96f...
```

Kesalahan ini menunjukkan bahwa satu atau beberapa lapisan untuk citra Anda gagal mengunduh. Beberapa kemungkinan alasan dan penjelasannya diberikan di bawah ini.

Anda menggunakan Docker versi lama

Kesalahan ini dapat terjadi dalam persentase kecil kasus saat menggunakan versi Docker kurang dari 1.10. Upgrade client Docker Anda menjadi 1.10 atau lebih tinggi.

Client Anda mengalami kesalahan jaringan atau disk

Disk penuh atau masalah jaringan dapat mencegah pengunduhan satu atau lebih lapisan, seperti yang dibahas sebelumnya tentang pesan `Filesystem verification failed`. Ikuti rekomendasi di atas untuk memastikan bahwa sistem file Anda tidak penuh, dan bahwa Anda telah mengaktifkan akses ke Amazon S3 dari dalam jaringan Anda.

Kesalahan HTTP 403 atau kesalahan "no basic auth credentials" ketika mendorong ke repositori

Ada kalanya Anda mungkin menerima kesalahan `HTTP 403 (Forbidden)`, atau pesan kesalahan `no basic auth credentials` dari perintah docker push atau docker pull, bahkan jika Anda telah berhasil diautentikasi ke Docker menggunakan perintah aws ecr get-login-password. Berikut ini adalah beberapa penyebab yang diketahui dari masalah ini:

Anda telah mengautentikasi ke wilayah yang berbeda

Permintaan autentikasi terkait dengan wilayah tertentu, dan tidak dapat digunakan di seluruh wilayah. Misalnya, jika Anda mendapatkan token otorisasi dari US West (Oregon), Anda tidak dapat menggunakannya untuk mengautentikasi terhadap repositori Anda di US East (N. Virginia). Untuk mengatasi masalah ini, pastikan bahwa Anda telah mengambil token autentikasi dari Wilayah yang sama dengan tempat repositori Anda berada. Untuk informasi selengkapnya, lihat the section called “Otentikasi registri”.

Anda telah mengautentikasi untuk mendorong ke repositori yang tidak memiliki izin

Anda tidak memiliki izin yang diperlukan untuk mendorong ke repositori. Untuk informasi selengkapnya, lihat Kebijakan repositori pribadi di Amazon ECR.

Token Anda kedaluwarsa

Masa kedaluwarsa token otorisasi default untuk token yang diperoleh dengan menggunakan operasi `GetAuthorizationToken` adalah 12 jam.

Bug di pengelola kredensial `wincred`

Beberapa versi Docker untuk Windows menggunakan pengelola kredensyal yang disebut`wincred`, yang tidak menangani perintah login Docker dengan benar yang dihasilkan oleh aws ecr get-login-password (untuk informasi lebih lanjut, lihat `CredsStore`gagal dengan repositori pribadi). Anda dapat menjalankan perintah masuk Docker yang merupakan output, tetapi ketika Anda mencoba untuk mendorong atau menarik citra, perintah tersebut gagal. Anda dapat bekerja di sekitar bug ini dengan menghapus skema `https://` dari argumen registri dalam perintah masuk Docker yang merupakan output dari aws ecr get-login-password. Contoh perintah masuk Docker tanpa skema HTTPS ditampilkan di bawah ini.

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Memecahkan masalah pesan kesalahan Amazon ECR

Dalam beberapa kasus, panggilan API yang telah Anda mulai melalui konsol Amazon ECR atau AWS CLI keluar dengan pesan kesalahan. Beberapa pesan kesalahan umum dan kemungkinan solusi dijelaskan di bawah ini.

HTTP 429: Terlalu Banyak Permintaan atau ThrottleException

Anda mungkin menerima `429: Too Many Requests` kesalahan atau `ThrottleException` kesalahan dari satu atau beberapa tindakan Amazon ECR atau panggilan API. Hal ini menunjukkan bahwa Anda memanggil titik akhir tunggal di Amazon ECR berulang kali dalam jangka waktu pendek, dan bahwa permintaan Anda semakin dibatasi (throttled). Throttling terjadi ketika panggilan ke titik akhir tunggal dari satu pengguna melebihi ambang batas tertentu selama periode waktu.

Setiap operasi API di Amazon ECR memiliki kecepatan kecepatan yang terkait dengannya. Misalnya, pembatasan untuk tindakan `GetAuthorizationToken` adalah 20 transaksi per detik (TPS), dengan lonjakan hingga 200 TPS diperbolehkan. Di setiap wilayah, setiap akun menerima bucket yang dapat menyimpan hingga 200 kredit `GetAuthorizationToken`. Kredit ini diisi ulang dengan kecepatan 20 per detik. Jika bucket Anda memiliki 200 kredit, Anda bisa mencapai 200 transaksi API `GetAuthorizationToken` per detik untuk satu detik, dan kemudian mempertahankan 20 transaksi per detik tanpa batas waktu. Untuk informasi selengkapnya tentang batas tarif Amazon ECR APIs, lihatKuota layanan Amazon ECR.

Untuk menangani kesalahan throttling, terapkan fungsi percobaan ulang dengan backoff tambahan ke dalam kode Anda. Untuk informasi selengkapnya, lihat Coba lagi perilaku di Panduan Referensi

Alat AWS SDKs dan Alat. Pilihan lainnya adalah meminta kenaikan batas tarif, yang dapat Anda lakukan menggunakan konsol Service Quotas. Untuk informasi lebih lanjut, lihatMengelola kuota layanan Amazon ECR Anda di Konsol Manajemen AWS. .

HTTP 403: "Pengguna [arn] tidak memiliki otoriasasi untuk melakukan [operasi]"

Anda mungkin menerima kesalahan berikut ketika mencoba untuk melakukan tindakan dengan Amazon ECR:

```
$ aws ecr get-login-password
A client error (AccessDeniedException) occurred when calling the GetAuthorizationToken
 operation:
    User: arn:aws:iam::account-number:user/username is not authorized to perform:
    ecr:GetAuthorizationToken on resource: *
```

Hal ini menunjukkan bahwa pengguna Anda tidak memiliki izin yang diberikan untuk menggunakan Amazon ECR, atau bahwa izin tersebut tidak diatur dengan benar. Secara khusus, jika Anda melakukan tindakan terhadap repositori Amazon ECR, verifikasi bahwa pengguna telah diberikan izin untuk mengakses repositori tersebut. Untuk informasi selengkapnya tentang membuat dan memverifikasi izin untuk Amazon ECR, lihat Identity and Access Management untuk Amazon Elastic Container Registry.

HTTP 404: kesalahan "Repositori Tidak Ada"

Jika Anda menentukan repositori Docker Hub yang saat ini tidak ada, Docker Hub membuatnya secara otomatis. Dengan Amazon ECR, repositori baru harus dibuat secara eksplisit sebelum dapat digunakan. Hal ini mencegah repositori baru dibuat secara tidak sengaja (misalnya, karena kesalahan ketik), dan juga memastikan bahwa kebijakan akses keamanan yang sesuai secara eksplisit ditetapkan ke repositori baru. Untuk informasi selengkapnya tentang membuat repositori, lihat Repositori pribadi Amazon ECR.

Kesalahan: Tidak dapat melakukan login interaktif dari perangkat non TTY

Jika Anda menerima kesalahan`Cannot perform an interactive login from a non TTY device`, langkah-langkah pemecahan masalah berikut akan membantu.

- Verifikasi bahwa Anda menggunakan AWS CLI versi 2 dan Anda tidak memiliki versi AWS CLI versi 1 yang bertentangan di sistem Anda. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui versi terbaru AWS CLI.
- Verifikasi bahwa Anda telah mengonfigurasi AWS CLI dengan kredensyal yang valid. Untuk informasi selengkapnya, lihat Menginstal atau memperbarui versi terbaru AWS CLI.
- Verifikasi bahwa sintaks AWS CLI perintah Anda benar.

Menggunakan Podman dengan Amazon ECR

Menggunakan Podman dengan Amazon ECR memungkinkan organisasi untuk memanfaatkan keamanan dan kesederhanaan Podman sambil memanfaatkan skalabilitas dan keandalan Amazon ECR untuk manajemen gambar kontainer. Dengan mengikuti langkah-langkah dan perintah yang diuraikan, pengembang dan administrator dapat merampingkan alur kerja kontainer mereka, meningkatkan keamanan, dan mengoptimalkan pemanfaatan sumber daya. Karena kontainerisasi terus mendapatkan momentum, menggunakan dan Podman Amazon ECR memberikan solusi yang kuat dan fleksibel untuk mengelola dan menerapkan aplikasi kontainer.

Menggunakan Podman untuk mengautentikasi dengan Amazon ECR

Sebelum berinteraksi dengan Amazon ECR menggunakanPodman, otentikasi diperlukan. Ini dapat dicapai dengan menjalankan `aws ecr get-login-password` perintah untuk mengambil token otentikasi, dan kemudian menggunakan token itu dengan `podman login` perintah untuk mengautentikasi dengan Amazon ECR.

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

Menggunakan pembantu kredensi Amazon ECR dengan Podman

Amazon ECR menyediakan pembantu kredenal Docker yang bekerja dengan Podman. Pembantu kredensi memudahkan penyimpanan dan penggunaan kredenal Docker saat mendorong dan menarik gambar ke Amazon ECR. Untuk langkah-langkah instalasi dan konfigurasi, lihat Amazon ECR Docker Credential Helper.

Important

Podman hanya mendukung sebagian docker-creds-helper spesifikasi. Podman mendukung `credHelpers` kata kunci dalam konfigurasi Docker tetapi tidak mendukung kata kunci. `credsStore`

Untuk menggunakan pembantu kredenal Amazon ECR dengan Podman, konfigurasikan file konfigurasi Docker Anda dengan format: `credHelpers`

```
{
    "credHelpers": {
        "public.ecr.aws": "ecr-login",
        "<aws_account_id>.dkr.ecr.<region>.amazonaws.com": "ecr-login"
    }
}
```

`credsStore`Konfigurasi berikut tidak didukung oleh Podman:

```
{
    "credsStore": "ecr-login"
}
```

> **Note**
>
> Pembantu kredensi Amazon ECR Docker tidak mendukung otentikasi multi-faktor (MFA) saat ini.

Menarik gambar dari Amazon ECR dengan Podman

Setelah otentikasi berhasil, gambar kontainer dapat ditarik dari Amazon ECR menggunakan `podman pull` perintah dengan URI repositori Amazon ECR lengkap.

```
podman pull aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Menjalankan wadah untuk Amazon ECR dengan Podman

Setelah gambar yang diinginkan telah ditarik, sebuah wadah dapat dipakai menggunakan perintah. `podman run`

```
podman run -d aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Mendorong gambar ke Amazon ECR dengan Podman

Untuk mendorong gambar lokal ke Amazon ECR, gambar harus terlebih dahulu ditandai dengan URI repositori Amazon ECR menggunakan`podman tag`, dan kemudian `podman push` perintah dapat digunakan untuk mengunggah gambar ke Amazon ECR.

```
podman
 tag local_image:tag aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
podman push aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir Amazon ECR. Kami juga rutin memperbarui dokumentasi untuk menjawab umpan balik yang Anda kirimkan kepada kami.

Perubahan	Deskripsi	Date
Pengakhiran pemindaian berbasis Clair selesai	Pada 2 Februari 2026, pemindaian gambar berbasis Clair telah tidak digunakan lagi dan semua akun ECR telah dimigrasikan ke pemindaian dasar asli. AWS Dokumentasi telah diperbarui untuk menghapus konten khusus CLAIR dan mencerminkan bahwa AWS native sekarang adalah satu-satunya implementasi pemindaian dasar.	2 Februari 2026
Penandatanganan terkelola ECR	Amazon ECR sekarang mendukung penandata nganan gambar kontainer terkelola untuk meningkat kan postur keamanan Anda dan menghilangkan overhead operasional pengaturan penandatanganan. Penandatanganan gambar kontainer memungkin kan Anda memverifikasi bahwa gambar berasal dari sumber tepercaya. Untuk informasi selengkapnya, lihat Penandatanganan terkelola.	21 November 2025
IPv6 dukungan untuk AWS PrivateLink (titik akhir VPC)	Menambahkan dukungan untuk konektivitas dual-stack (IPv4 dan IPv6) untuk titik akhir VPC Amazon ECR yang didukung oleh. AWS PrivateLink Anda sekarang dapat membuat titik akhir VPC dual-stack yang menangani lalu lintas melalui keduanya IPv4 dan alamat IP pribadi. IPv6 Untuk informasi selengkapnya, lihat Titik akhir VPC antarmuka Amazon ECR (AWS PrivateLink).	21 November 2025
Fitur ECR Archive	Amazon ECR telah menambahkan dukungan untuk pengarsipan gambar untuk retensi jangka panjang.	19 November 2025

Perubahan	Deskripsi	Date
	Untuk informasi selengkapnya, lihat Mengarsipkan gambar di Amazon ECR.	
Diperbarui untuk menyertakan dukungan untuk pola pengecualian immutabilitas tag gambar	Amazon ECR telah memperbarui kemampuan penandaan gambar yang diperbarui untuk menyertakan pola pengecualian immutabilitas tag gambar saat membuat dan memperbarui repositori. Anda sekarang dapat menentukan tag yang dapat diperbarui bahkan ketika kekekalan tag diaktifkan pada repositori dengan mendefinisikan pola wildcard (seperti, `latest`“,`v*.beta`, `dev-*`) untuk mengecualikan tag tertentu dari aturan kekekalan sambil mempertahankan kekekalan untuk semua tag lainnya. Untuk informasi selengkapnya, lihat Membuat repositori pribadi Amazon ECR untuk menyimpan gambar.	23 Juli 2025
Update Enhanced Image Scanning untuk memberikan wawasan penggunaan gambar	Amazon ECR telah memperbarui kemampuan Pemindaian Gambar yang Disempurnakan untuk menyertakan visibilitas tentang cara gambar digunakan di Amazon EKS dan Amazon ECS. Untuk informasi selengkapnya, lihat Memindai gambar untuk OS dan kerentanan paket bahasa pemrograman di Amazon ECR.	16 Juni 2025
IPv6 dukungan	Menambahkan dukungan untuk membuat permintaan ke pendaftar Amazon ECR menggunakan titik akhir IPv4 -only dan dual-stack (dan). IPv4 IPv6 Untuk informasi selengkapnya, lihat Membuat permintaan ke pendaftar Amazon ECR.	30 April 2025
Menambahkan dukungan registri pribadi Amazon ECR untuk menarik cache	Amazon ECR menambahkan dukungan untuk membuat aturan cache pull through untuk registri pribadi Amazon ECR. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR dan Peran terkait layanan Amazon ECR untuk menarik cache.	12 Maret 2025

Perubahan	Deskripsi	Date
Ditambahkan dukungan untuk pengaturan lingkup kebijakan registri	Amazon ECR menambahkan dukungan untuk mengonfigurasi cakupan kebijakan registri untuk registri pribadi Anda. Untuk informasi selengkapnya, lihat Izin registri pribadi di Amazon ECR dan registri pribadi Amazon ECR.	23 Desember 2024
Amazon EC2 Container RegistryPullOnly - Kebijakan baru	Amazon ECR menambahkan kebijakan baru yang memberikan izin khusus tarik ke Amazon ECR.	10 Oktober 2024
Operasi yang diproksi klien Docker/OCI dalam acara sekarang menunjuk ke CloudTrail `ecr.amazonaws.com`	Nilai `ecr.amazonaws.com` menggantikan `AWS Internal` kolom User Agent (`userAgent`) dan Source IP Address (`sourceIPAddress`) untuk CloudTrail peristiwa yang terkait dengan titik akhir Docker/OCI Klien. Sebagai contoh, lihat Contoh: Tindakan tarikan citra dan Contoh: Tindakan dorongan citra.	1 Juli 2024
Menambahkan deskripsi peran terkait layanan Amazon ECR baru untuk templat pembuatan repositori.	Amazon ECR menggunakan nama peran terkait layanan AWSServiceRoleForECRTemplateyang memberikan izin kepada Amazon ECR untuk melakukan tindakan atas nama Anda guna menyelesaikan tindakan template pembuatan repositori. Untuk informasi selengkapnya, lihat Peran terkait layanan Amazon ECR untuk templat pembuatan repositori.	20 Juni 2024
Menambahkan peran `ECRTemplateService RolePolicy` terkait layanan.	Menambahkan peran `ECRTemplateService RolePolicy` terkait layanan. Untuk informasi selengkapnya, lihat `ECRTemplateService RolePolicy`	20 Juni 2024
Menambahkan replikasi lintas wilayah dan lintas akun ke Wilayah Tiongkok.	Amazon ECR menambahkan dukungan ke Wilayah Tiongkok untuk memfilter repositori mana yang direplikasi. Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR	15 Mei 2024

Perubahan	Deskripsi	Date
Menambahkan registri GitLab kontainer untuk menarik aturan cache	Amazon ECR menambahkan dukungan untuk membuat aturan cache pull through untuk registri GitLab kontainer. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR.	8 Mei 2024
Pembaruan kebijakan siklus hidup Amazon ECR untuk menambahkan dukungan untuk menggunakan wildcard	Amazon ECR menambahkan dukungan untuk wildcard dalam kebijakan siklus hidup melalui penggunaan `tagPatternList` parameter dalam aturan kebijakan siklus hidup. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.	18 Desember 2023
Templat pembuatan repositori Amazon ECR	Amazon ECR menambahkan dukungan untuk template pembuatan repositori. Untuk informasi selengkapnya, lihat Template untuk mengontrol repositori yang dibuat selama pull through cache, membuat saat push, atau tindakan replikasi.	15 November 2023
Amazon ECR menarik cache ditambahkan didukung untuk registri upstream yang diautentikasi	Amazon ECR menambahkan dukungan untuk menggunakan pendaftar upstream yang memerlukan otentikasi untuk aturan cache pull through Anda. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR.	15 November 2023
AWSECRPullThroughCache_ServiceRolePolicy— Perbarui ke kebijakan yang ada	Amazon ECR menambahkan izin baru ke kebijakan `AWSECRPullThroughCache_ServiceRolePolicy` . Izin ini memungkinkan Amazon ECR untuk mengambil konten terenkripsi dari rahasia Secrets Manager. Ini diperlukan saat menggunakan aturan pull through cache untuk menyimpan gambar dari registri hulu yang memerlukan otentikasi.	15 November 2023

Perubahan	Deskripsi	Date
Penandatanganan gambar Amazon ECR	Amazon ECR dan AWS Signer menambahkan dukungan untuk membuat dan mendorong tanda tangan gambar kontainer menggunakan klien Notaris. Untuk informasi selengkapnya, lihat Menandatangani gambar di Amazon ECR.	6 Juni 2023
Menambahkan registri kontainer Kubernetes untuk menarik aturan cache	Amazon ECR menambahkan dukungan untuk membuat aturan cache pull through untuk registri container Kubernetes. Untuk informasi selengkapnya, lihat Sinkronkan registri hulu dengan registri pribadi Amazon ECR.	1 Juni 2023
Amazon ECR meningkatkan dukungan durasi pemindaian	Amazon Inspector menambahkan dukungan untuk menyetel durasi penyimpanan Anda dipantau saat pemindaian yang disempurnakan diaktifkan. Untuk informasi selengkapnya, lihat Mengubah durasi pemindaian yang disempurnakan untuk gambar di Amazon Inspector.	28 Juni 2022
Amazon ECR mengirimkan metrik jumlah tarik repositori ke Amazon CloudWatch	Amazon ECR mengirimkan metrik jumlah tarik repositori ke Amazon. CloudWatch Untuk informasi selengkapnya, lihat Metrik repositori Amazon ECR.	6 Januari 2022
Dukungan replikasi yang diperluas	Amazon ECR menambahkan dukungan untuk memfilter repositori mana yang direplikasi. Untuk informasi selengkapnya, lihat Replikasi image privat di Amazon ECR.	21 September 2021
AWS kebijakan terkelola untuk Amazon ECR	Amazon ECR menambahkan dokumentasi kebijakan AWS terkelola. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk Amazon Elastic Container Registry.	24 Juni 2021

Perubahan	Deskripsi	Date
Replikasi Lintas-Wilayah dan Lintas-Akun	Amazon ECR menambahkan support untuk mengkonfigurasi pengaturan replikasi untuk registrasi privat Anda. Untuk informasi selengkapnya, lihat Pengaturan registri pribadi di Amazon ECR.	8 Desember 2020
Support artefak OCI	Amazon ECR menambahkan support untuk mendorong dan menarik artefak Open Container Initiative (OCI). Parameter `artifactMediaType` baru ditambahkan ke Respons `DescribeImages` API untuk menunjukkan jenis artefak. Untuk informasi selengkapnya, lihat Mendorong bagan Helm ke repositori pribadi Amazon ECR.	24 Agustus 2020
Enkripsi saat tidak aktif	Amazon ECR menambahkan support untuk mengkonfigurasi enkripsi untuk repositori Anda menggunakan enkripsi server-side dengan kunci terkelola pelanggan yang disimpan di AWS Key Management Service (AWS KMS). Untuk informasi selengkapnya, lihat Enkripsi saat diam.	29 Juli 2020
citra multi-arsitektur	Amazon ECR menambahkan support untuk membuat dan mendorong daftar Docker manifest yang digunakan untuk citra multi-arsitektur. Untuk informasi selengkapnya, lihat Mendorong gambar multi-arsitektur ke repositori pribadi Amazon ECR.	28 April 2020

Perubahan	Deskripsi	Date
Metrik Penggunaan Amazon ECR	Amazon ECR menambahkan metrik CloudWatch penggunaan yang memberikan visibilitas ke penggunaan sumber daya akun Anda. Anda juga memiliki kemampuan untuk membuat CloudWatch alarm dari konsol Service Quotas CloudWatch dan Service Quotas untuk mendapatkan peringatan ketika penggunaan Anda mendekati kuota layanan yang diterapkan. Untuk informasi selengkapnya, lihat Metrik penggunaan Amazon ECR.	28 Feb 2020
Service quotas Amazon ECR yang diperbarui	Service quotas Amazon ECR yang diiperbarui untuk menyertakan kuota per-API. Untuk informasi selengkapnya, lihat Kuota layanan Amazon ECR.	19 Feb 2020
Perintah `get-login-password` yang ditambahkan	Support yang ditambahkan untuk get-login-password, yang menyediakan metode sederhana dan aman untuk mengambil token otorisasi. Untuk informasi selengkapnya, lihat Menggunakan token otorisasi.	4 Feb 2020
Pemindaian citra	Support yang ditambahkan untuk pemindaian citra, membantu dalam mengidentifikasi kerentanan perangkat lunak dalam citra kontainer Anda. Amazon ECR menggunakan database Common Vulnerabilities and Exposures (CVEs) dari proyek CoreOS Clair open source dan memberi Anda daftar temuan pemindaian. Untuk informasi selengkapnya, lihat Pindai gambar untuk kerentanan perangkat lunak di Amazon ECR.	24 Okt 2019

Perubahan	Deskripsi	Date
Kebijakan VPC Endpoint	Support yang ditambahkan untuk menetapkan kebijakan IAM pada antarmuka VPC endpoint Amazon ECR. Untuk informasi selengkapnya, lihat Buat kebijakan titik akhir untuk VPC endpoint Amazon ECR.	26 Sep 2019
Tanda Ketetapan Citra	Support yang ditambahkan untuk mengkonfigurasi repositori agar tetap untuk mencegah ketimpaan tanda citra. Untuk informasi selengkapnya, lihat Mencegah tag gambar ditimpa di Amazon ECR.	25 Juli 2019
VPC endpoint Antarmuka (AWS PrivateLink)	Menambahkan dukungan untuk mengonfigurasi titik akhir VPC antarmuka yang didukung oleh. AWS Priva teLink Hal ini memungkinkan Anda untuk membuat koneksi privat antara VPC Anda dan Amazon ECR tanpa memerlukan akses melalui internet, melalui instans NAT, koneksi VPN, atau Direct Connect. Untuk informasi selengkapnya, lihat Titik akhir VPC antarmuka Amazon ECR (AWS PrivateLink).	25 Jan 2019
Penandaan sumber daya	Amazon ECR menambahkan support untuk menambahkan tanda metadata ke repositori Anda. Untuk informasi selengkapnya, lihat Menandai repositori pribadi di Amazon ECR.	18 Des 2018
Peggantian Nama Amazon ECR	Nama Amazon Elastic Container Registry telah diganti (sebelumnya Amazon EC2 Container Registry).	21 Nov 2017

Perubahan	Deskripsi	Date
Kebijakan siklus hidup	Kebijakan siklus hidup Amazon ECR memungkinkan Anda untuk menentukan manajemen siklus hidup citra dalam repositori. Untuk informasi selengkapnya, lihat Mengotomatiskan pembersihan gambar dengan menggunakan kebijakan siklus hidup di Amazon ECR.	11 Okt 2017
Support Amazon ECR untuk Docker image manifest 2, skema 2	Kini Amazon ECR dapat digunakan pada Docker Image Manifest V2 Skema 2 (digunakan dengan Docker versi 1.10 dan yang lebih baru). Untuk informasi selengkapnya, lihat Dukungan format manifes gambar kontainer di Amazon ECR.	27 Jan 2017
Ketersediaan Umum Amazon ECR	Amazon Elastic Container Registry (Amazon ECR) adalah layanan registri Docker AWS terkelola yang aman, terukur, dan andal.	21 Des 2015

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