Manual do usuário

Amazon ECR

Versão da API 2015-09-21

Amazon ECR: Manual do usuário

Table of Contents

O que é o Amazon Elastic Container Registry?

O Amazon Elastic Container Registry (Amazon ECR) é AWS um serviço gerenciado de registro de imagens de contêineres que é seguro, escalável e confiável. O Amazon ECR oferece suporte a repositórios privados com permissões baseadas em recursos usando o IAM. AWS Isso é para que usuários ou instâncias do Amazon EC2 especificados possam acessar seus repositórios e imagens de contêiner. É possível usar a CLI preferida para enviar, extrair e gerenciar imagens do Docker, imagens da Open Container Initiative (OCI)e artefatos compatíveis com OCI.

> **Note**
>
> O Amazon ECR também suporta repositórios de imagens de contêiner público. Para obter mais informações, consulte O que é o Amazon ECR Public no Manual do usuário do Amazon ECR Public.

A equipe de serviços de AWS contêineres mantém um roteiro público em. GitHub Ele contém informações sobre o que as equipes estão trabalhando e permite que todos os AWS clientes forneçam feedback direto. Para obter mais informações, consulte Roteiro de contêineres da AWS.

Conceitos e componentes do Amazon ECR

O Amazon ECR é um serviço de registro de contêiner do Docker totalmente gerenciado fornecido pela AWS. Ele permite que você armazene, gerencie e implante imagens de contêineres do Docker de forma segura e confiável. Esses conceitos e componentes trabalham juntos para fornecer um serviço de registro de contêineres Docker seguro, escalável e confiável dentro do AWS, permitindo que você gerencie e implante com eficiência seus aplicativos em contêineres.

Aqui estão alguns dos principais conceitos e componentes do Amazon ECR:

Registro

Um registro do Amazon ECR é um repositório privado fornecido para cada AWS conta, onde você pode criar um ou mais repositórios. Esses repositórios permitem que você armazene e distribua imagens do Docker, imagens da Open Container Initiative (OCI) e outros artefatos compatíveis com o OCI em seu ambiente. AWS Para obter mais informações, consulte Registro privado do Amazon ECR.

Token de autorização

Seu cliente deve autenticar-se em um registro privado do Amazon ECR como um usuário AWS antes de poder enviar e receber imagens. Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.

Repositório

Um repositório no Amazon ECR é uma coleção lógica na qual você pode armazenar as imagens do Docker, imagens do Open Container Initiative (OCI) e outros artefatos compatíveis com o OCI. Em um único registro do Amazon ECR, você pode ter vários repositórios para organizar as imagens de contêineres. Para obter mais informações, consulte Repositórios privados do Amazon ECR.

Política de repositório

Você pode controlar o acesso aos repositórios e ao conteúdo contido neles com as políticas de repositório. Para obter mais informações, consulte Políticas de repositório privado no Amazon ECR.

Imagem

É possível enviar e extrair imagens de contêiner dos seus repositórios. Você pode usar essas imagens localmente no seu sistema de desenvolvimento ou nas definições de tarefas do Amazon ECS e especificações de pod do Amazon EKS. Para obter mais informações, consulte Uso de imagens do Amazon ECR com o Amazon ECS e Uso de imagens do Amazon ECR com o Amazon EKS.

Política de ciclo de vida

As políticas de ciclo de vida do Amazon ECR permitem que você gerencie o ciclo de vida das imagens definindo regras para reduzir e expirar imagens antigas ou não utilizadas. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.

Verificação de imagens

O Amazon ECR fornece um recurso integrado de verificação de imagens que ajuda a identificar vulnerabilidades de software nas imagens do contêineres. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no Amazon ECR.

Controle de acesso

O Amazon ECR usa o IAM para controlar o acesso aos repositórios. Você pode criar usuários, grupos e perfis do IAM com permissões específicas para enviar por push, extrair ou gerenciar

repositórios do Amazon ECR. Para obter mais informações, consulte Segurança no Amazon Elastic Container Registry.

Replicação entre regiões e entre contas

O Amazon ECR oferece suporte à replicação de imagens em várias AWS contas e regiões para aumentar a disponibilidade e reduzir a latência. Para obter mais informações, consulte Replicação de imagem privada no Amazon ECR.

Criptografia

O Amazon ECR é compatível com criptografia do lado do servidor das imagens do Docker em repouso usando o AWS KMS. Para obter mais informações, consulte Proteção de dados no Amazon ECR.

AWS Command Line Interface Integration

AWS CLI Ele fornece comandos para interagir com os repositórios do Amazon ECR, como criar, listar, enviar e extrair imagens.

Console de gerenciamento da AWS

O Amazon ECR também pode ser gerenciado por meio do Console de gerenciamento da AWS, fornecendo uma interface web fácil de usar para trabalhar com seus repositórios e imagens.

AWS CloudTrail

O Amazon ECR se integra AWS CloudTrail, permitindo que você registre e audite chamadas de API feitas para o Amazon ECR para fins de segurança e conformidade. Para obter mais informações, consulte Registrando ações do Amazon ECR com AWS CloudTrail.

Amazon CloudWatch

O Amazon ECR fornece métricas e logs que podem ser monitorados usando o Amazon CloudWatch, permitindo que você acompanhe a performance e o uso dos repositórios do Amazon ECR. Para obter mais informações, consulte Métricas do repositório do Amazon ECR.

Assinatura gerenciada

A assinatura gerenciada gera automaticamente assinaturas criptográficas quando as imagens são enviadas para o Amazon ECR, simplificando a assinatura de imagens de contêineres. Para obter mais informações, consulte Assinatura gerenciada.

Casos de uso comuns no Amazon ECR

O Amazon ECR é um serviço de registro de contêiner do Docker totalmente gerenciado fornecido pela AWS. Ele fornece um repositório seguro e escalável para armazenar e distribuir imagens de contêineres do Docker, tornando-o um componente essencial em implantações de aplicações em contêineres. O Amazon ECR simplifica o processo de criação, distribuição e execução de aplicativos em contêineres em vários AWS serviços e ambientes locais.

Veja alguns dos principais casos de uso do Amazon ECR:

Armazenamento e distribuição de imagens de contêineres

O Amazon ECR serve como um repositório centralizado para armazenar e distribuir imagens de contêineres do Docker em uma organização ou para consumo público. Os desenvolvedores podem enviar suas imagens de contêiner para o Amazon ECR e, em seguida, extraí-las de qualquer ambiente computacional interno AWS, como Amazon EC2 ou Amazon AWS Fargate EKS. Para obter mais informações, consulte Repositórios privados do Amazon ECR.

Integração e implantação contínuas (CI/CD)

O Amazon ECR se integra perfeitamente com AWS CodeBuild,, e outras CI/CD ferramentas AWS CodePipeline, permitindo a criação, o teste e a implantação automatizados de aplicativos em contêineres. As imagens do contêiner podem ser enviadas automaticamente para o Amazon ECR como parte do CI/CD pipeline, garantindo uma implantação consistente e confiável em diferentes ambientes.

Arquitetura de microsserviços

O Amazon ECR é adequado para arquiteturas de microsserviços, em que as aplicações são divididas em serviços menores e desacopladas, empacotadas como contêineres. Cada microsserviço pode ter sua própria imagem de contêiner armazenada no Amazon ECR, permitindo o desenvolvimento, a implantação e a escalabilidade independentes de serviços individuais.

Implantações híbridas e multinuvem

O Amazon ECR é compatível com a capacidade de extrair imagens de contêineres de outros registros de contêineres, como Docker Hub ou registros de terceiros. Isso permite que as organizações mantenham um modelo de implantação consistente em ambientes híbridos ou multinuvem, usando o Amazon ECR como repositório central para imagens de contêineres.

Segurança e controle de acesso

O Amazon ECR fornece mecanismos de controle de acesso refinados, permitindo que as organizações controlem quem pode enviar por push ou extrair imagens de contêineres do registro. Ele também se integra AWS Identity and Access Management para autenticação e autorização, garantindo acesso seguro às imagens do contêiner. Para obter mais informações, consulte Segurança no Amazon Elastic Container Registry.

Verificação de vulnerabilidades em imagens

O Amazon ECR oferece verificação automática de imagens de contêineres em busca de vulnerabilidades de software e possíveis erros de configuração, ajudando a manter um ambiente de contêiner seguro e compatível. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no Amazon ECR.

Registro de contêineres privados

Para organizações com requisitos rígidos de segurança ou conformidade, o Amazon ECR pode ser usado como um registro privado de contêineres, garantindo que imagens confidenciais de contêineres não sejam expostas a registros públicos e sejam acessíveis somente dentro do ambiente da AWS organização. Para obter mais informações, consulte Registro privado do Amazon ECR.

Implantação de aplicações distribuídas globalmente com a replicação do Amazon ECR

Aproveitando a capacidade de replicação do Amazon ECR, você pode centralizar suas imagens de aplicativos web em contêineres em um repositório principal, permitindo a distribuição automatizada em várias AWS regiões, garantindo implantações globais consistentes com baixa latência em todo o mundo e reduzindo a carga operacional. Para obter mais informações, consulte Replicação de imagem privada no Amazon ECR.

Limpeza automatizada de imagens obsoletas de contêineres

As políticas de ciclo de vida do Amazon ECR permitem a limpeza automatizada de imagens de contêineres obsoletas com base em regras definidas, como idade, contagem ou tags, otimizando os custos de armazenamento, mantendo um registro organizado, aprimorando a segurança e a conformidade e simplificando os fluxos de trabalho de desenvolvimento por meio da automação. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.

Recursos do Amazon ECR

O Amazon ECR fornece os seguintes recursos:

- As políticas de ciclo de vida ajudam a gerenciar o ciclo de vida das imagens em seus repositórios. Você define regras que resultam na limpeza das imagens não utilizadas. Você pode testar as regras antes de aplicá-las ao repositório. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.
- A verificação de imagens ajuda a identificar vulnerabilidades de software nas imagens de seu contêiner. Cada repositório pode ser configurado para verifica no envio. Isso garante que cada nova imagem enviada para o repositório seja verificada. Em seguida, você pode recuperar os resultados da verificação das imagens. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no Amazon ECR.
- A replicação entre regiões e entre contas torna mais fácil para você ter suas imagens onde precisa. Isso é definido como uma configuração do registro e é feito por região. Para obter mais informações, consulte Configurações de registro privado no Amazon ECR.
- As regras de cache de pull through fornecem uma maneira de armazenar em cache repositórios em um registro upstream no seu registro privado do Amazon ECR. Usando uma regra de cache pull through, o Amazon ECR entrará em contato com o registro upstream periodicamente para garantir que a imagem armazenada em cache no seu registro privado do Amazon ECR esteja atualizada. Para obter mais informações, consulte Sincronizar um registro upstream com um registro privado do Amazon ECR.
- Os modelos de criação de repositórios permitem que você defina as configurações dos repositórios criados pelo Amazon ECR em seu nome durante ações de pull through cache, create on push ou replicação. Você pode especificar imutabilidade de tags, configuração de criptografia, políticas de repositório, políticas de ciclo de vida e tags de recursos para repositórios criados automaticamente. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.
- A assinatura gerenciada gera automaticamente assinaturas criptográficas quando as imagens são enviadas para o Amazon ECR, simplificando a assinatura de imagens de contêineres. Para obter mais informações, consulte Assinatura gerenciada.

Como começar a usar o Amazon ECR

Se você estiver usando o Amazon Elastic Container Service (Amazon ECS) ou o Amazon Elastic Kubernetes Service (Amazon EKS), observe que a configuração desses dois serviços é semelhante à do Amazon ECR, pois ele é uma extensão de ambos os serviços.

Ao usar o AWS Command Line Interface com o Amazon ECR, use uma versão do AWS CLI que suporte os recursos mais recentes do Amazon ECR. Se você não encontrar suporte para um recurso do Amazon ECR no AWS CLI, atualize para a versão mais recente do AWS CLI. Para obter informações sobre como instalar a versão mais recente do AWS CLI, consulte Instalar ou atualizar para a versão mais recente do AWS CLI no Guia do AWS Command Line Interface Usuário.

Para saber como enviar por push uma imagem de contêiner para um repositório privado do Amazon ECR usando a AWS CLI e o Docker, consulte. Mover uma imagem ao longo do seu ciclo de vida no Amazon ECR

Preços do Amazon ECR

Com o Amazon ECR, você paga pela quantidade de dados que armazena em seus repositórios, pela transferência de dados de seus envios e retirados de imagens e pelas ações de imagem pelas quais você opta, como assinatura e replicação de imagens. Para obter mais informações, consulte a Definição de preço do Amazon ECR.

Mover uma imagem ao longo do seu ciclo de vida no Amazon ECR

Se você estiver usando o Amazon ECR pela primeira vez, use as etapas a seguir com a CLI do Docker e a AWS CLI para criar uma imagem de amostra, autenticar-se no registro padrão e criar um repositório privado. Em seguida, envie uma imagem por push e extraia uma imagem do repositório privado. Quando você terminar de usar a imagem de exemplo, exclua a imagem de exemplo e o repositório.

Para usar o Console de gerenciamento da AWS em vez do AWS CLI, consultethe section called "Criar um repositório para armazenar imagens".

Para obter mais informações sobre as outras ferramentas disponíveis para gerenciar seus AWS recursos, incluindo os diferentes AWS SDKs, kits de ferramentas do IDE e ferramentas de linha de PowerShell comando do Windows, consulte. http://aws.amazon.com/tools/

Pré-requisitos

Se você não tiver o Docker mais recente AWS CLI instalado e pronto para uso, use as etapas a seguir para instalar essas duas ferramentas.

Instale o AWS CLI

Para usar o AWS CLI com o Amazon ECR, instale a AWS CLI versão mais recente. Para obter informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface .

Instalar o Docker

O Docker está disponível em muitos sistemas operacionais diferentes, incluindo a maioria das distribuições modernas do Linux, como o Ubuntu, e até no MacOS e no Windows. Para obter mais informações sobre como instalar o Docker no seu sistema operacional, consulte o Guia de instalação do Docker.

Não é necessário um sistema de desenvolvimento local para usar o Docker. Se você já usa o Amazon EC2, pode iniciar uma instância do Amazon Linux 2023 e instalar o Docker para começar.

Se você já tiver um Docker instalado, vá para Etapa 1: criar uma imagem do Docker.

Para instalar o Docker em uma instância do Amazon EC2 usando uma AMI do Amazon Linux 2023

1. Inicie uma instância com a mais recente AMI do Amazon Linux 2023. Para obter mais informações, consulte Iniciar uma instância no Manual do usuário do Amazon EC2.
2. Conecte-se à sua instância. Para obter mais informações, consulte Conectar-se à sua instância do Linux no Manual do usuário do Amazon EC2.
3. Atualize os pacotes instalados e o cache de pacotes em sua instância.

```
sudo yum update -y
```

4. Instale o pacote do Docker Community Edition mais recente.

```
sudo yum install docker
```

5. Inicie o serviço Docker.

```
sudo service docker start
```

6. Adicione o `ec2-user` ao grupo `docker`, de modo que você possa executar comandos do Docker sem usar o `sudo`.

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

7. Faça logout e login novamente para selecionar as novas permissões do grupo `docker`. É possível fazer isso ao fechar a janela de terminal SSH atual e se reconectar à sua instância em outra janela. Sua nova sessão SSH terá as permissões de grupo `docker` apropriadas.
8. Verifique se o `ec2-user` pode executar comandos do Docker sem `sudo`.

```
docker info
```

> **Note**
>
> Em alguns casos, pode ser necessário reinicializar sua instância para fornecer permissões para o `ec2-user` acessar o daemon do Docker. Tente reinicializar sua instância se você vir o seguinte erro:

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

Etapa 1: criar uma imagem do Docker

Nesta etapa, crie uma imagem do Docker de uma aplicação Web simples e teste-a no sistema ou na instância do Amazon EC2 local.

Para criar uma imagem do Docker de um aplicativo web simples

1. Crie um arquivo chamado `Dockerfile`. Um Dockerfile é um manifesto que descreve a imagem básica a ser usada para a sua imagem do Docker e o que você deseja instalar e executar nela. Para obter mais informações sobre a Dockerfiles, visite Referência de Dockerfiles.

   ```
   touch Dockerfile
   ```

2. Edite o `Dockerfile` que você acabou de criar e adicione o conteúdo a seguir.

   ```
   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
   ```

Esse Dockerfile usa a imagem pública do Amazon Linux 2 hospedada no Amazon ECR Public. As instruções `RUN` atualizam os caches de pacotes, instalam alguns pacotes de software para o servidor Web e, em seguida, gravam o conteúdo de “Hello World!” na raiz do documento dos servidores Web. A instrução `EXPOSE` expõe a porta 80 do contêiner e a instrução `CMD` inicia o servidor Web.

3. Crie a imagem do Docker do seu Dockerfile.

> **Note**
>
> Algumas versões do Docker podem exigir o caminho completo para o seu Dockerfile no seguinte comando, em vez de o caminho relativo mostrado abaixo.

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

4. Liste a sua imagem do contêiner.

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

Saída:

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

5. Execute a imagem recém-criada. A opção `-p 80:80` mapeia a porta 80 exposta no contêiner para a porta 80 no sistema de host. Para obter mais informações sobre o docker run, acesse a Referência de execução do Docker.

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

> **Note**
>
> A saída do servidor Web Apache é exibida na janela do terminal. É possível ignorar a mensagem "`Could not reliably determine the fully qualified domain name`".

6. Abra um navegador e aponte para o servidor que está executando o Docker e hospedando seu contêiner.

   - Se você estiver usando uma instância do EC2, esse é o valor Public DNS para o servidor, que é o mesmo endereço usado para se conectar à instância com o SSH. Certifique-se de que o security group para sua instância permita o tráfego de entrada na porta 80.
   - Se você estiver executando o Docker localmente, aponte seu navegador para o. http://localhost/
   - Se você estiver usando docker-machine em um computador Windows ou Mac, encontre o endereço IP da VirtualBox VM que está hospedando o Docker com o docker-machine ip comando, substituindo-o *`machine-name`* pelo nome da máquina docker que você está usando.

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

   Você deve ver uma página da Web com seu “Hello, World!” instrução.

7. Interrompa o contêiner do Docker digitando Ctrl+c.

Etapa 2: criar um repositório

Agora que tem uma imagem para enviar ao Amazon ECR, você precisa criar um repositório para guardá-la. Neste exemplo, você cria um repositório chamado `hello-repository` para o qual enviará a imagem `hello-world:latest` posteriormente. Para criar um repositório, execute o seguinte comando:

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

Etapa 3: autenticar-se no registro padrão

Depois de instalar e configurar o AWS CLI, autentique a CLI do Docker em seu registro padrão. Desta forma, o comando docker pode enviar e extrair imagens com o Amazon ECR. O AWS CLI fornece um get-login-password comando para simplificar o processo de autenticação.

> **Note**
>
> Seu diretor do IAM deve ter `ecr:GetAuthorizationToken` permissão para se autenticar em um registro. Para obter mais informações, consulte AWS políticas gerenciadas para o Amazon Elastic Container Registry.

Para autenticar o Docker em um registro do Amazon ECR com get-login-password, execute o comando aws ecr get-login-password. Ao transmitir o token de autenticação para o comando docker login, use o valor `AWS` para o nome de usuário, e especifique o URI de registro do Amazon ECR para o qual deseja fazer a autenticação. Se autenticar em vários registros, você deverá repetir o comando para cada registro.

> **Important**
>
> Se você receber um erro, instale ou atualize para a versão mais recente da AWS CLI. Para obter mais informações, consulte Installing the AWS Command Line Interface (Instalar a AWS Command Line Interface) no User Guide (Guia do usuário da).

- 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
```

Etapa 4: enviar uma imagem ao Amazon ECR

Agora você pode enviar a imagem ao repositório do Amazon ECR que criou na seção anterior. Use a CLI do docker para enviar imagens por push após os seguintes pré-requisitos serem atendidos:

- A versão mínima do docker está instalada: 1.7.
- O token de autorização do Amazon ECR foi configurado com docker login.
- O repositório do Amazon ECR existe, e o usuário tem acesso para enviar imagens ao repositório.

Depois que esses pré-requisitos forem atendidos, você poderá enviar a imagem ao repositório recém-criado no registro padrão da sua conta.

Para marcar e enviar uma imagem para o Amazon ECR

1. Liste as imagens que você armazenou localmente para identificar a imagem a ser marcada e enviada.

   ```
   docker images
   ```

   Saída:

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

2. Marque a imagem a ser enviada ao seu repositório.

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

3. Envie a imagem.

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

   Saída:

   ```
   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
```

Etapa 5: extrair uma imagem do Amazon ECR

Depois que a imagem for enviada por push ao repositório do Amazon ECR, você poderá extraí-la de outros locais. Use a CLI do docker para extrair imagens após os seguintes pré-requisitos serem atendidos:

- A versão mínima do docker está instalada: 1.7.
- O token de autorização do Amazon ECR foi configurado com docker login.
- O repositório do Amazon ECR existe, e o usuário tem acesso para extrair imagens do repositório.

Depois que esses pré-requisitos forem atendidos, você poderá extrair a imagem. Para extrair a imagem de exemplo do Amazon ECR, execute o seguinte comando:

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

Saída:

```
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
```

Etapa 6: excluir uma imagem

Caso não precise mais de uma imagem em um dos repositórios, você poderá excluí-la. Para excluir uma imagem, especifique o repositório em que ela está e um valor de `imageTag` ou `imageDigest`

para imagem. O exemplo abaixo exclui uma imagem no repositório `hello-repository` com a tag de imagem `latest`. Para excluir a imagem de exemplo do repositório, execute o seguinte comando:

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

Etapa 7: excluir um repositório

Caso não precise mais de um repositório inteiro de imagens, você pode excluí-lo. O exemplo a seguir usa o sinalizador `--force` para excluir um repositório que contenha imagens. Para excluir um repositório que contém imagens (e todas as imagens contidas nele), execute o seguinte comando:

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

Otimização do performance para o Amazon ECR

Você pode usar as recomendações a seguir sobre configurações e estratégias para otimizar a performance ao usar o Amazon ECR.

Use o Docker 1.10 e versões posteriores para utilizar os uploads simultâneos da camada

As imagens de Docker são compostas por camadas, que são estágios de compilação intermediários da imagem. Cada linha em um Dockerfile resulta na criação de uma nova camada. Quando você usa o Docker 1.10 e versões posteriores, o Docker envia por padrão o maior número possível de camadas como carregamentos simultâneos ao Amazon ECR, o que resulta em tempos de carregamento mais rápidos.

Use uma imagem de base menor

As imagens padrão disponíveis por meio do Docker Hub podem conter muitas dependências das quais seu aplicativo não precisa. Considere o uso de uma imagem menor criada e mantida por outras pessoas da comunidade do Docker ou compile sua própria imagem de base usando a imagem mínima de scratch do Docker. Para obter mais informações, consulte Criar uma imagem de base na documentação do Docker.

Coloque as dependências que mudam menos no início do Dockerfile

O Docker armazena as camadas em cache, o que acelera os tempos de compilação. Se nada tiver sido alterado na camada desde a última compilação, o Docker usará a versão armazenada em cache, em vez de compilar a camada novamente. No entanto, cada camada depende das camadas que vieram antes dela. Se uma camada mudar, o Docker a compilará novamente, bem como todas as camadas que vierem depois dela.

Para minimizar o tempo necessário para compilar um arquivo de Dockerfile e fazer upload das camadas novamente, considere colocar as dependências que mudam com menos frequência no início do Dockerfile. E, aquelas que mudam rapidamente, (como o código-fonte do seu aplicativo) mais à frente na pilha.

Encadeie os comandos para evitar o armazenamento desnecessário de arquivos

Os arquivos intermediários criados em uma camada continuarão fazendo parte dela, mesmo que sejam excluídos em uma camada subsequente. Considere o seguinte exemplo:

```
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
```

Neste exemplo, as camadas criadas pelo primeiro e pelo segundo comando EXECUTAR contêm o arquivo original .tar.gz e todos os seus conteúdos descompactados. Embora o arquivo .tar.gz seja excluído pelo quarto comando EXECUTAR. Esses comandos podem ser encadeados em uma única instrução EXECUTAR para garantir que esses arquivos desnecessários não façam parte da imagem de Docker final:

```
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
```

Use o endpoint regional mais próximo

Você pode reduzir a latência na extração de imagens do Amazon ECR usando o endpoint regional mais próximo de onde seu aplicativo está sendo executado. Se seu aplicativo estiver sendo executado em uma instância do Amazon EC2, você pode usar o seguinte código de shell para obter a região da zona de disponibilidade da instância:

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

A região pode ser transmitida aos comandos da AWS CLI com o parâmetro --region ou definida como a região padrão de um perfil com o comando aws configure. Você também pode definir a região ao fazer chamadas usando o AWS SDK . Para obter mais informações, consulte a documentação do SDK para a sua linguagem de programação específica.

Fazer solicitações aos registros do Amazon ECR

Você pode enviar, extrair, excluir, visualizar e gerenciar imagens OCI, imagens Docker e artefatos compatíveis com OCI nos registros privados do Amazon ECR usando IPv4 apenas endpoints ou endpoints de pilha dupla (e). IPv4 IPv6 Para fazer solicitações de IPv4 redes, você pode usar pilha dupla ou endpoints. IPv4 Para fazer solicitações de uma IPv6 rede, use um endpoint de pilha dupla. Para obter mais informações sobre como fazer solicitações aos registros públicos do Amazon ECR usando IPv4 endpoints de pilha dupla, consulte Fazer solicitações aos registros públicos do Amazon ECR. Não há cobranças adicionais para acessar o Amazon ECR. IPv6 Para ter mais informações sobre preços, consulte Preços do Amazon Elastic Container Registry.

Os endpoints do Amazon ECR são designados por atributos além do suporte IPv4 exclusivo para endpoints ou endpoints de pilha dupla. Esses atributos podem incluir:

- Região – Cada endpoint é específico de uma região.
- Tipo — A seleção do endpoint depende se você está usando as interfaces de linha de comando Docker e compatíveis com AWS SDK ou OCI.
- Segurança – Em regiões selecionadas, o Amazon ECR oferece endpoints compatíveis com FIPS. Para obter uma lista de endpoints do Amazon ECR compatíveis com FIPS, consulte Federal Information Processing Standard (FIPS) 140-3.

Para obter mais informações sobre endpoints de serviço suportados pelo cliente dual-stack IPv4, Docker e OCI, que processa chamadas de API do Amazon ECR a partir da CLI AWS , consulte Service endpoints. AWS SDKs

Começando a fazer solicitações IPv6

Para fazer uma solicitação para um registro do Amazon ECR IPv6, você precisa usar um endpoint de pilha dupla. Antes de acessar um registro do Amazon ECR IPv6, verifique os seguintes requisitos:

- Seu cliente e sua rede devem oferecer suporte IPv6.
- O Amazon ECR oferece suporte aos seguintes tipos de solicitação em IPv6:
  - Solicitações de clientes OCI e Docker:

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

- AWS Solicitações de API:

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

- Você deve atualizar qualquer política AWS Identity and Access Management (IAM) ou de registro que use a filtragem de endereço IP de origem para incluir intervalos de IPv6 endereços. Para obter mais informações, consulte Usando IPv6 endereços nas políticas do IAM.
- Quando você usa IPv6, os registros de acesso ao servidor exibem `Remote IP` endereços em IPv6 formato. Atualize suas ferramentas, scripts e software existentes para analisar esses endereços IPv6 IP formatados.

> **Note**
>
> Se você tiver problemas relacionados à presença de IPv6 endereços nos arquivos de log, entre em contato com AWS Support.

Testar a compatibilidade com endereços IP

Se você estiver usando o use Linux/Unix ou o Mac OS X, poderá testar se é possível acessar um endpoint de pilha dupla IPv6 usando o `curl` comando, conforme mostrado no exemplo a seguir:

Example

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

Você recebe de volta informações semelhantes ao exemplo a seguir. Se você estiver conectado IPv6 pelo endereço IP conectado, será um IPv6 endereço.

```
* 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
```

Se você estiver usando o Microsoft Windows 7 ou o Windows 10, poderá testar se é possível acessar um endpoint de pilha dupla usando IPv4 ou IPv6 usando o `ping` comando, conforme mostrado no exemplo a seguir.

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

Fazer solicitações IPv6 usando endpoints de pilha dupla

Você pode fazer chamadas de API do Amazon ECR IPv6 usando endpoints de pilha dupla. A funcionalidade e o desempenho das operações da API do Amazon ECR permanecem consistentes, independentemente de você usar IPv4 ou IPv6.

Ao usar o AWS Command Line Interface (AWS CLI) e AWS SDKs, você pode habilitar IPv6 usando um parâmetro ou sinalizador para alternar para um endpoint de pilha dupla ou especificando diretamente o endpoint de pilha dupla em seu arquivo de configuração para substituir o endpoint padrão do Amazon ECR. Você também pode fazer alterações de configuração usando um comando, que define `use_dualstack_endpoint` como verdadeiro no perfil padrão. Para obter mais informações sobre `use_dualstack_endpoint`, consulte Endpoints FIPS e de pilha dupla.

Example Fazendo alterações na configuração usando um comando

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

Example Fazendo solicitações sobre o IPv6 uso AWS CLI

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

Usar endpoints do Amazon ECR a partir da CLI do Docker

Depois de entrar no repositório do Amazon ECR e marcar sua imagem, você pode enviar e extrair imagens OCI e imagens do Docker de e para os registros do Amazon ECR. Os exemplos a seguir demonstram os comandos de envio e extração do Docker com ambos os endpoints de pilha dupla.

Example Enviando imagens do docker usando o endpoint IPv4

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

Example Enviar imagens do Docker usando o endpoint de pilha dupla

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

Example Extraindo imagens do docker usando o endpoint IPv4

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

Example Extrair imagens do Docker usando o endpoint de pilha dupla

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

Usando IPv6 endereços nas políticas do IAM

Antes de acessar um registro usando IPv6, certifique-se de que seu usuário do IAM e as políticas de registro do Amazon ECR que usam a filtragem de endereços IP incluam intervalos de IPv6 endereços. Se as políticas de filtragem de endereços IP não forem atualizadas para lidar com IPv6 endereços, os clientes poderão perder ou obter acesso incorretamente ao registro quando começarem a usar. IPv6 Para obter mais informações sobre como gerenciar permissões de acesso com o IAM, consulte Gerenciamento de Identidade e Acesso para o Amazon Elastic Container Registry.

As políticas do IAM que filtram endereços IP usam Operadores de condição de endereço IP. O exemplo de política de registro a seguir mostra como identificar o `54.240.143.*` intervalo de IPv4 endereços permitidos usando operadores de condição de endereço IP. Todos os endereços IP fora deste intervalo terão o acesso ao registro (`exampleregistry`) negado. Como todos os IPv6 endereços estão fora do intervalo permitido, essa política impede que IPv6 os endereços sejam acessados`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"}
      }
    }
```

```
  ]
}
```

Para permitir os intervalos de endereços IPv4 IPv6 (`54.240.143.0/24`2001:DB8:1234:5678::/64) e (), modifique o elemento Condição da política de registro conforme mostrado no exemplo a seguir. Você pode usar esse formato de bloqueio `Condition` para atualizar as políticas de usuário do IAM e de registro.

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

Important

Antes de usar, IPv6 você deve atualizar todas as políticas relevantes de usuário e registro do IAM que usam filtragem de endereço IP. Não recomendamos usar filtros de endereços IP em políticas de registro.

Você pode revisar suas políticas de usuário do IAM usando o console do IAM em https://console.aws.amazon.com/iam/. Para obter mais informações sobre o IAM, consulte o Guia do usuário do IAM.

Registro privado do Amazon ECR

Um registro privado do Amazon ECR hospeda as imagens de contêiner em uma arquitetura altamente disponível e escalável. É possível usar o seu registro privado para gerenciar repositórios de imagens privados que consistem em imagens do Docker e da Open Container Initiative (OCI). Cada conta da AWS é fornecida com um registro privado padrão do Amazon ECR. Para obter mais informações sobre registros públicos do Amazon ECR, consulte Registros públicos no Manual do usuário do Amazon Elastic Container Registry.

Conceitos do registro privado

- O URL do seu registro privado padrão é `https://aws_account_id.dkr.ecr.region.amazonaws.com`.
- Por padrão, sua conta tem acesso de leitura e gravação aos repositórios no seu registro privado padrão. No entanto, os usuários precisam de permissões para fazer chamadas para as APIs do Amazon ECR e para enviar por push e extrair imagens de repositórios privados. O Amazon ECR fornece várias políticas gerenciadas da para controlar o acesso do usuário em diversos níveis. Para obter mais informações, consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry.
- Você deve autenticar seu cliente do Docker em um registro privado para poder usar os comandos docker push e docker pull para enviar para, e extrair imagens dos, repositórios nesse registro. Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.
- Os repositórios privados podem ser controlados com políticas de acesso de usuários do e políticas de repositório. Para obter mais informações sobre políticas de repositórios, consulte Políticas de repositório privado no Amazon ECR.
- Os repositórios em seu registro privado podem ser replicados entre as regiões da AWS em seu próprio registro privado e entre contas separadas. Para isso, configure a replicação para seu registro privado. Para obter mais informações, consulte Replicação de imagem privada no Amazon ECR.

Autenticação de registro privado no Amazon ECR

Você pode usar o Console de gerenciamento da AWS AWS CLI, o ou os AWS SDKs para criar e gerenciar repositórios privados. Você também pode usar esses métodos para realizar algumas ações em imagens, como listá-las ou excluí-las. Esses clientes usam métodos de AWS autenticação

padrão. Embora seja possível usar a API do Amazon ECR para enviar e extrair imagens, é muito mais provável que você use a CLI do Docker ou uma biblioteca do Docker específica para a linguagem.

A CLI do Docker não suporta métodos de autenticação nativos do IAM. É necessário realizar etapas adicionais para que o Amazon ECR possa autenticar e autorizar solicitações de extração e envio do Docker.

Os métodos de autenticação de registro a seguir estão detalhados nas seções a seguir estão disponíveis.

Uso de auxiliar de credenciais do Amazon ECR

O Amazon ECR fornece um auxiliar de credenciais do Docker que facilita o armazenamento e o uso de credenciais do Docker ao enviar e extrair imagens do Amazon ECR. Para obter as etapas de instalação e configuração, consulte Auxiliar de credenciais do Docker do Amazon ECR.

> **Note**
>
> No momento, o auxiliar de credencial do Amazon ECR Docker não oferece suporte a autenticação multifator (MFA).

Uso de um token de autorização

O escopo de permissão de um token de autorização corresponde ao do principal do IAM usado para recuperar o token de autenticação. Um token de autenticação é usado para acessar qualquer registro do Amazon ECR ao qual o principal do IAM tenha acesso e é válido por 12 horas. Para obter um token de autorização, você deve usar a operação da GetAuthorizationTokenAPI para recuperar um token de autorização codificado em base64 contendo o nome de usuário `AWS` e uma senha codificada. O AWS CLI `get-login-password` comando simplifica isso recuperando e decodificando o token de autorização, que você pode então canalizar para um docker login comando para autenticar.

Para autenticar o Docker para um registro privado do Amazon ECR com get-login

- Para autenticar o Docker em um registro do Amazon ECR com get-login-password, execute o comando aws ecr get-login-password. Ao transmitir o token de autenticação para o comando docker login, use o valor `AWS` para o nome de usuário, e especifique o URI de registro do

Amazon ECR para o qual deseja fazer a autenticação. Se autenticar em vários registros, você deverá repetir o comando para cada registro.

> **Important**
>
> Se você receber um erro, instale ou atualize para a versão mais recente da AWS CLI. Para obter mais informações, consulte Installing the AWS Command Line Interface (Instalar a AWS Command Line Interface) no User Guide (Guia do usuário da).

- 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
```

Uso da autenticação de API HTTP

O Amazon ECR suporta API HTTP de registro do Docker. No entanto, como o Amazon ECR é um registro privado, você deve fornecer um token de autorização com cada solicitação HTTP. Você pode adicionar um cabeçalho de autorização HTTP usando a `-H` opção for curl e passar o token de autorização fornecido pelo get-authorization-token AWS CLI comando.

Para autenticar com a API HTTP do Amazon ECR

1. Recupere um token de autorização com o AWS CLI e defina-o como uma variável de ambiente.

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

2. A fim de fazer a autenticação para a API, passe a variável `$TOKEN` para a opção `-H` de curl. Por exemplo, o comando a seguir lista as tags da imagem em um repositório do Amazon ECR. Para obter mais informações, consulte a documentação de referência da API HTTP de registro do Docker.

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

A saída é a seguinte:

```
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"]}
```

Configurações de registro privado no Amazon ECR

O Amazon ECR usa configurações do registro privado para configurar atributos no registro. As configurações de registro privado são definidas separadamente para cada região. Você pode usar as configurações do registro privado para configurar os seguintes recursos.

- Permissões de registro – Uma política de permissões de registro fornece controle sobre as permissões de replicação e de cache de pull through. Para obter mais informações, consulte Permissões de registro privado no Amazon ECR.
- Regras de cache de pull-through – Uma regra de cache de pull-through é usada para armazenar em cache imagens de um registro upstream no registro privado do Amazon ECR. Para obter mais informações, consulte Sincronizar um registro upstream com um registro privado do Amazon ECR.
- Configuração de replicação – A configuração de replicação é usada para controlar se seus repositórios são copiados entre Regiões da AWS ou Contas da AWS. Para obter mais informações, consulte Replicação de imagem privada no Amazon ECR.
- Modelos de criação de repositório – Um modelo de criação de repositório é usado para definir as configurações padrão a serem aplicadas quando novos repositórios são criados pelo Amazon ECR em seu nome. Por exemplo, repositórios criados por uma ação de cache de pull through, criação por push ou replicação. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.
- Configuração de verificação – Por padrão, seu registro está habilitado para verificação básica. Você pode habilitar a verificação avançada que fornece um modo de verificação automatizado e

contínuo que verifica as vulnerabilidades do sistema operacional e do pacote da linguagem de programação. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no Amazon ECR.

- Pull-time exclusão de atualização — Você pode configurar exclusões de atualização de tempo de extração para evitar que o último horário de extração seja atualizado para imagens específicas quando elas forem extraídas. Isso é útil para imagens usadas para testes ou para CI/CD fins em que você não deseja que o tempo de extração afete as decisões de política do ciclo de vida. Para obter mais informações, consulte Exclusões de atualizações em tempo integral.
- Configuração de montagem de blob — A configuração de montagem de blob é usada para controlar se os repositórios dentro do seu registro compartilham camadas comuns em vez de armazenar camadas duplicadas. Para obter mais informações, consulte Montagem de blobs no Amazon ECR.

Montagem de blobs no Amazon ECR

O Amazon ECR suporta um recurso chamado montagem de blobs para compartilhar camadas de imagens comuns entre repositórios dentro de um registro. Quando habilitados, os repositórios em um único registro podem referenciar camadas de outros repositórios no mesmo registro em vez de armazenar cópias duplicadas.

Quando a montagem de blobs do registro está ativada, o Amazon ECR verifica as camadas existentes em seu registro durante as operações push quando os parâmetros de montagem são incluídos. Se uma camada já existir em outro repositório dentro do mesmo registro, o Amazon ECR montará a camada existente em vez de fazer o upload de uma duplicata.

> **Note**
>
> Os clientes OCI incluem automaticamente parâmetros de montagem se detectarem que um blob pode já existir em um repositório diferente. O Amazon ECR tenta montar somente quando esses parâmetros estão presentes na solicitação POST do cliente.

Conceitos de montagem de bolhas

- A montagem de blobs só funciona no mesmo registro (mesma conta e região).
- Os repositórios devem usar chaves e tipos de criptografia idênticos.

- A montagem de blobs não é suportada para imagens criadas por meio do cache pull through.
- Se você decidir desativar a montagem de blob, as imagens existentes que foram enviadas com a montagem de blob configurada continuarão funcionando e as camadas permanecerão montadas.

Configuração de montagem de blob

Você pode usar o Console de gerenciamento da AWS ou AWS CLI para configurar a montagem de blob para seu registro.

> **Note**
>
> Os usuários precisam da permissão `ecr:GetDownloadUrlForLayer` do IAM em um repositório para montar camadas a partir dele.

Console de gerenciamento da AWS

Use as etapas a seguir para atualizar a configuração de montagem de blobs do seu registro usando o. Console de gerenciamento da AWS

Ative a configuração de montagem de blobs para seu registro privado

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/private-registry/repositories
2. Na barra de navegação, selecione a região.
3. No painel de navegação, escolha Registro privado, Recursos e configurações e, em seguida, escolha Montagem de blobs.
4. Na página de montagem do Blob, escolha Ativar.

   Um banner é exibido indicando que a configuração de montagem do blob foi atualizada para ser ativada.

AWS CLI

Use o comando a seguir para atualizar a configuração de montagem de blobs do registro usando o. AWS CLI

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

Permissões de registro privado no Amazon ECR

O Amazon ECR usa uma política de registro para conceder permissões a uma entidade principal da AWS no nível de registro privado.

O Amazon ECR permite todas as ações de ECR na política e aplica a política de registro em todas as solicitações de ECR. Você pode usar políticas de registro para conceder permissões para ações como configuração de replicação, criação de regras de cache pull-through e criação de repositórios. Para ver a lista completa de ações de API, consulte o Guia de API do Amazon ECR. Para obter informações sobre configurações gerais para o registro privado do Amazon ECR, consulte Configurações de registro privado no Amazon ECR.

> **Note**
>
> Embora seja possível adicionar a ação `ecr:*` a uma política de registro privado, é considerada uma prática recomendada adicionar apenas as ações específicas necessárias com base no atributo que você está usando, em vez de usar um curinga.

Tópicos

Exemplos de política de registro privado do Amazon ECR

Os exemplos a seguir mostram instruções de políticas de registro que você pode usar para controlar as permissões que os usuários têm em seu registro do Amazon ECR.
  ```

> **Note**
>
> Em cada exemplo, se a ação `ecr:CreateRepository` for removida da política do registro, a replicação ainda pode ocorrer. No entanto, para uma replicação ser bem-sucedida, você precisa criar repositórios com o mesmo nome em sua conta.

Exemplo: permitir que todos as entidades principais do IAM em uma conta de origem repliquem todos os repositórios

A política de permissões de registro a seguir permite que as entidades principais do IAM (usuários e funções) de uma conta de origem repliquem todos os repositórios.

Observe o seguinte:

- Importante: ao especificar uma ID da Conta da AWS como entidade principal em uma política, você concede acesso a todos os usuários e funções do IAM dentro dessa conta, não apenas ao usuário-raiz. Isso fornece amplo acesso em toda a conta.
- Consideração de segurança: Account-level as permissões concedem acesso a todas as entidades do IAM na conta especificada. Para um acesso mais restritivo, especifique usuários, funções ou instruções de condição de uso individuais do IAM para limitar ainda mais o acesso.

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/*"
```

```
            ]
        }
    ]
}
```

Exemplo: permitir entidades principais do IAM de várias contas

A política de permissões de registro a seguir tem duas declarações. Cada instrução permite que todas as entidades principais do IAM (usuários e funções) em uma conta de origem repliquem todos os repositórios.

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/*"
            ]
        }
    ]
}
```

Exemplo: permitir que todas as entidades principais do IAM em uma conta de origem repliquem todos os repositórios com o prefixo **`prod-`**.

A política de permissões de registro a seguir permite que as entidades principais do IAM (usuários e funções) de uma conta de origem repliquem todos os repositórios que começam com `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-*"
            ]
        }
    ]
}
```

Conceder permissões de registro para replicação entre contas no Amazon ECR

O tipo de política entre contas é usado para conceder permissões a um AWS principal, permitindo a replicação dos repositórios de um registro de origem para o seu registro. Por padrão, você tem

permissão para configurar a replicação entre regiões no seu próprio registro. Só é necessário configurar a política de registro se estiver concendendo outra permissão de conta para replicar conteúdo para o seu registro.

Uma política de registro deve conceder permissão para a ação de API `ecr:ReplicateImage`. Essa API é uma API interna do Amazon ECR que pode replicar imagens entre regiões ou contas. Você também pode conceder a permissão `ecr:CreateRepository`, que permite que o Amazon ECR crie repositórios em seu registro se eles ainda não existirem. Se a permissão `ecr:CreateRepository` não for fornecida, um repositório com o mesmo nome que o repositório de origem deve ser criado manualmente no seu registro. Se nenhuma das duas alternativas foi realizada, a replicação falha. Qualquer falha `CreateRepository` ou ação `ReplicateImage` da API aparece no CloudTrail.

Para configurar uma política de permissões para replicação (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região para configurar a sua política de registro.
3. No painel de navegação, escolha Registro privado, escolha Atributos e configurações e depois Permissões.
4. Na página Registry permissions (Permissões do registro), escolha Generate statement (Gerar declaração).
5. Realize as seguintes etapas para definir a instrução da política usando o gerador de políticas.
    a. Para Tipo de política, escolha Política entre contas.
    b. Para ID da instrução, insira uma ID de instrução exclusiva. Este campo é usado como o `Sid` na política de registro.
    c. Para Accounts (Contas), insira os IDs de conta para cada conta à qual você deseja conceder permissões. Ao especificar vários IDs de conta, separe-os com uma vírgula.
6. Escolha Salvar.

Para configurar uma política de permissões para replicação (AWS CLI)

1. Crie um arquivo denominado `registry_policy.json` e preencha-o com uma política de registro.

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. Crie a política de registro usando o arquivo de política.

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

3. Recupere a política para seu registro para confirmar.

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

Conceder permissões de registro para cache de pull-through no Amazon ECR

As permissões de registro privado do Amazon ECR podem ser usadas para dimensionar o escopo das permissões de entidades individuais do IAM para usar o cache de pull-through. Se uma entidade

do IAM tiver mais permissões concedidas por uma política do IAM do que a política de permissões do registro está concedendo, a política do IAM terá precedência.

Para criar uma política de permissões de um registro privado (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região na qual deseja configurar a sua declaração de permissões da política do registro.
3. No painel de navegação, escolha Registro privado, escolha Recursos e configurações e, em seguida, escolha Permissões.
4. Na página Registry permissions (Permissões do registro), escolha Generate statement (Gerar declaração).
5. Para cada declaração de política de permissões de cache de pull-through que você criar, faça o seguinte.

    a. Em Policy type (Tipo de política), escolha Pull through cache policy (Política de cache de pull-through).
    b. Em Statement id (ID da declaração), forneça um nome para a política de declaração do cache de pull-through.
    c. Em IAM entities (Entidades do IAM), especifique os usuários, grupos ou funções a serem incluídos na política.
    d. Em Namespace do cache selecione a regra de cache de pull-through à qual a política será associada.
    e. Em Repository names (Nomes de repositórios), especifique o nome da base do repositório ao qual a regra será aplicada. Por exemplo, se você quisesse especificar o repositório do Amazon Linux no Amazon ECR Public, o nome do repositório seria `amazonlinux`.

Repositórios privados do Amazon ECR

Um repositório privado do Amazon ECR contém as imagens do Docker, as imagens do Open Container Initiative (OCI) e os artefatos do OCI compatíveis. Você pode criar, monitorar e excluir repositórios de imagens e definir permissões que controlam quem pode acessá-los usando operações de API do Amazon ECR ou a seção Repositórios do console do Amazon ECR. O Amazon ECR também se integra à CLI do Docker para que você envie por push e extraia imagens dos ambientes de desenvolvimento para os repositórios.

Tópicos

Conceitos de repositório privado

- Por padrão, sua conta tem acesso de leitura e gravação aos repositórios no seu registro padrão (*aws_account_id*.dkr.ecr.*region*.amazonaws.com). No entanto, os usuários precisam de permissões para fazer chamadas para o Amazon ECR APIs e enviar ou extrair imagens de e para seus repositórios. O Amazon ECR fornece várias políticas gerenciadas da para controlar o acesso do usuário em diversos níveis. Para obter mais informações, consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry.
- Os repositórios podem ser controlados com políticas de acesso de usuários do e políticas de repositório individuais. Para obter mais informações, consulte Políticas de repositório privado no Amazon ECR.
- Os nomes de repositório podem oferecer suporte a namespaces, que você pode usar para agrupar repositórios semelhantes. Por exemplo, se houver diversas equipes usando o mesmo registro, a Equipe A poderia usar o namespace `team-a`, e a Equipe B poderia usar o namespace `team-b`. Ao fazer isso, cada equipe tem sua própria imagem chamada `web-app` com cada imagem prefaciada com o namespace da equipe. Essa configuração permite que essas imagens em cada

equipe sejam usadas simultaneamente sem interferência. A imagem da Equipe A é `team-a/web-app`, e a imagem da Equipe B é `team-b/web-app`.

- Suas imagens podem ser replicadas para outros repositórios nas regiões em seu próprio registro e em todas as contas. Você pode fazer isso especificando uma configuração de replicação nas configurações do Registro. Para obter mais informações, consulte Configurações de registro privado no Amazon ECR.
- Quando a montagem de blobs está habilitada no nível do registro, os repositórios podem compartilhar camadas de imagem comuns.

Criar um repositório privado do Amazon ECR para armazenar imagens

> **Important**
>
> A criptografia de camada dupla do lado do servidor com AWS KMS (DSSE-KMS) só está disponível nas regiões. AWS GovCloud (US)

Crie um repositório privado do Amazon ECR e, em seguida, use o repositório para armazenar as imagens de contêineres. Siga estas etapas para criar um repositório privado usando o Console de gerenciamento da AWS.

Como criar um repositório (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região na qual criará o seu repositório.
3. Selecione Repositórios privados e depois Criar repositório.
4. Em Repository name (Nome do repositório), insira um nome exclusivo para o repositório. O nome do repositório pode ser especificado isoladamente (por exemplo, `nginx-web-app`). Como alternativa, pode ter como prefixo um namespace para agrupar o repositório em uma categoria (por exemplo `project-a/nginx-web-app`).

> **Note**
>
> O nome do repositório pode conter até 256 caracteres. O nome de repositório deve começar com uma letra e só pode conter letras minúsculas, números, hifens, sublinhados, pontos e barras. O uso de uma barra dupla não é suportado.

5. Para Mutabilidade de tag de imagem, escolha uma das configurações de mutabilidade de tag a seguir para o repositório.

   - Mutável – Escolha essa opção se quiser que as tags de imagem sejam sobrescritas. É recomendada para repositórios que usam ações de cache de pull-through para garantir que o Amazon ECR possa atualizar imagens armazenadas em cache. Além disso, para desabilitar as atualizações de tag para algumas tags mutáveis, insira os nomes das tags ou use curingas (*) para combinar várias tags semelhantes na caixa de texto Exclusão de tag mutável.
   - Imutável – Selecione esta opção se quiser impedir que as tags de imagem sejam sobrescritas. Isso se aplica a todas as tags e exclusões no repositório ao enviar uma imagem com uma tag existente. O Amazon ECR retorna uma `ImageTagAlreadyExistsException` se você tentar enviar uma imagem com uma tag existente. Além disso, para habilitar as atualizações de tag para algumas tags imutáveis, insira os nomes das tags ou use curingas (*) para combinar várias tags semelhantes na caixa de texto Exclusão de tag imutável.

   > **Note**
   >
   > As configurações de mutabilidade de tags individuais não são suportadas.

6. Para configuração de criptografia, escolha entre AES-256 ou AWS KMS Para obter mais informações, consulte Criptografia em repouso.

   a. Se AWS KMS for escolhido, escolha entre criptografia de camada única e criptografia de camada dupla. Há cobranças adicionais pelo uso da AWS KMS criptografia de camada dupla. Para obter mais informações, consulte Preços do serviço do Amazon ECR.

   b. Por padrão, a chave AWS gerenciada com o alias `aws/ecr` é escolhida. Essa chave é criada na sua conta na primeira vez que você cria um repositório com a AWS KMS criptografia ativada. Selecione Chave gerenciada pelo cliente (avançado) para escolher sua própria chave do AWS KMS . A AWS KMS chave deve estar na mesma região do cluster.

Selecione Criar uma AWS KMS chave para navegar até o AWS KMS console e criar sua própria chave.

7. Em Configurações de verificação de imagens, embora você possa especificar as configurações de verificação no repositório para uma verificação básica, é prática recomendada especificar a configuração de verificação no registro privado. Definir as configurações de verificação no nível do registro privado permite escolher entre verificação aprimorada ou verificação básica, e também permite definir filtros para especificar quais repositórios devem ser verificados.
8. Escolha Criar.

Como criar um repositório (AWS CLI)

1. Você pode criar um repositório usando o aws ecr create-repository comando AWS CLI with the.

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

2. Se você tiver um modelo de criação de repositório definido, poderá criar um repositório enviando sua imagem usando comandos push familiares do Amazon ECR com o nome do repositório desejado. O Amazon ECR criará automaticamente o repositório para você usando as configurações predefinidas do seu modelo de criação de repositório. Se você ainda não tiver um modelo de criação de repositório definido, sua solicitação para o repositório de imagens inexistente falhará.

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

Próximas etapas

Para ver as etapas de como enviar por push uma imagem para o repositório, selecione o repositório e escolha Exibir comandos de push. Para obter mais informações sobre como enviar uma imagem para seu repositório, consulte Enviar por push uma imagem para um repositório privado do Amazon ECR.

Visualizar o conteúdo e os detalhes de um repositório privado no Amazon ECR

Depois de criar um repositório privado, você pode ver os detalhes sobre ele no Console de gerenciamento da AWS:

- Quais imagens são armazenadas em um repositório
- Detalhes sobre cada imagem armazenada no repositório, incluindo o tamanho e o resumo SHA para cada imagem
- A frequência de verificação especificada para o conteúdo do repositório
- Se o repositório tem uma regra de cache pull-through ativa associada a ele
- A configuração de criptografia para o repositório

> **Note**
>
> A partir do Docker versão 1.9, o cliente do Docker compacta camadas das imagens antes de enviá-las a um registro do Docker V2. A saída do comando docker images mostra o tamanho da imagem descompactada. Portanto, lembre-se de que o Docker pode retornar uma imagem maior do que a imagem mostrada no Console de gerenciamento da AWS.

Para visualizar as informações do repositório (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região que contém o repositório a ser visualizado.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositories (Repositórios), escolha a guia Private (Privado) e depois o repositório a ser visualizado.
5. Na página de detalhes do repositório, o console usa como padrão a visualização Images (imagens). Use o menu de navegação para visualizar outras informações sobre o repositório.
   - Selecione Summary (Resumo) para visualizar detalhes do repositório e os dados de contagem de extrações do repositório.
   - Escolha Images (Imagens) para visualizar informações sobre etiquetas de imagens no repositório. Para visualizar mais informações sobre a imagem, selecione a etiqueta da

imagem. Para obter mais informações, consulte Visualizar os detalhes da imagem no Amazon ECR.

Se houver imagens não marcadas que você deseja excluir, você pode selecionar a caixa à esquerda dos repositórios a serem excluídos e escolha Delete (Excluir). Para obter mais informações, consulte Excluir uma imagem no Amazon ECR.

- Escolha Permissões para visualizar as políticas de repositório aplicadas ao repositório. Para obter mais informações, consulte Políticas de repositório privado no Amazon ECR.
- Escolha Política de ciclo de vida para visualizar as regras de política de ciclo de vida que são aplicadas ao repositório. O histórico de eventos de ciclo de vida também são exibidos aqui. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.
- Selecione Tags para visualizar as tags de metadados que são aplicadas ao repositório.

Excluir um repositório privado do Amazon ECR

Se você já terminou de usar um repositório, pode excluí-lo. Quando você exclui um repositório no Console de gerenciamento da AWS, todas as imagens contidas no repositório também são excluídas; isso não pode ser desfeito.

> **Important**
>
> As imagens nos repositórios excluídos também são excluídas. Você não pode desfazer esta operação.

Para excluir um repositório (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região que contém o repositório a ser excluído.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositories (Repositórios), escolha a guia Private (Privado), depois selecione o repositório a excluir e escolha Delete (Excluir).
5. Na *repository_name* janela Excluir, verifique se os repositórios selecionados devem ser excluídos e escolha Excluir.

Políticas de repositório privado no Amazon ECR

O Amazon ECR usa permissões baseadas em recursos para controlar o acesso a repositórios. As permissões baseadas em recursos permitem especificar quais perfis ou usuários têm acesso a um repositório e quais ações eles podem realizar nele. Por padrão, somente a AWS conta que criou o repositório tem acesso ao repositório. Você pode aplicar uma política de repositório que permite acesso adicional ao repositório.

Tópicos

Políticas de repositório versus políticas do IAM

As políticas de repositório do Amazon ECR são um subconjunto de políticas do IAM que têm como escopo e são usadas especificamente para controlar o acesso a repositórios individuais do Amazon ECR. As políticas do IAM geralmente são usadas para aplicar permissões a todo o serviço Amazon ECR, mas também podem ser usadas para controlar o acesso a recursos específicos.

Tanto as políticas de repositório do Amazon ECR quanto as políticas do IAM são usadas ao determinar quais ações uma função ou um usuário específico pode executar em um repositório. Se uma função ou um usuário tiver permissão para executar uma ação por meio de uma política de repositório, mas tiver a permissão negada por uma política do IAM (ou vice-versa), a ação será negada. Uma função ou um usuário somente precisa ter permissão para uma ação por meio de uma política de repositório ou uma política do IAM, mas não ambas para que a ação seja permitida.

> **Important**
>
> O Amazon ECR exige que os usuários tenham permissão para fazer chamadas para a API `ecr:GetAuthorizationToken` por meio de uma política do IAM antes que possam fazer a autenticação para um registro e enviar e extrair qualquer imagem de um repositório do Amazon ECR. O Amazon ECR fornece várias políticas gerenciadas do IAM para controlar o acesso do usuário em diversos níveis. Para obter mais informações, consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry.

Você pode usar qualquer um desses tipos de política para controlar o acesso aos seus repositórios, conforme mostrado nos exemplos a seguir.

Este exemplo mostra uma política de repositório do Amazon ECR que permite que um usuário específico descreva o repositório e as imagens contidas nele.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ECRRepositoryPolicy",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::111122223333:user/username"},
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": "*"
        }
    ]
}
```

Este exemplo mostra uma política do IAM que atinge o mesmo objetivo que o acima definindo o escopo da política como um repositório (especificado pelo ARN completo do repositório) usando o parâmetro de recurso. Para obter mais informações sobre o formato do nome de recurso da Amazon (ARN), consulte Recursos.

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"]
        }
    ]
}
```

Exemplos de políticas de repositório privado no Amazon ECR

> **Important**
>
> Os exemplos de políticas de repositório nesta página destinam-se a ser aplicados a repositórios privados do Amazon ECR. Eles não funcionarão corretamente se forem usados diretamente com um entidade principal IAM, a menos que sejam modificados para especificar o repositório Amazon ECR como o recurso. Para obter mais informações sobre a definição de políticas de repositório, consulte Configurar uma declaração de política de repositório privado no Amazon ECR.

As políticas de repositório do Amazon ECR são um subconjunto de políticas do IAM que têm como escopo e são usadas especificamente para controlar o acesso a repositórios individuais do Amazon ECR. As políticas do IAM geralmente são usadas para aplicar permissões a todo o serviço Amazon ECR, mas também podem ser usadas para controlar o acesso a recursos específicos. Para obter mais informações, consulte Políticas de repositório versus políticas do IAM.

Os exemplos a seguir de políticas de repositório mostram declarações de permissão que você poderia usar para controlar o acesso aos seus repositórios privados do Amazon ECR.

> **Important**
>
> O Amazon ECR exige que os usuários tenham permissão para fazer chamadas para a API `ecr:GetAuthorizationToken` por meio de uma política do IAM antes que possam fazer a autenticação para um registro e enviar e extrair qualquer imagem de um repositório do Amazon ECR. O Amazon ECR fornece várias políticas gerenciadas do IAM para controlar o acesso do usuário em diversos níveis. Para obter mais informações, consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry.

Exemplo: permitir um ou mais usuários do

A política de repositório a seguir permite que um ou mais usuários do enviem e extraiam imagens de e para um repositório.

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": "*"
        }
    ]
}
```

Exemplo: permitir outra conta

A política de repositório a seguir permite que uma conta específica insira imagens.

> **Important**
>
> A conta para a qual você está concedendo permissões deve ter a região na qual você está criando a política de repositório ativada, caso contrário, ocorrerá um erro.

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": "*"
        }
    ]
}
```

A política de repositório a seguir permite que alguns usuários extraiam imagens (*pull-user-1*e*pull-user-2*) enquanto fornecem acesso total a outra (*admin-user*).

> **Note**
>
> Para políticas de repositório mais complicadas que atualmente não são suportadas no Console de gerenciamento da AWS, você pode aplicar a política com o set-repository-policy AWS CLI comando.

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": "*"
        }
    ]
}
```

Exemplo: negar tudo

A política de repositório a seguir nega a todos os usuários a capacidade de extrair imagens.

JSON

```
{
```

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

Exemplo: restringir o acesso a endereços IP específicos

O exemplo a seguir nega permissões a qualquer usuário para executar qualquer operação do Amazon ECR quando aplicada a um repositório de uma faixa específica de endereços.

A condição nesta declaração identifica o `54.240.143.*` intervalo de endereços IP permitidos do Protocolo de Internet versão 4 (IPv4).

O `Condition` bloco usa as `NotIpAddress` condições e a chave de `aws:SourceIp` condição, que é uma chave AWS de condição ampla. Para obter mais informações sobre chaves de condição, consulte Chaves de contexto de condição globais da AWS. Os `aws:sourceIp` IPv4 valores usam a notação CIDR padrão. Para obter mais informações, consulte Operadores de condição de endereço IP no Guia do usuário do 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"
                }
            }
        }
    ]
}
```

Exemplo: Permitir um AWS serviço

A política de repositório a seguir permite o AWS CodeBuild acesso às ações de API do Amazon ECR necessárias para a integração com esse serviço. Ao usar o exemplo a seguir, você deve usar as chaves de condição `aws:SourceArn` e `aws:SourceAccount` para definir o escopo de quais recursos que podem assumir essas permissões. Para obter mais informações, consulte a amostra do Amazon ECR CodeBuild no Guia do AWS CodeBuild usuário.

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"
```

```
            }
        }
    }
  ]
}
```

Configurar uma declaração de política de repositório privado no Amazon ECR

Você pode adicionar uma declaração de política de acesso a um repositório no Console de gerenciamento da AWS seguindo as etapas abaixo. Você pode adicionar várias instruções de política por repositório. Para obter exemplos de políticas, consulte Exemplos de políticas de repositório privado no Amazon ECR.

> **Important**
>
> O Amazon ECR exige que os usuários tenham permissão para fazer chamadas para a API `ecr:GetAuthorizationToken` por meio de uma política do IAM antes que possam fazer a autenticação para um registro e enviar e extrair qualquer imagem de um repositório do Amazon ECR. O Amazon ECR fornece várias políticas gerenciadas do IAM para controlar o acesso do usuário em diversos níveis. Para obter mais informações, consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry.

Para configurar uma instrução de política de repositório

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região que contém o repositório no qual será configurada uma instrução de política.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositories (Repositórios), escolha o repositório no qual definir uma instrução de política para visualizar o conteúdo do repositório.
5. Na exibição de lista de imagens do repositório, no painel de navegação, selecione Permissions (Permissões), Edit (Editar).

> **Note**
>
> Se você não vir a opção Permissions (Permissões) no painel de navegação, verifique se você está na exibição de lista de imagens do repositório.

6. Na página Editar permissões, selecione Adicionar declaração.
7. Em Statement name (Nome da instrução), insira um nome para a instrução.
8. Em Effect (Efeito), escolha se a instrução da política resultará em uma permissão ou negação explícita.
9. Em Principal, escolha o escopo ao qual aplicar a instrução da política. Para obter mais informações, consulte Elementos de política JSON da AWS : entidade principal no Manual do usuário do IAM.

   - Você pode aplicar a declaração a todos os AWS usuários autenticados marcando a caixa de seleção Todos (*).
   - Em Service principal (Principal do serviço), especifique o nome do principal do serviço (por exemplo, `ecs.amazonaws.com`) para aplicar a instrução a um serviço específico.
   - IDsEm AWS Conta, especifique um número de AWS conta (por exemplo,`111122223333`) para aplicar a declaração a todos os usuários em uma AWS conta específica. Várias contas podem ser especificadas usando uma lista delimitada por vírgulas.

     > **Important**
     >
     > A conta para a qual você está concedendo permissões deve ter a região na qual você está criando a política de repositório ativada, caso contrário, ocorrerá um erro.

   - Para entidades do IAM, selecione as funções ou os usuários em sua AWS conta aos quais aplicar a declaração.

     > **Note**
     >
     > Para políticas de repositório mais complicadas que atualmente não são suportadas no Console de gerenciamento da AWS, você pode aplicar a política com o set-repository-policy AWS CLI comando.

10. Em Actions (Ações), escolha o escopo das operações da API do Amazon ECR ao qual a declaração de política deve ser aplicada na lista de operações de API individuais.
11. Quando terminar, escolha Save (Salvar) para definir a política.
12. Repita a etapa anterior para cada política de repositório a ser adicionada.

Marcar um repositório privado no Amazon ECR

Para ajudá-lo a gerenciar seus repositórios Amazon ECR, você pode atribuir seus próprios metadados a repositórios Amazon ECR novos ou existentes usando tags de recursos. AWS Por exemplo, você pode definir um conjunto de tags para os repositórios do Amazon ECR da sua conta para ajudar a rastrear o proprietário de cada repositório.

Conceitos Básicos de Tags

As tags não têm significado semântico no Amazon ECR e são interpretadas estritamente como uma sequência dos caracteres. Tags não são automaticamente atribuídas aos recursos. É possível editar chaves de tags e valores, e é possível remover as tags de um recurso a qualquer momento. É possível definir o valor de uma tag a uma string vazia, mas não pode configurar o valor de um tag como nula. Se você adicionar uma tag que tenha a mesma chave de uma tag existente nesse recurso, o novo valor substituirá o antigo. Caso exclua um recurso, todas as respectivas tags também serão excluídas.

Você pode trabalhar com tags usando o console do Amazon ECR AWS CLI, o e a API do Amazon ECR.

Usando AWS Identity and Access Management (IAM), você pode controlar quais usuários em sua AWS conta têm permissão para criar, editar ou excluir tags. Para obter informações sobre tags nas políticas do IAM, consulte the section called “Usando o controle de Tag-Based acesso”.

Marcar recursos para faturamento

As tags que você adiciona aos repositórios do Amazon ECR são úteis para analisar a alocação de custos depois de habilitá-las em seu Relatório de custo e uso. Para obter mais informações, consulte Relatórios de uso do Amazon ECR.

Para ver o custo dos recursos combinados, é possível organizar as informações de faturamento com base nos recursos com os mesmos valores da chave da tag. Por exemplo, é possível etiquetar vários recursos com um nome de aplicação específico, e depois organizar suas informações de faturamento

para ver o custo total daquela aplicação em vários serviços. Para obter mais informações sobre como configurar um relatório de alocação de custos com tags, consulte Relatório mensal de alocação de custos no Manual do usuário do AWS Billing .

> **Note**
>
> Se você tiver acabado de habilitar a criação de relatórios, os dados do mês atual estarão disponíveis para visualização após 24 horas.

Adicionar tags a um repositório privado do Amazon ECR

Você pode adicionar tags a um repositório privado.

Para obter informações sobre nomes e práticas recomendadas para tags, consulte Limites e requisitos de nomenclatura de tags e Práticas recomendadas no Guia do usuário de AWS recursos de marcação.

Adicionar tags a um repositório (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região a ser usada.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositórios, marque a caixa de seleção ao lado do repositório que você deseja marcar.
5. No menu Ação, selecione Tags do repositório.
6. Na página Tags do repositório, selecione Adicionar tags, Adicionar tag.
7. Na página Editar tags do repositório, especifique a chave e o valor de cada tag e escolha Salvar.

Adicionar tags a um repositório (AWS CLI ou API)

Você pode adicionar ou substituir uma ou mais tags usando a AWS CLI ou uma API.

- AWS CLI - recurso de tag
- Ação da API - TagResource

Os exemplos a seguir mostram como adicionar tags usando a AWS CLI.

Exemplo 1: marcar um repositório

O comando a seguir marca um repositório.

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

Exemplo 2: marcar um repositório com várias tags

O comando a seguir adiciona três tags a um repositório.

```
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
```

Exemplo 3: listar tags de um repositório

O comando a seguir lista as tags associadas a um repositório.

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

Exemplo 4: criar um repositório e adicionar uma tag

O comando a seguir cria um repositório chamado `test-repo` e adiciona uma tag com a chave `team` e o valor `devs`.

```
aws ecr create-repository \
     --repository-name test-repo \
     --tags Key=team,Value=devs
```

Excluir tags de um repositório privado no Amazon ECR

Você pode excluir tags de um repositório privado.

Para excluir uma tag de um repositório privado (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região a ser usada.

3. Na página Repositórios, marque a caixa de seleção ao lado do repositório do qual você deseja remover uma tag.
4. No menu Ação, selecione Tags do repositório.
5. Na página Tags do repositório, selecione Editar.
6. Na página Editar tags do repositório, selecione Remover para cada tag que você desejar excluir e escolha Salvar.

Para excluir uma tag de um repositório privado (AWS CLI)

Você pode excluir uma ou mais tags usando a AWS CLI ou uma API.

- AWS CLI - recurso de desmarcação
- Ação da API - UntagResource

Os exemplos a seguir mostram como excluir uma tag de um repositório usando a AWS CLI.

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

Imagens privadas no Amazon ECR

O Amazon ECR armazena imagens do Docker, imagens do Open Container Initiative (OCI) e artefatos compatíveis com o OCI em repositórios privados. É possível usar a CLI do Docker, ou seu cliente preferido, para enviar e extrair imagens dos seus repositórios.

Com o Amazon ECR compatível com o OCI v1.1, você pode armazenar e gerenciar artefatos de referência que são definidos pela API Referrers do OCI. Os artefatos incluem assinaturas, lista de materiais de software (SBoMs), gráficos do Helm, resultados de escaneamento e atestados. Um conjunto de artefatos de uma imagem de contêiner é transferido com esse contêiner e armazenado como uma imagem separada que conta como uma imagem consumida pelo repositório.

As páginas Assine imagens no Amazon ECR e Exclusão de assinaturas e de outros artefatos de um repositório privado do Amazon ECR fornecem exemplos de como usar artefatos relacionados à assinatura. Para obter mais informações sobre como assinar imagens de contêineres, consulte Signing container images no Guia do desenvolvedor do AWS Signer .

Tópicos

Enviar por push uma imagem para um repositório privado do Amazon ECR

É possível enviar imagens do Docker, listas de manifesto e imagens da Open Container Initiative (OCI) e artefatos compatíveis para seus repositórios privados.

O Amazon ECR fornece uma maneira de replicar suas imagens em outros repositórios. Ao especificar uma configuração de replicação nas configurações do registro privado, você pode replicar entre regiões no seu próprio registro e em outras contas. Para obter mais informações, consulte Configurações de registro privado no Amazon ECR.

> **Note**
>
> Se você enviar uma imagem atualmente arquivada, essa imagem será automaticamente restaurada e removida do arquivamento. Para obter mais informações sobre arquivamento e restauração de imagens, consulte. Arquivamento de uma imagem no Amazon ECR

Quando a montagem de blobs de registro está ativada e os parâmetros de montagem são incluídos, o Amazon ECR verifica automaticamente as camadas existentes em seu registro durante as operações push. Se uma camada já existir em outro repositório dentro do mesmo registro, o Amazon ECR montará a camada existente em vez de fazer o upload de uma duplicata. Para obter mais informações, consulte Montagem de blobs no Amazon ECR.

Tópicos

- Permissões do IAM para enviar por push uma imagem para um repositório privado do Amazon ECR
- Envio por push de uma imagem do Docker para um repositório privado do Amazon ECR
- Como enviar uma imagem de multiarquitetura por push para um repositório do privado do Amazon ECR
- Para enviar um chart do Helm por push para um repositório privado do Amazon ECR

Permissões do IAM para enviar por push uma imagem para um repositório privado do Amazon ECR

Os usuários precisam de permissões do IAM para enviar imagens por push para os repositórios privados do Amazon ECR. Seguindo a prática recomendada de conceder privilégio mínimo, você pode conceder acesso a um repositório específico. Você também pode conceder acesso a todos os repositórios.

Um usuário deve se autenticar em cada registro do Amazon ECR para o qual deseja enviar imagens solicitando um token de autorização. O Amazon ECR fornece várias políticas AWS gerenciadas para controlar o acesso do usuário em vários níveis. Para obter mais informações, consulte AWS políticas gerenciadas para o Amazon Elastic Container Registry.

Você também pode criar suas próprias políticas do IAM. A política do IAM a seguir concede as permissões necessárias para enviar uma imagem por push a um repositório específico. Para limitar as permissões para um repositório específico, use o nome do recurso da Amazon (ARN) completo do repositório.

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": "*"
```

```
        }
    ]
}
```

A política do IAM a seguir concede as permissões necessárias para enviar uma imagem por push a todos os repositórios.

```
{
    "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/*"
        }
    ]
}
```

Envio por push de uma imagem do Docker para um repositório privado do Amazon ECR

Você pode enviar suas imagens de contêiner para um repositório do Amazon ECR com o comando docker push.

O Amazon ECR também é compatível com a criação e o envio por push de listas de manifestos do Docker que são usadas para imagens de multiarquitetura. Para mais informações, consulte Como enviar uma imagem de multiarquitetura por push para um repositório do privado do Amazon ECR.

Para enviar uma imagem do Docker a um repositório do Amazon ECR

O repositório Amazon ECR deve existir antes de você enviar a imagem, ou você deve ter um modelo de criação de repositório definido. Para obter mais informações, consulte Criar um repositório privado

do Amazon ECR para armazenar imagens e Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.

1. Autentique o cliente do Docker para o registro do Amazon ECR para o qual você pretende enviar a imagem. Os tokens de autenticação devem ser obtidos para cada registro usado e são válidos por 12 horas. Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.

   Para autenticar o Docker para um registro do Amazon ECR, execute o comando aws ecr get-login-password. Ao transmitir o token de autenticação para o comando docker login, use o valor `AWS` para o nome de usuário, e especifique o URI de registro do Amazon ECR para o qual deseja fazer a autenticação. Se autenticar em vários registros, você deverá repetir o comando para cada registro.

   > **Important**
   >
   > Se você receber um erro, instale ou atualize para a versão mais recente da AWS CLI. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da 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. Se o repositório de imagens ainda não existir no registro para o qual você pretende enviar e você tiver um modelo de criação de repositório definido, você poderá enviar sua imagem usando o prefixo do modelo de criação do repositório e o nome do repositório desejado. O ECR criará automaticamente o repositório para você usando as configurações predefinidas do seu modelo de criação de repositório.

   Se você não tiver um modelo de criação de repositório correspondente definido, precisará criar um repositório. Para acessar mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação ou Criar um repositório privado do Amazon ECR para armazenar imagens.

3. Identifique a imagem a ser enviada. Execute o comando docker images para listar as imagens do contêiner em seu sistema.

   ```
   docker images
   ```

Você pode identificar uma imagem com o *repository:tag* valor ou o ID da imagem na saída do comando resultante.

4. Marque a sua imagem com o registro do Amazon ECR, o repositório e a combinação opcional de nomes de tag de imagem a ser usada. O formato do registro é *aws_account_id*.dkr.ecr.*region*.amazonaws.com. O nome do repositório deve corresponder ao repositório que você criou para sua imagem. Se você omitir a tag de imagem, suporemos que a tag é `latest`.

   O exemplo a seguir marca uma imagem local com o ID *e9ae3c220b23* como*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. Envie a imagem usando o comando docker push:

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

6. (Opcional) Aplique quaisquer tags adicionais à sua imagem e envie essas tags ao Amazon ECR repetindo Step 4 e Step 5.

Como enviar uma imagem de multiarquitetura por push para um repositório do privado do Amazon ECR

Você pode enviar imagens de multiarquitetura por push para um repositório do Amazon ECR criando e enviando por push listas de manifestos do Docker. Uma lista de manifestos é uma lista de imagens criada com a especificação de um ou mais nomes de imagem. Na maioria dos casos, a lista de manifestos é criada de imagens que exercem a mesma função, mas que são para outros sistemas operacionais ou arquiteturas. A lista de manifestos não é obrigatória. Para obter mais informações, consulte manifesto do docker.

Uma lista de manifestos pode ser extraída ou referenciada em uma definição de tarefa do Amazon ECS ou especificação de pod do Amazon EKS como outras imagens do Amazon ECR.

Pré-requisitos

- Na CLI do Docker, ative os recursos experimentais. Para obter informações sobre recursos experimentais, consulte Experimental features na documentação do Docker.

- O repositório do Amazon ECR deve existir antes de enviar a imagem. Para obter mais informações, consulte the section called “Criar um repositório para armazenar imagens”.
- As imagens devem ser enviadas por push ao seu repositório antes de você criar o manifesto do Docker. Para obter informações sobre como enviar uma imagem, consulte Envio por push de uma imagem do Docker para um repositório privado do Amazon ECR.

Como enviar uma imagem de multiarquitetura do Docker para um repositório do Amazon ECR

1. Autentique o cliente do Docker para o registro do Amazon ECR para o qual você pretende enviar a imagem. Os tokens de autenticação devem ser obtidos para cada registro usado e são válidos por 12 horas. Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.

   Para autenticar o Docker para um registro do Amazon ECR, execute o comando aws ecr get-login-password. Ao transmitir o token de autenticação para o comando docker login, use o valor `AWS` para o nome de usuário, e especifique o URI de registro do Amazon ECR para o qual deseja fazer a autenticação. Se autenticar em vários registros, você deverá repetir o comando para cada registro.

   > **Important**
   >
   > Se você receber um erro, instale ou atualize para a versão mais recente da AWS CLI. Para obter mais informações, consulte Installing the AWS Command Line Interface (Instalar a AWS Command Line Interface) no User Guide (Guia do usuário da).

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

2. Listar as imagens no repositório, confirmando as tags de imagem.

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

3. Criar a lista de manifestos do Docker. O comando `manifest create` verifica se as imagens referenciadas já estão no repositório e cria o manifesto localmente.

   ```
   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. (Opcional) Inspecionar a lista de manifestos do Docker. Isso permite que você confirme o tamanho e o resumo de cada manifesto de imagem referenciado na lista de manifestos.

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

5. Enviar a lista de manifestos do Docker para seu repositório do Amazon ECR.

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

Para enviar um chart do Helm por push para um repositório privado do Amazon ECR

Você pode enviar por push artefatos da Open Container Initiative (OCI) para um repositório do Amazon ECR. Para ver um exemplo dessa funcionalidade, use as etapas a seguir para enviar por push um chart do Helm para o Amazon ECR.

Para obter mais informações sobre como usar os charts do Helm hospedados no Amazon ECR com o Amazon EKS, consulte Instalar um chart do Helm em um cluster do Amazon EKS.

Para enviar um chart do Helm para um repositório do Amazon ECR

1. Use a versão mais recente do cliente do Helm. Estas etapas foram escritas usando a versão `3.18.6` do Helm. Para compatibilidade com as versões do Kubernetes compatíveis com o Amazon EKS, use a versão 3.9 ou posterior do Helm. Para obter mais informações, consulte Instalação do Helm.
2. Use as etapas a seguir para criar um chart do Helm. Para obter mais informações, consulte o Documentos do Helm - Introdução.

   a. Crie um chart do Helm denominado `helm-test-chart` e limpe o conteúdo da caixa do diretório `templates`.

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

   b. Criae um `ConfigMap` na pasta `templates`.

```
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. Embalar o gráfico. A saída conterá o nome do arquivo do chart empacotado que você usa ao enviar o chart do 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. Crie um repositório para armazenar o chart do Helm. O nome do repositório deve corresponder ao nome utilizado ao criar o chart do Helm na etapa 2. Para obter mais informações, consulte Criar um repositório privado do Amazon ECR para armazenar imagens.

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

5. Autentique o cliente do Helm para o registro do Amazon ECR para o qual você pretende enviar o chart do Helm. Os tokens de autenticação devem ser obtidos para cada registro usado e são válidos por 12 horas. Para obter mais informações, consulte Autenticação de registro privado no 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. Envie o chart do Helm usando o comando helm push. A saída deve incluir o URI do repositório do Amazon ECR e o resumo do SHA.

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

7. Descreva seu chart do Helm.

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

Na saída, verifique se o parâmetro `artifactMediaType` indica o tipo de artefato apropriado.

```
{
    "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. (Opcional) Para etapas adicionais, instale o `ConfigMap` do Helm e comece a usar o Amazon EKS. Para obter mais informações, consulte Instalar um chart do Helm em um cluster do Amazon EKS.

Exclusão de assinaturas e de outros artefatos de um repositório privado do Amazon ECR

Você pode usar o cliente do ORAS para listar e excluir assinaturas e outros artefatos do tipo de referência de um repositório privado do Amazon ECR. A exclusão de assinaturas e outros artefatos de referência é semelhante à exclusão de uma imagem (consulte Excluir uma imagem no Amazon ECR). Veja como listar artefatos e excluir assinaturas:

Para gerenciar artefatos de imagem usando a CLI do ORAS

1. Instale e configure o cliente do ORAS.

   Para obter informações sobre como instalar e configurar o cliente do ORAS, consulte Installation na documentação do ORAS.

2. Para listar artefatos disponíveis para uma imagem do Amazon ECR, use `oras discover`, seguido por um nome de imagem:

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

   A saída deve ser semelhante a esta:

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

3. Para excluir uma assinatura usando a CLI do ORAS, no exemplo anterior, execute o seguinte comando:

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

   A saída deve ser semelhante a esta:

```
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. Pressione y. O artefato deve ser excluído.

Para solucionar problemas de exclusão de artefatos

Se uma exclusão de assinatura, como a que acabou de ser mostrada, falhar, será exibida uma saída semelhante à seguinte:

```
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]
```

Essa falha pode ocorrer ao excluir uma imagem enviada por push antes do lançamento do OCI 1.1. Conforme observado no erro, você deve excluir o manifesto que faz referência à imagem antes de excluir a imagem, da seguinte forma:

1. Para excluir o manifesto associado à assinatura que você deseja excluir, digite:

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

   A saída deve ser semelhante a esta:

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

2. Pressione y. O manifesto deve ser excluído.
3. Sem o manifesto, você pode excluir a assinatura:

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

A saída deve ser semelhante a esta. Pressione 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. Para ver se a assinatura foi excluída, digite:

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

A saída deve ser semelhante a esta:

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

Visualizar os detalhes da imagem no Amazon ECR

Depois que enviar uma imagem por push ao seu repositório, você poderá ver as informações sobre ela. Os detalhes incluídos são os seguintes:

- URI da imagem
- Tags da imagem
- Tipo de mídia do Artifact
- Tipo de manifesto da imagem
- Status da verificação
- O tamanho da imagem em MB

- Quando a imagem foi extraída para o repositório
- O status da replicação

Para ver os detalhes da imagem (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/repositories.
2. Na barra de navegação, selecione a região que contém o repositório de sua imagem.
3. No painel de navegação, em Registro privado, escolha Repositórios.
4. Na página Repositórios privados, escolha o repositório a ser visualizado.
5. Na *repository_name* página Repositórios:, escolha a imagem para ver os detalhes.

Extrair uma imagem de um repositório privado do Amazon ECR para o ambiente local

Se quiser executar uma imagem do Docker que está disponível no Amazon ECR, você pode extrai-la para seu ambiente local com o comando docker pull. Você pode fazer isso a partir do seu registro padrão ou de um registro associado a outra AWS conta.

Para usar uma imagem do Amazon ECR em uma definição de tarefa do Amazon ECS, consulte Uso de imagens do Amazon ECR com o Amazon ECS.

Important

Você não pode extrair uma imagem arquivada. As imagens arquivadas devem ser restauradas antes de serem extraídas. Para obter mais informações sobre arquivamento e restauração de imagens, consulte. Arquivamento de uma imagem no Amazon ECR

Important

O Amazon ECR exige que os usuários tenham permissão para fazer chamadas para a API `ecr:GetAuthorizationToken` por meio de uma política do IAM antes que possam fazer a autenticação para um registro e enviar e extrair qualquer imagem de um repositório do Amazon ECR. O Amazon ECR fornece várias políticas AWS gerenciadas para controlar o acesso do usuário em vários níveis. Para obter informações sobre as políticas AWS

gerenciadas do Amazon ECR, consulteAWS políticas gerenciadas para o Amazon Elastic Container Registry.

Para extrair uma imagem do Docker de um repositório do Amazon ECR

1. Autentique o cliente do Docker para o registro do Amazon ECR do qual você pretende extrair a imagem. Os tokens de autenticação devem ser obtidos para cada registro usado e são válidos por 12 horas. Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.
2. (Opcional) Identifique a imagem a ser extraída.
   - É possível listar os repositórios em um registro com o comando aws ecr describe-repositories:

     ```
     aws ecr describe-repositories
     ```

     O registro de exemplo acima tem um repositório chamado`amazonlinux`.
   - É possível descrever as imagens em um repositório com o comando aws ecr describe-images:

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

     O repositório de exemplo acima tem uma imagem marcada como `latest` e `2016.09`, com o resumo de imagem `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807`.
3. Extraia a imagem usando o comando docker pull. O formato de nome de imagem deve ser *`registry`*`/`*`repository`*` [:`*`tag`*`]` para extrair por tag ou *`registry`*`/`*`repository`*`[@`*`digest`*`]` para extrair por resumo.

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

   > **Important**
   >
   > Se receber um erro *`repository-url`*` not found: does not exist or no pull access`, pode ser necessário autenticar seu cliente do Docker com o Amazon ECR.

Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.

Extraia a imagem de contêiner do Amazon Linux

A imagem do contêiner do Amazon Linux é criada a partir dos mesmos componentes de software que são incluídos na AMI do Amazon Linux. A imagem de contêiner do Amazon Linux está disponível para uso em qualquer ambiente como uma imagem de base para workloads do Docker. Caso use a AMI do Amazon Linux para aplicações no Amazon EC2, você poderá colocar as aplicações em contêineres com a imagem de contêiner do Amazon Linux.

Você pode usar a imagem do contêiner Amazon Linux em seu ambiente de desenvolvimento local e, em seguida, enviar seu aplicativo para AWS usar o Amazon ECS. Para obter mais informações, consulte Uso de imagens do Amazon ECR com o Amazon ECS.

A imagem de contêiner do Amazon Linux está disponível no Amazon ECR Public no Docker Hub. Para obter suporte para a imagem de contêiner do Amazon Linux, acesse os fóruns de desenvolvedores da AWS.

Para extrair a imagem de contêiner do Amazon Linux do Amazon ECR Public

1. Autentique o cliente do Docker para seu registro do Amazon Linux. Os tokens de autenticação são válidos por 12 horas. Para obter mais informações, consulte Autenticação de registro privado no Amazon ECR.

   > **Note**
   >
   > Os comandos ecr-public estão disponíveis na AWS CLI a partir da versão `1.18.1.187`. No entanto, recomendamos usar a versão mais recente da AWS CLI. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface .

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

   A saída é a seguinte:

```
Login succeeded
```

2. Extraia a imagem do contêiner do Amazon Linux usando o comando docker pull. Para visualizar a imagem do contêiner do Amazon Linux na Galeria Pública do Amazon ECR, consulte Galeria pública do Amazon ECR - amazonlinux.

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

3. (Opcional) Execute o contêiner localmente.

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

Para extrair a imagem do contêiner do Amazon Linux a partir do Docker Hub

1. Extraia a imagem do contêiner do Amazon Linux usando o comando docker pull.

```
docker pull amazonlinux
```

2. (Opcional) Execute o contêiner localmente.

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

Excluir uma imagem no Amazon ECR

Se você já terminou de usar uma imagem, pode excluí-la do repositório. Se você já terminou de usar um repositório, pode excluir o repositório inteiro e todas as imagens contidas nele. Para obter mais informações, consulte Excluir um repositório privado do Amazon ECR.

Como alternativa a excluir as imagens manualmente, você pode criar políticas de ciclo de vida do repositório que fornecem mais controle sobre o gerenciamento do ciclo de vida das imagens em seus repositórios. As políticas de ciclo de vida automatizam esse processo para você. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.

> **Note**
>
> Se o repositório tiver uma combinação de imagens, algumas das quais tendo sido enviadas por push antes de o Amazon ECR ser compatível com o OCI v1.1, algumas assinaturas terão índices de imagens ou listas de manifestos apontando para elas. Como resultado, ao excluir uma imagem pré-OCI v1.1, talvez seja necessário excluir manualmente a lista de manifestos que faz referência à imagem para poder excluir o artefato.

Para excluir uma imagem (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/repositories.
2. Na barra de navegação, selecione a região que contém a imagem a ser excluída.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositórios, escolha o repositório que contém a imagem a ser excluída.
5. Na *repository_name* página Repositórios:, selecione a caixa à esquerda da imagem a ser excluída e escolha Excluir.
6. Na caixa de diálogo Excluir imagem(ns), verifique se as imagens selecionadas devem ser excluídas e escolha Excluir.

Para excluir uma imagem (AWS CLI)

1. Listar as imagens no seu repositório. As imagens marcadas terão um resumo de imagem, bem como uma lista de tags associadas. Imagens não marcadas só terão um resumo de imagem.

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

2. (Opcional) Exclua quaisquer tags indesejáveis da imagem especificando a tag da imagem que você deseja excluir. Quando você excluir a última tag de uma imagem, a imagem será excluída.

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

3. Exclua uma imagem marcada ou não marcada especificando o resumo da imagem. Quando você excluir uma imagem fazendo referência ao seu resumo, a imagem e todas as suas tags serão excluídas.

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

Para excluir várias imagens, você pode especificar várias tags de imagem ou resumos de imagem na solicitação.

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

Arquivamento de uma imagem no Amazon ECR

Qual é a classe de armazenamento de arquivamento ECR?

A classe de armazenamento arquivístico Amazon ECR é uma nova classe de armazenamento que fornece armazenamento de baixo custo e de longo prazo para imagens de contêineres. O Amazon ECR oferece duas classes de armazenamento:

- Classe de armazenamento padrão ECR — A classe de armazenamento padrão para imagens ativas que são acessadas regularmente.
- Classe de armazenamento de arquivamento ECR — uma classe de armazenamento de baixo custo para imagens que raramente são acessadas, mas que precisam ser mantidas para fins de conformidade ou referência de longo prazo. A classe de armazenamento arquivístico oferece economia de custos para grandes quantidades de imagens em comparação com a classe de armazenamento padrão para retenção de imagens a longo prazo. Para obter informações detalhadas sobre preços, consulte a definição de preço do Amazon ECR.

Para arquivar imagens, você tem duas opções. Primeiro, você pode configurar regras de ciclo de vida para arquivar imagens automaticamente com base em:

- Tempo desde que a imagem foi enviada

- Tempo desde a última vez que a imagem foi extraída
- Número de imagens no repositório

Você também pode definir as configurações para excluir permanentemente as imagens após elas terem sido arquivadas por um período especificado. Consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR para obter mais informações.

Você também pode arquivar imagens usando o console Amazon ECR ou AWS CLI. Consulte Arquivando uma imagem para obter mais informações.

Quando precisar usar uma imagem arquivada novamente, você pode restaurá-la de volta à classe de armazenamento ECR Standard. Você pode esperar que o ECR restaure a imagem em 20 minutos. As imagens restauradas se comportam como imagens recém-enviadas e ficam imediatamente disponíveis para uso quando a restauração for concluída. As imagens restauradas estão sujeitas às políticas de digitalização, replicação e ciclo de vida do repositório. Consulte Restaurando uma imagem para obter mais informações.

Arquivando uma imagem

Você pode arquivar imagens manualmente usando o console Amazon ECR ou AWS CLI automaticamente usando políticas de ciclo de vida. Quando uma imagem é arquivada:

- A imagem é movida para a classe de armazenamento arquivístico.
- As imagens arquivadas não podem ser extraídas. As solicitações para extrair a imagem arquivada falharão com um erro 404.
- Embora a imagem não possa ser extraída, ela ainda pode ser descrita usando o describe-images comando ou listada usando o list-images comando. O status da imagem será mostrado como`ARCHIVED`.
- As imagens arquivadas têm uma duração mínima de armazenamento de 90 dias. Você não pode configurar políticas de ciclo de vida que excluam imagens que estão arquivadas há menos de 90 dias. Se você precisar excluir imagens que foram arquivadas por menos de 90 dias, precisará usar a batch-delete-image API, mas você será cobrado pela duração mínima de armazenamento de 90 dias.
- A imagem aparece em uma guia Imagens arquivadas na visualização do repositório (essa guia aparecerá somente se pelo menos uma imagem estiver arquivada no repositório).
- A imagem pode ser restaurada como uma imagem ativa selecionando-a manualmente para ser restaurada ou empurrando-a novamente para o repositório.

- A imagem será excluída se o repositório tiver políticas de ciclo de vida que excluam a imagem com critérios como tempo de arquivamento.

Console de gerenciamento da AWS

Para arquivar uma imagem

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/repositories.
2. Na barra de navegação, escolha a região que contém o repositório com a imagem que você deseja arquivar.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositórios, escolha o repositório que contém a imagem que você deseja arquivar.
5. Selecione a imagem que você deseja arquivar. Você verá os detalhes da imagem.
6. Para arquivar a imagem, selecione o botão Arquivar e selecione Confirmar quando solicitado.
7. Se essa for a primeira imagem arquivada no repositório, uma nova guia Imagens arquivadas será exibida com a imagem recém-arquivada. Se houver outras imagens arquivadas, essa imagem será adicionada a essa guia.

AWS CLI

Para arquivar uma imagem

- Use o update-image-storage-class comando para arquivar uma imagem atualizando sua classe de armazenamento para`ARCHIVE`:

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

Para arquivar uma imagem usando políticas de ciclo de vida

- Você pode configurar regras de arquivamento para seus repositórios usando políticas de ciclo de vida para arquivar imagens automaticamente. As políticas de ciclo de vida permitem que você arquive imagens automaticamente com base em critérios como:

  - Tempo desde que a imagem foi enviada
  - Tempo desde a última vez que a imagem foi extraída
  - Número máximo de imagens a serem mantidas ativas

  Você também pode configurar políticas de ciclo de vida para excluir permanentemente imagens após elas terem sido arquivadas por um período especificado. Para obter mais informações e exemplos de políticas de ciclo de vida com ações de arquivamento, consulte. Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR

  > **Note**
  >
  > As imagens arquivadas têm uma duração mínima de armazenamento de 90 dias. Você não pode configurar políticas de ciclo de vida que excluam imagens que estão arquivadas há menos de 90 dias. Se você precisar excluir imagens que foram arquivadas por menos de 90 dias, precisará usar a batch-delete-image API, mas você será cobrado pela duração mínima de armazenamento de 90 dias.

Quando você descreve imagens usando o describe-images comando, as imagens arquivadas têm um `image-status` de`ARCHIVED`. Você pode filtrar imagens `image-status` para ver somente imagens arquivadas ou somente imagens ativas.

Restaurando uma imagem

Quando você restaura uma imagem arquivada, ela é movida da classe de armazenamento ECR Archive de volta para a classe de armazenamento ECR Standard. As imagens restauradas são cobradas de acordo com as taxas de armazenamento padrão. O processo de restauração executa ações semelhantes às que ocorrem quando uma nova imagem é criada:

- A imagem fica disponível para extração quando a restauração for concluída. A restauração normalmente leva até 20 minutos, embora possa ser concluída mais rapidamente.

- Se a digitalização por push estiver ativada para o repositório, a imagem restaurada será digitalizada. Observe que os resultados de digitalização anteriores anteriores ao arquivamento da imagem não estarão disponíveis.
- Se a replicação estiver configurada para o repositório, a imagem restaurada será replicada se a replicação estiver habilitada no momento da restauração.
- A imagem restaurada aparece na lista de imagens ativas.

A restauração de uma imagem normalmente leva até 20 minutos, embora possa ser concluída mais rapidamente. Durante o processo de restauração, a imagem permanece no estado arquivado e não pode ser extraída até que a restauração seja concluída.

Console de gerenciamento da AWS

Para restaurar uma imagem arquivada

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/repositories.
2. Na barra de navegação, escolha a região que contém o repositório com a imagem arquivada que você deseja restaurar.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositórios, escolha o repositório que contém a imagem arquivada.
5. Escolha a guia Imagens arquivadas.
6. Selecione a imagem arquivada que você deseja restaurar.
7. Escolha Restaurar e confirme a ação de restauração.
8. Aguarde a conclusão da restauração. A imagem aparecerá na lista de imagens ativas quando a restauração for concluída.

AWS CLI

Para restaurar uma imagem arquivada

- Use o update-image-storage-class comando para restaurar uma imagem arquivada atualizando sua classe de armazenamento para`STANDARD`:

```
aws ecr update-image-storage-class \
    --repository-name my-repository \
```

```
    --image-id
imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
    --target-storage-class STANDARD
```

Quando você descreve imagens usando o describe-images comando, as imagens que estão sendo restauradas têm um `image-status` de`ACTIVATING`. Você pode filtrar imagens `image-status` com o valor `ACTIVATING` para visualizar as imagens que estão sendo restauradas no momento.

Um método alternativo para restaurar uma imagem arquivada é reenviar a imagem para o repositório. Quando você envia uma imagem que está atualmente arquivada, essa imagem é imediatamente restaurada e removida do arquivamento.

Remarcação de uma imagem no Amazon ECR

Com as imagens do esquema 2 do manifesto de imagem do Docker V2, você pode usar a opção `--image-tag` do comando put-image para remarcar uma imagem existente. Você pode remarcar sem extrair ou enviar a imagem com Docker. Para imagens maiores, esse processo economiza uma quantidade considerável de largura de banda e de tempo necessário para remarcar uma imagem.

Como remarcar uma imagem (AWS CLI)

Para remarcar uma imagem com a AWS CLI

1. Use o comando batch-get-image para obter o manifesto da imagem para remarcá-la e gravá-la em um arquivo. Neste exemplo, o manifesto de uma imagem com a tag,*latest*, no repositório,*amazonlinux*, é gravado em uma variável de ambiente chamada*MANIFEST*.

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

2. Use a opção `--image-tag` do comando put-image para colocar o manifesto da imagem no Amazon ECR com uma nova tag. Neste exemplo, a imagem é marcada como*2017.03*.

   > **Note**
   >
   > Se a `--image-tag` opção não estiver disponível na sua versão do AWS CLI, atualize para a versão mais recente. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface .

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

3. Verifique se a sua nova tag de imagem está conectada à imagem. Na saída a seguir, a imagem têm as tags `latest` e `2017.03`.

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

A saída é a seguinte:

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

Como remarcar uma imagem (AWS Tools for Windows PowerShell)

Para remarcar uma imagem com a AWS Tools for Windows PowerShell

1. Use o cmdlet de Get-ECRImageBatch para obter a descrição da imagem para remarcá-la e gravá-la em uma variável de ambiente. Neste exemplo, uma imagem com a tag, *`latest`*, no repositório,*`amazonlinux`*, é gravada na variável de ambiente,*`$Image`*.

> **Note**
>
> Se você não tiver o cmdlet de Get-ECRImageBatch disponível no sistema, consulte Configuração do AWS Tools for Windows PowerShell no Ferramentas da AWS para PowerShell Manual do usuário.

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

2. Grave o manifesto da imagem na variável de *$Manifest* ambiente.

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

3. Use a opção `-ImageTag` do cmdlet de Write-ECRImage para colocar o manifesto da imagem no Amazon ECR com uma nova tag. Neste exemplo, a imagem é marcada como*2017.09*.

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

4. Verifique se a sua nova tag de imagem está conectada à imagem. Na saída a seguir, a imagem têm as tags `latest` e `2017.09`.

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

A saída é a seguinte:

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

Impedir que as tags de imagens sejam sobrescritas no Amazon ECR

Você pode impedir que as tags de imagens sejam sobrescritas ao ativar a imutabilidade de tags em um repositório. Depois que a imutabilidade de tags estiver ativada, um erro `ImageTagAlreadyExistsException` será retornado se você tentar enviar uma imagem por push com uma tag que já existe no repositório. A imutabilidade de tags afeta todas as tags. Você não pode fazer com que algumas tags sejam imutáveis e outras não.

Você pode usar as AWS CLI ferramentas Console de gerenciamento da AWS e para definir a mutabilidade da tag de imagem para um novo repositório ou para um repositório existente. Para criar um repositório usando as etapas do console, consulte Criar um repositório privado do Amazon ECR para armazenar imagens.

Configurar a mutabilidade de tags de imagens (Console de gerenciamento da AWS)

Para configurar a mutabilidade de tags de imagens

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/repositories.
2. Na barra de navegação, selecione a região que contém o repositório a ser editado.
3. No painel de navegação, selecione Repositórios em Registro privado.

   Se você não vê Repositórios, escolha Registro privado para expandir o menu e, em seguida, escolha Repositórios.
4. Na página Repositórios privados, selecione o botão de opção antes do nome do repositório para o qual você deseja definir as configurações de mutabilidade de tag de imagem.
5. Selecione Ações e, em seguida, Repositório em Editar.
6. Para Mutabilidade de tag de imagem, escolha uma das configurações de mutabilidade de tag a seguir para o repositório.

   - Mutável – Escolha essa opção se quiser que as tags de imagem sejam sobrescritas. É recomendada para repositórios que usam ações de cache de pull-through para garantir que o Amazon ECR possa atualizar imagens armazenadas em cache. Além disso, para desabilitar as atualizações de tag para algumas tags mutáveis, insira os nomes das tags ou use curingas (*) para combinar várias tags semelhantes na caixa de texto Exclusão de tag mutável.

- Imutável – Selecione esta opção se quiser impedir que as tags de imagem sejam sobrescritas. Isso se aplica a todas as tags e exclusões no repositório ao enviar uma imagem com uma tag existente. O Amazon ECR retorna uma `ImageTagAlreadyExistsException` se você tentar enviar uma imagem com uma tag existente. Além disso, para habilitar as atualizações de tag para algumas tags imutáveis, insira os nomes das tags ou use curingas (*) para combinar várias tags semelhantes na caixa de texto Exclusão de tag imutável.

7. Em Image scan settings (Configurações de verificação de imagens), embora você possa especificar as configurações de verificação no nível do repositório para uma verificação básica, é prática recomendada especificar a configuração de verificação no nível do registro privado. Especificar as configurações de verificação no registro privado permite habilitar a verificação avançada ou a verificação básica, e também definir filtros para especificar quais repositórios são verificados. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no Amazon ECR.
8. Em Encryption settings (Configurações de criptografia), este é um campo somente para visualização, pois as configurações de criptografia de um repositório não podem ser alteradas depois que ele é criado.
9. Escolha Save (Salvar) para atualizar as configurações do repositório.

Configurar a mutabilidade de tags de imagens (AWS CLI)

Como criar um repositório com tags imutáveis configuradas

Use um dos comandos a seguir para criar um novo repositório de imagens com tags imutáveis configuradas.

- create-repository (AWS CLI) com mutabilidade de tag de imagem

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

- create-repository (AWS CLI) com filtros de exclusão de mutabilidade de tag de imagem

```
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) com mutabilidade da tag de imagem

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

- New-ECRRepository(AWS Tools for Windows PowerShell) com filtros de exclusão de mutabilidade da tag de imagem

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

Para atualizar as configurações de mutabilidade de tags de imagens para um repositório

Use um dos comandos a seguir para atualizar as configurações de mutabilidade de tag de imagem de um repositório existente.

- put-image-tag-mutability () com mutabilidade da tag de imagem 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) com filtros de exclusão de mutabilidade de tag de imagem

```
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) com mutabilidade da tag de imagem

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

- Write-ECRImageTagMutability(AWS Tools for Windows PowerShell) com filtros de exclusão de mutabilidade da tag de imagem

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

O formato de manifesto de imagem de contêiner é compatível com o Amazon ECR

O Amazon ECR oferece suporte aos seguintes formatos de manifesto de imagem de contêiner:

- Schema 1 de manifesto V2 de imagem de Docker (usado com o Docker versão 1.9 e anteriores)
- Schema 2 de manifesto V2 de imagem de Docker (usado com o Docker versão 1.10 e posteriores)
- Especificações do Open Container Initiative (OCI) (v1.0 e v1.1)

O suporte para o schema 2 de manifesto V2 de imagem de Docker fornece a seguinte funcionalidade:

- A possibilidade de usar várias tags para uma única imagem.
- Suporte para armazenar imagens de contêiner do Windows.

Conversão de manifesto de imagem do Amazon ECR

Quando você envia imagens para o Amazon ECR e extrai imagens do Amazon ECR, o cliente de mecanismo de contêiner (por exemplo, Docker) se comunica com o registro para acordar um formato de manifesto que seja entendido pelo cliente e pelo registro para ser usado na imagem.

Quando você envia uma imagem ao Amazon ECR com o Docker versão 1.9 ou anterior, o formato do manifesto da imagem é armazenado como manifesto de imagem do Docker V2 esquema 1. Quando você envia uma imagem ao Amazon ECR com o Docker versão 1.10 ou posterior, o formato do manifesto da imagem é armazenado como manifesto de imagem do Docker V2 esquema 2.

Quando você extrai uma imagem do Amazon ECR por tag, o Amazon ECR retorna o formato de manifesto de imagem que está armazenado no repositório. Mas somente se o formato é entendido pelo cliente. Se o formato do manifesto de imagem armazenado não é entendido pelo cliente, o Amazon ECR converte o manifesto de imagem em um formato que seja entendido pelo cliente. Por exemplo, se um cliente do Docker 1.9 solicitar um manifesto de imagem armazenado como manifesto de imagem do Docker V2 esquema 2, o Amazon ECR devolve-o no formato de manifesto de imagem do Docker V2 esquema 1. A tabela a seguir descreve as conversões disponíveis suportadas pelo Amazon ECR quando uma imagem é extraída por tag:

Schema solicitado pelo cliente	Enviado ao ECR como V2, schema 1	Enviado ao ECR como V2, schema 2	Enviado ao ECR como OCI
V2, schema 1	Não é necessário converter	Convertido em V2, schema 1	Não há conversões disponíveis
V2, schema 2	Não há conversões disponíveis, o cliente volta para V2, schema 1	Não é necessário converter	Convertido em V2, schema 2
OCI	Não há conversões disponíveis	Convertido em OCI	Não é necessário converter

Important

Se você extrair uma imagem por resumo, não há tradução disponível. O cliente precisa entender o formato do manifesto de imagem armazenado no Amazon ECR. Se você solicitar uma imagem do schema 2 de manifesto V2 de imagem de Docker por resumo em um cliente do Docker 1.9 ou anterior, a extração da imagem falhará. Para obter mais informações, consulte Compatibilidade de registro na documentação do Docker.
Neste exemplo, se você solicitar a mesma imagem por tag, o Amazon ECR converte o manifesto da imagem em um formato que o cliente possa entender. A extração da imagem é bem-sucedida.

Uso de imagens do Amazon ECR com o Amazon ECS

Você pode usar seus repositórios privados do Amazon ECR para hospedar imagens e artefatos de contêiner dos quais suas tarefas do Amazon ECS podem ser extraídas. Para que isso funcione, o agente de contêiner Amazon ECS, ou Fargate, deve ter permissões para fazer `ecr:BatchGetImage` o`ecr:GetDownloadUrlForLayer`, e. `ecr:GetAuthorizationToken` APIs

Permissões obrigatórias do IAM

A tabela a seguir mostra o perfil do IAM a ser usado, para cada tipo de inicialização, que fornece as permissões necessárias para que suas tarefas sejam extraídas de um repositório privado do Amazon ECR. O Amazon ECS fornece políticas do IAM gerenciadas que incluem as permissões necessárias.

Tipo de inicialização	perfil do IAM	AWS política de IAM gerenciada
Amazon ECS em instâncias do Amazon EC2	Use o perfil do IAM de instância de contêiner associado à instância do Amazon EC2 registrada em seu cluster do Amazon ECS. Para obter mais informaçõ es, consulte Perfil do IAM de instância de contêiner no Guia do desenvolvedor do Amazon Elastic Container Service.	`AmazonEC2Container ServiceforEC2Role` Para obter mais informaçõ es, consulte AmazonEC2 ContainerServiceforEC2Role no Guia do desenvolvedor do Amazon Elastic Container Service
Amazon ECS no Fargate	Use o perfil do IAM de execução de tarefas que você referencia em sua definição de tarefa do Amazon ECS. Para obter mais informações, consulte Perfil do IAM para execução de tarefa no Guia do desenvolvedor do Amazon Elastic Container Service.	`AmazonECSTaskExecu tionRolePolicy` Para obter mais informaçõ es, consulte AmazonECS TaskExecutionRolePolicy no Guia do desenvolvedor do Amazon Elastic Container Service.
Amazon ECS em instâncias externas	Use o perfil do IAM de instância de contêiner que está associado ao servidor on-premises ou à máquina virtual (VM) registrada no cluster do Amazon ECS. Para obter mais informações, consulte	`AmazonEC2Container ServiceforEC2Role` Para obter mais informaçõ es, consulte AmazonEC2 ContainerServiceforEC2Role no Guia do desenvolvedor

Tipo de inicialização	perfil do IAM	AWS política de IAM gerenciada
	Perfil do Amazon ECS de instância de contêiner no Guia do desenvolvedor do Amazon Elastic Container Service.	do Amazon Elastic Container Service.

Important

As políticas AWS gerenciadas do IAM contêm permissões adicionais que talvez você não precise para seu uso. Nesse caso, estas são as permissões mínimas necessárias para extrair de um repositório privado do Amazon ECR.

JSON

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

Especificar uma imagem do Amazon ECR em uma definição de tarefa do Amazon ECS

Ao criar uma definição de tarefa do Amazon ECS, é possível especificar uma imagem de contêiner hospedada em um repositório privado do Amazon ECR. Na definição de tarefa, verifique se você está usando a nomenclatura completa de `registry/repository:tag` para as imagens do

Amazon ECR. Por exemplo, *.aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-repository*:*latest*

O trecho de definição de tarefa a seguir mostra a sintaxe a ser usada para especificar uma imagem de contêiner hospedada no Amazon ECR em sua definição de tarefa do Amazon ECS.

```
{
    "family": "task-definition-name",
    ...
    "containerDefinitions": [
        {
            "name": "container-name",
            "image": "aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:latest",
            ...
        }
    ],
    ...
}
```

Uso de imagens do Amazon ECR com o Amazon EKS

Você pode usar as imagens do Amazon ECR com o Amazon EKS.

Ao fazer referência a uma imagem do Amazon ECR, você deverá usar a nomenclatura de `registry/repository:tag` completa da imagem. Por exemplo, *.aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-repository*:*latest*

Permissões obrigatórias do IAM

Se você tiver cargas de trabalho do Amazon EKS hospedadas em nós gerenciados, nós autogerenciados ou AWS Fargate, analise o seguinte:

- Workloads do Amazon EKS hospedadas em nós gerenciados ou autogerenciados: o perfil do IAM do nó de processamento do Amazon EKS (`NodeInstanceRole`) é obrigatório. A função do IAM do nó de processamento do Amazon EKS deve conter as seguintes permissões de política do IAM para o Amazon ECR.

  JSON

  ```
  {
  ```

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

> **Note**
>
> Se você usou `eksctl` os CloudFormation modelos em Getting Started with Amazon EKS para criar seu cluster e grupos de nós de trabalho, essas permissões do IAM são aplicadas à sua função do IAM do nó de trabalho por padrão.

- Cargas de trabalho do Amazon EKS hospedadas em AWS Fargate: Use a função de execução de pods do Fargate, que fornece aos pods permissão para extrair imagens de repositórios privados do Amazon ECR. Para obter mais informações, consulte Criar uma função de execução do pod do Fargate.

Instalar um chart do Helm em um cluster do Amazon EKS

Os charts do Helm hospedados no Amazon ECR podem ser instalados nos clusters do Amazon EKS.

Pré-requisitos

- Use a versão mais recente do cliente do Helm. Estas etapas foram escritas usando a versão `3.9.0` do Helm. Para obter mais informações, consulte Instalação do Helm.
- Você tem pelo menos a versão `1.23.9` ou `2.6.3` da AWS CLI instalada em seu computador. Para obter mais informações, consulte Instalar ou atualizar a versão mais recente da AWS CLI.
- Você enviou um chart do Helm para o seu repositório do Amazon ECR. Para obter mais informações, consulte Para enviar um chart do Helm por push para um repositório privado do Amazon ECR.

- Você configurou `kubectl` para trabalhar com o Amazon EKS. Para obter mais informações, consulte Criar um `kubeconfig` para o Amazon EKS no Manual do usuário do Amazon EKS. Se os comados a seguir forem bem-sucedidos para o cluster, a configuração estará correta.

```
kubectl get svc
```

Para instalar um chart do Helm em um cluster do Amazon EKS

1. Autentique o cliente do Helm para o registro do Amazon ECR no qual o chart do Helm está hospedado. Os tokens de autenticação devem ser obtidos para cada registro usado e são válidos por 12 horas. Para obter mais informações, consulte Autenticação de registro privado no 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. Instale o chart. *helm-test-chart*Substitua pelo seu repositório e pela *0.1.0* tag do gráfico do Helm.

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

   A saída deve ser semelhante a esta:

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

3. Verifique a instalação do chart.

```
helm list -n default
```

   Resultado do exemplo:

```
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. (Opcional) Consulte o chart do Helm instalado `ConfigMap`.

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

5. Ao concluir, você pode remover a versão do chart do seu cluster.

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

Assine imagens no Amazon ECR

O Amazon ECR se integra AWS Signer para fornecer duas maneiras de você assinar suas imagens de contêiner: assinatura gerenciada (automática, recomendada) e assinatura manual (do lado do cliente). Você pode armazenar as imagens do contêiner e as assinaturas em seus repositórios privados.

Escolha um método de assinatura

O Amazon ECR oferece suporte a dois métodos para assinar imagens de contêineres:

Assinatura gerenciada (recomendada)

A assinatura gerenciada gera automaticamente assinaturas criptográficas quando as imagens são enviadas para o Amazon ECR. Esse método simplifica a configuração. A assinatura gerenciada é a abordagem recomendada para a maioria dos usuários. Para obter mais informações, consulte Assinatura gerenciada.

Assinatura manual

A assinatura manual usa a CLI AWS Signer e o plug-in do Notation para assinar imagens antes de enviá-las para o Amazon ECR. Esse método fornece mais controle sobre o processo de assinatura e é útil quando você precisa assinar imagens fora do fluxo de trabalho push ou exige um controle refinado sobre as operações de assinatura. Para obter mais informações, consulte Assinatura manual.

Considerações

O seguinte deve ser considerado ao usar a assinatura de imagens do Amazon ECR:

- As assinaturas armazenadas em seu repositório contam para a cota do serviço do número máximo de imagens por repositório. Cada assinatura conta como 1 artefato em relação às imagens por cota de repositório. Para obter mais informações, consulte Cotas de serviço do Amazon ECR.
- Quando artefatos de referência estão presentes em um repositório, as políticas de ciclo de vida do Amazon ECR limparão automaticamente esses artefatos em até 24 horas após a exclusão da imagem em questão.

Assinatura gerenciada

A assinatura gerenciada do Amazon ECR assina automaticamente suas imagens de contêiner gerando assinaturas criptográficas usando o AWS Signer quando as imagens são enviadas para o Amazon ECR. Isso elimina a necessidade de instalar e configurar ferramentas do lado do cliente e permite que você controle centralmente a assinatura como uma configuração de registro.

Pré-requisitos

Para configurar a assinatura gerenciada, você cria uma configuração de assinatura com o Amazon ECR que faz referência a um ou mais perfis de assinatura de signatários e, opcionalmente, filtros de repositório que restringem quais repositórios devem ter suas imagens assinadas. Depois de configurada, a assinatura gerenciada do Amazon ECR assina automaticamente as imagens à medida que elas são enviadas usando a identidade da entidade que está enviando a imagem.

Antes de configurar a assinatura gerenciada, você deve ter o seguinte:

- Um perfil de assinatura de signatário — Crie pelo menos um perfil de assinatura de signatário. Um perfil de assinatura é um recurso exclusivo do AWS signatário que você pode usar para realizar operações de assinatura no Amazon ECR. Os perfis de assinatura permitem que você assine e verifique artefatos de código, como imagens de contêineres e pacotes de AWS Lambda implantação. Cada perfil de assinatura designa a plataforma de assinatura a ser assinada, um ID da plataforma e outras informações específicas da plataforma. Por exemplo, um ARN de perfil de assinatura se parece com isso:. `arn:partition:signer:region:account-id:/signing-profiles/profile-name`
- Permissões do IAM — O diretor do IAM que envia a imagem deve ter as permissões do IAM necessárias para acessar o perfil de assinatura relevante do Signer e o repositório ECR relevante. Você precisa modificar a política baseada em identidade do IAM principal para incluir permissões tanto para as operações do repositório ECR quanto para as operações de assinatura do Signer. O exemplo de política a seguir mostra as permissões necessárias:

```
{
   "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"
    }
    ]
}
```

Com a assinatura gerenciada do Amazon ECR, você pode criar várias regras de assinatura (até 10 por registro) para criar limites de segurança mais fortes. Por exemplo, você pode executar vários pipelines de compilação e querer limitar quais repositórios cada pipeline pode assinar. Em cada regra, você configura um perfil de assinatura e especifica filtros de nome de repositório. Quando uma nova imagem é enviada, o Amazon ECR corresponde à regra de assinatura e ao perfil de assinatura que podem assinar a imagem. Se houver várias correspondências, o Amazon ECR gera várias assinaturas.

> **Note**
>
> Se você verificar as assinaturas manualmente, ainda precisará instalar a CLI do Notation.

> **Note**
>
> A assinatura gerenciada do Amazon ECR está disponível em todas as AWS regiões onde a assinatura de imagens de contêineres com o AWS Signer está disponível.

Introdução

Siga estas etapas para configurar a assinatura gerenciada. Você fornece ao Amazon ECR uma referência a um perfil de assinatura do signatário e, opcionalmente, filtros que restringem quais repositórios devem ter suas imagens assinadas.

Console de gerenciamento da AWS

Use as etapas a seguir para configurar a assinatura gerenciada usando Console de gerenciamento da AWS o.

1. Abra o console do Amazon ECR. No painel de navegação esquerdo, selecione Registro privado, Recursos e configurações, Assinatura gerenciada.
2. Na página Regras de assinatura, selecione Criar regra.
3. Na página Perfil de assinatura, em Selecionar um perfil de AWS signatário, escolha Criar novo perfil de AWS signatário, insira um nome de perfil e, opcionalmente, altere o período de validade da assinatura. Depois, selecione Próximo.
4. Na página Filtros, em Selecionar repositórios, insira um filtro de nome de repositório. Depois, selecione Próximo.
5. Na página Revisar e criar, verifique o perfil do AWS signatário e os filtros de nome do repositório que você inseriu. Se tudo estiver correto, selecione Salvar.

AWS CLI

Use os AWS CLI comandos a seguir para configurar a assinatura gerenciada.

- Crie uma regra de assinatura

  Crie uma configuração de assinatura com o ARN do seu perfil de assinatura. Crie um arquivo JSON com o seguinte conteúdo:

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

```
                    "filterType": "WILDCARD_MATCH"
                }
            ]
        }
    ]
}
```

Em seguida, execute o seguinte comando:

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

Você deve ver a resposta da API contendo a configuração de assinatura.

- Veja sua configuração de assinatura

  Recupere sua configuração de assinatura:

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

  Você deve ver a resposta da API contendo a configuração de assinatura.

- Verifique o status de assinatura da imagem

  Envie uma imagem para o seu repositório. Por exemplo:

  ```
  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
  ```

  Depois de pressionar, use sua tag de imagem para verificar o status da assinatura:

  ```
  aws ecr --region region \
      describe-image-signing-status \
      --repository-name repository-name \
  ```

```
    --image-id imageTag=test-1
```

Se o nome do repositório corresponder ao filtro do repositório definido na configuração de assinatura, você deverá ver o status da assinatura na resposta da API. Se o status for bem-sucedido, você deverá ver uma assinatura enviada ao seu repositório.

- Exclua sua configuração de assinatura

  Exclua sua configuração de assinatura:

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

  Você deve ver a resposta da API contendo a configuração de assinatura excluída.

Considerações

As seguintes limitações e recursos se aplicam à assinatura gerenciada:

- A assinatura entre regiões não é suportada — os perfis de assinatura devem estar na mesma região do seu registro do Amazon ECR. Você não pode usar um perfil de assinatura de uma região para assinar imagens em um registro localizado em uma região diferente.
- A assinatura entre contas é suportada — Os perfis de assinatura podem estar em contas diferentes das do seu registro do Amazon ECR. Isso permite que as organizações gerenciem centralmente os perfis de assinatura e, ao mesmo tempo, permite que desenvolvedores de outras contas os usem. Para obter mais informações, consulte Configurar a assinatura entre contas para o Signer no Guia do AWS Signer desenvolvedor.
- As assinaturas não podem ser assinadas — Você não pode assinar as assinaturas por si só. Somente imagens de contêiner podem ser assinadas.

Verificação de assinatura

Depois de assinar as imagens do contêiner, você pode verificar as assinaturas para garantir que as imagens não tenham sido adulteradas e venham de uma fonte confiável. O Amazon ECR oferece suporte a vários métodos de verificação de assinaturas:

Verificação gerenciada com o Amazon EKS

O Amazon EKS fornece integração nativa para verificação automática de assinaturas. Quando você configura a verificação de assinatura em seus clusters do Amazon EKS, o serviço verifica automaticamente as assinaturas das imagens antes de permitir que os contêineres sejam executados. Para obter mais informações sobre como configurar a verificação de assinatura, consulte Validar assinaturas de imagens de contêineres durante a implantação no Guia do usuário do Amazon EKS.

Controlador de admissão Lambda para Amazon ECS

O Amazon ECS fornece ganchos do ciclo de vida do serviço que permitem que você execute uma lógica personalizada durante as implantações do serviço. Esses ganchos podem acionar AWS Lambda funções em pontos específicos do processo de implantação, permitindo que você valide assinaturas de imagens de contêineres antes de permitir o início dos serviços. Para obter mais informações, consulte Verificar assinaturas de imagens de contêineres para o Amazon ECS no Guia do AWS Signer desenvolvedor.

Verificação manual com Notation CLI

Você pode verificar as assinaturas manualmente usando a CLI do Notation. Esse método exige que você instale e configure o Notation CLI em sua máquina local ou em seu ambiente de verificação. Para obter instruções detalhadas sobre como verificar uma imagem usando o Notation CLI, consulte Verificar uma imagem localmente após fazer login no Guia do desenvolvedor.AWS Signer

Configurar a autenticação para o cliente Notation

Se você usa a assinatura manual ou verifica assinaturas manualmente usando a CLI do Notation, você deve configurar o cliente do Notation para que ele possa se autenticar no Amazon ECR. Se o Docker estiver instalado no mesmo host em que você instalou o cliente do Notation, o Notation reutilizará o mesmo método de autenticação usado para o cliente do Docker. O Docker `login` e `logout` os comandos permitirão que o Notation `sign` e `verify` os comandos usem essas mesmas credenciais, e você não precisará autenticar separadamente o Notation. Para obter mais informações sobre como configurar o cliente do Notation para autenticação, consulte Authenticate with OCI-compliant registries na documentação do projeto Notary.

Se você não estiver usando o Docker ou outra ferramenta que use as credenciais do Docker, recomendamos que use o auxiliar de credenciais do Docker doo Amazon ECR como repositório de

credenciais. Para obter mais informações sobre como instalar e configurar o auxiliar de credenciais do Amazon ECR, consulte Amazon ECR Docker Credential Helper.

Assinatura manual

A assinatura manual usa a CLI AWS Signer e o plug-in do Notation para assinar imagens antes de enviá-las para o Amazon ECR. Esse método fornece mais controle sobre o processo de assinatura e é útil quando você precisa assinar imagens fora do fluxo de trabalho push ou exige um controle refinado sobre as operações de assinatura.

Para obter instruções detalhadas sobre como assinar imagens de contêiner usando a CLI AWS Signer do Notation, consulte Assinar imagens de contêiner no Signer e os tópicos relacionados no AWS Signer Guia do desenvolvedor.

Pré-requisitos

Antes de começar, certifique-se de que os seguintes pré-requisitos sejam atendidos.

- Instale e configure a versão mais recente do AWS CLI. Para obter mais informações, consulte Instalar ou atualizar a versão mais recente da AWS CLI no Guia do usuário do AWS Command Line Interface .
- Instale a CLI do Notation e o plug-in para AWS Signer o Notation. Para obter mais informações, consulte Pré-requisitos para assinar imagens de contêiner no Guia do desenvolvedor do AWS Signer .
- Tenha uma imagem de contêiner armazenada em um repositório privado do Amazon ECR para assinar. Para obter mais informações, consulte Enviar por push uma imagem para um repositório privado do Amazon ECR.

Verificar imagens quanto a vulnerabilidades do software no Amazon ECR

A verificação de imagens do Amazon ECR ajuda a identificar as vulnerabilidades do software nas imagens do contêineres. Os seguintes tipos de verificação são oferecidos.

> **Important**
>
> Alternar entre o escaneamento avançado e o escaneamento básico fará com que os escaneamentos previamente estabelecidos não estejam mais disponíveis. Você precisará configurar as verificações novamente. No entanto, se você voltar para o tipo de escaneamento anterior, os escaneamentos estabelecidos estarão disponíveis.

> **Note**
>
> As imagens arquivadas não podem ser digitalizadas. As imagens arquivadas devem ser restauradas antes de serem digitalizadas. Para obter mais informações sobre arquivamento e restauração de imagens, consulte. Arquivamento de uma imagem no Amazon ECR

- Verificação avançada – o Amazon ECR se integra ao Amazon Inspector para fornecer a verificação automatizada e contínua de seus repositórios. Suas imagens de contêiner são verificadas quanto a vulnerabilidades em sistemas operacionais e pacotes de linguagem de programação. À medida que novas vulnerabilidades aparecem, os resultados da verificação são atualizados e o Amazon Inspector emite um evento EventBridge para notificá-lo. A verificação avançada fornece o seguinte:
  - Vulnerabilidades dos pacotes de sistemas operacionais e linguagens de programação
  - Duas frequências de verificação: verificação por push e verificação contínua
- Escaneamento básico — O Amazon ECR usa tecnologia AWS nativa com o banco de dados Common Vulnerabilities and Exposures (CVEs) para verificar as vulnerabilidades do sistema operacional.

  Com a verificação básica, você pode configurar seus repositórios para verificação durante o envio ou executar verificações manuais, e o Amazon ECR fornece uma lista de descobertas da verificação. A verificação básica fornece o seguinte:

- Verificações do sistema operacional
- Duas frequências de verificação: manual e verificação por push

> **Important**
>
> A nova versão de verificação básica do Amazon ECR não usa os atributos `imageScanFindingsSummary` e `imageScanStatus` da resposta da API `DescribeImages` para retornar os resultados da verificação. No lugar dela, use a API `DescribeImageScanFindings`. Para obter mais informações, consulte `DescribeImageScanFindings`.

Filtros para escolher quais repositórios são verificados no Amazon ECR

Ao configurar a verificação de imagens para o registro privado, você pode usar filtros para escolher quais repositórios serão verificados.

Quando a digitalização básica é usada, você pode especificar digitalização em filtros push para especificar quais repositórios estão definidos para fazer uma digitalização de imagem quando novas imagens forem enviadas. Qualquer repositório que não corresponda a um escaneamento básico no filtro push será configurado para a frequência de escaneamento manual, o que significa que, para realizar um escaneamento, você deve acionar manualmente o escaneamento.

Quando a digitalização aprimorada é usada, você pode especificar filtros separados para digitalização em push e digitalização contínua. Quaisquer repositórios que não correspondam a um filtro de digitalização aprimorada terão a digitalização desativada. Se você estiver usando a digitalização aprimorada e especificar filtros separados para digitalização em push e digitalização contínua, onde vários filtros correspondem ao mesmo repositório, o Amazon ECR impõe o filtro de digitalização contínua em vez da digitalização no filtro push para esse repositório.

Filtrar curingas

Quando um filtro é especificado, um filtro sem curinga corresponderá a todos os nomes de repositórios que contêm o filtro. Um filtro com um curinga (*) corresponde a qualquer nome de repositório em que o curinga substitui zero ou mais caracteres no nome do repositório.

A tabela a seguir fornece exemplos em que os nomes de repositórios podem ser visualizados no eixo horizontal e os filtros de exemplo são especificados no eixo vertical.

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod	Sim	Sim	Sim	Sim	Sim
*prod	Sim	Sim	Não	Não	Não
prod*	Sim	Não	Sim	Não	Sim
prod	Sim	Sim	Sim	Sim	Sim
prod*repo	Não	Não	Sim	Não	Sim

Verificar imagens em busca de vulnerabilidades dos pacotes de sistemas operacionais e de linguagens de programação no Amazon ECR

A verificação avançada do Amazon ECR é uma integração com o Amazon Inspector que permite a verificação de vulnerabilidades para suas imagens de contêiner. Suas imagens de contêiner são verificadas quanto a vulnerabilidades em sistemas operacionais e pacotes de linguagem de programação. Você pode ver as descobertas da verificação com o Amazon ECR e com o Amazon Inspector diretamente. Para obter mais informações sobre o Amazon Inspector, consulte Verificação de imagens de contêiner com o Amazon Inspector no Guia do usuário do Amazon Inspector.

Com a verificação avançada, você pode escolher quais repositórios são configurados para verificação automática e contínua e quais são configurados para verificação ao enviar. Isso é feito com a configuração de filtros de verificação.

Considerações sobre a verificação avançada

Os aspectos a seguir devem ser considerados antes de habilitar a verificação avançada do Amazon ECR.

- Não há custo adicional do Amazon ECR para usar esse recurso, no entanto, há um custo do Amazon Inspector para a digitalização das suas imagens. Esse atributo está disponível nas regiões com suporte do Amazon Inspector. Para obter mais informações, consulte:
  - Preços do Amazon Inspector – Preços do Amazon Inspector.
  - Regiões com suporte pelo Amazon Inspector – Regiões e endpoints.
- A verificação aprimorada do Amazon ECR mostra como as imagens são usadas no Amazon EKS e no Amazon ECS. Você pode ver quando as imagens foram usadas pela última vez e identificar quantos clusters usam cada imagem. Essas informações ajudam a priorizar a correção de vulnerabilidade das imagens usadas ativamente. Você pode determinar rapidamente quais clusters podem ser afetados por vulnerabilidades recém-descobertas. Para obter mais informações sobre como solicitar essas informações e visualizar a resposta, consulte `DescribeImageScanFindings`.
- O Amazon Inspector oferece suporte à verificação para sistemas operacionais específicos. Para obter uma lista completa, consulte Sistemas operacionais compatíveis: verificação do Amazon ECR no Guia do usuário do Amazon Inspector.
- O Amazon Inspector usa uma função do IAM vinculada ao serviço que fornece as permissões necessárias para disponibilizar a verificação avançada para seus repositórios. O perfil do IAM vinculado ao serviço é criado automaticamente pelo Amazon Inspector quando a verificação avançada é ativada para seu registro privado. Para obter mais informações, consulte Usar funções vinculadas ao serviço do Amazon Inspector no Guia do usuário do Amazon Inspector.
- Quando você ativa inicialmente a digitalização aprimorada para seu registro privado, o Amazon Inspector reconhece apenas as imagens enviadas para o Amazon ECR nos últimos 14 dias, com base na data e hora do envio da imagem. Imagens mais antigas terão o status de verificação `SCAN_ELIGIBILITY_EXPIRED`. Se desejar que essas imagens sejam verificadas pelo Amazon Inspector, você deverá enviá-las novamente para o seu repositório.
- Quando a verificação avançada está ativada no registro privado do Amazon ECR, todos os repositórios que correspondem aos filtros de verificação são verificados usando somente a verificação avançada. Todos os repositórios que não corresponderem a um filtro terão uma frequência de digitalização `Off`, mas não serão verificados. Não há suporte a digitalizações manuais com a digitalização avançada. Para obter mais informações, consulte Filtros para escolher quais repositórios são verificados no Amazon ECR.
- Se você especificar filtros separados para digitalização em push e digitalização contínua, onde vários filtros correspondem ao mesmo repositório, o Amazon ECR impõe o filtro de digitalização contínua em vez da digitalização no filtro push para esse repositório.

- Quando a verificação aprimorada é ativada, o Amazon ECR envia um evento para EventBridge quando a frequência de varredura de um repositório é alterada. O Amazon Inspector emite eventos para EventBridge quando uma varredura inicial é concluída e quando uma descoberta de digitalização de imagem é criada, atualizada ou fechada.

Alterar a duração da verificação avançada de imagens no Amazon Inspector

Após a ativação da verificação aprimorada, o Amazon ECR verifica continuamente as imagens enviadas recentemente durante o período configurado. Por padrão, o Amazon Inspector monitora seus repositórios até que as imagens sejam excluídas ou a verificação aprimorada seja desativada. Você pode configurar a duração da data de envio (até a vida útil) e a duração da nova verificação no console do Amazon Inspector para atender às necessidades do seu ambiente. Quando a duração da verificação de um repositório terminar, o status da verificação é exibido como `SCAN_ELIGIBILITY_EXPIRED`. Para obter mais informações sobre como definir as configurações de duração da nova verificação para o Amazon ECR no Amazon Inspector, consulte Configurar a duração da nova verificação do Amazon ECR no Guia do usuário do Amazon Inspector.

Permissões do IAM necessárias para a verificação avançada no Amazon ECR

O escaneamento aprimorado do Amazon ECR requer uma função do IAM vinculada ao serviço Amazon Inspector e que o principal do IAM que habilita e usa o escaneamento aprimorado tenha permissões para chamar o Amazon Inspector necessário para o escaneamento. APIs O perfil do IAM vinculado ao serviço do Amazon Inspector é criado automaticamente pelo Amazon Inspector quando a verificação avançada é ativada para seu registro privado. Para obter mais informações, consulte Usar funções vinculadas ao serviço do Amazon Inspector no Guia do usuário do Amazon Inspector.

A política do IAM a seguir concede as permissões necessárias para habilitar e usar a verificação avançada. Ela inclui a permissão necessária para que o Amazon Inspector crie o perfil do IAM vinculado ao serviço, bem como as permissões da API do Amazon Inspector necessárias para ativar e desativar a verificação avançada e recuperar as descobertas da verificação.

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"
                    ]
                }
            }
        }
    ]
}
```

Configurar a verificação avançada para imagens do Amazon ECR

Configure a verificação avançada por região para o registro privado.

Verifique se você tem as permissões do IAM adequadas para configurar a verificação avançada. Para mais informações, consulte Permissões do IAM necessárias para a verificação avançada no Amazon ECR.

Console de gerenciamento da AWS

Para ativar a verificação avançada para o registro privado

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.

2. Na barra de navegação, selecione a região para a qual deseja definir a configuração de verificação.
3. No painel de navegação, escolha Registro privado e depois Configurações.
4. Na página Scanning configuration (Configuração da verificação), em Scan type (Tipo de verificação), escolha Enhanced scanning (Verificação avançada).

   Por padrão, quando a Verificação avançada está selecionada, todos os repositórios são verificados continuamente.
5. Para escolher repositórios específicos para verificar continuamente, desmarque a caixa Verificar continuamente todos os repositórios e defina os filtros:

   > **Important**
   >
   > Um filtro sem curinga corresponderá a todos os nomes de repositórios que contêm o filtro. Filtros com curingas (*) correspondem a qualquer nome de repositório em que o curinga substitui zero ou mais caracteres no nome do repositório. Para ver exemplos de como os filtros se comportam, consulte the section called “Filtrar curingas”.

   a. Insira um filtro com base nos nomes dos repositórios e escolha Adicionar filtro.
   b. Decida quais repositórios verificar quando uma imagem for enviada por push:

      - Para verificar todos os repositórios por push, selecione Verificar por push todos os repositórios.
      - Para escolher repositórios específicos para verificar por push, insira um filtro com base nos nomes dos repositórios e escolha Adicionar filtro.
6. Escolha Salvar.
7. Repita estas etapas em cada região na qual deseja ativar a verificação avançada.

AWS CLI

Use o AWS CLI comando a seguir para ativar a verificação aprimorada do seu registro privado usando AWS CLI o. Você pode especificar filtros de verificação usando o objeto `rules`.

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

O exemplo a seguir ativa a verificação avançada para seu registro privado. Por padrão, quando `rules` não são especificadas, o Amazon ECR define a configuração de verificação para verificação contínua para todos os repositórios.

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

O exemplo a seguir ativa a verificação avançada para seu registro privado e especifica um filtro de verificação. O filtro de verificação no exemplo ativa a verificação contínua para todos os repositórios com `prod` em seu nome.

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

O exemplo a seguir ativa a verificação avançada para seu registro privado e especifica vários filtros de verificação. Os filtros de verificação no exemplo ativam a verificação contínua para todos os repositórios com `prod` em seu nome e ativam a verificação somente ao enviar para todos os demais repositórios.

```
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 eventos enviados para digitalização aprimorada no Amazon ECR

Quando a verificação aprimorada é ativada, o Amazon ECR envia um evento para EventBridge quando a frequência de varredura de um repositório é alterada. O Amazon Inspector envia eventos para EventBridge quando uma varredura inicial é concluída e quando uma descoberta de digitalização de imagem é criada, atualizada ou fechada.

Evento para uma alteração de frequência de verificação do repositório

Quando a verificação avançada está ativada para seu registro, o evento a seguir é enviado pelo Amazon ECR quando há uma alteração em um recurso que tem a verificação avançada ativada. Isso inclui novos repositórios sendo criados, a frequência de verificação de um repositório sendo alterada ou quando as imagens são criadas ou excluídas em repositórios com a verificação avançada ativada. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no 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"
 }
```

```
}
```

Evento para uma verificação de imagem inicial (verificação avançada)

Quando a verificação avançada está ativada para seu registro, o evento a seguir é enviado pelo Amazon Inspector quando a verificação de imagem inicial é concluída. O parâmetro `finding-severity-counts` só retornará um valor de um nível de gravidade se existir algum. Por exemplo, se a imagem não contiver descobertas no nível `CRITICAL`, não será retornada uma contagem crítica. Para obter mais informações, consulte Verificar imagens em busca de vulnerabilidades dos pacotes de sistemas operacionais e de linguagens de programação no Amazon ECR.

Padrão de evento:

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

Resultado do exemplo:

```
{
    "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"
        ]
    }
}
```

Evento para uma atualização de descoberta de verificação de imagem (verificação avançada)

Quando a verificação avançada está ativada para seu registro, o evento a seguir é enviado pelo Amazon Inspector quando a descoberta da verificação de imagem é criada, atualizada ou fechada. Para obter mais informações, consulte Verificar imagens em busca de vulnerabilidades dos pacotes de sistemas operacionais e de linguagens de programação no Amazon ECR.

Padrão de evento:

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

Resultado do exemplo:

```
{
    "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"
    }
}
```

Recuperar as descobertas de verificações avançadas no Amazon ECR

Você pode recuperar as descobertas de verificação da última verificação avançada de imagens concluída e, em seguida, abri-las no Amazon Inspector para ver mais detalhes. As vulnerabilidades de software que foram descobertas são listadas por gravidade com base no banco de dados Common Vulnerabilities and Exposures (). CVEs

Para obter detalhes de solução de problemas para alguns problemas comuns ao digitalizar imagens, consulte Solucionar problemas de verificação de imagens no Amazon ECR.

Console de gerenciamento da AWS

Use as etapas a seguir para recuperar as descobertas da verificação de imagem usando o Console de gerenciamento da AWS.

Para recuperar descobertas da verificação de imagens

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região em que seu repositório reside.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositories (Repositórios), escolha o repositório que contém a imagem para a qual recuperar as descobertas da verificação.
5. Na página Imagens, na coluna Tag da imagem, selecione a tag da imagem para recuperar as descobertas da verificação.
6. Para ver mais detalhes no console do Amazon Inspector, escolha o nome da vulnerabilidade na coluna Nome.

AWS CLI

Use o AWS CLI comando a seguir para recuperar os resultados da digitalização de imagens usando o. AWS CLIÉ possível especificar uma imagem usando a `imageTag` ou o `imageDigest`. Ambos podem ser obtidos usando o comando list-images da CLI.

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

  O exemplo a seguir usa uma tag de imagem.

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

  O exemplo a seguir usa um resumo de imagem.

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

Verificar imagens quanto a vulnerabilidades do sistema operacional no Amazon ECR

O escaneamento básico do Amazon ECR usa tecnologia AWS nativa para escanear as imagens do seu contêiner em busca de vulnerabilidades de software. O escaneamento básico fornece detecção de vulnerabilidades em um amplo conjunto de sistemas operacionais populares, fornecendo mais de 50 feeds de dados para gerar descobertas sobre vulnerabilidades e exposições comuns (). CVEs Essas fontes incluem recomendações de segurança de fornecedores, feeds de dados, feeds de inteligência de ameaças, bem como o National Vulnerability Database (NVD) e o MITRE.

O escaneamento básico do Amazon ECR é suportado em todas as regiões listadas em AWS Serviços por região.

O Amazon ECR usa a severidade de um CVE da fonte de distribuição upstream, se disponível. Do contrário, é usada a pontuação do Common Vulnerability Scoring System (CVSS). A pontuação do

CVSS pode ser usada para obter a classificação de gravidade de vulnerabilidade do NVD. Para obter mais informações, consulte Classificações de gravidade de vulnerabilidade do NVD.

O escaneamento básico do Amazon ECR suporta filtros para especificar quais repositórios devem ser escaneados por push. Todos os repositórios que não correspondam a um escaneamento no filtro push são configurados para a frequência de escaneamento manual, o que significa que você deve iniciar o escaneamento manualmente. Uma imagem só pode ser verificada uma vez a cada 24 horas. Essas 24 horas incluem a verificação inicial por push, se configurada, e quaisquer verificações manuais. Com a verificação básica, você pode verificar até 100.000 imagens por 24 horas em um determinado registro.

As últimas descobertas da verificação de imagem concluídas podem ser recuperadas para cada imagem. Quando uma digitalização de imagem é concluída, o Amazon ECR envia um evento para a Amazon EventBridge. Para obter mais informações, consulte Eventos do Amazon ECR e EventBridge.

Suporte do sistema operacional para digitalização básica

Como prática recomendada de segurança e para cobertura contínua, é recomendável continuar usando as versões compatíveis de um sistema operacional. No entanto, de acordo com a política dos provedores, sistemas operacionais descontinuados não são mais atualizados com patches e, em muitos casos, novos avisos de segurança não são mais lançados para eles. Além disso, alguns fornecedores removem os alertas e detecções de segurança existentes de seus feeds quando um sistema operacional afetado chega ao fim do suporte padrão. Quando uma distribuição perde o suporte de seu provedor, o Amazon ECR pode não mais ser compatível com a verificação quanto a vulnerabilidades dessa distribuição. Qualquer descoberta que o Amazon ECR gerar para um sistema operacional descontinuado deverá ser usada apenas para fins informativos. Para obter uma lista completa dos sistemas operacionais e versões compatíveis, consulte Sistemas operacionais compatíveis - Análise do Amazon Inspector no Guia do usuário do Amazon Inspector.

Configurar a verificação avançada para imagens no Amazon ECR

Por padrão, o Amazon ECR ativa a verificação básica em todos os registros privados. Como resultado, a menos que você tenha alterado as configurações de verificação no registro privado, não há necessidade de ativar a verificação básica.

Você pode usar as etapas a seguir para definir um ou mais filtros de verificação por push.

Para ativar a verificação básica para o registro privado

1. Abra o console do Amazon ECR em registros https://console.aws.amazon.com/ecr/ privados/ repositórios
2. Na barra de navegação, selecione a região para a qual deseja definir a configuração de verificação.
3. No painel de navegação, escolha Registro privado, Digitalização.
4. Na página Scanning configuration (Configuração da verificação), em Scan type (Tipo de verificação), escolha Basic scanning (Verificação básica).
5. Por padrão, todos os repositórios estão definidos como verificação Manual. Você também pode configurar a verificação ao enviar especificando Filtros de verificação ao enviar. Você pode definir a verificação ao enviar para todos os repositórios ou para repositórios individuais. Para obter mais informações, consulte Filtros para escolher quais repositórios são verificados no Amazon ECR.

> **Note**
>
> Se a digitalização por push estiver habilitada para um repositório, as digitalizações também serão feitas em imagens que são restauradas após serem arquivadas. Nenhuma digitalização antiga estará disponível na imagem restaurada.

Verificação manual de uma imagem quanto a vulnerabilidades do sistema operacional no Amazon ECR

Se os repositórios não estiverem configurados para verificação por push, você poderá iniciar manualmente as verificações de imagens. Uma imagem só pode ser verificada uma vez a cada 24 horas. Essas 24 horas incluem a verificação inicial por push, se configurada, e quaisquer verificações manuais.

Para obter detalhes de solução de problemas para alguns problemas comuns ao digitalizar imagens, consulte Solucionar problemas de verificação de imagens no Amazon ECR.

Console de gerenciamento da AWS

Use as etapas a seguir para iniciar uma verificação manual de imagem usando o Console de gerenciamento da AWS.

1. Abra o console do Amazon ECR em registros https://console.aws.amazon.com/ecr/ privados/repositórios
2. Na barra de navegação, selecione a região na qual criará o seu repositório.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositories (Repositórios), escolha o repositório que contém a imagem a ser verificada.
5. Na página Images (Imagens) selecione a imagem a ser verificada e escolha Scan (Verificar).

AWS CLI

- start-image-scan (AWS CLI)

  O exemplo a seguir usa uma tag de imagem.

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

  O exemplo a seguir usa um resumo de imagem.

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

AWS Tools for Windows PowerShell

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

  O exemplo a seguir usa uma tag de imagem.

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

  O exemplo a seguir usa um resumo de imagem.

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

Recuperar as descobertas de verificações básicas no Amazon ECR

Você pode recuperar as descobertas de verificação da última verificação de imagens concluída. As vulnerabilidades de software que foram descobertas são listadas por gravidade com base no banco de dados Common Vulnerabilities and Exposures (). CVEs

Para obter detalhes de solução de problemas para alguns problemas comuns ao digitalizar imagens, consulte Solucionar problemas de verificação de imagens no Amazon ECR.

Console de gerenciamento da AWS

Use as etapas a seguir para recuperar as descobertas da verificação de imagem usando o Console de gerenciamento da AWS.

Para recuperar descobertas da verificação de imagens

1. Abra o console do Amazon ECR em registros https://console.aws.amazon.com/ecr/ privados/ repositórios
2. Na barra de navegação, selecione a região na qual criará o seu repositório.
3. No painel de navegação, escolha Repositories (Repositórios).
4. Na página Repositories (Repositórios), escolha o repositório que contém a imagem para a qual recuperar as descobertas da verificação.
5. Na página Imagens, na coluna Tag da imagem, selecione a tag da imagem para recuperar as descobertas da verificação.

AWS CLI

Use o AWS CLI comando a seguir para recuperar os resultados da digitalização de imagens usando o. AWS CLIÉ possível especificar uma imagem usando a `imageTag` ou o `imageDigest`. Ambos podem ser obtidos usando o comando list-images da CLI.

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

  O exemplo a seguir usa uma tag de imagem.

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

  O exemplo a seguir usa um resumo de imagem.

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

AWS Tools for Windows PowerShell

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

  O exemplo a seguir usa uma tag de imagem.

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

  O exemplo a seguir usa um resumo de imagem.

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

Solucionar problemas de verificação de imagens no Amazon ECR

Veja a seguir falhas comuns de verificação de imagem. Você pode visualizar erros como esses no console do Amazon ECR exibindo os detalhes da imagem ou por meio da API ou AWS CLI usando a `DescribeImageScanFindings` API.

UnsupportedImageError

Você pode receber um erro de `UnsupportedImageError` ao tentar realizar uma verificação básica em uma imagem que foi criada usando um sistema operacional para o qual o Amazon ECR não oferece suporte à verificação de imagens. O Amazon ECR suporta verificação de vulnerabilidades de pacotes para as versões principais de distribuições do Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine e RHEL Linux. Quando uma distribuição perde o suporte de seu fornecedor, o Amazon ECR pode não suportar a verificação de vulnerabilidades dessa distribuição. O Amazon ECR não suporta verificação de imagens criadas a partir de imagem de scratch do Docker.

> **Important**
>
> Ao usar a verificação avançada, o Amazon Inspector oferece suporte à verificação para sistemas operacionais e tipos de mídias específicos. Para obter uma lista completa, consulte Sistemas operacionais compatíveis: verificação do Amazon ECR no Guia do usuário do Amazon Inspector.

Um nível de severidade `UNDEFINED` é retornado

Você poderá receber uma descoberta de verificação que tenha um nível de severidade `UNDEFINED`. As causas comuns para isso são as seguintes:

- A origem do CVE não atribuiu uma prioridade à vulnerabilidade.
- A vulnerabilidade recebeu uma prioridade que o Amazon ECR não reconhece.

Para determinar a gravidade e a descrição de uma vulnerabilidade, você pode exibir o CVE diretamente da origem.

Noções básicas do status de verificação **`SCAN_ELIGIBILITY_EXPIRED`**

Quando a verificação aprimorada usando o Amazon Inspector estiver habilitada para seu registro privado e você estiver visualizando suas vulnerabilidades de verificação, poderá ver um status de verificação de `SCAN_ELIGIBILITY_EXPIRED`. As causas comuns para isso são as seguintes:

- Quando você ativa inicialmente a verificação avançada para seu registro privado, o Amazon Inspector reconhece apenas imagens enviadas para o Amazon ECR nos últimos 30 dias com base na data e hora do envio da imagem. Imagens mais antigas terão o status de verificação `SCAN_ELIGIBILITY_EXPIRED`. Se desejar que essas imagens sejam verificadas pelo Amazon Inspector, você deverá enviá-las novamente para o seu repositório.
- Se ECR re-scan duration (Duração da nova verificação do ECR) for alterada no console do Amazon Inspector e, quando esse tempo decorre, o status da verificação da imagem é alterado para `inactive` com um código de motivo `expired`, e todas as descobertas associadas à imagem são programadas para serem fechadas. Com isso, o console do Amazon ECR lista o status da verificação como `SCAN_ELIGIBILITY_EXPIRED`.

Sincronizar um registro upstream com um registro privado do Amazon ECR

Com as regras de cache de pull-through, você pode sincronizar o conteúdo de um registro upstream com o registro privado do Amazon ECR.

No momento, o Amazon ECR oferece suporte à criação de regras de cache de pull-through para os seguintes registros upstream:

- Amazon ECR Public, o registro de imagens de contêineres do Kubernetes e o Quay (não requer autenticação)
- Docker Hub, Microsoft Azure Container Registry, GitHub Container Registry, GitLab Container Registry e Chainguard Registry (requer autenticação com segredo) AWS Secrets Manager
- Amazon ECR (requer autenticação com a função AWS IAM)

Para o GitLab Container Registry, o Amazon ECR oferece suporte ao cache de pull through somente com GitLab a oferta de Software as a Service (SaaS). Para obter mais informações sobre como usar GitLab a oferta de SaaS, consulte GitLab .com.

Para registros upstream que exigem autenticação com segredos (como o Docker Hub), você deve armazenar suas credenciais em um segredo. AWS Secrets Manager Você pode usar o console do Amazon ECR para criar segredos do Secrets Manager para cada registro upstream autenticado. Para obter mais informações sobre como criar um segredo no Secrets Manager usando o console do Secrets Manager, consulte Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager.

Para o Amazon ECR, você deve criar uma função do IAM se os registros upstream e downstream do Amazon ECR pertencerem a contas diferentes. AWS Para obter mais informações sobre como criar uma função do IAM, consulte Políticas do IAM necessárias para cache de pull-through ECR para ECR entre contas.

Após criar uma regra de cache de pull-through para o registro upstream, extraia uma imagem desse registro upstream usando o URI de registro privado do Amazon ECR. Em seguida, o Amazon ECR cria um repositório e armazena essa imagem em cache no seu registro privado. Nas solicitações subsequentes de extração da imagem armazenada em cache com uma determinada tag, o Amazon ECR verifica o registro upstream para ver se há uma nova versão da imagem com essa tag específica e tenta atualizar a imagem no registro privado pelo menos uma vez a cada 24 horas.

Modelos de criação de repositório

O Amazon ECR adicionou suporte para modelos de criação de repositório, o que lhe dá o controle para especificar configurações iniciais para novos repositórios criados pelo Amazon ECR em seu nome usando regras de cache de pull-through. Cada modelo contém um prefixo de namespace do repositório que é usado para associar novos repositórios a um modelo específico. Os modelos podem especificar a configuração para todas as configurações do repositório, incluindo políticas de acesso baseadas em recursos, imutabilidade de tags, criptografia e políticas de ciclo de vida. As configurações em um modelo de criação de repositório são aplicadas apenas durante a criação do repositório e não têm efeito sobre repositórios existentes ou repositórios criados usando qualquer outro método. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.

Considerações sobre o uso do cache de pull-through

Os pontos a seguir devem ser considerados ao usar as regras do cache de pull-through do Amazon ECR.

- A criação de regras de cache de pull-through não é aceita nas seguintes Regiões:
  - China (Pequim) (`cn-north-1`)
  - China (Ningxia) (`cn-northwest-1`)
  - AWS GovCloud (Leste dos EUA) (`us-gov-east-1`)
  - AWS GovCloud (Oeste dos EUA) (`us-gov-west-1`)
- AWS Lambda não suporta a extração de imagens de contêineres do Amazon ECR usando uma regra de cache pull through.
- Ao extrair imagens usando o cache de pull-through, os endpoints de serviço FIPS do Amazon ECR não são suportados na primeira vez que uma imagem é extraída. No entanto, usar os endpoints de serviço FIPS do Amazon ECR funciona em extrações subsequentes.
- Para repositórios upstream que exigem autenticação, quando uma imagem é extraída pela URI de registro privado do Amazon ECR pela primeira vez ou para atualizar o cache, a extração da imagem é iniciada pelo usuário associado às credenciais configuradas na regra de extração de cache. As extrações subsequentes retornarão a imagem diretamente do cache no registro privado do cliente.

- Para repositórios upstream que não exigem autenticação, quando uma imagem é extraída por meio do URI de registro privado do Amazon ECR, a extração da imagem é iniciada por endereços IP. AWS
- Quando um cliente extrai uma imagem em cache por meio do URI de registro privado do Amazon ECR, o Amazon ECR verifica se validou a imagem em relação ao registro upstream nas últimas 24 horas. Se a janela de 24 horas tiver expirado, o Amazon ECR envia uma solicitação upstream para verificar se há uma versão mais recente e atualiza o cache, se houver. Se a janela não tiver expirado, o Amazon ECR veicula a imagem em cache sem entrar em contato com o upstream.
- Chamar a `ListImageReferrers` API para um repositório criado pelo cache pull through retorna os artefatos de referência compatíveis com OCI para o cache privado.
- O Amazon ECR verifica se os artefatos do referenciador foram atualizados nas últimas 6 horas. Se a janela de 6 horas tiver expirado, o Amazon ECR envia uma solicitação upstream para verificar as versões mais recentes e atualizar o cache, se elas existirem.
- Se o Amazon ECR não conseguir atualizar a imagem do registro upstream por qualquer motivo e a imagem for extraída, a última imagem em cache ainda será extraída.
- Ao criar o segredo do Secrets Manager que contém as credenciais do registro upstream, o nome do segredo deve usar o prefixo `ecr-pullthroughcache/`. O segredo também deve estar na mesma conta e região em que a regra de cache de pull-through foi criada.
- Quando uma imagem multiarquitetura é extraída por meio de uma regra de cache de pull-through, a lista de manifestos e cada imagem referenciada na lista de manifesto são extraídas para o repositório do Amazon ECR. Se desejar apenas extrair uma arquitetura específica, você poderá extrair a imagem usando o resumo da imagem ou a tag associada à arquitetura, em vez da tag associada à lista de manifesto.
- O Amazon ECR utiliza um perfil do IAM vinculada ao serviço, que fornece as permissões necessárias para o Amazon ECR criar o repositório, recuperar o valor do segredo do Secrets Manager para autenticação e enviar a imagem armazenada em cache em seu nome. A função do IAM vinculada ao serviço é criada automaticamente quando uma regra de cache de pull-through é criada. Para obter mais informações, consulte Função vinculada ao serviço do Amazon ECR para cache de pull-through.
- Por padrão, a entidade principal do IAM que está puxando a imagem armazenada em cache tem as permissões concedidas a ele por meio de sua política do IAM. Você pode usar a política de permissões de registro privado do Amazon ECR para aumentar o escopo das permissões de uma entidade do IAM. Para obter mais informações, consulte Usar permissões de registro.

- Os repositórios do Amazon ECR criados usando o fluxo de trabalho de cache de pull-through são tratados como qualquer outro repositório do Amazon ECR. Todos os recursos do repositório, como replicação e verificação de imagens, são compatíveis.
- Quando o Amazon ECR cria um novo repositório em seu nome usando uma ação de cache de pull-through, as seguintes configurações padrão são aplicadas ao repositório, a menos que haja um modelo correspondente de criação de repositório. Você pode usar um modelo de criação de repositório para definir as configurações aplicadas aos repositórios criados pelo Amazon ECR em seu nome. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.
  - Imutabilidade da tag – A imutabilidade da tag especifica se as tags de imagem podem ser substituídas. Por padrão, as tags de imagem são mutáveis (podem ser substituídas). Você pode modificar o comportamento da tag configurando os filtros de exclusão da tag na caixa de texto Exclusão da tag mutável, quando a opção Mutável estiver selecionada, ou na caixa de texto Exclusão da tag imutável quando a opção Imutável estiver selecionada.
  - Criptografia – A criptografia padrão do `AES256` é usada.
  - Permissões do repositório – Omitidas, nenhuma política de permissões do repositório é aplicada.
  - Política de ciclo de vida – Omitida, nenhuma política de ciclo de vida é aplicada.
  - Tags de recursos – Omitidas, nenhuma tag de recurso é aplicada.
- Ativar a imutabilidade da tag de imagem para repositórios usando uma regra de cache de pull-through impedirá que o Amazon ECR atualize imagens usando a mesma tag.
- Quando uma imagem é extraída usando a regra de cache de pull-through pela primeira vez, uma rota para a internet pode ser necessária. Há certas circunstâncias em que uma rota para a internet é necessária, então é melhor configurar uma rota para evitar falhas. Portanto, se você configurou o Amazon ECR para usar uma interface VPC endpoint AWS PrivateLink , precisará garantir que o primeiro pull tenha uma rota para a Internet. Uma maneira de fazer isso é criar uma sub-rede pública na mesma VPC, com um gateway da internet. Em seguida, é preciso rotear todo o tráfego de saída da sub-rede privada para a sub-rede pública. As extrações subsequentes de imagem usando a regra de cache de pull-through não exigem isso. Para obter mais informações, consulte Opções de rotas de exemplos no Guia do usuário da Amazon Virtual Private Cloud.

Permissões do IAM necessárias para sincronizar um registro upstream com um registro privado do Amazon ECR

Além das permissões da API do Amazon ECR necessárias para autenticar em um registro privado e enviar e extrair imagens, as seguintes permissões adicionais são necessárias para usar regras de cache de pull-through de forma efetiva.

- `ecr:CreatePullThroughCacheRule`: concede permissão para criar regra de cache de pull-through. Essa permissão deve ser concedida por meio de uma política do IAM baseada em identidade.
- `ecr:BatchImportUpstreamImage`: concede permissão para recuperar a imagem externa e importá-la para o registro privado. Essa permissão pode ser concedida usando a política de permissões do registro privado, uma política do IAM baseada em identidade ou a política de permissões de repositório baseadas em recursos. Para obter mais informações sobre o uso de permissões de repositório, consulte Políticas de repositório privado no Amazon ECR.
- `ecr:CreateRepository`: concede permissão para criar um repositório em um registro privado. Essa permissão é necessária se o repositório de armazenamento de imagens em cache ainda não existir. Essa permissão pode ser concedida por uma política do IAM baseada em identidade ou pela política de permissões do registro privado.

Usar permissões de registro

As permissões de registro privado do Amazon ECR podem ser usadas para dimensionar o escopo das permissões de entidades individuais do IAM para usar o cache de pull-through. Se uma entidade do IAM tiver mais permissões concedidas por uma política do IAM do que a política de permissões do registro está concedendo, a política do IAM terá precedência. Por exemplo, se um usuário já tiver as permissões `ecr:*`, não serão necessárias permissões adicionais no nível do registro.

Para criar uma política de permissões de um registro privado (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região na qual deseja configurar a sua declaração de permissões da política do registro.
3. No painel de navegação, escolha Private registry (Registro privado), Registry permissions (Permissões do registro).

4. Na página Registry permissions (Permissões do registro), escolha Generate statement (Gerar declaração).
5. Para cada declaração de política de permissões de cache de pull-through que você criar, faça o seguinte.

   a. Em Policy type (Tipo de política), escolha Pull through cache policy (Política de cache de pull-through).

   b. Em Statement id (ID da declaração), forneça um nome para a política de declaração do cache de pull-through.

   c. Em IAM entities (Entidades do IAM), especifique os usuários, grupos ou funções a serem incluídos na política.

   d. Em Repository namespace (Namespace do repositório), selecione a regra de cache de pull-through para associar à política.

   e. Em Repository names (Nomes de repositórios), especifique o nome da base do repositório ao qual a regra será aplicada. Por exemplo, se você quisesse especificar o repositório do Amazon Linux no Amazon ECR Public, o nome do repositório seria `amazonlinux`.

Para criar uma política de permissões de um registro privado (AWS CLI)

Use o AWS CLI comando a seguir para especificar as permissões do registro privado usando AWS CLI o.

1. Crie um arquivo local denominado `ptc-registry-policy.json` com o conteúdo da política do registro. O exemplo a seguir concede a permissão `ecr-pull-through-cache-user` para criar um repositório e extrair uma imagem do Amazon ECR Public, que é a fonte upstream associada à regra de cache pull-through criada anteriormente.

   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**
>
> A permissão `ecr-CreateRepository` é necessária se o repositório de armazenamento de imagens em cache ainda não existir. Por exemplo, se a ação de criação do repositório e as ações de extração de imagem forem realizadas por entidades separadas do IAM, como um administrador e um desenvolvedor.

2. Use o put-registry-policycomando para definir a política de registro.

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

Próximas etapas

Quando você estiver pronto para começar a usar as regras de cache de pull-through, as próximas etapas são apresentadas a seguir.

- Crie uma regra de cache de pull-through. Para obter mais informações, consulte Criar uma regra de cache de pull-through no Amazon ECR.
- Crie um modelo de criação de repositório. Um modelo de criação de repositório oferece a você o controle para definir as configurações a serem usadas para novos repositórios criados pelo Amazon ECR em seu nome durante uma ação do cache de pull-through. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.

Como configurar permissões para ECR entre contas e ECR PTC

O recurso de cache pull through do Amazon ECR para o Amazon ECR (ECR para ECR) permite a sincronização automática de imagens entre regiões, AWS contas ou ambas. Com o PTC ECR para ECR, você pode enviar imagens para seu registro primário do Amazon ECR e configurar uma regra de cache de pull-through para armazenar imagens em registros downstream do Amazon ECR.

Políticas do IAM necessárias para cache de pull-through ECR para ECR entre contas

Para armazenar imagens em cache entre registros do Amazon ECR em AWS contas diferentes, crie uma função do IAM na conta downstream e configure as políticas nesta seção para fornecer as seguintes permissões:

- O Amazon ECR precisa de permissões para extrair imagens do registro upstream do Amazon ECR em seu nome. Você pode conceder essas permissões criando um perfil do IAM e especificando-o na regra de cache de pull-through.
- O proprietário do registro upstream também deve conceder ao proprietário do registro de cache as permissões necessárias para inserir as imagens nas políticas de recursos.

Políticas

- Criação de um perfil do IAM para definir as permissões de cache de pull-through
- Como criar uma política de confiança para o perfil do IAM
- Criação de uma política de recursos no registro upstream do Amazon ECR

Criação de um perfil do IAM para definir as permissões de cache de pull-through

O exemplo a seguir mostra uma política de permissões que concede a um perfil do IAM permissão para extrair imagens do registro upstream do Amazon ECR em seu nome. Quando o Amazon ECR assumir a função, ele terá as permissões que você especificar nessa política.

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": "*"
        }
    ]
}
```

Como criar uma política de confiança para o perfil do IAM

O exemplo a seguir mostra uma política de confiança que identifica o cache pull through do Amazon ECR como o principal AWS de serviço que pode assumir a função.

JSON

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

Criação de uma política de recursos no registro upstream do Amazon ECR

O proprietário do registro upstream do Amazon ECR também deve adicionar uma política de registro ou uma política de repositório para conceder ao proprietário do registro downstream as permissões necessárias para realizar as ações a seguir.

> **Note**
>
> A política de recursos a seguir é necessária somente para configurações de cache pull through de ECR para ECR entre contas. Para extrair o cache entre regiões da mesma conta, você só precisa da política de função do IAM e da política de confiança mostradas nas seções anteriores. A permissão principal da conta raiz não é necessária quando os registros upstream e downstream estão na mesma conta. 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/*"
}
```

Criar uma regra de cache de pull-through no Amazon ECR

Para cada registro upstream que contenha imagens que você deseja armazenar em cache no registro privado do Amazon ECR, é necessário criar uma regra de cache de pull-through.

Para registros upstream que exigem autenticação com segredos, você deve armazenar as credenciais em um segredo do Secrets Manager. Você pode usar uma senha existente do ou criar outra. Você pode criar o segredo do Secrets Manager no console do Amazon ECR ou no do Secrets Manager. Para criar um segredo do Secrets Manager usando o console do Secrets Manager em vez

do console do Amazon ECR, consulte Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager.

Pré-requisitos

- Verifique se você tem as permissões adequadas do IAM para criar regras de cache de pull-through. Para mais informações, consulte Permissões do IAM necessárias para sincronizar um registro upstream com um registro privado do Amazon ECR.
- Para registros upstream que exigem autenticação com segredos: se você quiser usar um segredo existente, verifique se o segredo do Secrets Manager atende aos seguintes requisitos:
  - O nome do segredo começa com `ecr-pullthroughcache/`. O Console de gerenciamento da AWS só exibe segredos do Secrets Manager com o prefixo `ecr-pullthroughcache/`.
  - A conta e a região em que o segredo está devem corresponder à conta e à região em que a regra de cache de pull-through está.

Para criar uma regra de cache de pull-through (Console de gerenciamento da AWS)

As etapas a seguir mostram como criar uma regra do cache de pull-through e um segredo do Secrets Manager usando o console do Amazon ECR. Para criar um segredo usando o console do Secrets Manager, consulte Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager.

Para Amazon ECR Public, Kubernetes Container Registry ou Quay

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.
3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).
4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).
5. Na Etapa 1: especificar uma página de origem, em Registro, escolha Amazon ECR Public, Kubernetes ou Quay na lista de registros upstream e, em seguida, escolha Avançar.
6. Na Etapa 2: especificar uma página de destino, para o prefixo do repositório Amazon ECR, especifique o prefixo do namespace do repositório a ser usado ao armazenar em cache imagens

retiradas do registro público de origem e escolha Avançar. Um namespace é preenchido por padrão, mas também é possível especificar um namespace personalizado.

7. Na página Etapa 3: Revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.
8. Repita a etapa anterior para cada cache de pull-through que deseja criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para o Docker Hub

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.
3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).
4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).
5. Na Etapa 1: especificar uma página de origem, em Registro, escolha Docker Hub, Avançar.
6. Na página Etapa 2: configurar autenticação, para credenciais do Upstream, você deve armazenar suas credenciais de autenticação para o Docker Hub em um segredo AWS Secrets Manager . Você pode especificar um segredo existente ou usar o console do Amazon ECR para criar um novo segredo.

   a. Para usar um segredo existente, escolha Usar um AWS segredo existente. Em Nome secreto, use o menu suspenso para selecionar seu segredo existente e, em seguida, escolha Avançar.

      > **Note**
      >
      > Exibe Console de gerenciamento da AWS apenas segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo. O segredo também deve estar na mesma conta e região em que a regra de cache de pull-through foi criada.

   b. Para criar um novo segredo, escolha Criar um AWS segredo, faça o seguinte e escolha Avançar.

i. Em Nome secreto, especifique um nome descritivo para o segredo. Os nomes de segredos devem conter de 1 a 512 caracteres Unicode.

ii. Em E-mail do Docker Hub, especifique o e-mail do Docker Hub.

iii. Para o token de acesso do Docker Hub, especifique seu token de acesso do Docker Hub. Para obter mais informações sobre como criar um token de acesso do Docker Hub, consulte Criar e gerenciar tokens de acesso na documentação do Docker.

7. Na Etapa 3: especificar uma página de destino, para o prefixo do repositório Amazon ECR, especifique o namespace do repositório a ser usado ao armazenar em cache imagens retiradas do registro público de origem e escolha Avançar.

   Um namespace é preenchido por padrão, mas também é possível especificar um namespace personalizado.

8. Na página Etapa 4: revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.

9. Repita a etapa anterior para cada cache de pull-through que deseja criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para GitHub Container Registry

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.

2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.

3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).

4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).

5. Na Etapa 1: Especificar uma página de origem, em Registro, escolha Registro de GitHub contêiner, Avançar.

6. Na página Etapa 2: Configurar autenticação, para credenciais do Upstream, você deve armazenar suas credenciais de autenticação para o GitHub Container Registry em um segredo. AWS Secrets Manager Você pode especificar um segredo existente ou usar o console do Amazon ECR para criar um novo segredo.

a. Para usar um segredo existente, escolha Usar um AWS segredo existente. Em Nome secreto, use o menu suspenso para selecionar seu segredo existente e, em seguida, escolha Avançar.

> **Note**
>
> Exibe Console de gerenciamento da AWS apenas segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo. O segredo também deve estar na mesma conta e região em que a regra de cache de pull-through foi criada.

b. Para criar um novo segredo, escolha Criar um AWS segredo, faça o seguinte e escolha Avançar.

   i. Em Nome secreto, especifique um nome descritivo para o segredo. Os nomes de segredos devem conter de 1 a 512 caracteres Unicode.

   ii. Para nome de usuário do GitHub Container Registry, especifique seu nome de usuário do GitHub Container Registry

   iii. Para o token de acesso do GitHub Container Registry, especifique seu token de acesso do GitHub Container Registry. Para obter mais informações sobre a criação de um token de GitHub acesso, consulte Gerenciando seus tokens de acesso pessoais na GitHub documentação.

7. Na Etapa 3: especificar uma página de destino, para o prefixo do repositório Amazon ECR, especifique o namespace do repositório a ser usado ao armazenar em cache imagens retiradas do registro público de origem e escolha Avançar.

   Um namespace é preenchido por padrão, mas também é possível especificar um namespace personalizado.

8. Na página Etapa 4: revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.

9. Repita a etapa anterior para cada cache de pull-through que deseja criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para o Registro de Contêiner do Microsoft Azure

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.

2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.
3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).
4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).
5. Na Etapa 1: especificar uma página de origem, faça o seguinte.

   a. Em Registro, escolha Microsoft Azure Container Registry

   b. Em URL do registro de origem, especifique o nome do seu registro de contêiner do Microsoft Azure e escolha Avançar.

   > **Important**
   >
   > Você só precisa especificar o prefixo, pois o sufixo `.azurecr.io` é preenchido em seu nome.

6. Na página Etapa 2: configurar autenticação, para credenciais do Upstream, você deve armazenar suas credenciais de autenticação para o Microsoft Azure Container Registry em um segredo AWS Secrets Manager . Você pode especificar um segredo existente ou usar o console do Amazon ECR para criar um novo segredo.

   a. Para usar um segredo existente, escolha Usar um AWS segredo existente. Em Nome secreto, use o menu suspenso para selecionar seu segredo existente e, em seguida, escolha Avançar.

   > **Note**
   >
   > Exibe Console de gerenciamento da AWS apenas segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo. O segredo também deve estar na mesma conta e região em que a regra de cache de pull-through foi criada.

   b. Para criar um novo segredo, escolha Criar um AWS segredo, faça o seguinte e escolha Avançar.

      i. Em Nome secreto, especifique um nome descritivo para o segredo. Os nomes de segredos devem conter de 1 a 512 caracteres Unicode.

ii. Para o nome de usuário do Registro de Contêiner do Microsoft Azure, especifique seu nome de usuário do Registro de Contêiner do Microsoft Azure.

iii. Para o token de acesso do Registro de Contêiner do Microsoft Azure, especifique seu token de acesso do Registro de Contêiner do Microsoft Azure. Para obter mais informações sobre a criação de um token de acesso ao Registro de Contêiner do Microsoft Azure, consulte Criar token - portal na documentação do Microsoft Azure.

7. Na Etapa 3: especificar uma página de destino, para o prefixo do repositório Amazon ECR, especifique o namespace do repositório a ser usado ao armazenar em cache imagens retiradas do registro público de origem e escolha Avançar.

   Um namespace é preenchido por padrão, mas também é possível especificar um namespace personalizado.

8. Na página Etapa 4: revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.

9. Repita a etapa anterior para cada cache de pull-through que deseja criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para GitLab Container Registry

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.

2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.

3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).

4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).

5. Na Etapa 1: Especificar uma página de origem, em Registro, escolha Registro de GitLab contêiner, Avançar.

6. Na página Etapa 2: Configurar autenticação, para credenciais do Upstream, você deve armazenar suas credenciais de autenticação para o GitLab Container Registry em um segredo. AWS Secrets Manager Você pode especificar um segredo existente ou usar o console do Amazon ECR para criar um novo segredo.

   a. Para usar um segredo existente, escolha Usar um AWS segredo existente. Em Nome secreto, use o menu suspenso para selecionar seu segredo existente e, em seguida,

escolha Avançar. Para obter mais informações sobre como criar um segredo no Secrets Manager usando o console do Secrets Manager, consulte Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager.

> **Note**
>
> Exibe Console de gerenciamento da AWS apenas segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo. O segredo também deve estar na mesma conta e região em que a regra de cache de pull-through foi criada.

b. Para criar um novo segredo, escolha Criar um AWS segredo, faça o seguinte e escolha Avançar.

   i. Em Nome secreto, especifique um nome descritivo para o segredo. Os nomes de segredos devem conter de 1 a 512 caracteres Unicode.

   ii. Para nome de usuário do GitLab Container Registry, especifique seu nome de usuário do GitLab Container Registry

   iii. Para o token de acesso do GitLab Container Registry, especifique seu token de acesso do GitLab Container Registry. Para obter mais informações sobre a criação de um token de acesso ao GitLab Container Registry, consulte Tokens de acesso pessoal, Tokens de acesso de grupo ou Tokens de acesso ao projeto, na GitLab documentação.

7. Na Etapa 3: especificar uma página de destino, para o prefixo do repositório Amazon ECR, especifique o namespace do repositório a ser usado ao armazenar em cache imagens retiradas do registro público de origem e escolha Avançar.

   Um namespace é preenchido por padrão, mas também é possível especificar um namespace personalizado.

8. Na página Etapa 4: revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.

9. Repita a etapa anterior para cada cache de pull-through que deseja criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para registro privado do Amazon ECR em sua conta AWS

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.

2. Na barra de navegação, selecione a região na qual você deseja ajustar as configurações do registro privado.
3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).
4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).
5. Na página Etapa 1: especificar upstream, em Registro, selecione Amazon ECR Private e Esta conta. Em Região, selecione a região para o registro upstream do Amazon ECR e, em seguida, escolha Avançar.
6. Na página Etapa 2: especificar namespaces, em Namespace do cache, escolha se deseja criar repositórios de cache de pull-through com Um prefixo específico ou Sem prefixo. Se você selecionar Um prefixo específico, deverá especificar um nome de prefixo a ser usado como parte do namespace para armazenar imagens do registro upstream em cache.
7. Para o Namespace upstream, escolha se deseja extrair de Um prefixo específico que existe no registro upstream. Se você Sem prefixo, poderá extrair de qualquer repositório no registro upstream. Especifique o prefixo do repositório upstream, se solicitado, e escolha Avançar.

   > **Note**
   >
   > Para saber mais sobre como personalizar o cache e os namespaces upstream, consulte Personalização de prefixos de repositório para cache de pull-through de ECR para ECR.

8. Na página Etapa 3: Revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.
9. Repita essas etapas para cada cache de pull-through que desejar criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para o registro privado do Amazon ECR de outra conta AWS

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.
3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).

4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).
5. Na página Etapa 1: especificar upstream, em Registro, escolha Amazon ECR Private e Entre contas. Em Região, selecione a região para o registro upstream do Amazon ECR. Em Conta, especifique o ID da AWS conta para o registro upstream do Amazon ECR e, em seguida, escolha Avançar.
6. Na página Etapa 2: especificar permissões, em Perfil do IAM, selecione uma função a ser usada para obter acesso ao cache de pull-through entre contas e, em seguida, escolha Criar.

   > **Note**
   >
   > Certifique-se de selecionar o perfil do IAM que usa as permissões criadas em Políticas do IAM necessárias para cache de pull-through ECR para ECR entre contas.

7. Na página Etapa 3: especificar namespaces, em Namespace do cache, escolha se deseja criar repositórios de cache de pull-through com Um prefixo específico ou Sem prefixo. Se você selecionar Um prefixo específico, deverá especificar um nome de prefixo a ser usado como parte do namespace para armazenar imagens do registro upstream em cache.
8. Para o Namespace upstream, escolha se deseja extrair de Um prefixo específico que existe no registro upstream. Se você Sem prefixo, poderá extrair de qualquer repositório no registro upstream. Especifique o prefixo do repositório upstream, se solicitado, e escolha Avançar.

   > **Note**
   >
   > Para saber mais sobre como personalizar o cache e os namespaces upstream, consulte Personalização de prefixos de repositório para cache de pull-through de ECR para ECR.

9. Na página Etapa 4: revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.
10. Repita essas etapas para cada cache de pull-through que desejar criar. As regras de cache de pull-through são criadas separadamente para cada região.

Para Chainguard Registry

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.

2. Na barra de navegação, selecione a região para a qual deseja ajustar as configurações do registro privado.
3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).
4. Na página Pull through cache configuration (Configuração do cache de pull-through), escolha Add rule (Adicionar regra).
5. Na Etapa 1: Especificar uma página de origem, em Registro, escolha Chainguard Registry, Avançar.
6. Na página Etapa 2: Configurar autenticação, para credenciais Upstream, você deve armazenar suas credenciais de autenticação para o Chainguard Registry em um segredo. AWS Secrets Manager Você pode especificar um segredo existente ou usar o console do Amazon ECR para criar um novo segredo.

    a. Para usar um segredo existente, escolha Usar um AWS segredo existente. Em Nome secreto, use o menu suspenso para selecionar seu segredo existente e, em seguida, escolha Avançar. Para obter mais informações sobre como criar um segredo no Secrets Manager usando o console do Secrets Manager, consulte Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager.

    > **Note**
    >
    > Exibe Console de gerenciamento da AWS apenas segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo. O segredo também deve estar na mesma conta e região em que a regra de cache de pull-through foi criada.

    b. Para criar um novo segredo, escolha Criar um AWS segredo, faça o seguinte e escolha Avançar.

        i. Em Nome secreto, especifique um nome descritivo para o segredo. Os nomes de segredos devem conter de 1 a 512 caracteres Unicode.

        ii. Para nome de usuário do Chainguard Registry, especifique seu nome de usuário do Chainguard Registry.

        iii. Para o token pull do Chainguard Registry, especifique seu token pull do Chainguard Registry. Para obter mais informações sobre como criar um token pull do Chainguard Registry, consulte Autenticação com um token pull na documentação do Chainguard.

7. Na Etapa 3: Especificar uma página de destino, para o prefixo do repositório Amazon ECR, especifique o namespace do repositório a ser usado ao armazenar em cache imagens retiradas do registro de origem e escolha Avançar.

   Um namespace é preenchido por padrão, mas também é possível especificar um namespace personalizado.
8. Na página Etapa 4: revisar e criar, revise a configuração da regra de cache de pull-through e escolha Criar.
9. Repita a etapa anterior para cada cache de pull-through que deseja criar. As regras de cache pull-through são criadas separadamente para cada região.

Para criar uma regra de cache de pull-through (AWS CLI)

Use o AWS CLI comando create-pull-through-cache-rule para criar uma regra de cache pull through para um registro privado do Amazon ECR. Para registros de upstream que exigem autenticação com segredos, você deve armazenar as credenciais em um segredo do Secrets Manager. Para criar um segredo usando o console do Secrets Manager, consulte Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager.

Os exemplos a seguir são fornecidos para cada registro upstream compatível.

Para o Amazon ECR Public

O exemplo a seguir cria uma regra de cache de pull-through para o registro público do Amazon ECR. Ele especifica um prefixo de repositório de `ecr-public`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `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
```

Para registro de contêineres Kubernetes

O exemplo a seguir cria uma regra de cache de pull-through para o registro público do Kubernetes. Ele especifica um prefixo de repositório de `kubernetes`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `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
```

Para Quay

O exemplo a seguir cria uma regra de cache de pull-through para o registro público do Quay. Ele especifica um prefixo de repositório de `quay`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `quay/`*`upstream-repository-name`*.

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

Para o Docker Hub

O exemplo a seguir cria uma regra de cache de pull-through para o registro do Docker Hub. Ele especifica um prefixo de repositório de `docker-hub`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `docker-hub/`*`upstream-repository-name`*. É necessário especificar o nome completo do Amazon Resource Name (ARN) do segredo que contém suas credenciais do Docker Hub.

```
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
```

Para GitHub Container Registry

O exemplo a seguir cria uma regra de cache pull through para o GitHub Container Registry. Ele especifica um prefixo de repositório de `github`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `github/`*`upstream-repository-name`*. Você deve especificar o Amazon Resource Name (ARN) completo do segredo que contém suas credenciais do GitHub Container Registry.

```
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
```

Para o Registro de Contêiner do Microsoft Azure

O exemplo a seguir cria uma regra de cache de pull-through para o Microsoft Azure Container Registry. Ele especifica um prefixo de repositório de `azure`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `azure/`*`upstream-repository-name`*. É necessário especificar o nome completo do Amazon Resource Name (ARN) do segredo que contém suas credenciais do Docker Hub.

```
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
```

Para GitLab Container Registry

O exemplo a seguir cria uma regra de cache pull through para o GitLab Container Registry. Ele especifica um prefixo de repositório de `gitlab`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `gitlab/`*`upstream-repository-name`*. Você deve especificar o Amazon Resource Name (ARN) completo do segredo que contém suas credenciais do GitLab Container Registry.

```
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
```

Para registro privado do Amazon ECR em sua conta AWS

O exemplo a seguir cria uma regra de cache pull through para o registro privado do Amazon ECR para várias regiões dentro da mesma AWS conta. Ele especifica um prefixo de repositório de `ecr`, o

que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `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
```

Para o registro privado do Amazon ECR de outra conta AWS

O exemplo a seguir cria uma regra de cache pull through para o registro privado do Amazon ECR para várias regiões dentro da mesma AWS conta. Ele especifica um prefixo de repositório de `ecr`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `ecr/`*`upstream-repository-name`*. É necessário especificar o nome do recurso da Amazon (ARN) completo do perfil do IAM com as permissões criadas em Criar uma regra de cache de pull-through no 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
```

Para Chainguard Registry

O exemplo a seguir cria uma regra de cache pull through para o Chainguard Registry. Ele especifica um prefixo de repositório de `chainguard`, o que faz com que cada repositório criado usando a regra de cache de pull-through receba o esquema de nomes de `chainguard/`*`upstream-repository-name`*. Você deve especificar o Amazon Resource Name (ARN) completo do segredo que contém suas credenciais do Chainguard Registry.

```
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
```

Próximas etapas

Depois de criar as regras de cache de pull-through, as próximas etapas são as seguintes:

- Crie um modelo de criação de repositório. Um modelo de criação de repositório oferece a você o controle para definir as configurações a serem usadas para novos repositórios criados pelo Amazon ECR em seu nome durante uma ação do cache de pull-through. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.
- Valide suas regras de cache de pull-through. Ao validar uma regra de cache de pull-through, o Amazon ECR faz uma conexão de rede com o registro upstream, verifica se ele pode acessar o segredo do Secrets Manager contendo as credenciais do registro upstream e se a autenticação foi bem-sucedida. Para obter mais informações, consulte Validar regras de cache de pull-through no Amazon ECR.
- Comece a usar suas regras de cache de pull-through. Para obter mais informações, consulte Extrair uma imagem com uma regra de cache de pull-through no Amazon ECR.

Validar regras de cache de pull-through no Amazon ECR

Após uma regra de cache de pull-through ser criada, para registros upstream que exigem autenticação, você pode validar o funcionamento correto da regra. Ao validar uma regra de cache de pull-through, o Amazon ECR faz uma conexão de rede com o registro upstream, verifica se ele pode acessar o segredo do Secrets Manager que contém as credenciais do registro upstream e verifica se a autenticação teve êxito.

Antes de começar a trabalhar com as regras de cache de pull-through, verifique se você tem as permissões adequadas do IAM. Para obter mais informações, consulte Permissões do IAM necessárias para sincronizar um registro upstream com um registro privado do Amazon ECR.

Para validar uma regra de cache de pull-through (Console de gerenciamento da AWS)

Os seguintes passos mostram como validar uma regra de cache de pull-through usando o console do Amazon ECR.

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a Região que contém a regra do cache de pull-through a ser validada.

3. No painel de navegação, escolha Private registry (Registro privado), Pull through cache (Cache de pull-through).
4. Na página de Configuração do cache de pull-through, selecione a regra de pull through cache para validar. Em seguida, use o menu suspenso Ações e escolha Exibir detalhes.
5. Na página de detalhes da regra de cache de pull-through, use o menu suspenso Ações e escolha Verificar autenticação. O Amazon ECR exibirá um banner com o resultado.
6. Repita essas etapas para cada regra de cache de pull-through que deseja validar.

Para validar uma regra de cache de pull-through (AWS CLI)

O AWS CLI comando validate-pull-through-cache-rule é usado para validar uma regra de cache pull through para um registro privado do Amazon ECR. O exemplo a seguir usa o prefixo do `ecr-public` namespace. Substitua esse valor pelo valor do prefixo para validação da regra de cache de pull-through.

```
aws ecr validate-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --region us-east-2
```

Na resposta, o parâmetro `isValid` indica se a validação foi bem-sucedida ou não. Se`true`, o Amazon ECR conseguisse acessar o registro upstream e a autenticação fosse bem-sucedida. Se `false` houve um problema e a validação falhou. O parâmetro `failure` indica a causa.

Extrair uma imagem com uma regra de cache de pull-through no Amazon ECR

Os exemplos a seguir mostram a sintaxe do comando a ser usada ao extrair uma imagem usando uma regra de cache de pull- through. Se você receber um erro ao extrair uma imagem upstream usando uma regra de cache de pull-through, consulte Solução de problemas de cache de pull-through no Amazon ECR para ver os erros mais comuns e como resolvê-los.

Antes de começar a trabalhar com as regras de cache de pull-through, verifique se você tem as permissões adequadas do IAM. Para obter mais informações, consulte Permissões do IAM necessárias para sincronizar um registro upstream com um registro privado do Amazon ECR.

> **Note**
>
> Os exemplos a seguir usam os valores de namespace padrão do repositório Amazon ECR que eles usam. Console de gerenciamento da AWS Certifique-se de usar o URI do repositório privado do Amazon ECR que você configurou.

Para o Amazon ECR Public

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/
image_name:tag
```

Registro de contêineres Kubernetes

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/kubernetes/repository_name/
image_name:tag
```

Quay

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/quay/repository_name/
image_name:tag
```

Docker Hub

Para imagens oficiais do Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

> **Note**
>
> Para imagens oficiais do Docker Hub, o orefixo `/library` deve ser incluído. Para todos os outros repositórios do Docker Hub, você deve omitir o prefixo `/library`.

Para todas as outras imagens do Docker Hub:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

GitHub Registro de contêiner

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/
image_name:tag
```

Microsoft Azure Container Registry

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/
image_name:tag
```

GitLab Registro de contêiner

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

Registro Chainguard

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/chainguard/repository_name/
image_name:tag
```

Armazenando suas credenciais do repositório upstream em segredo AWS Secrets Manager

Ao criar uma regra de cache de pull-through para um repositório upstream que requer autenticação, você deve armazenar as credenciais em um segredo do Secrets Manager. Pode haver um custo para usar um segredo do Secrets Manager. Para obter mais informações, consulte Preços do AWS Secrets Manager.

Os procedimentos a seguir explicam como criar um segredo do Secrets Manager para cada repositório upstream compatível. Opcionalmente, você pode usar o fluxo de trabalho para criar regras de cache de pull-through no console do Amazon ECR para criar o segredo, em vez de criar o segredo usando o console do Secrets Manager. Para obter mais informações, consulte Criar uma regra de cache de pull-through no Amazon ECR.

Docker Hub

Para criar um segredo do Secrets Manager para suas credenciais do Docker Hub (Console de gerenciamento da AWS)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Escolher o tipo de segredo, faça o seguinte:
   a. Em Tipo de segredo, escolha Outro tipo de segredo.
   b. Em pares de chave/valor, crie duas linhas para suas credenciais do Docker Hub. É possível armazenar até 65536 bytes no segredo.
      i. Para o primeiro key/value par, especifique `username` como chave e seu nome de usuário do Docker Hub como valor.
      ii. Para o segundo key/value par, especifique `accessToken` como chave e seu token de acesso do Docker Hub como valor. Para obter mais informações sobre como criar um token de acesso do Docker Hub, consulte Criar e gerenciar tokens de acesso na documentação do Docker.
   c. Em Chave de criptografia, mantenha o AWS KMS key valor padrão aws/secretsmanager e escolha Avançar. Não há custo para o uso dessa chave. Para obter mais informações, consulte Criptografia e decodificação secretas no Secrets Manager no Guia do usuário do AWS Secrets Manager .

   > **Important**
   >
   > Você deve usar a chave de `aws/secretsmanager` criptografia padrão para criptografar seu segredo. O Amazon ECR não oferece suporte ao uso de uma chave gerenciada pelo cliente (CMK) para isso.

4. Na página Configurar segredo, faça o seguinte:
   a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição). Os nomes de segredos devem conter de 1 a 512 caracteres Unicode e ter prefixo com `ecr-pullthroughcache/`.

> **Important**
>
> O Amazon ECR exibe Console de gerenciamento da AWS somente segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo.

b. (Opcional) Na seção Tags, adicione tags ao segredo. Para estratégias de marcação, consulte os segredos do Tag Secrets Manager no Guia do usuário do AWS Secrets Manager . Não armazene informações sigilosas em tags porque elas não são criptografadas.

c. (Opcional) Em Resource permissions (Permissões do recurso), para adicionar uma política de recursos ao segredo, escolha Edit permissions (Editar permissões). Para obter mais informações, consulte Anexo de uma política de permissões a um segredo do Secrets Manager no Guia do usuário do AWS Secrets Manager .

d. (Opcional) Em Replicar segredo, para replicar seu segredo para outro Região da AWS, escolha Replicar segredo. É possível replicar seu segredo agora ou voltar e replicá-lo mais tarde. Para obter mais informações, consulte Replicar um segredo para outras Regiões no Guia do usuário do AWS Secrets Manager .

e. Escolha Próximo.

5. (Opcional) Na página Configure rotation (Configurar alternância), habilite alternância automática para os segredos. Você também pode manter a alternância desabilitada por enquanto e habilitá-la mais tarde. Para obter mais informações, consulte o Guia do usuário do Secrets Manager no Guia do usuário do AWS Secrets Manager . Escolha Próximo.

6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

   O Secrets Manager retorna para a lista de segredos. Se o segredo não aparecer, escolha Refresh (Atualizar).

GitHub Container Registry

Para criar um segredo do Secrets Manager para suas credenciais do GitHub Container Registry ()Console de gerenciamento da AWS

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Escolher o tipo de segredo, faça o seguinte:

a. Em Tipo de segredo, escolha Outro tipo de segredo.

b. Em pares de chave/valor, crie duas linhas para suas GitHub credenciais. É possível armazenar até 65536 bytes no segredo.

   i. Para o primeiro key/value par, especifique `username` como chave e seu GitHub nome de usuário como valor.

   ii. Para o segundo key/value par, especifique `accessToken` como chave e seu token de GitHub acesso como valor. Para obter mais informações sobre a criação de um token de GitHub acesso, consulte Gerenciando seus tokens de acesso pessoais na GitHub documentação.

c. Em Chave de criptografia, mantenha o AWS KMS key valor padrão aws/secretsmanager e escolha Avançar. Não há custo para o uso dessa chave. Para obter mais informações, consulte Criptografia e decodificação secretas no Secrets Manager no Guia do usuário do AWS Secrets Manager .

> **Important**
>
> Você deve usar a chave de `aws/secretsmanager` criptografia padrão para criptografar seu segredo. O Amazon ECR não oferece suporte ao uso de uma chave gerenciada pelo cliente (CMK) para isso.

4. Na página Configure secret (Configurar segredo), faça o seguinte:

a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição). Os nomes de segredos devem conter de 1 a 512 caracteres Unicode e ter prefixo com `ecr-pullthroughcache/`.

> **Important**
>
> O Amazon ECR exibe Console de gerenciamento da AWS somente segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo.

b. (Opcional) Na seção Tags, adicione tags ao segredo. Para estratégias de marcação, consulte os segredos do Tag Secrets Manager no Guia do usuário do AWS Secrets Manager . Não armazene informações sigilosas em tags porque elas não são criptografadas.

c. (Opcional) Em Resource permissions (Permissões do recurso), para adicionar uma política de recursos ao segredo, escolha Edit permissions (Editar permissões). Para obter mais informações, consulte Anexo de uma política de permissões a um segredo do Secrets Manager no Guia do usuário do AWS Secrets Manager .

d. (Opcional) Em Replicar segredo, para replicar seu segredo para outro Região da AWS, escolha Replicar segredo. É possível replicar seu segredo agora ou voltar e replicá-lo mais tarde. Para obter mais informações, consulte Replicar um segredo para outras Regiões no Guia do usuário do AWS Secrets Manager .

e. Escolha Próximo.

5. (Opcional) Na página Configure rotation (Configurar alternância), habilite alternância automática para os segredos. Você também pode manter a alternância desabilitada por enquanto e habilitá-la mais tarde. Para obter mais informações, consulte o Guia do usuário do Secrets Manager no Guia do usuário do AWS Secrets Manager . Escolha Próximo.

6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

   O Secrets Manager retorna para a lista de segredos. Se o segredo não aparecer, escolha Refresh (Atualizar).

Microsoft Azure Container Registry

Para criar um segredo do Secrets Manager para suas credenciais do Microsoft Azure Container Registry (Console de gerenciamento da AWS)

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Escolher o tipo de segredo, faça o seguinte:

   a. Em Tipo de segredo, escolha Outro tipo de segredo.

   b. Em pares de chave/valor, crie duas linhas para suas credenciais do GitHub. É possível armazenar até 65536 bytes no segredo.

      i. Para o primeiro key/value par, especifique `username` como chave e seu nome de usuário do Registro de Contêiner do Microsoft Azure como valor.

      ii. Para o segundo key/value par, especifique `accessToken` como chave e seu token de acesso ao Registro de Contêiner do Microsoft Azure como valor. Para obter mais

informações sobre a criação de um token de acesso do Microsoft Azure, consulte Criar token - portal na documentação do Microsoft Azure.

c. Em Chave de criptografia, mantenha o AWS KMS key valor padrão aws/secretsmanager e escolha Avançar. Não há custo para o uso dessa chave. Para obter mais informações, consulte Criptografia e decodificação secretas no Secrets Manager no Guia do usuário do AWS Secrets Manager .

> **Important**
>
> Você deve usar a chave de `aws/secretsmanager` criptografia padrão para criptografar seu segredo. O Amazon ECR não oferece suporte ao uso de uma chave gerenciada pelo cliente (CMK) para isso.

4. Na página Configure secret (Configurar segredo), faça o seguinte:

a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição). Os nomes de segredos devem conter de 1 a 512 caracteres Unicode e ter prefixo com `ecr-pullthroughcache/`.

> **Important**
>
> O Amazon ECR exibe Console de gerenciamento da AWS somente segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo.

b. (Opcional) Na seção Tags, adicione tags ao segredo. Para estratégias de marcação, consulte os segredos do Tag Secrets Manager no Guia do usuário do AWS Secrets Manager . Não armazene informações sigilosas em tags porque elas não são criptografadas.

c. (Opcional) Em Resource permissions (Permissões do recurso), para adicionar uma política de recursos ao segredo, escolha Edit permissions (Editar permissões). Para obter mais informações, consulte Anexo de uma política de permissões a um segredo do Secrets Manager no Guia do usuário do AWS Secrets Manager .

d. (Opcional) Em Replicar segredo, para replicar seu segredo para outro Região da AWS, escolha Replicar segredo. É possível replicar seu segredo agora ou voltar e replicá-lo mais tarde. Para obter mais informações, consulte Replicar um segredo para outras Regiões no Guia do usuário do AWS Secrets Manager .

e. Escolha Próximo.

5. (Opcional) Na página Configure rotation (Configurar alternância), habilite alternância automática para os segredos. Você também pode manter a alternância desabilitada por enquanto e habilitá-la mais tarde. Para obter mais informações, consulte o Guia do usuário do Secrets Manager no Guia do usuário do AWS Secrets Manager . Escolha Próximo.
6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

   O Secrets Manager retorna para a lista de segredos. Se o segredo não aparecer, escolha Refresh (Atualizar).

GitLab Container Registry

Para criar um segredo do Secrets Manager para suas credenciais do GitLab Container Registry ()Console de gerenciamento da AWS

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Escolher o tipo de segredo, faça o seguinte:
   a. Em Tipo de segredo, escolha Outro tipo de segredo.
   b. Em pares de chave/valor, crie duas linhas para suas GitLab credenciais. É possível armazenar até 65536 bytes no segredo.
      i. Para o primeiro key/value par, especifique `username` como chave e seu nome de usuário do GitLab Container Registry como valor.
      ii. Para o segundo key/value par, especifique `accessToken` como chave e seu token de acesso do GitLab Container Registry como valor. Para obter mais informações sobre a criação de um token de acesso ao GitLab Container Registry, consulte Tokens de acesso pessoal, Tokens de acesso de grupo ou Tokens de acesso ao projeto, na GitLab documentação.
   c. Em Chave de criptografia, mantenha o AWS KMS key valor padrão aws/secretsmanager e escolha Avançar. Não há custo para o uso dessa chave. Para obter mais informações, consulte Criptografia e decodificação secretas no Secrets Manager no Guia do usuário do AWS Secrets Manager .

> **Important**
>
> Você deve usar a chave de `aws/secretsmanager` criptografia padrão para criptografar seu segredo. O Amazon ECR não oferece suporte ao uso de uma chave gerenciada pelo cliente (CMK) para isso.

4. Na página Configure secret (Configurar segredo), faça o seguinte:

   a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição). Os nomes de segredos devem conter de 1 a 512 caracteres Unicode e ter prefixo com `ecr-pullthroughcache/`.

   > **Important**
   >
   > O Amazon ECR exibe Console de gerenciamento da AWS somente segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo.

   b. (Opcional) Na seção Tags, adicione tags ao segredo. Para estratégias de marcação, consulte os segredos do Tag Secrets Manager no Guia do usuário do AWS Secrets Manager . Não armazene informações sigilosas em tags porque elas não são criptografadas.

   c. (Opcional) Em Resource permissions (Permissões do recurso), para adicionar uma política de recursos ao segredo, escolha Edit permissions (Editar permissões). Para obter mais informações, consulte Anexo de uma política de permissões a um segredo do Secrets Manager no Guia do usuário do AWS Secrets Manager .

   d. (Opcional) Em Replicar segredo, para replicar seu segredo para outro Região da AWS, escolha Replicar segredo. É possível replicar seu segredo agora ou voltar e replicá-lo mais tarde. Para obter mais informações, consulte Replicar um segredo para outras Regiões no Guia do usuário do AWS Secrets Manager .

   e. Escolha Próximo.

5. (Opcional) Na página Configure rotation (Configurar alternância), habilite alternância automática para os segredos. Você também pode manter a alternância desabilitada por enquanto e habilitá-la mais tarde. Para obter mais informações, consulte o Guia do usuário do Secrets Manager no Guia do usuário do AWS Secrets Manager . Escolha Próximo.

6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

O Secrets Manager retorna para a lista de segredos. Se o segredo não aparecer, escolha Refresh (Atualizar).

Chainguard Registry

Para criar um segredo do Secrets Manager para suas credenciais do Chainguard ()Console de gerenciamento da AWS

1. Abra o console do Secrets Manager em https://console.aws.amazon.com/secretsmanager/.
2. Selecione Armazenar um novo segredo.
3. Na página Escolher o tipo de segredo, faça o seguinte:
   a. Em Tipo de segredo, escolha Outro tipo de segredo.
   b. Em pares de chave/valor, crie duas linhas para suas credenciais do Chainguard. É possível armazenar até 65536 bytes no segredo.
      i. Para o primeiro key/value par, especifique `username` como chave e seu nome de usuário do Chainguard Registry como valor.
      ii. Para o segundo key/value par, especifique `accessToken` como chave e seu token de acesso ao Chainguard Registry como valor. Para obter mais informações sobre como criar um token pull do Chainguard Registry, consulte Autenticação com um token pull na documentação do Chainguard.
   c. Em Chave de criptografia, mantenha o AWS KMS key valor padrão aws/secretsmanager e escolha Avançar. Não há custo para o uso dessa chave. Para obter mais informações, consulte Criptografia e decodificação secretas no Secrets Manager no Guia do usuário do AWS Secrets Manager .

   > **Important**
   >
   > Você deve usar a chave de `aws/secretsmanager` criptografia padrão para criptografar seu segredo. O Amazon ECR não oferece suporte ao uso de uma chave gerenciada pelo cliente (CMK) para isso.

4. Na página Configure secret (Configurar segredo), faça o seguinte:

a. Insira um Secret name (Nome de segredo) descritivo e uma Description (Descrição). Os nomes de segredos devem conter de 1 a 512 caracteres Unicode e ter prefixo com `ecr-pullthroughcache/`.

> **Important**
>
> O Amazon ECR exibe Console de gerenciamento da AWS somente segredos do Secrets Manager com nomes usando o `ecr-pullthroughcache/` prefixo.

b. (Opcional) Na seção Tags, adicione tags ao segredo. Para estratégias de marcação, consulte os segredos do Tag Secrets Manager no Guia do usuário do AWS Secrets Manager . Não armazene informações sigilosas em tags porque elas não são criptografadas.

c. (Opcional) Em Resource permissions (Permissões do recurso), para adicionar uma política de recursos ao segredo, escolha Edit permissions (Editar permissões). Para obter mais informações, consulte Anexo de uma política de permissões a um segredo do Secrets Manager no Guia do usuário do AWS Secrets Manager .

d. (Opcional) Em Replicar segredo, para replicar seu segredo para outro Região da AWS, escolha Replicar segredo. É possível replicar seu segredo agora ou voltar e replicá-lo mais tarde. Para obter mais informações, consulte Replicar um segredo para outras Regiões no Guia do usuário do AWS Secrets Manager .

e. Escolha Próximo.

5. (Opcional) Na página Configure rotation (Configurar alternância), habilite alternância automática para os segredos. Você também pode manter a alternância desabilitada por enquanto e habilitá-la mais tarde. Para obter mais informações, consulte o Guia do usuário do Secrets Manager no Guia do usuário do AWS Secrets Manager . Escolha Próximo.

6. Na página Review (Revisar), revise os detalhes do segredo e escolha Store (Armazenar).

   O Secrets Manager retorna para a lista de segredos. Se o segredo não aparecer, escolha Refresh (Atualizar).

Personalização de prefixos de repositório para cache de pull-through de ECR para ECR

As regras de cache pull-through oferecem suporte tanto ao prefixo do repositório ecr quanto ao prefixo do repositório upstream. O prefixo do repositório ecr é o prefixo do namespace do repositório no registro de cache do Amazon ECR que está associado à regra. Todos os repositórios que usam esse prefixo se tornam repositórios habilitados por cache para pull-through para o registro upstream definido na regra. Por exemplo, um prefixo de `prod` se aplica a todos os repositórios que começam com `prod/`. Para aplicar um modelo a todos os repositórios em seu registro que não têm uma regra para cache de pull-through associada, use `ROOT` como prefixo.

> **Important**
>
> Sempre há uma suposição `/` aplicada ao fim do prefixo. Se você especificar `ecr-public` como prefixo, o Amazon ECR tratará isso como `ecr-public/`.

O prefixo do repositório upstream corresponde ao nome do repositório upstream. Por padrão, ele é definido como `ROOT`, o que permite a correspondência com qualquer repositório upstream. Você pode definir o prefixo do repositório upstream somente quando o prefixo do repositório Amazon ECR não tiver o valor `ROOT`.

A tabela a seguir mostra o mapeamento entre nomes de repositórios de cache e nomes de repositórios upstream com base em suas configurações de prefixo nas regras de cache de pull-through.

Namespace do cache	Namespace upstream	Relacionamento de mapeamento (repositório de cache → repositório upstream)

ecr-public	ROOT (padrão)	`ecr-public/my-app/image1` → `my-app/image1`
		
		`ecr-public/my-app/image2` → `my-app/image2`

Namespace do cache	Namespace upstream	Relacionamento de mapeamento (repositório de cache → repositório upstream)
ROOT	ROOT	`my-app/image1` → `my-app/image1`
team-a	team-a	`team-a/myapp/image1` → `team-a/myapp/image1`
my-app	aplicação upstream	`my-app/image1` → `upstream-app/image1`

Solução de problemas de cache de pull-through no Amazon ECR

Ao extrair uma imagem upstream usando uma regra de cache pull-through, os erros a seguir são os mais comuns que você pode receber.

Repository does not exist

Um erro que indica que o repositório não existe é mais frequentemente causado pela inexistência do repositório no seu registro privado do Amazon ECR ou porque a permissão `ecr:CreateRepository` não foi concedida à entidade principal do IAM que está extraindo a imagem upstream. Para resolver esse erro, você deve verificar se o URI do repositório no comando pull está correto, as permissões do IAM necessárias foram concedidas à entidade do IAM que está extraindo a imagem upstream ou se o repositório para a imagem upstream a ser enviada foi criado no registro privado do Amazon ECR antes da extração da imagem upstream. Para obter mais informações sobre as permissões necessárias do IAM, consulte Permissões do IAM necessárias para sincronizar um registro upstream com um registro privado do Amazon ECR

A seguir, temos um exemplo desse erro.

```
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'
```

Requested image not found

Um erro que indica que o repositório não pode ser encontrado é mais frequentemente causado pela inexistência do repositório upstream ou porque a permissão `ecr:BatchImportUpstreamImage` não foi concedida à entidade principal do IAM que está extraindo a imagem upstream, mas o repositório já está sendo criado no seu registro privado do Amazon ECR. Para resolver esse erro, você deve verificar se o nome da imagem upstream e da etiqueta da imagem está correto, se ela existe e se as permissões do IAM necessárias foram concedidas à entidade principal do IAM que está extraindo a imagem upstream. Para obter mais informações sobre as permissões necessárias do IAM, consulte Permissões do IAM necessárias para sincronizar um registro upstream com um registro privado do Amazon ECR.

A seguir, temos um exemplo desse erro.

```
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 Forbidden when pulling from a Docker Hub repository

Ao puxar de um repositório Docker Hub marcado como uma imagem oficial do Docker, você deve incluir o `/library/` no URI que você usa. Por exemplo, *.aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/docker-hub/`**`library/`***image_name:tag* Se você omitir a `/library/` das imagens oficiais do Docker Hub, um erro `403 Forbidden` será retornado quando você tentar extrair a imagem usando uma regra de cache de pull-through. Para obter mais informações, consulte Extrair uma imagem com uma regra de cache de pull-through no Amazon ECR.

A seguir, temos um exemplo desse erro.

```
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
```

Replicação de imagem privada no Amazon ECR

É possível configurar o registro privado do Amazon ECR para oferecer suporte à replicação dos seus repositórios. O Amazon ECR oferece suporte à replicação entre regiões e entre contas. Para que ocorra replicação entre contas, a conta de destino deverá configurar uma política de permissões de registro para permitir que a replicação do registro de origem ocorra. Para obter mais informações, consulte Permissões de registro privado no Amazon ECR.

Tópicos

Requisitos da política de replicação entre contas

Para que a replicação do ECR entre contas funcione adequadamente, você precisa entender qual conta precisa de quais políticas configuradas. Esta seção esclarece os requisitos de política para contas de origem e de destino.

Visão geral da configuração da política

A replicação do ECR entre contas requer a configuração da política somente na conta de destino. A conta de origem não exige nenhuma política especial de repositório ou registro.

- Conta de origem: defina as regras de replicação nas configurações do registro. Não são necessárias políticas adicionais nos repositórios de origem.
- Conta de destino: configure uma política de permissões de registro para permitir que a conta de origem replique imagens.

Requisitos da política de registro de destino

A conta de destino deve configurar uma política de permissões de registo que conceda à conta de origem a permissão para realizar as seguintes ações:

- `ecr:ReplicateImage` - Permite que a conta de origem replique imagens para o registro de destino
- `ecr:CreateRepository` - Permite que o ECR crie automaticamente repositórios no registro de destino, caso eles ainda não existam

> **Important**
>
> Se não conceder a permissão `ecr:CreateRepository`, você precisa criar manualmente repositórios com os mesmos nomes na conta de destino para que a replicação seja bem-sucedida.

Exemplo de política de registro de destino:

JSON

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

Requisitos da conta de origem

A conta de origem só precisa:

- Definir as regras de replicação nas configurações do registro para especificar a conta e as regiões de destino
- Certificar que a entidade principal do IAM que configura a replicação tenha as permissões do ECR necessárias

Não são necessárias políticas adicionais nos repositórios de origem. Os repositórios de origem não precisam de políticas de repositório que concedam permissões de replicação.

Equívocos comuns

A seguir estão os equívocos comuns sobre as políticas de replicação entre contas do ECR:

- Equívoco: o repositório de origem precisa de uma política que permita que a conta de destino replique imagens.

  Realidade: os repositórios de origem não precisam de nenhuma política especial para replicação.
- Equívoco: tanto as contas de origem quanto as de destino precisam de políticas de registro.

  Realidade: somente a conta de destino precisa de uma política de permissões de registro.
- Equívoco: políticas de repositório e políticas de registro são a mesma coisa.

  Realidade: as políticas de repositório controlam o acesso a repositórios individuais, enquanto as políticas de registro controlam as operações de registro, como a replicação.

Solução de problemas de falhas de replicação

Se a replicação entre contas estiver falhando, verifique o seguinte:

- Verifique se a conta de destino tem uma política de permissões de registro configurada
- Certifique-se de que a política de registro inclua ambas as ações `ecr:ReplicateImage` e `ecr:CreateRepository`
- Confirme se a ID da conta de origem está especificada corretamente na política de registro de destino
- Verifique se os repositórios de destino existem (se `ecr:CreateRepository` não foi concedido)
- Revise CloudTrail os registros de falhas `CreateRepository` ou chamadas de `ReplicateImage` API

Considerações sobre a replicação de imagem privada

As seguintes informações devem ser consideradas ao usar replicação de imagem privada.

- Somente o conteúdo do repositório enviado ou restaurado em um repositório após a configuração da replicação é replicado. Nenhum conteúdo preexistente em um repositório é replicado. Se uma imagem for restaurada após a ativação da replicação, ela será replicada. Se for restaurado antes da ativação da replicação, não será replicado.
- O nome do repositório permanecerá o mesmo em diferentes regiões e contas quando a replicação ocorrer. O Amazon ECR não suporta a alteração do nome do repositório durante a replicação.
- Na primeira vez que você configura seu registro privado para replicação, o Amazon ECR cria um perfil do IAM vinculada ao serviço em seu nome. A função do IAM vinculada ao serviço concede ao serviço de replicação do Amazon ECR a permissão necessária para criar repositórios e replicar imagens em seu registro. Para obter mais informações, consulte Uso de funções vinculadas ao serviço para o Amazon ECR.
- Para que a replicação entre contas ocorra, o destino do registro privado deve conceder permissão para permitir que o registro de origem replique suas imagens. Isso é feito definindo uma política de permissões de registro privado. Para obter mais informações, consulte Permissões de registro privado no Amazon ECR.
- Se a política de permissão para um registro privado for alterada para remover uma permissão, todas as replicações em andamento concedidas anteriormente poderão ser concluídas.
- Para que a replicação entre regiões ocorra, tanto a conta de origem quanto a conta de destino devem estar ativadas na região antes que qualquer ação de replicação ocorra dentro da região ou tendo a região como destino. Para obter mais informações, consulte Como gerenciar regiões da AWS no Referência geral da Amazon Web Services.
- A replicação entre regiões não é suportada entre AWS partições. Por exemplo, um repositório em `us-west-2` não pode ser replicado para `cn-north-1`. Para obter mais informações sobre AWS partições, consulte o formato ARN AWS na Referência geral.
- A configuração de replicação para um registro privado pode conter até 25 destinos exclusivos em todas as regras, com um máximo de 10 regras no total. Cada regra pode conter até 100 filtros. Isso permite especificar regras separadas para repositórios contendo imagens usadas para produção e teste, por exemplo.
- A configuração de replicação oferece suporte à filtragem de quais repositórios em um registro privado são replicados especificando um prefixo de repositório. Para ver um exemplo, consulte Exemplo: configurar a replicação entre regiões usando um filtro de repositório.

- Uma ação de replicação ocorre somente uma vez por envio de imagem ou restauração de imagem. Por exemplo, se você configurou a replicação entre regiões do `us-west-2` para `us-east-1` e do `us-east-1` para `us-east-2`, uma imagem enviada para `us-west-2` replica somente para `us-east-1`, ela não é replicada novamente para `us-east-2`. Esse comportamento se aplica à replicação entre regiões e entre contas.
- A maioria das imagens replica-se em menos de 30 minutos, mas em casos raros a replicação pode demorar mais.
- A replicação do registro não executa nenhuma ação de exclusão ou arquivamento. Imagens e repositórios replicados podem ser excluídos ou arquivados quando não estiverem mais sendo usados.
- Se a imagem a ser replicada for arquivada no destino, ela será restaurada no destino.
- Quando uma imagem é arquivada em uma região de origem, ela não será arquivada em uma região de destino especificada pela configuração de replicação.
- As políticas de repositório, incluindo políticas do IAM e políticas de ciclo de vida não são replicadas e não têm nenhum efeito além do repositório para o qual estão definidas.
- As configurações do repositório não são replicadas por padrão. Você pode replicar as configurações do repositório usando modelos de criação de repositórios. Essas configurações incluem mutabilidade de tags, criptografia, permissões de repositório e políticas de ciclo de vida. Para obter mais informações sobre modelos de criação de repositórios, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.
- Se a imutabilidade da etiqueta estiver habilitada em um repositório e for replicada uma imagem que usa a mesma marca de uma imagem existente, a imagem será replicada, mas não conterá a etiqueta duplicada. Isso pode resultar na imagem ficar sem etiqueta.
- Ao replicar imagens, se a montagem de blob tiver sido configurada, o ECR verificará se todas as camadas do repositório de origem já existem no registro de destino. Se alguma camada já existir no registro de destino, o ECR montará essas camadas.

> **Note**
>
> Se o registro de origem for diferente do registro de destino, a montagem de blobs precisará ser habilitada em ambos os registros para que o ECR monte camadas replicadas.

Exemplos de replicação de imagens privadas do Amazon ECR

Os exemplos a seguir mostram casos de uso comuns para replicação de imagens privadas. Se você configurar a replicação usando o AWS CLI, poderá usar os exemplos de JSON como ponto de partida ao criar seu arquivo JSON. Caso configure a replicação usando o Console de gerenciamento da AWS, você verá um JSON semelhante ao revisar a regra de replicação na página Revisar e enviar.

Exemplo: configurar a replicação entre regiões para uma única região de destino

A seguir, é mostrado um exemplo para configurar a replicação entre regiões em um único registro. Este exemplo pressupõe que o ID da conta seja `111122223333` e que você está especificando essa configuração de replicação em uma região diferente de `us-west-2`.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Exemplo: configurar a replicação entre regiões usando um filtro de repositório

O exemplo a seguir mostra um exemplo para configurar a replicação entre regiões para repositórios que correspondam a um valor de nome de prefixo. Este exemplo pressupõe que o ID da conta seja `111122223333` e que você está especificando essa configuração de replicação em uma região diferente de `us-west-1` e tem repositórios com prefixo `prod`.

```
{
 "rules": [{
```

```
  "destinations": [{
   "region": "us-west-1",
   "registryId": "111122223333"
  }],
  "repositoryFilters": [{
   "filter": "prod",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

Exemplo: configurar a replicação entre regiões para várias regiões de destino

A seguir, é mostrado um exemplo para configurar a replicação entre regiões em um único registro. Este exemplo pressupõe que o ID da sua conta seja `111122223333` e que você esteja especificando essa configuração de replicação em uma região diferente de ou. `us-west-1` `us-west-2`

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-1",
                    "registryId": "111122223333"
                },
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Exemplo: configurar replicação entre contas

A seguir, é mostrado um exemplo para configurar a replicação entre contas para o registo. Este exemplo configura a replicação para a conta `444455556666` e para a região `us-west-2`.

> **Important**
>
> Para que ocorra replicação entre contas, a conta de destino deve configurar uma política de permissões de registo para permitir que a replicação ocorra. Para obter mais informações, consulte Permissões de registro privado no Amazon ECR.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "444455556666"
                }
            ]
        }
    ]
}
```

Exemplo: especificar várias regras em uma configuração

A seguir, é mostrado um exemplo para configurar várias regras de replicação para o seu registro. Este exemplo configura a replicação para a conta *111122223333* com uma regra que replica repositórios com prefixo `prod` para a região `us-west-2` e repositórios com prefixo `test` para a região `us-east-2`. Uma configuração de replicação pode conter até 10 regras, com cada regra especificando até 25 destinos.

```
{
 "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"
   }]
  }
 ]
}
```

Exemplo: remover todas as configurações de replicação

A seguir, é mostrado um exemplo para remover todas as configurações de replicação do seu registro. Para remover as configurações de replicação, você deve configurar uma matriz de regras vazia.

```
{
    "rules": []
}
```

> **Important**
>
> A remoção das configurações de replicação não exclui nenhum repositório ou imagem replicado anteriormente. Você precisa excluir manualmente o conteúdo replicado caso ele não seja mais necessário.

Configurar replicação de imagem privada no Amazon ECR

Configurar a replicação por região para o registro privado. Você pode configurar a replicação entre regiões ou entre contas.

Para obter exemplos de como a replicação é comumente usada, consulte Exemplos de replicação de imagens privadas do Amazon ECR.

Para configurar as definições de replicação do registo (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.

2. Na barra de navegação, selecione a região para a qual definirá as configurações de replicação do registro.
3. No painel de navegação, escolha Private registry (Registro privado).
4. Na página Registro privado, escolha Configurações e, em seguida, escolha Editar em Configuração de replicação.
5. Na página Replication (Replicação), selecione Add replication rule (Adicionar regra de replicação).
6. Na página Destination types (Tipos de destino), escolha se deseja ativar a replicação entre regiões, a replicação entre contas ou ambas e escolha Next (Próximo).
7. Se a replicação entre regiões estiver habilitada, então para Configure destination regions (Configurar regiões de destino), escolha um ou mais Destination regions (Regiões de destino) e, depois, escolha Next (Próximo).
8. Se a replicação entre contas estiver habilitada, então para Cross- accpint replication (Replicação entre contas), escolha a configuração de replicação entre contas para o registro. Para Destination account (Conta de destino), insira o ID da conta para a conta de destino e uma ou mais Destination regions (Regiões de destino) para replicar. Selecione Destination account + (Conta de destino +) para configurar contas adicionais como destinos de replicação.

   > **Important**
   >
   > Para que ocorra replicação entre contas, a conta de destino deve configurar uma política de permissões de registo para permitir que a replicação ocorra. Para obter mais informações, consulte Permissões de registro privado no Amazon ECR.

9. (Opcional) Na página Add filters (Adicionar filtros), especifique um ou mais filtros para a regra de replicação e escolhaAdd (Adicionar). Repita essa etapa para cada filtro que deseja associar à ação de replicação. Um filtro deve ser especificado como prefixo do nome do repositório. Se nenhum filtro for adicionado, o conteúdo de todos os repositórios será replicado. Selecione Next (Próximo) quando todos os filtros tiverem sido adicionados.
10. Na página Review and submit (Analisar e enviar), analise a configuração da regra de replicação e selecione Submit rule (Enviar regra).

Para configurar as definições de replicação do registo (AWS CLI)

1. Crie um arquivo JSON contendo as regras de replicação a serem definidas para o registro. Uma configuração de replicação pode conter até 10 regras, com até 25 destinos exclusivos entre todas as regras e 100 filtros por regra. Para configurar a replicação entre regiões em sua própria conta, especifique seu próprio ID de conta. Para obter mais exemplos, consulte Exemplos de replicação de imagens privadas do Amazon ECR.

```
{
 "rules": [{
  "destinations": [{
   "region": "destination_region",
   "registryId": "destination_accountId"
  }],
  "repositoryFilters": [{
   "filter": "repository_prefix_name",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

2. Crie uma configuração de replicação para o seu registro.

```
aws ecr put-replication-configuration \
     --replication-configuration file://replication-settings.json \
     --region us-west-2
```

3. Confirme as configurações do seu registro.

```
aws ecr describe-registry \
     --region us-west-2
```

Remoção das configurações de replicação de imagem privada no Amazon ECR

Para remover ou desativar as configurações de replicação do seu registro privado, você precisa definir uma configuração de replicação vazia. Não há nenhum comando de remoção dedicado no AWS CLI.

Para remover as configurações de replicação do registo (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região da qual serão removidas as configurações de replicação do registro.
3. No painel de navegação, escolha Private registry (Registro privado).
4. Na página Registro privado, escolha Configurações e, em seguida, escolha Editar em Configuração de replicação.
5. Remova todas as regras de replicação existentes escolhendo a opção de exclusão para cada regra.
6. Selecione Salvar para aplicar a configuração de replicação vazia.

Para remover as configurações de replicação do registo (AWS CLI)

1. Crie um arquivo JSON com uma matriz de regras vazia para remover todas as configurações de replicação.

```
{
    "rules": []
}
```

2. Aplique a configuração de replicação vazia ao seu registro.

```
aws ecr put-replication-configuration \
     --replication-configuration file://empty-replication-settings.json \
     --region us-west-2
```

3. Confirme que as configurações de replicação foram removidas.

```
aws ecr describe-registry \
     --region us-west-2
```

O resultado deve ser um `replicationConfiguration` vazio sem regras.

> **Important**
>
> A remoção das configurações de replicação não exclui nenhum repositório ou imagem replicado anteriormente. Você precisa excluir manualmente o conteúdo replicado caso ele não seja mais necessário.

Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação

Use modelos de criação de repositório do Amazon ECR para definir as configurações aplicadas aos repositórios criados pelo Amazon ECR em seu nome. As configurações em um modelo de criação de repositório são aplicadas apenas durante a criação do repositório e não têm efeito sobre repositórios existentes ou repositórios criados usando qualquer outro método. Atualmente, os modelos de criação de repositório podem ser usados para aplicar configurações durante a criação do repositório para estes recursos:

- Cache de pull-through
- Crie por push
- Replicação

Como funcionam os modelos de criação de repositórios

Há momentos em que o Amazon ECR precisa criar um novo repositório privado em seu nome. Por exemplo:

- Na primeira vez que você usar uma regra de cache de pull-through para recuperar o conteúdo de um repositório upstream e armazená-lo no registro privado do Amazon ECR.
- Quando você envia uma imagem para um repositório que ainda não existe.
- Quando você quiser que o Amazon ECR replique um repositório para outra região ou conta.

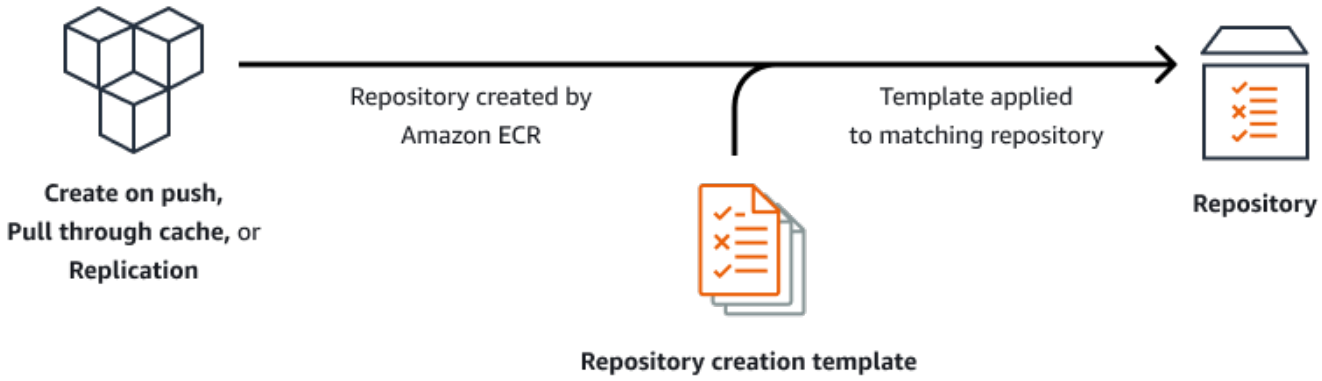
Quando não houver um modelo de criação de repositório que corresponda à regra de cache de pull-through ou ao repositório replicado, o Amazon ECR utilizará as configurações padrão para o novo repositório. Essas configurações padrão incluem desativar a imutabilidade da tag, usar criptografia `AES-256` e não aplicar nenhuma política de repositório ou ciclo de vida.

Quando não há um modelo de criação de repositório que corresponda ao repositório de destino para um envio de imagem, o Amazon ECR não criará um repositório com configurações padrão.

O uso de um modelo de criação de repositório permite definir as configurações que o Amazon ECR aplica aos novos repositórios criados por meio das ações de pull through cache, create on push e

replicação. É possível definir a imutabilidade da tag, a configuração de criptografia, as permissões do repositório, a política de ciclo de vida e as tags de recursos para os novos repositórios.

O diagrama a seguir mostra o fluxo de trabalho que o Amazon ECR usa quando um modelo de criação de repositório é usado.



A seguir, descrevemos detalhadamente cada parâmetro no modelo de criação de repositório.

Prefixo

O prefixo é o prefixo do namespace do repositório a ser associado ao modelo. Todos os repositórios criados usando esse prefixo terão as configurações aplicadas que estão definidas nesse modelo. Por exemplo, um prefixo de se `prod` aplicaria a todos os repositórios começando com `prod/`. Por exemplo, um prefixo de `prod/team` se aplicaria a todos os repositórios começando com `prod/team/`. Em um registro contendo dois modelos, se um modelo tiver o prefixo “prod” e o outro tiver o prefixo "prod/team", the template with the prefix "prod/team" will be applied to all repositories whose names start with "prod/team/”.

Para aplicar um modelo a todos os repositórios em seu registro que não têm um modelo de criação associado, você pode usar `ROOT` como prefixo.

> **Important**
>
> Sempre há uma suposição `/` aplicada ao fim do prefixo. Se você especificar `ecr-public` como prefixo, o Amazon ECR tratará isso como `ecr-public/`. Ao usar uma regra de cache de pull-through, o prefixo do repositório que você especifica durante a criação da regra é o que você também deve especificar como prefixo do modelo de criação do repositório.

Description

Essa descrição do modelo é opcional e é usada para descrever a finalidade do modelo de criação de repositório.

Aplicado para

A configuração aplicado para determina quais repositórios criados pelo Amazon ECR serão criados com esse modelo. Os valores válidos são `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH` e `REPLICATION`. Por exemplo, a primeira vez que você usa uma regra de cache de pull-through para recuperar o conteúdo de um repositório upstream e armazená-lo em seu registro privado do Amazon ECR. Quando não há um modelo de criação de repositório que corresponda à sua regra de cache de pull-through, o Amazon ECR utiliza as configurações padrão para o novo repositório.

Perfil de criação de repositório

O perfil de criação de repositórios é um ARN do perfil do IAM que será considerado pelo Amazon ECR ao criar e configurar repositórios por meio de modelos de criação de repositórios. Essa função deve ser fornecida ao usar as tags de repositório and/or KMS no modelo, caso contrário, a criação do repositório falhará.

Mutabilidade de tag de imagem

A configuração de mutabilidade da tag a ser usada para repositórios criados usando o modelo. Se este parâmetro for omitido, será usada a configuração padrão de MUTÁVEL, o que permitirá que as tags de imagem sejam substituídas. Essa é a configuração recomendada a ser usada para modelos usados em repositórios criados por ações de cache de pull-through. Isso garante que o Amazon ECR possa atualizar as imagens em cache quando as tags forem as mesmas.

Se IMUTÁVEL for especificada, todas as tags de imagem dentro do repositório serão imutáveis, o que impedirá que elas sejam substituídas.

Configuração de criptografia

> **Important**
>
> A criptografia de camada dupla do lado do servidor com AWS KMS (DSSE-KMS) só está disponível nas regiões. AWS GovCloud (US)

A configuração de criptografia a ser usada para repositórios criados usando o modelo.

Se você usar o tipo de criptografia KMS, o conteúdo do repositório será criptografado usando criptografia do lado do servidor com uma chave AWS Key Management Service armazenada em AWS KMS. Ao usar AWS KMS para criptografar seus dados, você pode usar a AWS KMS chave AWS gerenciada padrão para o Amazon ECR ou especificar sua própria AWS KMS chave, que você já criou. Você também pode optar por usar criptografia de camada única ou de camada dupla com. AWS KMS Para obter mais informações, consulte Criptografia em repouso. Caso esteja usando o tipo de criptografia do KMS e com a replicação entre regiões, você talvez precise de permissões adicionais. Para obter mais informações, consulte Creating a KMS key policy for replication.

Se você usar o tipo de criptografia AES256, o Amazon ECR usará criptografia no lado do servidor com chaves de criptografia gerenciadas pelo Amazon S3 que criptografa as imagens no repositório usando um algoritmo de criptografia AES-256. Para obter mais informações, consulte Proteção de dados usando criptografia do lado do servidor com chaves de criptografia gerenciadas pelo Amazon S3 (SSE-S3) no Manual do usuário do Amazon Simple Storage Service.

Permissões do repositório

A política de repositório a ser aplicada aos repositórios criados usando o modelo. Uma política de repositório utiliza permissões baseadas em recursos para controlar o acesso a um repositório. As permissões baseadas em recursos permitem especificar quais usuários ou funções do IAM têm acesso a um repositório e quais ações podem realizar nele. Por padrão, somente a AWS conta que criou o repositório tem acesso a um repositório. Você pode aplicar um documento de política que concede ou nega permissões adicionais ao repositório. Para obter mais informações, consulte Políticas de repositório privado no Amazon ECR.

Política de ciclo de vida do repositório

A política de ciclo de vida a ser usada para repositórios criados usando o modelo. Uma política de ciclo de vida oferece mais controle sobre o gerenciamento do ciclo de vida das imagens em um repositório privado. Uma política de ciclo de vida é um conjunto de uma ou mais regras em que cada regra define uma ação do Amazon ECR. Isso permite a automação da limpeza de imagens de suas imagens de contêiner por imagens com validade prestes a expirar baseadas em idade ou número. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.

Tags de recursos

As tags de recurso são metadados a serem aplicados ao repositório para ajudá-lo a categorizá-los e organizá-los. Cada tag consiste de uma chave e um valor opcional, que podem ser

definidos. Essa permissão precisa ser aplicada na política de registro de destino caso esteja usando modelos de criação de repositórios com replicação entre regiões.

Criar um modelo de criação de repositório no Amazon ECR

Você pode criar um modelo de criação de repositório para definir as configurações a serem usadas para repositórios criados pelo Amazon ECR em seu nome durante ações de pull through cache, create on push ou replicação. Depois que o modelo de criação do repositório for criado, todos os novos repositórios criados terão as configurações aplicadas. Isso não tem efeito em repositórios criados anteriormente.

Ao configurar um repositório com modelos, você tem a opção de especificar chaves do KMS e tags de recursos. Caso pretenda usar chaves do KMS, tags de recursos ou uma combinação das duas em um ou mais modelos, você precisa:

- Criar uma política personalizada para modelos de criação de repositórios.
- Criar um perfil do IAM para modelos de criação de repositórios.

Depois de configurado, você pode anexar o perfil personalizado a modelos específicos no registro.

Permissões do IAM para criar modelos de criação de repositórios

As permissões a seguir são necessárias para que uma entidade principal do IAM gerencie os modelos de criação de repositórios. Essas permissões devem ser concedidas usando uma política do IAM baseada em identidade.

- `ecr:CreateRepositoryCreationTemplate` - Concede permissão para criar o modelo de criação do repositório
- `ecr:UpdateRepositoryCreationTemplate`: concede permissão para atualizar um modelo de criação de repositório.
- `ecr:DescribeRepositoryCreationTemplates`: concede permissão para listar modelos de criação de repositórios em um registro.
- `ecr:DeleteRepositoryCreationTemplate` - Concede permissão para excluir o modelo de criação do repositório
- `ecr:CreateRepository`: concede permissão para criar um repositório do Amazon ECR.

- `ecr:PutLifecyclePolicy` - Concede permissão para criar uma política de ciclo de vida e aplicá-la a um repositório. Esta permissão é necessária apenas se o modelo de criação de repositório incluir uma política de ciclo de vida.
- `ecr:SetRepositoryPolicy` - Concede permissão para criar uma política de permissões para um repositório. Esta permissão é necessária apenas se o modelo de criação de repositório incluir uma política de repositório.
- `iam:PassRole`: concede permissão para permitir que uma entidade passe um perfil para um serviço ou aplicação. Essa permissão é necessária para serviços e aplicações que precisam assumir um perfil para executar ações em seu nome.

Criar uma política personalizada para modelos de criação de repositórios

Você pode usar o Console de gerenciamento da AWS para definir uma política que será posteriormente associada a uma função do IAM. Esse perfil do IAM pode então ser utilizado como um perfil de criação de repositório ao configurar um modelo de criação de repositório.

Console de gerenciamento da AWS

Para usar o editor de políticas JSON para criar uma política personalizada para modelos de criação de repositórios.

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Selecione Criar política.
4. Na seção Editor de políticas, escolha a opção JSON.
5. Insira a política a seguir no campo 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. Resolva os avisos de segurança, as mensagens erros ou os avisos gerais gerados durante a validação de política, e depois escolha Próximo.
7. Quando terminar de adicionar as permissões à política, escolha Avançar.
8. Na página Revisar e criar, digite um nome de política e uma descrição (opcional) para a política que você está criando. Revise Permissões definidas nessa política para ver as permissões que são concedidas pela política.
9. Escolha Criar política para salvar sua nova política.
10. Crie um perfil para atribuir essa política ao modelo de criação. Consulte Criar um perfil do IAM para modelos de criação de repositórios.

Criar um perfil do IAM para modelos de criação de repositórios

Você pode usar o Console de gerenciamento da AWS para criar uma função que pode ser usada pelo Amazon ECR ao especificar a função de criação de repositório em um modelo de criação de repositório que está usando tags de repositório ou KMS em um modelo.

Console de gerenciamento da AWS

Para criar um perfil.

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.

2. No painel de navegação do console, escolha Roles (Perfis) e, em seguida, clique em Create role (Criar perfil).
3. Escolha o tipo de perfil Política de confiança personalizada.
4. Na seção Política de confiança personalizada, cole a política de confiança personalizada listada abaixo:

   JSON

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

5. Escolha Próximo.
6. Na página Adicionar permissões, marque a caixa de seleção ao lado da política personalizada que você criou anteriormente na lista de políticas de permissões, e escolha Próximo.
7. Em Role name (Nome da função), digite um nome para sua função. Os nomes das funções devem ser exclusivos em seu Conta da AWS. Ao ser usada em uma política ou como parte de um ARN, o nome da função diferencia maiúsculas de minúsculas. Quando exibida para os clientes no console, por exemplo, como durante o processo de login, o nome de função não diferencia maiúsculas de minúsculas. Como várias entidades podem fazer referência à função, não é possível editar o nome da função depois de criada.
8. (Opcional) Em Descrição da função, insira uma descrição para a nova função.
9. Revise a função e escolha Criar função.

Criar um modelo de criação de repositório

Depois de concluir os pré-requisitos necessários para os modelos, você pode continuar criando os modelos de criação do repositório.

Console de gerenciamento da AWS

Para criar um modelo de criação de repositório (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região para criar o modelo de criação de repositório.
3. No painel de navegação, escolha Registro privado, Modelos de criação de repositório.
4. Na página Modelos de criação de repositório, escolha Criar modelo.
5. Na página Etapa 1: definir modelo, para Detalhes do modelo, escolha Um prefixo específico para aplicar o modelo a um prefixo de namespace de repositório específico ou escolha Qualquer prefixo em seu registro ECR para aplicar o modelo a todos os repositórios que não correspondam a nenhum outro modelo na região.

   a. Se você escolher Um prefixo específico, em Prefixo, especifique o prefixo do namespace do repositório ao qual aplicar o modelo. Sempre há uma suposição `/` aplicada ao fim do prefixo. Por exemplo, um prefixo de se `prod` aplicaria a todos os repositórios começando com `prod/`. Por exemplo, um prefixo de `prod/team` se aplicaria a todos os repositórios começando com `prod/team/`.

   b. Se você escolher Qualquer prefixo em seu registro do ECR, o prefixo será definido como `ROOT`.
6. Em Aplicado para, especifique a quais fluxos de trabalho do Amazon ECR esse modelo se aplicará. As opções são `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH` e `REPLICATION`.
7. Em Descrição do modelo, especifique uma descrição opcional para o modelo e escolha Avançar.
8. Na página Etapa 2: adicionar configuração de criação de repositório, especifique a configuração de configuração do repositório a ser aplicada aos repositórios criados usando o modelo.

   a. Para Mutabilidade de tag de imagem, escolha a configuração de mutabilidade de tags a ser usada. Para obter mais informações, consulte Impedir que as tags de imagens sejam sobrescritas no Amazon ECR.

- Mutável – Escolha essa opção se quiser que as tags de imagem sejam sobrescritas. É recomendada para repositórios que usam ações de cache de pull-through para garantir que o Amazon ECR possa atualizar imagens armazenadas em cache. Além disso, para desabilitar as atualizações de tag para algumas tags mutáveis, insira os nomes das tags ou use curingas (*) para combinar várias tags semelhantes na caixa de texto Exclusão de tag mutável.
- Imutável – Selecione esta opção se quiser impedir que as tags de imagem sejam sobrescritas. Isso se aplica a todas as tags e exclusões no repositório ao enviar uma imagem com uma tag existente. O Amazon ECR retorna uma `ImageTagAlreadyExistsException` se você tentar enviar uma imagem com uma tag existente. Além disso, para habilitar as atualizações de tag para algumas tags imutáveis, insira os nomes das tags ou use curingas (*) para combinar várias tags semelhantes na caixa de texto Exclusão de tag imutável.

b. Para Configuração de criptografia, escolha a configuração de criptografia a ser usada. Para obter mais informações, consulte Criptografia em repouso.

Quando AES-256 é selecionado, o Amazon ECR utiliza a criptografia do lado do servidor com chaves de criptografia gerenciadas pelo Amazon Simple Storage Service, o que criptografa seus dados em repouso usando o padrão de criptografia AES-256. Este é oferecido sem custo adicional.

Quando o KMS do AWS é selecionado, o Amazon ECR usa criptografia do lado do servidor com chaves armazenadas em AWS Key Management Service (AWS KMS). Ao usar AWS KMS para criptografar seus dados, você pode usar a chave AWS gerenciada padrão, que é gerenciada pelo Amazon ECR, ou especificar sua própria AWS KMS chave, chamada de chave gerenciada pelo cliente.

> **Note**
>
> As configurações de criptografia para um repositório não podem ser alteradas após a criação do repositório.

c. Para permissões do repositório, especifique a política de permissões do repositório a ser aplicada aos repositórios criados usando esse modelo. Opcionalmente, você pode usar o menu suspenso para selecionar uma das amostras de JSON para os casos de uso mais comuns. Para obter mais informações, consulte Políticas de repositório privado no Amazon ECR.

d. Para a Política do ciclo de vida do repositório, especifique a política de ciclo de vida do repositório a ser aplicada aos repositórios criados usando esse modelo. Opcionalmente, você pode usar o menu suspenso para selecionar uma das amostras de JSON para os casos de uso mais comuns. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.

e. Para AWS tags de repositório, especifique os metadados, na forma de pares de valores-chave, a serem associados aos repositórios criados usando esse modelo e escolha Avançar. Para obter mais informações, consulte Marcar um repositório privado no Amazon ECR.

f. Em Perfil de criação de repositório, selecione um perfil personalizado do IAM no menu suspenso a ser usado para modelos de criação de repositório ao usar tags de repositório ou o KMS no modelo (consulte Criar um perfil do IAM para modelos de criação de repositórios para obter detalhes). Em seguida, escolha Próximo.

9. Na página Etapa 3: revisar e criar, revise as configurações que você especificou para o modelo de criação do repositório. Escolha a opção Editar para fazer alterações. Escolha Criar quando você terminar.

AWS CLI

O create-repository-creation-template AWS CLI comando é usado para criar um modelo de criação de repositório para seu registro privado.

Para criar um modelo de criação de repositório (AWS CLI)

1. Use o AWS CLI para gerar um esqueleto para o create-repository-creation-templatecomando.

```
aws ecr create-repository-creation-template \
    --generate-cli-skeleton
```

A saída do comando exibe a sintaxe completa do modelo de criação de repositório.

```
{
"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. Criar um arquivo denominado `repository-creation-template.json` com a saída da etapa anterior. Este modelo define uma chave de criptografia do KMS para qualquer repositório criado em `prod/*` com uma política de repositório que permite enviar por push e extrair imagens para repositórios futuros, define uma política de ciclo de vida que expirará imagens com mais de duas semanas e define um perfil personalizado que permitirá que o ECR acesse a chave do KMS e atribua a tag de recurso `examplekey` a repositórios futuros.

```
{
"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. Use o comando a seguir para criar um modelo de criação de repositório. Certifique-se de especificar o nome do arquivo de configuração criado na etapa anterior no lugar do `repository-creation-template.json` no exemplo a seguir.

```
aws ecr create-repository-creation-template \
    --cli-input-json file://repository-creation-template.json
```

Como atualizar um modelo de criação de repositório

Você pode editar o modelo de criação de repositório caso precise alterar as configurações. Depois que o modelo de criação de repositório for editado, as novas configurações serão aplicadas ao modelo existente.

> **Important**
>
> Isso não tem efeito em repositórios criados anteriormente.

Console de gerenciamento da AWS

Para editar um modelo de criação de repositório (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região onde o modelo de criação de repositório que deseja editar está localizado.
3. No painel de navegação, escolha Registro privado e depois Configurações.
4. Na barra de navegação, escolha os Modelos de criação de repositório.
5. Na página Modelos de criação de repositório, selecione o modelo de criação de repositório a ser editado.
6. No menu suspenso Ações, escolha Editar.
7. Revise e atualize as configurações.
8. Opte por atualizar para aplicar as novas configurações do modelo de criação.

AWS CLI

Para editar um modelo de criação de repositório (AWS CLI)

- Use o update-repository-creation-templatecomando para atualizar um modelo de criação de repositório existente. Você deve especificar o valor do `prefix` do modelo. O exemplo a seguir atualiza um modelo de criação de repositório com o prefixo `prod`.

```
aws ecr update-repository-creation-template \
     --prefix prod \
     --image-tag-mutability="IMMUTABLE_WITH_EXCLUSION" \
     --image-tag-mutability-exclusion-filters filterType=WILDCARD, filter=latest
```

A saída do comando exibe os detalhes do modelo atualizado de criação de repositório.

Excluir um modelo de criação de repositório no Amazon ECR

É possível excluir um modelo de criação de repositório se terminar de usá-lo. Depois que um modelo de criação de repositório for excluído, todos os repositórios recém-criados sob o prefixo associado durante uma ação de cache de pull-through e de replicação herdarão as configurações padrão, a

menos que outro modelo correspondente seja encontrado. Consulte Como funcionam os modelos de criação de repositórios.

> **Important**
>
> Isso não tem efeito em repositórios criados anteriormente.

Console de gerenciamento da AWS

Para excluir um modelo de criação de repositório ()Console de gerenciamento da AWS

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região onde o modelo de criação de repositório que deseja excluir está localizado.
3. No painel de navegação, escolha Registro privado, Modelos de criação de repositório.
4. Na página Modelos de criação de repositório, selecione o modelo de criação de repositório a ser excluído.
5. No menu suspenso Ações, escolha Excluir.

AWS CLI

Para excluir um modelo de criação de repositório ()AWS CLI

- Use o comando delete-repository-creation-template.html para excluir um modelo de criação de repositório existente. Você deve especificar o valor do `prefix` do modelo. O exemplo a seguir exclui um modelo de criação de repositório com o prefixo `prod`.

```
aws ecr delete-repository-creation-template \
     --prefix prod
```

A saída do comando exibe os detalhes do modelo de criação de repositório excluído.

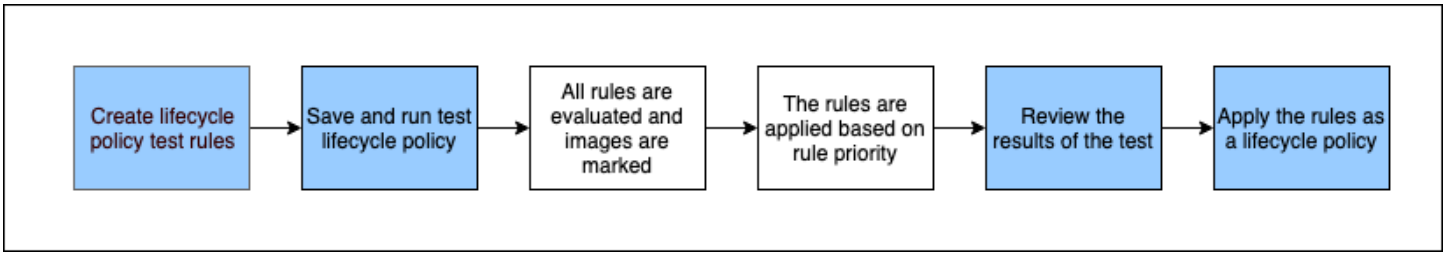
Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR

As políticas de ciclo de vida do Amazon ECR fornecem mais controle sobre o gerenciamento de ciclo de vida de imagens em um repositório privado. Uma política de ciclo de vida contém uma ou mais regras, e cada regra define uma ação do Amazon ECR. Com base nos critérios de expiração da política de ciclo de vida, as imagens podem ser arquivadas ou expiradas com base nos critérios especificados na política de ciclo de vida em 24 horas. Quando o Amazon ECR executa uma ação com base em uma política de ciclo de vida, ela é capturada como um evento no AWS CloudTrail. Para obter mais informações, consulte Registrando ações do Amazon ECR com AWS CloudTrail.

Como funcionam as políticas de ciclo

Uma política de ciclo de vida consiste em uma ou mais regras que determinam quais imagens em um repositório devem perder a validade. Ao considerar o uso de políticas de ciclo de vida, é importante usar a visualização da política de ciclo de vida para confirmar de quais imagens a política de ciclo de vida expira a validade antes de aplicá-la a um repositório. Depois que uma política de ciclo de vida é aplicada a um repositório, você deve esperar que as imagens expirem dentro de 24 horas após atenderem aos critérios de expiração. Quando o Amazon ECR executa uma ação com base em uma política de ciclo de vida, ela é capturada como um evento no AWS CloudTrail. Para obter mais informações, consulte Registrando ações do Amazon ECR com AWS CloudTrail.

O diagrama a seguir mostra um fluxo de trabalho da política de ciclo de vida.



1. Crie uma ou mais regras de teste.
2. Salve as regras de teste e execute a visualização.
3. O avaliador de políticas de ciclo de vida percorre todas as regras e marca as imagens que cada regra afeta.
4. O avaliador da política de ciclo de vida então aplica as regras, com base na prioridade da regra, e exibe quais imagens no repositório estão definidas para serem expiradas ou arquivadas. Um

número de prioridade de regra mais baixo significa maior prioridade. Por exemplo, uma regra com prioridade 1 tem precedência sobre uma regra com prioridade 2.

5. Analise os resultados do teste, garantindo que as imagens marcadas como expiradas ou arquivadas sejam as desejadas.
6. Aplique as regras de teste como política de ciclo de vida para o repositório.
7. Depois que a política de ciclo de vida for criada, você deve esperar que as imagens expirem ou sejam arquivadas dentro de 24 horas após atenderem aos critérios de expiração.

Regras de avaliação de política de ciclo de vida

O avaliador da política de ciclo de vida é responsável por analisar o JSON de texto simples da política de ciclo de vida, avaliando todas as regras e aplicando essas regras com base na prioridade da regra às imagens no repositório. A seguir encontra-se a explicação mais detalhada da lógica do avaliador de políticas de ciclo de vida. Para obter exemplos, consulte Exemplos de políticas de ciclo de vida no Amazon ECR.

- Quando artefatos de referência estão presentes em um repositório, as políticas de ciclo de vida do Amazon ECR expiram automaticamente ou arquivam esses artefatos dentro de 24 horas após a exclusão ou arquivamento da imagem em questão.
- Todas as regras são avaliadas ao mesmo tempo, independentemente da prioridade da regra. Depois que todas as regras são avaliadas, elas são aplicadas com base na prioridade da regra.
- Uma imagem é expirada ou arquivada por exatamente uma ou zero regras.
- Uma imagem que corresponda aos requisitos de marcação de uma regra não pode ser expirada ou arquivada por uma regra com prioridade mais baixa.
- As regras nunca podem marcar imagens marcadas por regras de prioridade mais alta, mas ainda podem identificá-las como se não tivessem expirado ou arquivadas.
- O conjunto de todas as regras que selecionam uma classe de armazenamento específica deve conter um conjunto exclusivo de prefixos.
- Somente uma regra de seleção de uma classe de armazenamento específica pode selecionar imagens não marcadas.
- Se uma imagem for referenciada por uma lista de manifestos, ela não poderá ser expirada ou arquivada sem que a lista de manifestos seja excluída ou arquivada primeiro.
- A expiração é sempre ordenada por `pushed_at_time` ou `transitioned_at_time` e sempre expira as imagens mais antigas antes das mais novas. Se uma imagem foi arquivada e depois

restaurada em algum momento no passado, a imagem `last_activated_at` é usada em vez de`pushed_at_time`.

- Uma regra de política de ciclo de vida pode especificar um `tagPatternList` ou `tagPrefixList`, mas não ambos. Porém, uma política de ciclo de vida pode conter várias regras em que regras diferentes usam listas de padrões e prefixos. Uma imagem será correspondida com êxito se todas as tags no valor `tagPatternList` ou `tagPrefixList` corresponderem a qualquer tag da imagem.
- Os parâmetros `tagPatternList` ou `tagPrefixList` apenas poderão ser usados se o `tagStatus` for `tagged`.
- Ao usar `tagPatternList`, uma imagem será correspondida com êxito se corresponder ao filtro de curinga. Por exemplo, se um filtro de `prod*` for aplicado, ele corresponderá às tags de imagem cujo nome comece com `prod`, como `prod`, `prod1` ou `production-team1`. Da mesma maneira, se um filtro de `*prod*` for aplicado, ele corresponderá às tags de imagem cujo nome contenha `prod`, como `repo-production` ou `prod-team`.

  > **Important**
  >
  > Existe um limite máximo de quatro curingas (*) por string. Por exemplo, `["*test*1*2*3", "test*1*2*3*"]` é válido, mas `["test*1*2*3*4*5*6"]` é inválido.

- Ao usar `tagPrefixList`, uma imagem será correspondida com êxito se todos os filtros de curinga no valor `tagPrefixList` corresponderem a qualquer tag da imagem.
- O `countUnit` parâmetro só é usado se `countType` for `sinceImagePushed`ou`sinceImagePulled`, ou`sinceImageTransitioned`.
- Com`countType = imageCountMoreThan`, as imagens são classificadas da mais nova para a mais antiga com base em `pushed_at_time` e, em seguida, todas as imagens maiores que a contagem especificada expiram ou são arquivadas.
- Com`countType = sinceImagePushed`, todas as imagens mais `pushed_at_time` antigas do que o número especificado de dias com base em `countNumber` expiram ou são arquivadas.
- Com`countType = sinceImagePulled`, todas as imagens mais `last_recorded_pulltime` antigas do que o número de dias especificado com base em `countNumber` são arquivadas. Se uma imagem nunca foi extraída, a imagem `pushed_at_time` é usada em vez da`last_recorded_pulltime`. Se uma imagem foi arquivada e depois restaurada em algum

momento no passado, mas nunca foi retirada desde que a imagem foi restaurada, a imagem `last_activated_at` é usada em vez da`last_recorded_pulltime`.

- Com`countType = sinceImageTransitioned`, todas as imagens arquivadas `last_archived_at` mais antigas do que o número especificado de dias com base em `countNumber` expiram.
- A expiração é sempre ordenada por `pushed_at_time` e sempre expira as imagens mais antigas antes das novas.

Criar uma pré-visualização de política de ciclo de vida no Amazon ECR

Você pode usar uma pré-visualização da política de ciclo de vida para ver o impacto de uma política de ciclo de vida em um repositório de imagens antes de aplicá-la. É prática recomendada fazer uma visualização antes de aplicar uma política de ciclo de vida a um repositório.

> **Note**
>
> Se você estiver usando a replicação do Amazon ECR para fazer cópias de um repositório em diferentes regiões ou contas, tenha em mente que uma política de ciclo de vida só pode realizar uma ação em repositórios na região em que foi criada. Portanto, se você tiver a replicação ativada, convém criar uma política de ciclo de vida em cada região e conta para a qual estiver replicando os repositórios.

Para criar uma política de ciclo de vida (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região que contém o repositório no qual a visualização de uma política de ciclo de vida será executada.
3. No painel de navegação, em Registro privado, escolha Repositórios.
4. Na página Repositórios privados, selecione um repositório e use o menu suspenso Ações para escolher Políticas de ciclo de vida.
5. Na página de regras de política de ciclo de vida do repositório, selecione Editar regras de teste, Criar regra.
6. Especifique os seguintes detalhes para cada regra da política de ciclo de vida de teste.

a. Em Prioridade de regra, digite um número para a prioridade da regra. A prioridade da regra determina em que ordem as regras de políticas de ciclo de vida são aplicadas. Um número mais baixo significa maior prioridade. Por exemplo, uma regra com prioridade 1 tem precedência sobre uma regra com prioridade 2.

b. Em Descrição da regra, digite uma descrição para a regra de política de ciclo de vida.

c. Em Status da imagem, escolha Marcado (correspondência de curingas), Marcado (correspondência de prefixo), Sem etiqueta ou Qualquer.

> **Important**
>
> Se você especificar várias tags, apenas imagens com todas as tags especificadas serão selecionadas.

d. Se escolher Marcado (correspondência de curingas) para Status da imagem, em Especificar etiquetas para correspondência de curingas, você poderá especificar uma lista de etiquetas de imagem com um curinga (*) para agir com sua política de ciclo de vida. Por exemplo, se as suas imagens forem marcadas como `prod`, `prod1`, `prod2` e assim por diante, você especificaria `prod*` para agir em todas elas. Se você especificar várias tags, apenas imagens com todas as tags especificadas serão selecionadas.

> **Important**
>
> Existe um limite máximo de quatro curingas (*) por string. Por exemplo, `["*test*1*2*3", "test*1*2*3*"]` é válido, mas `["test*1*2*3*4*5*6"]` é inválido.

e. Se escolher Marcado (correspondência de prefixo) para Status da imagem e, em Especificar etiquetas para correspondência de prefixo, você poderá especificar uma lista de etiquetas de imagem nas quais agir com sua política de ciclo de vida.

f. Em Critérios de correspondência, escolha Dias desde a criação da imagem, Dias desde a última data de extração registrada, Dias desde o arquivamento da imagem ou Contagem de imagens e, em seguida, especifique um valor.

g. Em Ação de regra, escolha Expirar ou Arquivar.

h. Escolha Salvar.

7. Crie regras de política de ciclo de vida de teste adicionais repetindo as etapas de 5 a 7.

8. Para executar a visualização da política de ciclo de vida, escolha Salvar e executar teste.
9. Em Combinações de imagem para regras de ciclo de vida de teste, avalie o impacto da visualização da política de ciclo de vida.
10. Se você estiver satisfeito com os resultados da visualização, escolha Aplicar como política de ciclo de vida para criar uma política de ciclo de vida com as regras especificadas. Você deve esperar que, depois de aplicar uma política de ciclo de vida, as imagens afetadas expirem ou sejam arquivadas em 24 horas.
11. Se não estiver satisfeito com os resultados da pré-visualização, poderá excluir uma ou mais regras de ciclo de vida de teste e criar uma ou mais regras para substituí-las e, em seguida, repetir o teste.

Criar uma política de ciclo de vida para um repositório no Amazon ECR

Use uma política de ciclo de vida para criar um conjunto de regras que expiram ou arquivam imagens de repositório não utilizadas. Depois de criar uma política de ciclo de vida, as imagens afetadas expiram ou são arquivadas em 24 horas.

> **Note**
>
> Se você estiver usando a replicação do Amazon ECR para fazer cópias de um repositório em diferentes regiões ou contas, tenha em mente que uma política de ciclo de vida só pode realizar uma ação em repositórios na região em que foi criada. Portanto, se você tiver a replicação ativada, convém criar uma política de ciclo de vida em cada região e conta para a qual estiver replicando os repositórios.

Pré-requisito

Prática recomendada: crie uma prévia da política de ciclo de vida para verificar se as imagens expiradas ou arquivadas de acordo com suas regras de política de ciclo de vida são o que você pretende. Para instruções, consulte Criar uma pré-visualização de política de ciclo de vida no Amazon ECR.

Como criar uma política de ciclo de vida (Console de gerenciamento da AWS)

1. Abra o console do Amazon ECR nos https://console.aws.amazon.com/ecr/repositórios.
2. Na barra de navegação, selecione a região que contém o repositório para o qual uma política de ciclo de vida será criada.
3. No painel de navegação, em Registro privado, escolha Repositórios.
4. Na página Repositórios privados, selecione um repositório e use o menu suspenso Ações para escolher Políticas de ciclo de vida.
5. Na página de regras de política de ciclo de vida do repositório, selecione Criar regra.
6. Insira os seguintes detalhes para sua a regra da política de ciclo de vida.

    a. Em Prioridade de regra, digite um número para a prioridade da regra. A prioridade da regra determina em que ordem as regras de políticas de ciclo de vida são aplicadas. Um número de prioridade de regra mais baixo significa maior prioridade. Por exemplo, uma regra com prioridade 1 tem precedência sobre uma regra com prioridade 2.

    b. Em Descrição da regra, digite uma descrição para a regra de política de ciclo de vida.

    c. Em Status da imagem, escolha Marcado (correspondência de curingas), Marcado (correspondência de prefixo), Sem etiqueta ou Qualquer.

    > **Important**
    >
    > Se você especificar várias tags, apenas imagens com todas as tags especificadas serão selecionadas.

    d. Se escolher Marcado (correspondência de curingas) para Status da imagem, em Especificar etiquetas para correspondência de curingas, você poderá especificar uma lista de etiquetas de imagem com um curinga (*) para agir com sua política de ciclo de vida. Por exemplo, se as suas imagens forem marcadas como `prod`, `prod1`, `prod2` e assim por diante, você especificaria `prod*` para agir em todas elas. Se você especificar várias tags, apenas imagens com todas as tags especificadas serão selecionadas.

    > **Important**
    >
    > Existe um limite máximo de quatro curingas (*) por string. Por exemplo, `["*test*1*2*3", "test*1*2*3*"]` é válido, mas `["test*1*2*3*4*5*6"]` é inválido.

e. Se escolher Marcado (correspondência de prefixo) para Status da imagem e, em Especificar etiquetas para correspondência de prefixo, você poderá especificar uma lista de etiquetas de imagem nas quais agir com sua política de ciclo de vida.

f. Em Critérios de correspondência, escolha Dias desde a criação da imagem, Dias desde a última data de extração registrada, Dias desde o arquivamento da imagem ou Contagem de imagens e, em seguida, especifique um valor.

g. Em Ação de regra, escolha Expirar ou Arquivar.

h. Escolha Salvar.

7. Crie regras de política de ciclo de vida adicionais repetindo as etapas de 5 a 7.

Como criar uma política de ciclo de vida (AWS CLI)

1. Obtenha o nome do repositório para o qual a política de ciclo de vida será criada.

```
aws ecr describe-repositories
```

2. Crie um arquivo local chamado `policy.json` com o conteúdo da política de ciclo de vida. Para ver exemplos de política do ciclo de vida, consulte Exemplos de políticas de ciclo de vida no Amazon ECR.

3. Crie uma política de ciclo de vida especificando o nome do repositório e referencie o arquivo JSON da política de ciclo de vida criado.

```
aws ecr put-lifecycle-policy \
      --repository-name repository-name \
      --lifecycle-policy-text file://policy.json
```

Exemplos de políticas de ciclo de vida no Amazon ECR

Veja a seguir exemplos de políticas de ciclo de vida mostrando a sintaxe.

Para ver mais informações sobre as propriedades da política, consulte Propriedades da política de ciclo de vida no Amazon ECR. Para obter instruções sobre como criar uma política de ciclo de vida usando o AWS CLI, consulte. Como criar uma política de ciclo de vida (AWS CLI)

Modelo de política do ciclo de vida

O conteúdo da política de ciclo de vida é avaliado antes de ser associado a um repositório. Veja a seguir o modelo de sintaxe JSON de política de ciclo de vida.

```
{
        "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"
                }
            }
        ]
    }
```

Filtrar pela idade da imagem

O exemplo a seguir mostra a sintaxe da política de ciclo de vida de uma política que expira imagens com uma etiqueta que começa com `prod` usando um `tagPatternList` de `prod*` que também tenha mais de 14 dias.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
```

```
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtrando na hora da última extração

O exemplo a seguir mostra a sintaxe da política de ciclo de vida de uma política que faz a transição de imagens para armazenamento de arquivos que não são retiradas há dias. `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

O tipo de `sinceImagePulled` contagem deve ser usado com a `transition` ação. Ele não pode ser usado com a `expire` ação. Para excluir imagens com base na atividade de extração, primeiro transfira-as para armazenamento de arquivos usando `esinceImagePulled`, em seguida, use `sinceImageTransitioned` com uma `expire`

ação para excluí-las. As imagens devem estar armazenadas em arquivo por no mínimo 90 dias antes da exclusão.

Filtragem no tempo de transição do arquivamento

O exemplo a seguir mostra a sintaxe da política de ciclo de vida de uma política que expira imagens arquivadas que estão armazenadas há mais de dias. 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

O tipo de `sinceImageTransitioned` contagem deve ser usado com a `expire` ação e a classe `archive` de armazenamento. As imagens devem estar armazenadas em arquivo por no mínimo 90 dias antes da exclusão.

Filtrar pela contagem da imagem

O exemplo a seguir mostra a sintaxe de política de ciclo de vida para uma política que mantém apenas uma imagem não marcada e expira todas as outras.

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtrar por várias regras

Os exemplos a seguir usam várias regras em uma política de ciclo de vida. São fornecidos um repositório e uma política de ciclo de vida de exemplo com uma explicação do resultado.

Exemplo A

Conteúdo do repositório:

- Imagem A, Taglist: ["beta-1", "prod-1"], Enviada: 10 dias atrás
- Imagem B, Taglist: ["beta-2", "prod-2"], Enviada: 9 dias atrás
- Imagem C, Taglist: ["beta-3"], Enviada: 8 dias atrás

Texto da política de ciclo de vida:

```
{
    "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"
            }
        }
    ]
}
```

A lógica dessa política de ciclo de vida seria:

- A regra 1 identifica as imagens marcadas com o prefixo `prod`. Ela deve marcar imagens, começando com a mais antiga, até que haja uma ou menos imagem restante correspondente. Ela marca a imagem A para expiração.
- A regra 2 identifica as imagens marcadas com o prefixo `beta`. Ela deve marcar imagens, começando com a mais antiga, até que haja uma ou menos imagem restante correspondente. Ela marca as imagens A e B para expiração. No entanto, a imagem A já foi vista pela Regra 1 e se a imagem B fosse expirada, ela violaria a Regra. Portanto, é ignorada.
- Resultado: a imagem A é expirada.

Exemplo B

Este é o mesmo repositório do exemplo anterior mas a solicitação de prioridade de regra é alterada para ilustrar o resultado.

Conteúdo do repositório:

- Imagem A, Taglist: ["beta-1", "prod-1"], Enviada: 10 dias atrás

- Imagem B, Taglist: ["beta-2", "prod-2"], Enviada: 9 dias atrás
- Imagem C, Taglist: ["beta-3"], Enviada: 8 dias atrás

Texto da política de ciclo de vida:

```
{
    "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"
            }
        }
    ]
}
```

A lógica dessa política de ciclo de vida seria:

- A regra 1 identifica as imagens marcadas com o prefixo `beta`. Ela deve marcar imagens, começando com a mais antiga, até que haja uma ou menos imagem restante correspondente. Ela vê todas as três imagens e marcaria as imagens A e B para expiração.

- A regra 2 identifica as imagens marcadas com o prefixo `prod`. Ela deve marcar imagens, começando com a mais antiga, até que haja uma ou menos imagem restante correspondente. A Regra 2 não veria nenhuma imagem porque todas as imagens disponíveis já teriam sido vistas pela Regra 1. Portanto, nenhuma imagem adicional seria marcada.
- Resultado: as imagens A e B são expiradas.

Filtrar por várias tags em uma única regra

Os seguintes exemplos especificam a sintaxe de política de ciclo de vida para vários padrões de etiqueta em uma única regra. São fornecidos um repositório e uma política de ciclo de vida de exemplo com uma explicação do resultado.

Exemplo A

Quando vários padrões de etiquetas são especificados em uma única regra, as imagens devem corresponder a todos os padrões de etiquetas listados.

Conteúdo do repositório:

- Imagem A, Taglist: ["alpha-1"], Enviada: 12 dias atrás
- Imagem B, Taglist: ["beta-1"], Enviada: 11 dias atrás
- Imagem C, Taglist: ["alpha-2", "beta-2"], Enviada: 10 dias atrás
- Imagem D, Taglist: ["alpha-3"], Enviada: 4 dias atrás
- Imagem E, Taglist: ["beta-3"], Enviada: 3 dias atrás
- Imagem F, Taglist: ["alpha-4", "beta-4"], Enviada: 2 dias atrás

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
```

```
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

A lógica dessa política de ciclo de vida seria:

- A regra 1 identifica as imagens marcadas com o prefixo `alpha` e `beta`. Ela vê as imagens C e F. A Regra 1 deve marcar imagens que têm mais de cinco dias, que seria a imagem C.
- Resultado: a imagem C é expirada.

Exemplo B

O exemplo a seguir ilustra as tags que não são exclusivas.

Conteúdo do repositório:

- Imagem A, Taglist: ["alpha-1", "beta-1", "gamma-1"], Enviada: 10 dias atrás
- Imagem B, Taglist: ["alpha-2", "beta-2"], Enviada: 9 dias atrás
- Imagem C, Taglist: ["alpha-3", "beta-3", "gamma-2"], Enviada: 8 dias atrás

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
```

```
}
```

A lógica dessa política de ciclo de vida seria:

- A regra 1 identifica as imagens marcadas com o prefixo `alpha` e `beta`. Ela vê todas as imagens. Ela deve marcar imagens, começando com a mais antiga, até que haja uma ou menos imagem restante correspondente. E marca as imagens A e B para expiração.
- Resultado: as imagens A e B são expiradas.

Filtrar todas as imagens

Os exemplos de política de ciclo de vida a seguir especificam todas as imagens com filtros diferentes. São fornecidos um repositório e uma política de ciclo de vida de exemplo com uma explicação do resultado.

Exemplo A

O exemplo a seguir mostra a sintaxe de política de ciclo de vida para uma política que é aplicada a todas as regras, mas mantém apenas uma imagem e expira todas as outras.

Conteúdo do repositório:

- Imagem A, Taglist: ["alpha-1"], Enviada: 4 dias atrás
- Imagem B, Taglist: ["beta-1"], Enviada: 3 dias atrás
- Imagem C, Taglist: [], Enviada: 2 dias atrás
- Imagem D, Taglist: ["alpha-2"], Enviada: 1 dia atrás

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
```

```
                "type": "expire"
            }
        }
    ]
}
```

A lógica dessa política de ciclo de vida seria:

- A regra 1 identifica todas as imagens. Ela visualiza as imagens A, B, C e. D. Deve expirar todas as imagens, exceto a mais recente. E marca as imagens A, B e C para expiração.
- Resultado: as imagens A, B e C são expiradas.

Exemplo B

O exemplo a seguir ilustra uma política de ciclo de vida que combina todos os tipos de regra em uma única política.

Conteúdo do repositório:

- Imagem A, Taglist: ["alpha-", "beta-1", "-1"], Enviada: 4 dias atrás
- Imagem B, Taglist: [], Enviada: 3 dias atrás
- Imagem C, Taglist: ["alpha-2"], Enviada: 2 dias atrás
- Imagem D, Taglist: ["git hash"], Enviada: 1 dia atrás
- Imagem E, Taglist: [], Enviada: 1 dia atrás

```
{
    "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"
            }
        }
    ]
}
```

A lógica dessa política de ciclo de vida seria:

- A regra 1 identifica as imagens marcadas com o prefixo `alpha`. Ela identifica as imagens A e C. Deve manter a imagem mais recente e marcar o restante para expiração. Ela marca a imagem A para expiração.
- A regra 2 identifica as imagens não marcadas. Ela identifica as imagens B e E. Deve marcar todas as imagens com mais de um dia para expiração. E marca a imagem B para expiração.
- A regra 3 identifica todas as imagens. Ela identifica as imagens A, B, C, D e E. Deve manter a imagem mais recente e marcar o restante para expiração. No entanto, ela não pode marcar as imagens A, B, C ou E, pois elas foram identificadas por regras de maior prioridade. E marca a imagem D para expiração.
- Resultado: as imagens A, B e D são expiradas.

Exemplos de arquivamento

Os exemplos a seguir mostram políticas de ciclo de vida que arquivam imagens em vez de excluí-las.

Arquivamento de imagens com mais de um determinado número de dias

O exemplo a seguir mostra uma política de ciclo de vida que arquiva imagens com tags `prod` que começam com mais de 30 dias:

```
{
    "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"
            }
        }
    ]
}
```

Arquivamento de imagens não extraídas em um determinado número de dias

O exemplo a seguir mostra uma política de ciclo de vida que arquiva imagens que não foram retiradas em 90 dias:

```
{
    "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"
            }
        }
    ]
}
```

Combinando regras de arquivamento e expiração

O exemplo a seguir mostra uma política de ciclo de vida que arquiva imagens com mais de 30 dias e, em seguida, expira permanentemente as imagens que foram arquivadas por mais de 365 dias:

> **Note**
>
> As imagens arquivadas têm uma duração mínima de armazenamento de 90 dias. Você não pode configurar políticas de ciclo de vida que excluam imagens que estão arquivadas há menos de 90 dias. Se você precisar excluir imagens que foram arquivadas por menos de 90 dias, precisará usar a batch-delete-image API, mas você será cobrado pela duração mínima de armazenamento de 90 dias.

```
{
    "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"
            }
        }
    ]
}
```

Propriedades da política de ciclo de vida no Amazon ECR

As políticas de ciclo de vida têm as propriedades a seguir.

Para ver exemplos de políticas de ciclo de vida, consulte Exemplos de políticas de ciclo de vida no Amazon ECR. Para obter instruções sobre como criar uma política de ciclo de vida usando o AWS CLI, consulte. Como criar uma política de ciclo de vida (AWS CLI)

Prioridade das regras

`rulePriority`

Tipo: inteiro

Obrigatório: sim

Define a ordem em que as regras são avaliadas, da menor para a maior. Uma regra de política de ciclo de vida com prioridade 1 será aplicada primeiro, uma regra com prioridade 2 será a próxima, e assim por diante. Ao adicionar regras a uma política de ciclo de vida, você deve dar a elas um valor exclusivo para `rulePriority`. Os valores não precisam ser sequenciais entre regras em uma política. Uma regra com um valor `tagStatus` de `any` deve ter o valor o mais alto para `rulePriority` e ser avaliada por último.

Description

`description`

Tipo: string

Obrigatório: não

(Opcional) Descreve a finalidade de uma regra em uma política de ciclo de vida.

Status da tag

`tagStatus`

Tipo: string

Obrigatório: sim

Determina se a regra da política de ciclo de vida que você está adicionando especifica uma tag para uma imagem. As opções aceitáveis são `tagged`, `untagged` ou `any`. Se você especificar `any`, todas as regras serão avaliadas segundo a regra Se você especificar`tagged`, também deverá especificar um `tagPrefixList` valor ou um `tagPatternList` valor. Se você especificar`untagged`, deverá omitir os dois `tagPrefixList` e. `tagPatternList`

Lista de padrões de etiquetas

`tagPatternList`

Tipo: list[string]

Obrigatório: sim, se `tagStatus` estiver definido como marcado e se `tagPrefixList` não for especificado

Ao criar uma política de ciclo de vida para imagens marcadas, uma prática recomendada é usar um `tagPatternList` para especificar as etiquetas que expirarão. Você especifica uma lista separada por vírgulas de padrões de etiquetas de imagem que podem conter curingas (`*`) sobre os quais agir com sua política de ciclo de vida. Por exemplo, se suas imagens forem marcadas como `prod`, `prod1`, `prod2` e assim por diante, você deve usar a lista de padrões de etiquetas `prod*` para especificá-las. Se você especificar várias tags, apenas imagens com todas as tags especificadas serão selecionadas.

> **Important**
>
> Existe um limite máximo de quatro curingas (*) por string. Por exemplo, `["*test*1*2*3", "test*1*2*3*"]` é válido, mas `["test*1*2*3*4*5*6"]` é inválido.

Lista de prefixos de tags

`tagPrefixList`

Tipo: list[string]

Obrigatório: sim, se `tagStatus` estiver definido como marcado e se `tagPatternList` não for especificado

Usado somente se você tiver especificado `"tagStatus": "tagged"` e não estiver especificando `tagPatternList`. Você deve especificar uma lista separada por vírgulas de prefixos de tags de imagem na qual agir com política de ciclo de vida. Por exemplo, se suas imagens forem marcadas como `prod`, `prod1`, `prod2` e assim por diante, você deve usar o prefixo de tag `prod` para especificá-las. Se você especificar várias tags, apenas imagens com todas as tags especificadas serão selecionadas.

Classe de armazenamento

`storageClass`

Tipo: string

Obrigatório: sim, se `countType` for `sinceImageTransitioned`

A regra selecionará somente imagens dessa classe de armazenamento. Ao usar um `countType` of`imageCountMoreThan`,`sinceImagePushed`, ou`sinceImagePulled`, o único valor suportado é`standard`. Ao usar um tipo de contagem de`sinceImageTransitioned`, isso é obrigatório, e o único valor suportado é`archive`. Se você omitir isso, o valor de `standard` será usado.

Tipo de contagem

`countType`

Tipo: sequência

Obrigatório: sim

Especifique um tipo de contagem a ser aplicado às imagens.

Se `countType` for definido como `imageCountMoreThan`, você também especificará `countNumber` para criar uma regra que define um limite no número de imagens que existem no repositório. Se `countType` estiver definido como`sinceImagePushed`, ou `sinceImagePulledsinceImageTransitioned`, você também especifica `countUnit` e especifica um limite de tempo `countNumber` para as imagens que existem no seu repositório.

Unidade de contagem

`countUnit`

Tipo: string

Obrigatório: sim, somente se `countType` estiver definido como `sinceImagePushedsinceImagePulled`, ou `sinceImageTransitioned`

Especifique uma unidade de contagem de `days` para indicar como a unidade de tempo, além de `countNumber`, que é o número de dias.

Isso só deve ser especificado quando `countType` é`sinceImagePushed`,`sinceImagePulled`, ou`sinceImageTransitioned`; um erro ocorrerá se você especificar uma unidade de contagem quando `countType` for qualquer outro valor.

Contagem numérica

`countNumber`

Tipo: inteiro

Obrigatório: sim

Especifique um número de contagem. Os valores aceitáveis são inteiros positivos (`0` não é um valor aceito).

Se o `countType` usado for `imageCountMoreThan`, o valor será o número máximo de imagens que você deseja manter no repositório. Se o `countType` usado for `sinceImagePushed`, o valor será o limite de idade máximo das imagens. Se for `countType` usado`sinceImagePulled`, o valor será o número máximo de dias desde a última vez que a imagem foi extraída. Se for `countType` usado`sinceImageTransitioned`, o valor será o número máximo de dias desde que a imagem foi arquivada.

Ação

`type`

Tipo: string

Obrigatório: sim

Especifique um tipo de ação. Os valores suportados são `expire` (para excluir imagens) e `transition` (para mover imagens para o armazenamento de arquivos).

`targetStorageClass`

Tipo: string

Obrigatório: sim, se `type` for `transition`

A classe de armazenamento para a qual você deseja que a política de ciclo de vida faça a transição da imagem. `archive`é o único valor suportado.

Exclusões de atualizações em tempo integral

O Amazon ECR atualiza o `LastRecordedPullTime` timestamp em todas as retiradas, exceto nas retiradas do Inspector. AWS As exclusões de atualização em tempo integral permitem que você especifique uma função do IAM ARNs que não deve atualizar os tempos de extração das imagens ao extrair imagens, como as extraídas de scanners de terceiros (como Crowdstrike, Snyk e Trivy). Isso é útil para imagens usadas para testes ou para CI/CD fins em que você não deseja que o tempo de extração afete as decisões de política do ciclo de vida.

Quando uma função na lista de exclusão extrai uma imagem, o tempo de extração permanece inalterado. Qualquer outra função continua atualizando o tempo de pull (comportamento atual). Você pode configurar até 100 exclusões por conta.

Gerenciando exclusões de atualizações em tempo integral

Para gerenciar as exclusões de atualizações em tempo integral, você precisa das seguintes permissões do IAM:

- `ecr:CreatePullTimeUpdateExclusion`— Concede permissão para adicionar um ARN de função à lista de exclusão.
- `ecr:DeletePullTimeUpdateExclusion`— Concede permissão para remover um ARN de função da lista de exclusão.
- `ecr:ListPullTimeUpdateExclusions`— Concede permissão para listar todas as funções ARNs na lista de exclusão.

> **Note**
>
> Você não precisa de `iam:PassRole` permissão. O Amazon ECR não assume a função de realizar uma ação; ele usa apenas a configuração de exclusão ARNs para determinar se o tempo de extração da imagem deve ser atualizado.

Você pode gerenciar exclusões de atualizações em tempo integral usando o console Amazon ECR ou a CLI. AWS

Console de gerenciamento da AWS

Para gerenciar exclusões de atualizações em tempo integral ()Console de gerenciamento da AWS

1. Abra o console do Amazon ECR em registros https://console.aws.amazon.com/ecr/ privados/repositórios
2. Na barra de navegação, selecione a região.
3. No painel de navegação, escolha Registro privado, Recursos e configurações e, em seguida, escolha Exclusões de atualização pull-time.
4. Para adicionar uma exclusão, escolha Adicionar exclusão, insira o ARN da função e escolha Adicionar.
5. Para remover uma exclusão, selecione o ARN da função na lista e escolha Excluir.
6. Para ver todas as exclusões, a lista exibe todas as funções ARNs configuradas.

AWS CLI

Para criar uma exclusão de atualização em tempo integral

- Use o create-pull-time-update-exclusion comando para adicionar um ARN de função à lista de exclusão:

```
aws ecr create-pull-time-update-exclusion \
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

O comando retorna o ARN da função e o timestamp de criação:

```
{
   "roleArn": "arn:aws:iam::123456789012:role/scanner-role",
   "createdAt": 1745531331.0
}
```

Para excluir uma exclusão de atualização em tempo integral

- Use o delete-pull-time-update-exclusion comando para remover um ARN de função da lista de exclusão:

```
aws ecr delete-pull-time-update-exclusion \
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

O comando retorna o ARN da função que foi excluído:

```
{
  "roleArn": "arn:aws:iam::123456789012:role/scanner-role"
}
```

Para listar as exclusões de atualizações em tempo integral

1. Use o list-pull-time-update-exclusions comando para listar todas as funções ARNs na lista de exclusão:

   ```
   aws ecr list-pull-time-update-exclusions
   ```

   Se nenhuma exclusão for configurada, o comando retornará uma lista vazia:

   ```
   {
      "pullTimeUpdateExclusions": []
   }
   ```

   Se as exclusões forem configuradas, o comando retornará a lista de funções ARNs:

   ```
   {
      "pullTimeUpdateExclusions": [
           "arn:aws:iam::123456789012:role/security-role"
       ]
   }
   ```

2. Para paginar os resultados, use os parâmetros `--max-results` e`--next-token`:

   ```
   aws ecr list-pull-time-update-exclusions \
       --max-results 4
   ```

   O comando retorna até o número especificado de resultados e `nextToken` se houver mais resultados disponíveis:

```
{
   "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..."
}
```

Para recuperar a próxima página de resultados, use `nextToken` a resposta anterior:

```
aws ecr list-pull-time-update-exclusions \
    --max-results 4 \
    --next-token ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79...
```

Considerações sobre exclusões de atualizações em tempo integral

Considere o seguinte ao usar exclusões de atualização em tempo integral:

- O tamanho padrão da página para exclusões de anúncios é 100. Você pode usar paginação com parâmetros `maxResults` e. `nextToken`
- Somente funções do IAM válidas ARNs no formato ARN correto são aceitas.
- Se você tentar criar uma exclusão que já existe, você receberá um `ExclusionAlreadyExistsException` erro. Se você tentar excluir uma exclusão que não existe, você receberá uma `ExclusionNotFoundException` mensagem de erro.

Segurança no Amazon Elastic Container Registry

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem.

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Third-partyauditores testam e verificam regularmente a eficácia de nossa segurança como parte dos programas de AWS conformidade. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon ECR, consulte Serviços da AWS no escopo pelo programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon ECR. Os tópicos a seguir mostram como configurar o Amazon ECR para atender aos seus objetivos de segurança e conformidade. Você também aprende a usar outros AWS serviços que ajudam você a monitorar e proteger seus recursos do Amazon ECR.

Tópicos

Gerenciamento de Identidade e Acesso para o Amazon Elastic Container Registry

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para usar recursos do Amazon ECR. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas de identidade e acesso para o Amazon Elastic Container Registry).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como o Amazon Elastic Container Registry funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry)

Autenticação com identidades

A autenticação é como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Identity-based políticas

Identity-based políticas são documentos de políticas de permissões JSON que você anexa a uma identidade (usuário, grupo ou função). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Identity-based as políticas podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Resource-based políticas

Resource-based políticas são documentos de política JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Resource-based políticas são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de Controle de Serviços (SCPs): as SCPs especificam o número máximo de permissões para uma organização ou uma unidade organizacional no AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs): definem o número máximo de permissões disponíveis para recursos em suas contas. Consulte mais informações em Resource control policies (RCPs) no Guia do usuário do AWS Organizations .
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o Amazon Elastic Container Registry funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon ECR, você deve entender quais recursos do IAM estão disponíveis para uso com o Amazon ECR. Para obter uma visão de alto nível de como o Amazon ECR e outros AWS serviços funcionam com o IAM, consulte AWS Serviços que funcionam com o IAM no Guia do usuário do IAM.

Tópicos

- Políticas do Amazon ECR Identity-based

- Políticas baseadas em recurso do Amazon ECR
- Autorização baseada em tags do Amazon ECR
- Perfis do IAM no Amazon ECR

Políticas do Amazon ECR Identity-based

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. O Amazon ECR oferece suporte a ações, chaves de condição e recursos específicos. Para conhecer todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de política no Amazon ECR usam o seguinte prefixo antes da ação: `ecr:`. Por exemplo, para conceder a alguém permissão para criar um repositório do Amazon ECR com a operação de API `CreateRepository` do Amazon ECR, inclua a ação `ecr:CreateRepository` na política da pessoa. As instruções de política devem incluir um elemento `Action` ou `NotAction`. O Amazon ECR define seu próprio conjunto de ações que descrevem as tarefas que você pode executar com esse serviço.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "ecr:Describe*"
```

Para ver uma lista das ações do Amazon ECR, consulte Ações, recursos e chaves de condição do Amazon Elastic Container Registry no Manual do usuário do IAM.

Recursos

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Um recurso do repositório do Amazon ECR tem o seguinte ARN:

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

Para obter mais informações sobre o formato dos ARNs, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces.

Por exemplo, para especificar o repositório `my-repo` na região `us-east-1` em sua instrução, use o seguinte ARN:

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

Para especificar todos os repositórios que pertencem a uma conta específica, use o caractere curinga (*):

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

Para especificar vários recursos em uma única declaração, separe os ARNs com vírgulas.

```
"Resource": [
      "resource1",
      "resource2"
```

Para ver uma lista dos tipos de recursos do Amazon ECR e seus ARNs, consulte Recursos definidos pelo Amazon Elastic Container Registry no Manual do usuário do IAM. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Elastic Container Registry.

Chaves de condição

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

O Amazon ECR define seu próprio conjunto de chaves de condição e também oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição AWS globais, consulte Chaves de contexto de condição AWS global no Guia do usuário do IAM.

A maioria das ações do Amazon ECR oferecem suporte às chaves de condição `aws:ResourceTag` e `ecr:ResourceTag`. Para obter mais informações, consulte Usando o controle de Tag-Based acesso.

Para ver uma lista das chaves de condição do Amazon ECR, consulte Chaves de condição definidas pelo Amazon Elastic Container Registry noManual do usuário do IAM. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Container Registry.

Exemplos

Para ver exemplos de políticas baseadas em identidade do Amazon ECR, consulte Exemplos de Identity-based políticas do Amazon Elastic Container Registry.

Políticas baseadas em recurso do Amazon ECR

Resource-based políticas são documentos de política JSON que especificam quais ações um diretor específico pode realizar em um recurso do Amazon ECR e sob quais condições. O Amazon ECR oferece suporte a políticas de permissões baseadas em recursos para repositórios do Amazon ECR. Resource-based as políticas permitem que você conceda permissão de uso a outras contas por

recurso. Também é possível usar uma política baseada em recurso para permitir que um serviço da AWS acesse seus repositórios do Amazon ECR.

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recurso. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso estão em AWS contas diferentes, você também deve conceder permissão à entidade principal para acessar o recurso. Conceda permissão anexando uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, não serão necessárias permissões adicionais do repositório do Amazon ECR na política baseada em identidade. Para obter mais informações, consulte Como as funções do IAM diferem das Resource-based políticas no Guia do usuário do IAM.

O serviço Amazon ECR oferece suporte somente a um tipo de política baseada em recurso, denominada política de repositório, que é anexada a um repositório. Essa política define quais entidades principais (contas, usuários, funções e usuários federados) podem realizar ações no repositório. Para saber como anexar uma política baseada em recurso a um repositório, consulte Políticas de repositório privado no Amazon ECR.

> **Note**
>
> Em uma política de repositório do Amazon ECR, o elemento de política `Sid` aceita caracteres e espaçamento adicionais que as políticas do IAM não aceitam.

Exemplos

Para ver exemplos de políticas baseadas em recursos do Amazon ECR, consulte Exemplos de políticas de repositório privado no Amazon ECR.

Autorização baseada em tags do Amazon ECR

É possível anexar tags a recursos do Amazon ECR ou informar tags em uma solicitação para o Amazon ECR. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `ecr:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`. Para obter mais informações sobre recursos de marcação do Amazon ECR, consulte Marcar um repositório privado no Amazon ECR.

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Usando o controle de Tag-Based acesso.

Perfis do IAM no Amazon ECR

Uma função do IAM é uma entidade dentro da sua AWS conta que tem permissões específicas.

Usar credenciais temporárias com o Amazon ECR

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de AWS STS API, como AssumeRoleou GetFederationToken.

O Amazon ECR suporta o uso de credenciais temporárias.

Service-linked funções

Service-linked as funções permitem que os AWS serviços acessem recursos em outros serviços para concluir uma ação em seu nome. Service-linked as funções aparecem na sua conta do IAM e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados a serviço.

O Amazon ECR oferece suporte as funções vinculadas ao serviço. Para obter mais informações, consulte Uso de funções vinculadas ao serviço para o Amazon ECR.

Exemplos de Identity-based políticas do Amazon Elastic Container Registry

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do Amazon ECR. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte Criar políticas do IAM (no console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recurso definidos pelo Amazon ECR, inclusive o formato dos ARNs para cada um dos tipos de recurso, consulte Ações, recursos e chaves de condição do Amazon Elastic Container Registry na Referência de autorização do serviço.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Tópicos

Melhores práticas de políticas

Identity-based as políticas determinam se alguém pode criar, acessar ou excluir recursos do Amazon ECR em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar

a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.

- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon ECR

Para acessar o console da Amazon ECR, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon ECR em sua AWS conta. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Para garantir que essas entidades ainda possam usar o console do Amazon ECR, adicione a política `AmazonEC2ContainerRegistryReadOnly` AWS gerenciada às entidades. Para obter mais informações, consulte Adição de permissões a um usuário no Manual do usuário do IAM:

Para visualizar as permissões para esta política, consulte AmazonElasticContainerRegistryPublicReadOnly na Referência de políticas gerenciadas pela AWS .

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente às ações que correspondem à operação da API que você está tentando executar.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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": "*"
        }
    ]
}
```

Acesso a um repositório do Amazon ECR

Neste exemplo, você deseja conceder a um usuário da sua AWS conta acesso a um dos seus repositórios do Amazon ECR,. `my-repo` Você também quer permitir que o usuário envie, extraia e liste imagens.

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"
         }
      ]
}
```

Usando o controle de Tag-Based acesso

A ação de API `CreateRepository` do Amazon ECR permite especificar tags ao criar o repositório. Para obter mais informações, consulte Marcar um repositório privado no Amazon ECR.

Para permitir que os usuários marquem repositórios na criação, eles devem ter permissões para usar a ação que cria o recurso (por exemplo, `ecr:CreateRepository`). Se as tags forem especificadas na ação resource-creating, a Amazon executará autorização adicional na ação `ecr:CreateRepository` para verificar se os usuários têm permissões para criar tags.

É possível usar controle de acesso baseado em tags por meio de políticas do IAM. Veja os exemplos a seguir.

A política a seguir só permitiria que um usuário criasse ou marcasse um repositório como `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"
                }
            }
        }
    ]
}
```

A política a seguir permitiria que um usuário extraísse imagens de todos os repositórios, a menos que eles estivessem marcados como `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 políticas gerenciadas para o Amazon Elastic Container Registry

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para saber mais, consulte AWS Políticas gerenciadas pela no Guia do usuário do IAM.

O Amazon ECR fornece várias políticas gerenciadas que você pode anexar às identidades do IAM ou às instâncias do Amazon EC2. Essas políticas gerenciadas permitem habilitar diferentes níveis de controle sobre o acesso às operações de API e aos recursos do Amazon ECR. Para obter mais informações sobre cada operação de API mencionada nessas políticas, consulteAções na Referência da API do Amazon Elastic Container Registry.

Tópicos

- AmazonEC2ContainerRegistryFullAccess
- AmazonEC2ContainerRegistryPowerUser
- AmazonEC2ContainerRegistryPullOnly
- AmazonEC2ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- ECRTemplateServiceRolePolicy
- Atualizações do Amazon ECR para políticas AWS gerenciadas

AmazonEC2ContainerRegistryFullAccess

É possível anexar a política `AmazonEC2ContainerRegistryFullAccess` às suas identidades do IAM. Essa política concede acesso administrativo aos recursos do Amazon ECR e concede a uma identidade do IAM (como usuário, grupo ou função) acesso aos AWS serviços aos quais o Amazon ECR está integrado para usar todos os recursos do Amazon ECR. O uso dessa política permite o acesso a todos os atributos do Amazon ECR que estão disponíveis no Console de gerenciamento da AWS.

Para visualizar as permissões para esta política, consulte AmazonEC2ContainerRegistryFullAccess na Referência de políticas gerenciadas pela AWS .

AmazonEC2ContainerRegistryPowerUser

É possível anexar a política `AmazonEC2ContainerRegistryPowerUser` às suas identidades do IAM. Essa política concede permissões administrativas que permitem que os usuários do IAM leiam e gravem nos repositórios, mas não permitem que eles excluam repositórios ou alterem os documentos de política aplicados a eles.

Para visualizar as permissões para esta política, consulte AmazonEC2ContainerRegistryPowerUser na Referência de políticas gerenciadas pela AWS .

AmazonEC2ContainerRegistryPullOnly

É possível anexar a política `AmazonEC2ContainerRegistryPullOnly` às suas identidades do IAM. Essa política concede permissão para extrair imagens de contêineres do Amazon ECR. Se o registro estiver habilitado para cache de pull-through, ele também permitirá que as extrações importem uma imagem de um registro upstream.

Para visualizar as permissões para esta política, consulte AmazonEC2ContainerRegistryPullOnly na Referência de políticas gerenciadas pela AWS .

AmazonEC2ContainerRegistryReadOnly

É possível anexar a política `AmazonEC2ContainerRegistryReadOnly` às suas identidades do IAM. Esta política concede permissões de acesso somente para leitura ao Amazon ECR. Isso inclui a capacidade de listar repositórios e imagens dentro dos repositórios. Inclui também a capacidade de extrair imagens do Amazon ECR com a CLI do Docker.

Para visualizar as permissões para esta política, consulte AmazonEC2ContainerRegistryReadOnly na Referência de políticas gerenciadas pela AWS .

`AWSECRPullThroughCache_ServiceRolePolicy`

Não é possível anexar a política do IAM gerenciada `AWSECRPullThroughCache_ServiceRolePolicy` às suas entidades do IAM. Essa política é anexada a uma função vinculada a serviço que permite que o Amazon ECR envie imagens para seus repositórios por meio do fluxo de trabalho do cache de pull-through. Para obter mais informações, consulte Função vinculada ao serviço do Amazon ECR para cache de pull-through.

`ECRReplicationServiceRolePolicy`

Não é possível anexar a política do IAM gerenciada `ECRReplicationServiceRolePolicy` às suas entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon ECR realizar ações em seu nome. Para obter mais informações, consulte Uso de funções vinculadas ao serviço para o Amazon ECR.

`ECRTemplateServiceRolePolicy`

Não é possível anexar a política do IAM gerenciada `ECRTemplateServiceRolePolicy` às suas entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon ECR realizar ações em seu nome. Para obter mais informações, consulte Uso de funções vinculadas ao serviço para o Amazon ECR.

Atualizações do Amazon ECR para políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Amazon ECR desde o momento em que esse serviço começou a rastrear essas alterações. Para obter alertas automáticos sobre alterações feitas nesta página, inscreva-se no feed RSS na página Histórico de documentos do Amazon ECR.

Alteração	Descrição	Data
Função vinculada ao serviço do Amazon ECR para cache de pull-through: atualizar para uma política existente	O Amazon ECR adicionou novas permissões à política `AWSECRPullThroughCache_ServiceRolePolicy` . Essas permissões permitem que o Amazon ECR extraia imagens do registro privado do ECR. Isso é necessário ao usar uma regra de cache de pull-through para armazenar em cache imagens de outro registro privado do Amazon ECR.	12 de março de 2025
AmazonEC2ContainerRegistryPullOnly – Nova política	O Amazon ECR adicionou uma nova política que concede permissões somente de extração para o Amazon ECR.	10 de outubro de 2024
ECRTemplateServiceRolePolicy – Nova política	O Amazon ECR adicionou uma nova política. Esta	20 de junho de 2024

Alteração	Descrição	Data
	política está associada ao perfil vinculado ao serviço `ECRTemplateService RolePolicy` para o recurso de modelo de criação de repositório.	
AWSECRPullThroughCache_ ServiceRolePolicy — Atualização de uma política existente	O Amazon ECR adicionou novas permissões à política `AWSECRPullThroughC ache_ServiceRolePo licy` . Essas permissões permitem que o Amazon ECR recupere o conteúdo criptografado de um segredo do Secrets Manager. Isso é necessário ao usar uma regra de cache de pull-through para armazenar em cache imagens de um registro upstream que requer autenticação.	15 de novembro de 2023
AWSECRPullThroughCache_ ServiceRolePolicy — Nova política	O Amazon ECR adicionou uma nova política. Essa política está associada à função vinculada ao serviço `AWSServiceRoleForE CRPullThroughCache` para o recurso de cache de pull-through.	29 de novembro de 2021

Alteração	Descrição	Data
ECRReplicationServiceRolePolicy – Nova política	O Amazon ECR adicionou uma nova política. Essa política está associada à função vinculada ao serviço `AWSServiceRoleForECRReplication` para o recurso de replicação.	4 de dezembro de 2020
AmazonEC2ContainerRegistryFullAccess – atualização para uma política existente	O Amazon ECR adicionou novas permissões à política `AmazonEC2ContainerRegistryFullAccess`. Essas permissões permitem que os principais criem a função vinculada ao serviço do Amazon ECR.	4 de dezembro de 2020
AmazonEC2ContainerRegistryReadOnly – atualização para uma política existente	O Amazon ECR adicionou novas permissões à política `AmazonEC2ContainerRegistryReadOnly` que permite que os principais leiam políticas de ciclo de vida, listem tags e descrevam as descobertas de digitalização para as imagens.	10 de dezembro de 2019

Alteração	Descrição	Data
AmazonEC2ContainerRegistryPowerUser – atualização para uma política existente	O Amazon ECR adicionou novas permissões à política `AmazonEC2ContainerRegistryPowerUser`. Elas permitem que os principais leiam políticas de ciclo de vida, listem tags e descrevam as descobertas de digitalização para as imagens.	10 de dezembro de 2019
AmazonEC2ContainerRegistryFullAccess – atualização para uma política existente	O Amazon ECR adicionou novas permissões à política `AmazonEC2ContainerRegistryFullAccess`. Eles permitem que os diretores consultem eventos de gerenciamento ou eventos do AWS CloudTrail Insights que são capturados por CloudTrail.	10 de novembro de 2017
AmazonEC2ContainerRegistryReadOnly – atualização para uma política existente	O Amazon ECR adicionou novas permissões à política `AmazonEC2ContainerRegistryReadOnly`. Elas permitem que os principais descrevam imagens do Amazon ECR.	11 de outubro de 2016

Alteração	Descrição	Data
AmazonEC2ContainerRegistryPowerUser – atualização para uma política existente	O Amazon ECR adicionou novas permissões à política `AmazonEC2ContainerRegistryPowerUser` . Elas permitem que os principais descrevam imagens do Amazon ECR.	11 de outubro de 2016
AmazonEC2ContainerRegistryReadOnly – Nova política	O Amazon ECR adicionou uma nova política que concede permissões somente de leitura para o Amazon ECR. Essas permissões incluem a capacidade de listar repositórios e imagens nos repositórios. Incluem também a capacidade de extrair imagens do Amazon ECR com a CLI do Docker.	21 de dezembro de 2015
AmazonEC2ContainerRegistryPowerUser – Nova política	O Amazon ECR adicionou uma nova política que concede permissões administrativas para que os usuários leiam e gravem nos repositórios, mas não permitem que eles excluam repositórios ou alterem os documentos de política aplicados a eles.	21 de dezembro de 2015
AmazonEC2ContainerRegistryFullAccess – Nova política	O Amazon ECR adicionou uma nova política. Essa política concede ao acesso total ao Amazon ECR.	21 de dezembro de 2015

Alteração	Descrição	Data
O Amazon ECR passou a monitorar alterações	O Amazon ECR começou a monitorar as alterações nas políticas AWS gerenciadas.	24 de junho de 2021

Uso de funções vinculadas ao serviço para o Amazon ECR

O Amazon Elastic Container Registry (Amazon ECR) AWS Identity and Access Management usa funções vinculadas ao serviço (IAM) para fornecer as permissões necessárias para usar a replicação e aproveitar os recursos de cache. A função vinculada ao serviço é um tipo especial de função do IAM vinculada diretamente ao Amazon ECR. A função vinculada ao serviço é predefinida pelo Amazon ECR. Ele inclui todas as permissões que o serviço requer para oferecer suporte à replicação e recursos de cache para o seu registro privado. Após você configurar a replicação ou o cache de pull-through para o registro, uma função vinculada ao serviço é criada automaticamente em seu nome. Para obter mais informações, consulte Configurações de registro privado no Amazon ECR.

Uma função vinculada ao serviço facilita a configuração da replicação e do cache de pull-through com o Amazon ECR. Isso porque, assim, você não precisa adicionar manualmente todas as permissões necessárias. O Amazon ECR define as permissões de suas funções vinculadas ao serviço e, a não ser que definido de outra forma, somente o Amazon ECR pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões. A política de permissões não pode ser anexada a nenhuma outra entidade do IAM.

Você só poderá excluir a função vinculada ao serviço após desabilitar a replicação ou o cache de pull-through no seu registro. Isso garante que você não remova inadvertidamente as permissões que o Amazon ECR requer para esses recursos.

Para obter informações sobre outros produtos que oferecem suporte a funções vinculadas a serviços, consulte Serviços da AWS que funcionam com o IAM. Nessa página vinculada, procure os serviços que têm Sim na coluna Service-linked Função. Escolha um Yes (Sim) com um link para ver a documentação da função vinculada a serviço para esse serviço.

Tópicos

- Regiões suportadas para a funções vinculadas a serviço do Amazon ECR
- Função vinculada ao serviço do Amazon ECR para replicação
- Função vinculada ao serviço do Amazon ECR para cache de pull-through

- Perfil vinculado ao serviço do Amazon ECR para modelos de criação de repositório

Regiões suportadas para a funções vinculadas a serviço do Amazon ECR

O Amazon ECR oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço do Amazon ECR está disponível. Para obter mais informações sobre a disponibilidade regional do Amazon ECR, consulte Regiões e endpoints da AWS.

Função vinculada ao serviço do Amazon ECR para replicação

O Amazon ECR usa uma função vinculada ao serviço chamada AWSServiceRoleForECRReplicationque permite que o Amazon ECR replique imagens em várias contas.

Service-linked permissões de função para o Amazon ECR

A função AWSServiceRoleForECRReplication vinculada ao serviço confia nos seguintes serviços para assumir a função:

- `replication.ecr.amazonaws.com`

A política de permissões da função do `ECRReplicationServiceRolePolicy` a seguir permite que o Amazon ECR use as seguintes ações em todos os recursos:

JSON

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

> **Note**
>
> `ReplicateImage` é uma API interna que o Amazon ECR usa para replicação e não pode ser chamada diretamente.

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte Permissões de Service-Linked função no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon ECR

Você não precisa criar manualmente a função vinculada ao serviço Amazon ECR. Quando você define as configurações de replicação do seu registro na Console de gerenciamento da AWS, na ou na AWS API AWS CLI, o Amazon ECR cria a função vinculada ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você define as configurações de replicação para o registro, o Amazon ECR cria a função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon ECR

O Amazon ECR não permite a edição manual da função vinculada ao AWSServiceRoleForECRReplication serviço. Depois que você criar um perfil vinculado ao serviço, não poderá alterar o nome do perfil, pois várias entidades podem fazer referência ao perfil. No entanto, você poderá editar a descrição do perfil usando o IAM. Para saber mais, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Exclusão de função vinculada ao serviço para o Amazon ECR

Se você não precisar mais usar um atributo ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ou mantida ativamente. Contudo, você deve remover a configuração de replicação do seu registro em todas as regiões para poder excluir manualmente a função vinculada ao serviço.

> **Note**
>
> Se você tentar excluir recursos enquanto o serviço Amazon ECR ainda está usando as funções, sua ação de exclusão pode falhar. Se isso acontecer, aguarde alguns minutos e tente novamente.

Para excluir os recursos do Amazon ECR usados pelo AWSServiceRoleForECRReplication

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região na qual a configuração de replicação está definida.
3. No painel de navegação, escolha Private registry (Registro privado).
4. Na página Private registry (Registro privado), na seção Replicação configuration (Configuração de replicação), selecione Edit (Editar).
5. Para excluir todas as suas regras de replicação, escolha Delete all (Excluir tudo). Essa etapa requer confirmação.

Como excluir manualmente o perfil vinculado ao serviço usando o IAM

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSServiceRoleForECRReplicationvinculada ao serviço. Para obter mais informações, consulte Excluir uma Service-Linked função no Guia do usuário do IAM.

Função vinculada ao serviço do Amazon ECR para cache de pull-through

O Amazon ECR usa uma função vinculada ao serviço chamada AWSServiceRoleForECRPullThroughCacheque dá permissão para o Amazon ECR realizar ações em seu nome para concluir ações de pull through cache. Para obter mais informações sobre o cache de pull-through, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.

Service-linked permissões de função para o Amazon ECR

A função AWSServiceRoleForECRPullThroughCachevinculada ao serviço confia no serviço a seguir para assumir a função.

- `pullthroughcache.ecr.amazonaws.com`

Detalhes das permissões

A política de permissões da `AWSECRPullThroughCache_ServiceRolePolicy` está vinculada à função vinculada ao serviço. Essa política gerenciada concede permissão ao Amazon ECR para realizar as seguintes ações. Para obter mais informações, consulte `AWSECRPullThroughCache_ServiceRolePolicy`.

- `ecr` – Permite que o serviço Amazon ECR extraia e envie imagens para um repositório privado.
- `secretsmanager:GetSecretValue`— Permite que o serviço Amazon ECR recupere o conteúdo criptografado de um AWS Secrets Manager segredo. Isso é necessário ao usar uma regra de cache de pull-through para armazenar em cache imagens de um registro upstream que requer autenticação em seu registro privado. Essa permissão se aplica apenas a segredos com o prefixo de nome `ecr-pullthroughcache/`.

A política da `AWSECRPullThroughCache_ServiceRolePolicy` contém os elementos a seguir:

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}"
                }
            }
        }
    ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte as permissões de Service-linked função no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon ECR

Você não precisa criar manualmente a função vinculada ao serviço do Amazon ECR para cache de pull-through. Quando você cria uma regra de cache pull through para seu registro privado na Console de gerenciamento da AWS, na ou na AWS API AWS CLI, o Amazon ECR cria a função vinculada ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma regra de cache de pull-through para seu registro privado, o Amazon ECR cria a função vinculada ao serviço para você novamente, caso ela ainda não exista.

Editar uma função vinculada ao serviço do Amazon ECR

O Amazon ECR não permite a edição manual da função vinculada ao AWSServiceRoleForECRPullThroughCacheserviço. Após a criação da função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência à função. No entanto, você poderá editar a descrição do perfil usando o IAM. Para saber mais, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Exclusão de função vinculada ao serviço para o Amazon ECR

Se você não precisar mais usar um atributo ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ou mantida ativamente. Contudo, você deve remover as regras de cache de pull-through do seu registro em todas as regiões para poder excluir manualmente a função vinculada ao serviço.

> **Note**
>
> Se você tentar excluir recursos enquanto o serviço Amazon ECR ainda estiver usando as funções, sua ação de exclusão poderá falhar. Se isso acontecer, aguarde alguns minutos e tente novamente.

Para excluir os recursos do Amazon ECR usados pela função vinculada ao serviço AWSServiceRoleForECRPullThroughCache

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, selecione a região na qual suas regras de cache de pull-through são criadas.
3. No painel de navegação, escolha Private registry (Registro privado).
4. Na página Private registry (Registro privado), na seção Pull through cache configuration (Configuração de cache de pull-through), escolha Edit (Editar).
5. Para cada regra de cache de pull-through que você criou, selecione a regra e escolha Delete rule (Excluir regra).

Como excluir manualmente o perfil vinculado ao serviço usando o IAM

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSServiceRoleForECRPullThroughCachevinculada ao serviço. Para obter mais informações, consulte Excluir uma Service-Linked função no Guia do usuário do IAM.

Perfil vinculado ao serviço do Amazon ECR para modelos de criação de repositório

O Amazon ECR usa uma função vinculada ao serviço chamada AWSServiceRoleForECRTemplateque dá permissão para o Amazon ECR realizar ações em seu nome para concluir ações de modelo de criação de repositórios.

Service-linked permissões de função para o Amazon ECR

A função AWSServiceRoleForECRTemplatevinculada ao serviço confia no serviço a seguir para assumir a função.

- `ecr.amazonaws.com`

Detalhes das permissões

A política de permissões da `ECRTemplateServiceRolePolicy` está vinculada à função vinculada ao serviço. Essa política gerenciada concede permissão ao Amazon ECR para realizar ações de criação de repositórios em seu nome.

A política da `ECRTemplateServiceRolePolicy` contém os elementos a seguir:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
     "Sid": "CreateRepositoryWithTemplate",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte as permissões de Service-linked função no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon ECR

Você não precisa criar manualmente o perfil vinculado ao serviço do Amazon ECR para o modelo de criação de repositório. Quando você cria uma regra de modelo de criação de repositório para seu registro privado na Console de gerenciamento da AWS, na ou na AWS API AWS CLI, o Amazon ECR cria a função vinculada ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma regra de modelo de criação de repositório para seu registro privado, o Amazon ECR cria o perfil vinculado ao serviço para você novamente, caso ele já não exista.

Editar uma função vinculada ao serviço do Amazon ECR

O Amazon ECR não permite a edição manual da função vinculada ao AWSServiceRoleForECRTemplateserviço. Após a criação da função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência à função. No entanto, você poderá editar a descrição do perfil usando o IAM. Para saber mais, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Exclusão de função vinculada ao serviço para o Amazon ECR

Se você não precisar mais usar um atributo ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ou mantida ativamente. Contudo, você deve excluir as regras de criação de repositório do registro em todas as regiões antes de poder excluir manualmente o perfil vinculado ao serviço.

> **Note**
>
> Se você tentar excluir recursos enquanto o serviço Amazon ECR ainda estiver usando as funções, sua ação de exclusão poderá falhar. Se isso acontecer, aguarde alguns minutos e tente novamente.

Para excluir os recursos do Amazon ECR usados pela função vinculada ao serviço AWSServiceRoleForECRTemplate

1. Abra o console do Amazon ECR em https://console.aws.amazon.com/ecr/.
2. Na barra de navegação, escolha a região em que as regras de criação de repositórrio foram criadas.
3. No painel de navegação, escolha Private registry (Registro privado).
4. Na página Registro privado, na seção Modelos de criação de repositório, escolha Editar.
5. Para cada regra de criação de repositório que você criou, selecione a regra e escolha Excluir regra.

Como excluir manualmente o perfil vinculado ao serviço usando o IAM

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSServiceRoleForECRTemplatevinculada ao serviço. Para obter mais informações, consulte Excluir uma Service-Linked função no Guia do usuário do IAM.

Solução de problemas de identidade e acesso para o Amazon Elastic Container Registry

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você possa encontrar ao trabalhar com a Amazon ECR e o IAM.

Tópicos

Não tenho autorização para executar uma ação na Amazon ECR

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `ecr:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `ecr:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar o meu pedido: PassRole

Se receber uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, suas políticas devem ser atualizadas para permitir a transmissão de um perfil ao Amazon ECR.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação na Amazon ECR. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas de fora da minha Conta da AWS acessem meus recursos do Amazon ECR

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon ECR suporta a esses recursos, consulte Como o Amazon Elastic Container Registry funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Proteção de dados no Amazon ECR

O modelo de responsabilidade AWS compartilhada O modelo de se aplica à proteção de dados no Amazon Elastic Container Service. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Perguntas frequentes sobre privacidade de dados . Para obter informações sobre proteção de dados na Europa, consulte o Centro de Regulamentação Geral de Proteção de Dados (GDPR).

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Amazon ECS ou outros Serviços da AWS usando o console, a API ou os AWS SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de

formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Tópicos

Criptografia em repouso

> **Important**
>
> Dual-layer a criptografia do lado do servidor com AWS KMS (DSSE-KMS) só está disponível nas regiões. AWS GovCloud (US)

O Amazon ECR armazena imagens em buckets do Amazon S3 gerenciados pelo Amazon ECR. Por padrão, o Amazon ECR usa criptografia do lado do servidor com chaves de criptografia da Amazon, que criptografam seus dados em repouso S3-managed usando um algoritmo de criptografia. AES-256 Isso não requer nenhuma ação da sua parte e é oferecido sem custo adicional. Para obter mais informações, consulte Proteção de dados usando Server-Side criptografia com chaves de S3-Managed criptografia da Amazon (SSE-S3) no Guia do usuário do Amazon Simple Storage Service.

Para obter mais controle sobre a criptografia dos seus repositórios Amazon ECR, você pode usar a criptografia do lado do servidor com chaves KMS armazenadas em (). AWS Key Management Service AWS KMS Ao usar AWS KMS para criptografar seus dados, você pode usar o padrão Chave gerenciada pela AWS, que é gerenciado pelo Amazon ECR, ou especificar sua própria chave KMS (chamada de chave gerenciada pelo cliente). Para obter mais informações, consulte Proteção de dados usando Server-Side criptografia com chaves KMS armazenadas em AWS KMS (SSE-KMS) no Guia do usuário do Amazon Simple Storage Service.

Você pode optar por aplicar duas camadas de criptografia às suas imagens do Amazon ECR usando criptografia de duas camadas no lado do servidor com (). AWS KMS DSSE-KMS DSSE-KMSA opção é semelhante aSSE-KMS, mas aplica duas camadas individuais de criptografia em vez de uma camada. Para obter mais informações, consulte Usando criptografia de camada dupla no lado do servidor com chaves (). AWS KMS DSSE-KMS

Cada repositório do Amazon ECR tem uma configuração de criptografia, que é definida quando o repositório é criado. Você pode usar configurações de criptografia diferentes em cada repositório.

Para obter mais informações, consulte Criar um repositório privado do Amazon ECR para armazenar imagens.

Quando um repositório é criado com a AWS KMS criptografia ativada, uma chave KMS é usada para criptografar o conteúdo do repositório. Além disso, o Amazon ECR adiciona uma AWS KMS concessão à chave KMS com o repositório Amazon ECR como principal beneficiário.

A próxima seção fornece uma compreensão de alto nível de como o Amazon ECR é integrado com o AWS KMS para criptografar e descriptografar seus repositórios:

1. Ao criar um repositório, o Amazon ECR envia uma DescribeKeychamada para AWS KMS validar e recuperar o Amazon Resource Name (ARN) da chave KMS especificada na configuração de criptografia.
2. O Amazon ECR envia duas CreateGrantsolicitações AWS KMS para criar concessões na chave KMS para permitir que o Amazon ECR criptografe e descriptografe dados usando a chave de dados.
3. Ao enviar uma imagem, é feita uma GenerateDataKeysolicitação AWS KMS que especifica a chave KMS a ser usada para criptografar a camada e o manifesto da imagem.
4. AWS KMS gera uma nova chave de dados, a criptografa sob a chave KMS especificada e envia a chave de dados criptografada para ser armazenada com os metadados da camada de imagem e o manifesto da imagem.
5. Ao extrair uma imagem, é feita uma solicitação de Decrypt AWS KMS, especificando a chave de dados criptografada.
6. AWS KMS descriptografa a chave de dados criptografada e envia a chave de dados descriptografada para o Amazon S3.
7. A chave de dados é usada para descriptografar a camada de imagem antes que a camada de imagem seja extraída.
8. Quando um repositório é excluído, o Amazon ECR envia duas RetireGrantsolicitações AWS KMS para retirar as concessões criadas para o repositório.

Considerações

Os seguintes pontos devem ser considerados ao usar criptografia AWS KMS baseada (SSE-KMSouDSSE-KMS) com o Amazon ECR.

- Se você criar seu repositório Amazon ECR com criptografia KMS e não especificar uma chave KMS, o Amazon ECR usa um Chave gerenciada pela AWS com o alias por padrão. `aws/ecr` Essa

chave KMS é criada em sua conta na primeira vez que você cria um repositório com criptografia KMS habilitada.

- Não é possível alterar a configuração da criptografia do repositório após a sua criação.
- Quando você usa a criptografia KMS com sua própria chave KMS, a chave deve existir na mesma região que seu repositório.
- As concessões que o Amazon ECR cria em seu nome não devem ser revogadas. Se você revogar a concessão que dá permissão ao Amazon ECR para usar as AWS KMS chaves em sua conta, o Amazon ECR não poderá acessar esses dados, criptografar novas imagens enviadas ao repositório ou descriptografá-las quando forem retiradas. Quando você revoga uma concessão do Amazon ECR, a alteração ocorre imediatamente. Para revogar direitos de acesso, você deve excluir o repositório em vez de revogar a concessão. Quando um repositório é excluído, o Amazon ECR retira as concessões em seu nome.
- Há um custo associado ao uso de AWS KMS chaves. Para obter mais informações, consulte Preços do AWS Key Management Service.
- Há um custo associado ao uso da criptografia de camada dupla do servidor. Para obter mais informações, consulte a Definição de preço do Amazon ECR

Permissões obrigatórias do IAM

Ao criar ou excluir um repositório do Amazon ECR com criptografia no lado do servidor usando o AWS KMS, as permissões necessárias dependem da chave KMS específica que você está usando.

Permissões do IAM necessárias ao usar o Chave gerenciada pela AWS for Amazon ECR

Por padrão, quando a AWS KMS criptografia está habilitada para um repositório Amazon ECR, mas nenhuma chave KMS é especificada, a para Chave gerenciada pela AWS Amazon ECR é usada. Quando a chave KMS AWS gerenciada do Amazon ECR é usada para criptografar um repositório, qualquer diretor que tenha permissão para criar um repositório também pode ativar a criptografia no repositório. AWS KMS No entanto, o principal do IAM que exclui o repositório deve ter a permissão `kms:RetireGrant`. Isso permite a retirada das concessões que foram adicionadas à AWS KMS chave quando o repositório foi criado.

O exemplo a seguir da política do IAM pode ser adicionado como uma política em linha a um usuário para garantir que ele tenha as permissões mínimas necessárias para excluir um repositório que tenha a criptografia habilitada. A chave do KMS usada para criptografar o repositório pode ser especificada usando o parâmetro do recurso.

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"
        }
    ]
}
```

Permissões do IAM obrigatórias ao usar uma chave gerenciada pelo cliente

Ao criar um repositório com AWS KMS criptografia habilitada usando uma chave gerenciada pelo cliente, há permissões necessárias tanto para a política de chaves do KMS quanto para a política do IAM para o usuário ou função que está criando o repositório.

Ao criar sua própria chave KMS, você pode usar a política de chave padrão que o AWS KMS cria ou pode especificar o seu próprio valor. Para garantir que a chave gerenciada pelo cliente permaneça gerenciável pelo proprietário da conta, a política de chaves da chave KMS deve permitir todas as AWS KMS ações para o usuário raiz da conta. Permissões de escopo adicionais podem ser adicionadas à política de chave, mas pelo menos o usuário-raiz deve receber permissões para gerenciar a chave KMS. Para permitir que a chave KMS seja usada somente para solicitações originadas no Amazon ECR, você pode usar a chave de ViaService condição kms: com o valor. `ecr.<region>.amazonaws.com`

O exemplo de política de chaves a seguir dá à AWS conta (usuário raiz) que possui a chave KMS acesso total à chave KMS. Para obter mais informações sobre esse exemplo de política de chaves, consulte Permite acesso à AWS conta e ativa políticas do IAM no Guia do AWS Key Management Service desenvolvedor.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

O usuário do IAM, a função do IAM ou a AWS conta que está criando seus repositórios devem ter a `kms:DescribeKey` permissão `kms:CreateGrantkms:RetireGrant`, e, além das permissões necessárias do Amazon ECR.

> **Note**
>
> A permissão `kms:RetireGrant` deve ser adicionada à política do IAM do usuário ou função que cria o repositório. As permissões `kms:CreateGrant` e `kms:DescribeKey` podem ser adicionadas à política de chave para a chave KMS ou à política do IAM de usuário ou função que cria o repositório. Para obter mais informações sobre como AWS KMS as permissões funcionam, consulte Permissões de AWS KMS API: referência de ações e recursos no Guia do AWS Key Management Service desenvolvedor.

O exemplo a seguir de política do IAM pode ser adicionado como uma política em linha a um usuário para garantir que ele tenha as permissões mínimas necessárias para criar um repositório com criptografia habilitada e excluir o repositório quando não precisar mais dele. A AWS KMS key usada para criptografar o repositório pode ser especificada usando o parâmetro do recurso.

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"
        }
    ]
}
```

Permitir que um usuário liste chaves KMS no console ao criar um repositório

Ao usar o console do Amazon ECR para criar um repositório, você pode conceder permissões para que um usuário liste as chaves KMS gerenciadas pelo cliente na região quando habilitar a criptografia para o repositório. O exemplo de política do IAM a seguir mostra as permissões necessárias para listar suas chaves e aliases do KMS ao usar o console.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
```

```
  }
}
```

Monitorando a interação do Amazon ECR com AWS KMS

Você pode usar AWS CloudTrail para rastrear as solicitações que o Amazon ECR envia AWS KMS em seu nome. As entradas de registro no CloudTrail registro contêm uma chave de contexto de criptografia para torná-las mais facilmente identificáveis.

Contexto de criptografia do Amazon ECR

Um contexto de criptografia é um conjunto de pares de chave/valor que contém dados arbitrários não secretos. Quando você inclui um contexto de criptografia em uma solicitação para criptografar dados, vincula AWS KMS criptograficamente o contexto de criptografia aos dados criptografados. Para descriptografar os dados, é necessário passar o mesmo contexto de criptografia.

Em suas solicitações GenerateDataKeye Decrypt para, o AWS KMS Amazon ECR usa um contexto de criptografia com dois pares de nome e valor que identificam o repositório e o bucket do Amazon S3 que estão sendo usados. Isso é mostrado no exemplo a seguir. Os nomes não variam, mas os valores de contexto de criptografia combinados serão diferentes para cada valor.

```
"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"
}
```

Você pode usar o contexto de criptografia para identificar essas operações criptográficas em registros e registros de auditoria, como AWS CloudTrailAmazon CloudWatch Logs, e como condição para autorização em políticas e concessões.

O contexto de criptografia do Amazon ECR consiste em dois pares de nome e valor.

- aws:s3:arn – O par de nome e valor identifica o bucket. A chave é `aws:s3:arn`. O valor de nome do recurso da Amazon (ARN) do bucket do Amazon S3

```
"aws:s3:arn": "ARN of an Amazon S3 bucket"
```

Por exemplo, se o ARN de um bucket fosse `arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`, o contexto de criptografia incluiria o seguinte par.

```
"arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
```

- aws:ecr:arn – O segundo par de nome e valor identifica nome do recurso da Amazon (ARN) do repositório. A chave é `aws:ecr:arn`. O valor é o ARN do repositório.

  ```
  "aws:ecr:arn": "ARN of an Amazon ECR repository"
  ```

  Por exemplo, se o ARN do repositório fosse `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`, o contexto de criptografia incluiria o seguinte par.

  ```
  "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
  ```

Solução de problemas

Ao excluir um repositório do Amazon ECR com o console, se o repositório for excluído com sucesso, mas o Amazon ECR não conseguir retirar as concessões adicionadas à sua chave KMS para seu repositório, você receberá o seguinte erro.

```
The repository [{repository-name}] has been deleted successfully but the grants created
 by the kmsKey [{kms_key}] failed to be retired
```

Quando isso ocorrer, você mesmo poderá retirar as AWS KMS concessões do repositório.

Para retirar manualmente os AWS KMS subsídios de um repositório

1. Liste as concessões para a AWS KMS chave usada no repositório. O valor `key-id` é incluído no erro que você recebe do console. Você também pode usar o `list-keys` comando para listar as chaves KMS gerenciadas pelo cliente Chaves gerenciadas pela AWS e as chaves do KMS em uma região específica da sua conta.

```
aws kms list-grants \
     --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
     --region us-west-2
```

A saída inclui um `EncryptionContextSubset` com o nome do recurso da Amazon (ARN) do seu repositório. Isso pode ser usado para determinar qual é a concessão adicionada à chave que você deseja retirar. O valor `GrantId` será usado quando for retirada a concessão na próxima etapa.

2. Retire cada concessão da AWS KMS chave adicionada ao repositório. Substitua o valor de pelo *`GrantId`* ID da concessão da saída da etapa anterior.

```
aws kms retire-grant \
     --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
     --grant-id GrantId \
     --region us-west-2
```

Validação da conformidade do Amazon Elastic Container Registry

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a Documentação AWS de segurança.

Segurança de infraestrutura no Amazon Elastic Container Registry

Como um serviço gerenciado, o Amazon Elastic Container Registry é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores

práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar o Amazon ECR pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Suítes de criptografia com sigilo direto perfeito (PFS), como DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

É possível chamar essas operações de API de qualquer local da rede, mas o Amazon ECR não é compatível com políticas de acesso baseadas em recursos, que podem incluir restrições com base no endereço IP de origem. Também é possível usar políticas do Amazon ECR para controlar o acesso a partir de endpoints da Amazon Virtual Private Cloud (Amazon VPC) ou de VPCs específicos. Efetivamente, isso isola o acesso à rede a um determinado recurso do Amazon ECR somente da VPC específica dentro da rede. AWS Para obter mais informações, consulte Endpoints VPC da interface Amazon ECR ()AWS PrivateLink.

Endpoints VPC da interface Amazon ECR ()AWS PrivateLink

É possível melhorar a postura de segurança da sua VPC configurando o Amazon ECR para usar um endpoint de interface da VPC. Os VPC endpoints são alimentados por AWS PrivateLink uma tecnologia que permite acessar de forma privada as APIs do Amazon ECR por meio de endereços IP privados (IPv4 e IPv6). AWS PrivateLink restringe todo o tráfego de rede entre sua VPC e o Amazon ECR para a rede Amazon. Você não precisa de um gateway da Internet, de um dispositivo NAT ou de um gateway privado virtual.

Para obter mais informações sobre AWS PrivateLink VPC endpoints, consulte VPC Endpoints no Guia do usuário da Amazon VPC.

Considerações endpoints da VPC do Amazon ECR

Antes de configurar endpoints da VPC para o Amazon ECR, fique atento às seguintes considerações:

- Para permitir que as tarefas do Amazon ECS hospedadas nas instâncias do Amazon EC2 extraiam imagens privadas do Amazon ECR, crie endpoints da VPC da interface para o Amazon ECS. Para

obter mais informações, consulte Interface VPC Endpoints (AWS PrivateLink) no Amazon Elastic Container Service Developer Guide.

- As tarefas do Amazon ECS hospedadas no Fargate que extraem imagens do Amazon ECR podem restringir o acesso à VPC específica que as tarefas usam e ao endpoint da VPC que o serviço usa, adicionando chaves de condição à função do IAM para a tarefa. Para obter mais informações, consulte Permissões opcionais do IAM para tarefas do Fargate que extraem imagens do Amazon ECR por endpoints de interface no Guia do desenvolvedor do Amazon Elastic Container Service.
- O grupo de segurança anexado ao endpoint da VPC deve permitir conexões de entrada na porta 443 na sub-rede privada da VPC.
- Os endpoints VPC do Amazon ECR oferecem suporte à conectividade de pilha dupla (IPv4 e IPv6). Quando você cria um endpoint VPC de pilha dupla, ele pode lidar com o tráfego em endereços IP privados IPv4 e IPv6.
- Os VPC endpoints oferecem suporte aos repositórios públicos do Amazon ECR por meio do endpoint AWS API SDK no Leste dos EUA (Norte da Virgínia).
- Os VPC endpoints oferecem suporte somente ao DNS AWS fornecido por meio do Amazon Route 53. Se quiser usar seu próprio DNS, você pode usar o encaminhamento de DNS condicional. Para obter mais informações, consulte Conjuntos de opções de DHCP no Manual do usuário da Amazon VPC.
- Se os contêineres tiverem conexões existentes com o Amazon S3, as conexões poderão ser interrompidas brevemente quando você adicionar o endpoint do gateway do Amazon S3. Se você quiser evitar essa interrupção, crie uma VPC que usa o endpoint de gateway do Amazon S3 e migre o cluster do Amazon ECS e seus contêineres para a nova VPC.
- Quando uma imagem é extraída usando uma regra de cache de pull-through pela primeira vez, se você configurou o Amazon ECR para usar um endpoint de VPC de interface usando AWS PrivateLink , então é necessário criar uma sub-rede pública na mesma VPC, com um gateway NAT e, em seguida, rotear todo o tráfego de saída para a Internet de sua sub-rede privada para o gateway NAT para que a extração funcione. As extrações de imagem subsequentes não exigem isso. Para obter mais informações, consulte Cenário: Acessar a Internet de uma sub-rede privada no Guia do usuário da Amazon Virtual Private Cloud.
- Para cargas de trabalho que exigem módulos criptográficos validados pelo FIPS 140-3, o Amazon ECR oferece suporte a endpoints FIPS para endpoints VPC.

Considerações para imagens do Windows

As imagens baseadas no sistema operacional Windows incluem artefatos que são restringidos, pela licença, de serem distribuídos. Por padrão, quando você envia imagens do Windows para um repositório do Amazon ECR, as camadas que incluem esses artefatos não são enviadas, pois elas são consideradas camadas externas. Quando os artefatos são fornecidos pela Microsoft, as camadas externas são recuperadas da infraestrutura do Microsoft Azure. Por esse motivo, para permitir que seus contêineres extraiam essas camadas externas do Azure, etapas adicionais são necessárias além da criação dos endpoints da VPC.

É possível substituir esse comportamento ao enviar imagens do Windows ao Amazon ECR usando o sinalizador `--allow-nondistributable-artifacts` no daemon do Docker. Quando habilitado, esse sinalizador envia as camadas licenciadas para o Amazon ECR, o que permite que essas imagens sejam extraídas do Amazon ECR por meio do endpoint da VPC sem necessidade de acesso adicional ao Azure.

> **Important**
>
> Usar o sinalizador `--allow-nondistributable-artifacts` não exclui sua obrigação de cumprir os termos da licença de imagem base de contêiner do Windows. Você não pode postar conteúdo do Windows para redistribuição pública ou de terceiros. O uso dentro do seu próprio ambiente é permitido.

Para habilitar o uso desse sinalizador para a instalação do Docker, você deve modificar o arquivo de configuração do daemon do Docker que, dependendo da instalação do Docker, normalmente pode ser configurado no menu de configurações ou preferências na seção Engine do Docker ou editando a seção do arquivo `C:\ProgramData\docker\config\daemon.json` diretamente.

Veja a seguir um exemplo da configuração necessária: Substitua o valor pelo URI do repositório para o qual você está enviando imagens.

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
    ]
}
```

Depois de modificar o arquivo de configuração do daemon do Docker, você deve reiniciar o daemon do Docker antes de tentar enviar sua imagem. Confirme se o push funcionou verificando se a camada base foi enviada ao seu repositório.

> **Note**
>
> As camadas base para imagens do Windows são grandes. O tamanho da camada resulta em mais tempo de envio e custos adicionais de armazenamento no Amazon ECR para essas imagens. Por esses motivos, recomendamos usar essa opção somente quando for estritamente necessário reduzir os tempos de construção e os custos contínuos de armazenamento. Por exemplo, a imagem `mcr.microsoft.com/windows/servercore` tem aproximadamente 1,7 GiB de tamanho quando compactada no Amazon ECR.

Criação dos endpoints da VPC para o Amazon ECR

Para criar os endpoints da VPC para o serviço do Amazon ECR, use o procedimento de Criação de um endpoint de interface no Manual do usuário da Amazon VPC.

Os endpoints VPC do Amazon ECR oferecem suporte à conectividade de pilha dupla (IPv4 e IPv6). Quando você cria um endpoint VPC de pilha dupla, ele processa automaticamente o tráfego em endereços IP privados IPv4 e IPv6. O endpoint roteará o tráfego usando a versão IP apropriada com base na configuração de rede do seu cliente e nos recursos do endpoint.

Se você já tem IPv4-only VPC endpoints e deseja migrar para endpoints de pilha dupla, pode modificar seus endpoints existentes para oferecer suporte à conectividade de pilha dupla ou criar novos endpoints de pilha dupla. Ao criar ou modificar endpoints, certifique-se de que sua VPC e sub-redes sejam compatíveis com a versão IP que você deseja usar. Depois de criar endpoints de pilha dupla, os endpoints suportarão IPv4 e IPv6.

As tarefas do Amazon ECS hospedadas em instâncias do Amazon EC2 exigem endpoints do Amazon ECR e o endpoint do gateway do Amazon S3.

As Tarefas do Amazon ECS hospedadas no Fargate usando a versão `1.4.0` da plataforma ou posterior exigem endpoints da VPC do Amazon ECR e endpoints de gateway do Amazon S3.

As tarefas do Amazon ECS hospedadas no Fargate que usam a **`1.3.0`** versão da plataforma ou anterior exigem apenas o com.amazonaws. ***`region`***.ecr.dkr Endpoint Amazon ECR VPC e endpoints do gateway Amazon S3.

> **Note**
>
> A ordem em que os endpoints são criados não importa.

com.amazonaws. ***region***.ecr.dkr

Esse endpoint é usado para as APIs de registro do Docker. Os comandos de cliente do Docker, como `push` e `pull`, usam esse endpoint.

Ao criar esse endpoint, você deve habilitar um nome de host DNS privado. Para fazer isso, verifique se a opção Enable Private DNS Name (Habilitar nome DNS privado) está selecionada no console da Amazon VPC ao criar o endpoint da VPC.

Para conexões compatíveis com FIPS 140-3, use o nome do endpoint FIPS com.amazonaws. ***region***.ecr-fips.dkr

com.amazonaws. ***region***.ecr.api

> **Note**
>
> O especificado *region* representa o identificador de região para uma AWS região suportada pelo Amazon ECR, como `us-east-2` para a região Leste dos EUA (Ohio).

Para conexões compatíveis com FIPS 140-3, use os nomes dos endpoints FIPS: com.amazonaws. *region*.ecr-fips.dkr e com.amazonaws. ***region***.ecr-fips.api.

Esse endpoint é usado para chamadas à API do Amazon ECR. As opções da API, como `DescribeImages` e `CreateRepository` são enviadas para esse endpoint.

Quando esse endpoint é criado, você tem a opção de habilitar um nome de host DNS privado. Habilite essa configuração selecionando Habilitar nome DNS privado no console da VPC ao criar o VPC endpoint. Se você habilitar um nome de host DNS privado para o VPC endpoint, atualize seu SDK ou AWS CLI para a versão mais recente para que não seja necessário especificar uma URL de endpoint ao usar o SDK ou não. AWS CLI

Para conexões compatíveis com FIPS 140-3, use o nome do endpoint FIPS com.amazonaws. ***region***.ecr-fips.api.

Se você habilitar um nome de host DNS privado e estiver usando um SDK ou uma AWS CLI versão lançada antes de 24 de janeiro de 2019, deverá usar o `--endpoint-url` parâmetro para especificar os endpoints da interface. O exemplo a seguir mostra o formato do URL do endpoint.

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

Se você não habilitar um nome de host DNS privado para o VPC endpoint, deverá usar o parâmetro `--endpoint-url` especificando o ID do VPC endpoint para o endpoint de interface. O exemplo a seguir mostra o formato do URL do endpoint.

```
aws ecr create-repository --repository-name name --endpoint-url
 https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

Para conexões compatíveis com FIPS 140-3, use o URL do endpoint FIPS:

```
aws ecr create-repository --repository-name name --endpoint-url https://api.ecr-
fips.region.amazonaws.com
```

Criar o endpoint do gateway do Amazon S3

Para que as tarefas do Amazon ECS extraiam imagens privadas do Amazon ECR, crie um endpoint de gateway para o Amazon S3. O endpoint do gateway é necessário porque o usa o Amazon ECR usa o Amazon S3 para armazenar as camadas de imagem. Quando os contêineres baixam imagens do Amazon ECR, eles devem acessar o Amazon ECR para obter o manifesto da imagem e o Amazon S3 para baixar as camadas reais da imagem. Este é o nome do recurso da Amazon (ARN) do bucket do Amazon S3 que contém as camadas de cada imagem do Docker.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

> **Note**
>
> Se estiver criando um endpoint VPC de pilha dupla para o Amazon ECR, você também precisará criar um Amazon S3 Gateway ou um endpoint de interface do Amazon S3 de pilha dupla. Consulte a documentação do S3 para obter detalhes.

Usar o procedimento Criação de um endpoint de gateway no Manual do usuário da Amazon VPC para criar o seguinte endpoint de gateway do Amazon S3 para o Amazon ECR. Ao criar o endpoint, selecione as tabelas de rotas para sua VPC.

com.amazonaws. ***region***.s3

> O endpoint de gateway do Amazon S3 usa um documento de política do IAM para limitar o acesso ao serviço. A política de Acesso total pode ser usada, pois qualquer restrição colocada nas funções do IAM de sua tarefa ou outras políticas de usuário do IAM ainda serão aplicadas além dessa política. Se você quiser limitar o acesso ao bucket do Amazon S3 para as permissões mínimas exigidas para usar o Amazon ECR, consulte Permissões mínimas do bucket do Amazon S3 para o Amazon ECR.

Permissões mínimas do bucket do Amazon S3 para o Amazon ECR

O endpoint de gateway do Amazon S3 usa um documento de política do IAM para limitar o acesso ao serviço. Para conceder apenas as permissões mínimas do bucket do Amazon S3 para o Amazon ECR, restrinja o acesso ao bucket do Amazon S3 usado pelo Amazon ECR ao criar o documento de política do IAM para o endpoint.

A tabela a seguir descreve as permissões de política do bucket do Amazon S3 exigidas pelo Amazon ECR.

Permissão	Description
`arn:aws:s3:::prod-` *`region`*`-starport-layer-bucket/*`	Fornece acesso ao bucket do Amazon S3 que contém as camadas de cada imagem do Docker. Representa o identificador da região para uma região da AWS suportada pelo Amazon ECR, como `us-east-2` para a região Leste dos EUA (Ohio).

Exemplo

O exemplo a seguir ilustra como conceder acesso aos buckets do Amazon S3 exigidos para as operações do Amazon ECR.

```
{
```

```
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

Crie o endpoint do CloudWatch Logs

As tarefas do Amazon ECS que usam o tipo de execução Fargate que usam uma VPC sem um gateway de internet e que também usam **`awslogs`** o driver de log para enviar informações de log para o Logs exigem que CloudWatch você crie o com.amazonaws. ***`region`***.logs Interface VPC endpoint CloudWatch para Logs. Para obter mais informações, consulte Como usar CloudWatch registros com endpoints VPC de interface no Guia do usuário do Amazon CloudWatch Logs.

Criar uma política de endpoint para os endpoints da VPC do Amazon ECR

Uma política de endpoint da VPC é uma política de recursos do IAM que você anexa a um endpoint quando cria ou modifica o endpoint. Se você não anexar uma política ao criar um endpoint, AWS anexará uma política padrão que permita acesso total ao serviço. Uma política de endpoint não substitui políticas de usuário do ou políticas de serviço específicas. É uma política separada para controlar o acesso do endpoint ao serviço especificado. Políticas de endpoint devem ser gravadas em formato JSON. Para obter mais informações, consulte Controlar o acesso a serviços com VPC endpoints no Manual do usuário da Amazon VPC.

Recomendamos criar uma única política de recursos do IAM e anexá-la a ambos os endpoints da VPC do Amazon ECR.

A seguir temos um exemplo de uma política de endpoint para o Amazon ECR Essa política permite que uma função do IAM específica extraia imagens do Amazon ECR.

```
{
 "Statement": [{
  "Sid": "AllowPull",
  "Principal": {
```

```
    "AWS": "arn:aws:iam::1234567890:role/role_name"
   },
   "Action": [
    "ecr:BatchGetImage",
    "ecr:GetDownloadUrlForLayer",
                    "ecr:GetAuthorizationToken"
   ],
   "Effect": "Allow",
   "Resource": "*"
  }]
}
```

O exemplo de política de endpoint a seguir impede que um repositório especificado seja excluído.

```
{
 "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"
  }
 ]
}
```

O exemplo de política de endpoint a seguir combina os dois exemplos anteriores em uma única política.

```
{
 "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": "*"
  }
 ]
}
```

Para modificar a política endpoint da VPC para o Amazon ECR

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. No painel de navegação, escolha Endpoints.
3. Se você ainda não criou os endpoints da VPC para o Amazon ECR, consulte Criação dos endpoints da VPC para o Amazon ECR.
4. Selecione endpoint da VPC do Amazon ECR ao qual deseja adicionar uma política e escolha a guia Policy (Política) na parte inferior da tela.
5. Selecione Edit policy (Editar política) e faça as alterações na política.
6. Escolha Save (Salvar) para salvar a política.

Sub-redes compartilhadas

Você não pode criar, descrever, modificar ou excluir endpoints da VPC em sub-redes que são compartilhadas com você. No entanto, é possível usar os endpoints da VPC em sub-redes que são compartilhadas com você.

Cross-service prevenção delegada confusa

O problema do “confused deputy” é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executar a ação. Em AWS, a falsificação de identidade entre serviços pode resultar no problema confuso do deputado. Cross-service a representação pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em políticas de recursos para limitar as permissões que o Amazon ECR concede a outro serviço no recurso para o recurso. Use `aws:SourceArn` se quiser que apenas um recurso seja associado ao acesso entre serviços. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curinga (*) para as partes desconhecidas do ARN. Por exemplo, .`arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*`

Se o valor de `aws:SourceArn` não contiver o ID da conta, como um ARN de bucket do Amazon S3, você deverá usar ambas as chaves de contexto de condição global para limitar as permissões.

O valor de `aws:SourceArn` deve ser ResourceDescription.

O exemplo a seguir mostra como você pode usar as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto em uma política de repositório do Amazon ECR para permitir o AWS CodeBuild acesso às ações de API do Amazon ECR necessárias para a integração com esse serviço e, ao mesmo tempo, evitar o problema confuso do substituto.

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"
            }
         }
      }
   ]
}
```

Monitoramento do Amazon ECR

Você pode monitorar o uso da API do Amazon ECR com a Amazon CloudWatch, que coleta e processa dados brutos do Amazon ECR em métricas legíveis e quase em tempo real. Essas estatísticas são registradas por um período de duas semanas para que você possa acessar as informações do histórico e obter uma perspectiva melhor sobre o uso da API. Os dados métricos do Amazon ECR são enviados automaticamente CloudWatch em períodos de um minuto. Para obter mais informações sobre CloudWatch, consulte o Guia CloudWatch do usuário da Amazon.

O Amazon ECR fornece métricas com base no uso de API para ações de autorização, envio de imagem e extração de imagem.

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho do Amazon ECR e de suas AWS soluções. Recomendamos que você colete dados de monitoramento dos recursos que compõem sua AWS solução para poder depurar com mais facilidade uma falha de vários pontos, caso ocorra. No entanto, antes de começar a monitorar o Amazon ECR, é necessário criar um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer uma linha de base de performance normal do Amazon ECR no seu ambiente, medindo o performance em vários momentos e em diferentes condições de carga. À medida que você monitorar o Amazon ECR, armazene dados de monitoramento históricos para compará-los com os novos dados de performance, identificar padrões de performance normais e anomalias de performance, além de elaborar métodos para resolver problemas.

Tópicos

- Relatórios de uso do Amazon ECR
- Métricas do repositório do Amazon ECR
- Eventos do Amazon ECR e EventBridge
- Registrando ações do Amazon ECR com AWS CloudTrail

Visualizar as Service Quotas e definir alarmes

Você pode usar o CloudWatch console para visualizar suas cotas de serviço e ver como seu uso atual se compara às cotas de serviço. Também é possível definir alarmes para que você seja notificado ao se aproximar de uma cota.

Como visualizar uma cota de serviço e opcionalmente definir um alarme

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, selecione Métricas.
3. Na guia Todas as métricas, escolha Uso e depois Por recurso da AWS .

   A lista das métricas de uso da cota de serviço é exibida.
4. Marque a caixa de seleção ao lado de uma das métricas.

   O gráfico mostra seu uso atual desse AWS recurso.
5. Para adicionar a cota de serviço ao gráfico, faça o seguinte:

   a. Escolha a guia Graphed metrics (Métricas em gráfico).

   b. Selecione Math expression (Expressão matemática), Start with an empty expression (Começar com uma expressão vazia). Depois, na nova linha, em Details (Detalhes), insira **`SERVICE_QUOTA(m1)`**.

      Uma nova linha é adicionada ao gráfico, exibindo a cota de serviço do recurso representado na métrica.
6. Para ver o uso atual como uma porcentagem da cota, adicione uma nova expressão ou altere a expressão SERVICE_QUOTA atual. Para a nova expressão, use **`m1/60/SERVICE_QUOTA(m1)*100`**.
7. (Opcional) Para definir um alarme que notifique se você caso se aproxime da cota de serviço, faça o seguinte:

a. Na linha **`m1/60/SERVICE_QUOTA(m1)*100`**, em Actions (Ações), selecione o ícone de alarme. Ele se parece com um sino.

   A página de criação de alarmes é exibida.

b. Em Conditions (Condições), verifique se o Threshold type (Tipo de limite) é Static (Estático) e se Whenever Expression1 is (Sempre que a Expression1 for) esteja definido como Greater (Maior). Em than (que), insira **`80`**. Isso cria um alarme que entrará no estado ALARM (ALARME) quando seu uso exceder 80% da cota.

c. Escolha Próximo.

d. Na próxima página, selecione um tópico do Amazon SNS ou crie um. Esse tópico será notificado quando o alarme entrar no estado ALARM (ALARME). Escolha Próximo.

e. Na próxima página, insira um nome e uma descrição para o alarme e selecione Next (Próximo).

f. Selecione Criar alarme.

Métricas de uso do Amazon ECR

Você pode usar métricas de CloudWatch uso para dar visibilidade ao uso dos recursos da sua conta. Use essas métricas para visualizar seu uso atual do serviço em CloudWatch gráficos e painéis.

As métricas de uso do Amazon ECR correspondem às cotas AWS de serviço. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações sobre cotas de serviço do Amazon ECR, consulte Cotas de serviço do Amazon ECR.

O Amazon ECR publica as seguintes métricas no namespace `AWS/Usage`.

Métrica	Description
`CallCount`	O número de chamadas de ação de API da sua conta. Os recursos são definidos pelas dimensões associadas à métrica. A estatística mais útil para essa métrica é `SUM`, que representa a soma dos valores de todos os colaboradores durante o período definido.

Métrica	Description
`ResourceCount`	O número dos recursos especificado em sua conta. Os recursos são definidos pelas dimensões associadas à métrica. A estatística mais útil para essa métrica é `MAXIMUM` a que representa o número máximo de recursos usados durante o período de 5 minutos.

As dimensões a seguir são usadas para refinar as métricas de uso da API publicadas pelo Amazon ECR.

Dimensão	Description
`Service`	O nome do AWS serviço que contém o recurso. Para as métricas de uso do Amazon ECR, o valor dessa dimensão é `ECR`.
`Type`	O tipo de entidade que está sendo relatado. Atualmente, o único valor válido para as métricas de uso da API Amazon ECR é`API`.
`Resource`	O tipo de recurso que está em execução. No momento, o Amazon ECR retorna informações sobre o uso da API para as ações de API a seguir. • `GetAuthorizationToken` • `BatchCheckLayerAvailability` • `InitiateLayerUpload` • `UploadLayerPart` • `CompleteLayerUpload` • `PutImage` • `BatchGetImage` • `GetDownloadUrlForLayer`
`Class`	A classe do recurso sob acompanhamento. No momento, o Amazon ECR não usa a dimensão de classe.

As dimensões a seguir são usadas para refinar as métricas de uso de recursos publicadas pelo Amazon ECR.

Dimensão	Description
`Service`	O nome do AWS serviço que contém o recurso. Para as métricas de uso do Amazon ECR, o valor dessa dimensão é `ECR`.
`Type`	O tipo de entidade que está sendo relatado. Atualmente, o único valor válido para as métricas de uso de recursos do Amazon ECR é`RESOURCE`.
`Resource`	O tipo de recurso que está em execução. Atualmente, o Amazon ECR retorna informações sobre seu uso de recursos para as seguintes métricas. • `RepositoryCount` • `ImagesPerRepositoryCount`
`ResourceId`	O identificador do recurso que gerou o uso. Atualmente, só ResourceId é relevante para `ImagesPerRepositoryCount` e seu valor é formatado como “repository/your_repository_name. For example: "repository/my-repo” retorna o número de imagens no repositório com o nome “my-repo”.

Relatórios de uso do Amazon ECR

AWS fornece uma ferramenta de geração de relatórios gratuita chamada Cost Explorer, que permite analisar o custo e o uso dos recursos do Amazon ECR.

Use o Cost Explorer para visualizar gráficos de uso e de custos. É possível visualizar dados dos últimos 13 meses e prever o valor que você provavelmente gastará nos próximos três meses. É possível usar o Cost Explorer para ver padrões de gastos de recursos da AWS ao longo do tempo, identificar áreas que precisam de uma investigação mais profunda e ver tendências que é possível usar para entender seus custos. Também é possível especificar os períodos dos dados e visualizar os dados de tempo por dia ou por mês.

Os dados de medição nos Relatórios de uso e de custo mostram o uso em todos os repositórios do Amazon ECR. Para obter mais informações, consulte Marcar recursos para faturamento.

Para obter mais informações sobre a criação de um relatório de AWS custo e uso, consulte Relatório de AWS custo e uso no Guia AWS Billing do usuário.

Métricas do repositório do Amazon ECR

O Amazon ECR envia métricas de contagem de pull do repositório para a Amazon. CloudWatch Os dados métricos do Amazon ECR são enviados automaticamente CloudWatch em períodos de 1 minuto. Para obter mais informações sobre CloudWatch, consulte o Guia CloudWatch do usuário da Amazon.

Tópicos

Habilitando CloudWatch métricas

O Amazon ECR envia métricas de repositório automaticamente para todos os repositórios. Não é preciso realizar nenhuma etapa manual.

Métricas e dimensões disponíveis

As seções a seguir listam as métricas e dimensões que o Amazon ECR envia para a Amazon CloudWatch.

Métricas do Amazon ECR

O Amazon ECR fornece métricas para você monitorar seus repositórios. Você pode medir o número de solicitações pull.

O namespace `AWS/ECR` inclui as métricas a seguir.

`RepositoryPullCount`

O número total de solicitações pull das imagens no repositório.

Dimensões válidas: `RepositoryName`.

Estatísticas válidas: média, mínima, máxima, soma, contagem de exemplo. A estatística mais útil é Sum.

Unit: Integer.

Dimensões para métricas do Amazon ECR

As métricas do Amazon ECR usam o namespace `AWS/ECR` e fornecem métricas para as dimensões a seguir.

`RepositoryName`

Essa dimensão filtra os dados solicitados para todas as imagens do contêiner em um repositório especificado.

Visualizando métricas do Amazon ECR usando o console CloudWatch

Você pode visualizar as métricas do repositório Amazon ECR no CloudWatch console. O CloudWatch console fornece uma exibição refinada e personalizável de seus recursos. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.

Eventos do Amazon ECR e EventBridge

A Amazon EventBridge permite que você automatize seus AWS serviços e responda automaticamente a eventos do sistema, como problemas de disponibilidade de aplicativos ou alterações de recursos. Os eventos dos AWS serviços são entregues quase EventBridge em tempo real. É possível escrever regras simples para indicar quais eventos são do seu interesse e incluir ações automatizadas que deverão ser realizadas quando um evento corresponder à regra. Ações que podem ser automaticamente acionadas incluem:

- Adicionar eventos a grupos de CloudWatch registros no Logs
- Invocando uma função AWS Lambda
- Invocar o comando de execução do Amazon EC2
- Transmitir o evento Amazon Kinesis Data Streams
- Ativando uma máquina de AWS Step Functions estado
- Notificar um tópico do Amazon SNS ou uma fila do Amazon SQS

Para obter mais informações, consulte Getting Started with Amazon EventBridge no Guia EventBridge do usuário da Amazon.

Amostra de eventos do Amazon ECR

Veja a seguir exemplos de eventos do Amazon ECR. Os eventos são emitidos com base no melhor esforço.

Evento para um envio de imagem concluído

O evento a seguir é enviado quando cada envio de imagem é concluído. Para obter mais informações, consulte Envio por push de uma imagem do Docker para um repositório privado do 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"
    }
}
```

Evento para uma ação de cache de pull-through

Quando uma tentativa de ação de cache de pull-through é feita, o seguinte evento é enviado. Para obter mais informações, consulte Sincronizar um registro upstream com um registro privado do 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",
    }
}
```

Evento para uma verificação de imagem concluída (verificação básica)

Quando a verificação básica está habilitada para seu registro, o evento a seguir é enviado quando cada verificação de imagem é concluída. O parâmetro `finding-severity-counts` só retornará um valor de um nível de gravidade se existir algum. Por exemplo, se a imagem não contiver descobertas no nível `CRITICAL`, não será retornada uma contagem crítica. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do sistema operacional no Amazon ECR.

> **Note**
>
> Para obter detalhes sobre eventos que o Amazon Inspector emite quando a verificação avançada está habilitada, consulteEventBridge eventos enviados para digitalização aprimorada no 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": []
    }
}
```

Evento para uma notificação de alteração em um recurso com verificação avançada habilitada (verificação avançada)

Quando a verificação avançada é habilitada para seu registro, o evento a seguir é enviado pelo Amazon ECR quando há uma alteração em um recurso que tem a verificação avançada habilitada. Isso inclui novos repositórios sendo criados, a frequência de verificação de um repositório sendo alterada ou quando as imagens são criadas ou excluídas em repositórios com a verificação avançada ativada. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no 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"
 }
}
```

Evento para uma exclusão de imagem

O evento a seguir é enviado quando uma imagem é excluída. Para obter mais informações, consulte Excluir uma imagem no 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"
    }
}
```

Evento para uma ação de arquivamento de imagens

O evento a seguir é enviado quando uma imagem é arquivada. O `target-storage-class` campo será definido como`ARCHIVE`. O evento inclui os tipos de mídia de manifesto e artefato para identificar o tipo de conteúdo que está sendo arquivado.

```
{
    "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"
    }
}
```

Evento para uma ação de restauração de imagem

O evento a seguir é enviado quando uma imagem arquivada é restaurada. O `target-storage-class` campo será definido como`STANDARD`. O evento inclui um `last-activated-at` campo mostrando quando a imagem foi restaurada pela última vez.

```
{
    "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"
    }
}
```

Evento para uma ação de restauração do referenciador

O evento a seguir é enviado quando um referenciador arquivado (artefato de referência, como um SBOM, assinatura ou atestado) é restaurado. Observe que `detail-type` é `ECR Referrer Action` para diferenciá-lo das ações regulares de imagem. Os `artifact-media-type` campos `manifest-media-type` e identificam o tipo específico de referenciador que está sendo restaurado. Neste exemplo, um artefato SBOM está sendo restaurado.

```
{
    "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"
    }
}
```

Evento para uma replicação de imagem concluída

O evento a seguir é enviado quando cada verificação de imagem é concluída. Para obter mais informações, consulte Replicação de imagem privada no 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"
  }
}
```

Evento para uma replicação de imagem com falha

O evento a seguir é enviado quando a replicação de imagem falha. O campo `result` conterá `FAILED` e informações adicionais sobre erros poderão ser incluídas nos detalhes do evento.

```
{
  "version": "0",
  "id": "d9c244c2-7130-ff84-f3b2-5g577c9cb000",
  "detail-type": "ECR Replication Action",
  "source": "aws.ecr",
```

```
  "account": "123456789012",
  "time": "2024-05-08T20:45:12Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecr:us-east-1:123456789012:repository/my-app"
  ],
  "detail": {
    "result": "FAILED",
    "repository-name": "my-app",
    "image-digest":
 "sha256:8g6c3751gf7gc5g47603ege4511d5a80ead5g90f5893543f1489bde2345",
    "source-account": "123456789012",
    "action-type": "REPLICATE",
    "source-region": "us-west-2",
    "image-tag": "latest"
  }
}
```

Registrando ações do Amazon ECR com AWS CloudTrail

O Amazon ECR é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um AWS serviço no Amazon ECR. CloudTrail captura as seguintes ações do Amazon ECR como eventos:

- Todas as chamadas de API, incluindo chamadas do console do Amazon ECR
- Todas as ações tomadas devido às configurações de criptografia em seus repositórios
- Todas as ações tomadas devido às regras de política de ciclo de vida, incluindo ações bem-sucedidas e malsucedidas

> **Important**
>
> Devido às limitações de tamanho de CloudTrail eventos individuais, para ações de política de ciclo de vida em que 10 ou mais imagens expiram, o Amazon ECR envia vários eventos para. CloudTrail Além disso, o Amazon ECR inclui no máximo 100 etiquetas por imagem.

Quando uma trilha é criada, você pode habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o Amazon ECR. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando

essas informações, é possível determinar a solicitação feita ao Amazon ECR, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para obter mais informações, consulte o Manual do usuário do AWS CloudTrail.

Informações do Amazon ECR em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Amazon ECR, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo de eventos em sua AWS conta, incluindo eventos para o Amazon ECR, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Ao criar uma trilha no console, você pode aplicá-la a uma única região ou a todas as regiões. A trilha registra eventos na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- Criando uma trilha para sua AWS conta
- AWS integrações de serviços com registros CloudTrail
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e Recebendo arquivos de CloudTrail log de várias contas

Todas as ações da API do Amazon ECR são registradas CloudTrail e documentadas na Amazon Elastic Container Registry API Reference. Quando você executa tarefas comuns, as seções são geradas nos arquivos de CloudTrail log para cada ação da API que faz parte dessa tarefa. Por exemplo, quando você cria um repositório,`GetAuthorizationToken`, `CreateRepository` e `SetRepositoryPolicy` seções são geradas nos arquivos de CloudTrail log. Quando você envia uma imagem para um repositório,,,, `InitiateLayerUploadUploadLayerPart`, e `CompleteLayerUploadPutImage`, se a montagem de blob estiver ativada, `MountLayer` seções são geradas. Quando você extrai uma imagem, são geradas as seções `GetDownloadUrlForLayer` e `BatchGetImage`. Quando você arquiva ou restaura, uma `UpdateImageStorageClass` seção de imagem é gerada. Quando OCI os clientes que oferecem

suporte à OCI 1.1 especificação buscam a lista de referenciadores ou artefatos de referência de uma imagem usando a API Referrers, um evento é emitido. `ListImageReferrers` CloudTrail Para ver exemplos dessas tarefas comuns, consulte CloudTrail exemplos de entrada de registro.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário raiz ou do
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado
- Se a solicitação foi feita por outro AWS serviço

Para obter mais informações, consulte Elemento do CloudTrail `userIdentity`.

Noções básicas sobre entradas do arquivo de log do Amazon ECR

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e outras informações. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

CloudTrail exemplos de entrada de registro

Veja a seguir exemplos de entrada de CloudTrail registro para algumas tarefas comuns do Amazon ECR.

Estes exemplos foram formatados para obter melhor legibilidade. Em um arquivo de CloudTrail log, todas as entradas e eventos são concatenados em uma única linha. Além disso, este exemplo foi limitado a uma única entrada do Amazon ECR. Em um arquivo de CloudTrail log real, você vê entradas e eventos de vários AWS serviços.

> **Important**
>
> A origem IPAddress é o endereço IP do qual a solicitação foi feita. Para ações originadas do console de serviço, o endereço informado é do recurso subjacente, não do servidor Web do console. Para serviços em AWS, somente o nome DNS é exibido. Ainda avaliamos a

autenticação com o IP de origem do cliente, mesmo que esteja editado para o nome DNS do serviço da AWS .

Tópicos

Exemplo: criar ação de repositório

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a `CreateRepository` ação.


```
{
    "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"
}
```

Exemplo: ação de API **`CreateGrant`** do AWS KMS ao criar um repositório do Amazon ECR

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a AWS KMS `CreateGrant` ação ao criar um repositório Amazon ECR com a criptografia KMS ativada. Para cada repositório criado com a criptografia KMS ativada, você deverá ver duas entradas de `CreateGrant` registro. 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"
}
```

Exemplo: ação de envio de imagem

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra um push de imagem que usa a `PutImage` ação.

> **Note**
>
> Ao enviar uma imagem, você também verá `InitiateLayerUploadUploadLayerPart`, e `CompleteLayerUpload` referências nos CloudTrail registros.

```
{
    "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"
}
```

Exemplo: ação de extração de imagem

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra uma extração de imagem que usa a `BatchGetImage` ação.

> **Note**
>
> Ao extrair uma imagem, se você ainda não tiver a imagem armazenada localmente, também serão exibidas as referências `GetDownloadUrlForLayer` nos logs do CloudTrail .

```
{
```

```
    "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"
}
```

Exemplo: ação da política de ciclo de vida da imagem

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra quando uma imagem expira devido a uma regra de política de ciclo de vida. Esse tipo de evento pode ser localizado filtrando `PolicyExecutionEvent` para o campo de nome do evento.

Quando você testa uma prévia da política de ciclo de vida, o Amazon ECR gera uma entrada de CloudTrail registro com o campo do nome do evento de`DryRunEvent`, com exatamente a mesma estrutura do. `PolicyExecutionEvent` Ao alterar o nome do evento para `DryRunEvent`, você pode filtrar os eventos de simulação.

> **Important**
>
> Devido às limitações de tamanho de CloudTrail eventos individuais, para ações de política de ciclo de vida em que 10 ou mais imagens expiram, o Amazon ECR envia vários eventos para. CloudTrail Além disso, o Amazon ECR inclui no máximo 100 etiquetas por imagem.

```
{
    "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]"
            }
        ]
    }
}
```

Exemplo: ação de arquivamento de imagens

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra uma imagem sendo arquivada usando a `UpdateImageStorageClass` ação com `targetStorageClass` definido como. `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"
}
```

Exemplo: ação de restauração de imagem

Os exemplos a seguir mostram entradas de CloudTrail registro que demonstram uma imagem sendo restaurada. Quando você restaura uma imagem arquivada, dois eventos são gerados:

1. Um evento de chamada de API quando a restauração é iniciada
2. Um evento de serviço quando a operação de restauração assíncrona é concluída

Evento de chamada de API (início da restauração)

O exemplo a seguir mostra a chamada inicial da API para restaurar uma imagem usando a `UpdateImageStorageClass` ação com `targetStorageClass` definido como`STANDARD`. A resposta mostra o status da imagem como`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"
}
```

Evento de serviço (conclusão da restauração)

O exemplo a seguir mostra o evento de serviço gerado quando a operação de restauração assíncrona é concluída. Esse tipo de evento pode ser localizado filtrando `ImageActivationEvent` para o campo de nome do evento. A `serviceEventDetails` seção contém o resultado da restauração e o status final da imagem.

```
{
    "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"
}
```

Exemplo: ação de referência de imagem

O exemplo a seguir mostra uma entrada de AWS CloudTrail registro que demonstra quando um cliente OCI 1.1 compatível busca uma lista de referenciadores ou artefatos de referência para uma imagem usando a 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"
}
```

Usando o Amazon ECR com um SDK AWS

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK para Go	AWS SDK para Go exemplos de código
AWS SDK para Java	AWS SDK para Java exemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK para .NET	AWS SDK para .NET exemplos de código
AWS SDK para PHP	AWS SDK para PHP exemplos de código
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell exemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP exemplos de código
AWS SDK para Swift	AWS SDK para Swift exemplos de código

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

Exemplos de código para Amazon ECR usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon ECR com um kit de desenvolvimento AWS de software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código

- Exemplos básicos do Amazon ECR usando AWS SDKs
  - Hello Amazon ECR
  - Aprenda as noções básicas do Amazon ECR com um SDK AWS
  - Ações para o Amazon ECR usando AWS SDKs
    - Use CreateRepository com um AWS SDK ou CLI
    - Use DeleteRepository com um AWS SDK ou CLI
    - Use DescribeImages com um AWS SDK ou CLI
    - Use DescribeRepositories com um AWS SDK ou CLI
    - Use GetAuthorizationToken com um AWS SDK ou CLI
    - Use GetRepositoryPolicy com um AWS SDK ou CLI
    - Use ListImages com um AWS SDK ou CLI
    - Use PushImageCmd com um AWS SDK
    - Use PutLifeCyclePolicy com um AWS SDK ou CLI
    - Use SetRepositoryPolicy com um AWS SDK ou CLI
    - Use StartLifecyclePolicyPreview com um AWS SDK ou CLI

Exemplos básicos do Amazon ECR usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Elastic Container Registry com SDKs da AWS .

Exemplos

Hello Amazon ECR

Os exemplos de código a seguir mostram como começar a usar o Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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()));
    }
}
```

- Consulte detalhes da API em listImages na Referência de APIs do AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- Consulte detalhes da API em listImages na Referência de APIs do AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
```

- Consulte detalhes da API em listImages na Referência de API do AWS SDK para Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Aprenda as noções básicas do Amazon ECR com um SDK AWS

Os exemplos de código a seguir mostram como:

- Crie um repositório do Amazon ECR.
- Defina políticas de repositório.
- Recupere URIs de repositórios.
- Obtenha tokens de autorização do Amazon ECR.
- Defina políticas de ciclo de vida para repositórios do Amazon ECR.
- Envie por push uma imagem do Docker para um repositório do Amazon ECR.
- Verifique a existência de uma imagem em um repositório do Amazon ECR.
- Liste os repositórios do Amazon ECR da conta e verifique os detalhes sobre eles.
- Exclua repositórios do Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do 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.");
           }
       }
   }
}
```

Uma classe de wrapper para os métodos do SDK do Amazon ECR.

```
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;
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do 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.")
        }
    }
}
```

Uma classe de wrapper para os métodos do SDK do Amazon ECR.

```
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)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
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)
```

Classe ECRWrapper que encapsula ações do 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
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para o Amazon ECR usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon ECR com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API Amazon ECR e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para o Amazon ECR usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do Amazon Elastic Container Registry.

Exemplos

- Use CreateRepository com um AWS SDK ou CLI
- Use DeleteRepository com um AWS SDK ou CLI
- Use DescribeImages com um AWS SDK ou CLI
- Use DescribeRepositories com um AWS SDK ou CLI
- Use GetAuthorizationToken com um AWS SDK ou CLI
- Use GetRepositoryPolicy com um AWS SDK ou CLI
- Use ListImages com um AWS SDK ou CLI
- Use PushImageCmd com um AWS SDK
- Use PutLifeCyclePolicy com um AWS SDK ou CLI
- Use SetRepositoryPolicy com um AWS SDK ou CLI
- Use StartLifecyclePolicyPreview com um AWS SDK ou CLI

Use **`CreateRepository`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateRepository`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do Amazon ECR

CLI

AWS CLI

Exemplo 1: criar um repositório

O exemplo `create-repository` a seguir cria um repositório dentro do namespace especificado no registro padrão de uma conta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Para obter mais informações, consulte Creating a repository no Guia do usuário do Amazon ECR.

Exemplo 2: criar um repositório configurado com imutabilidade da tag de imagem

O exemplo `create-repository` a seguir cria um repositório configurado para imutabilidade de tags no registro padrão de uma conta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Para obter mais informações, consulte Mutabilidade da tag de imagem no Guia do usuário do Amazon ECR.

Exemplo 3: criar um repositório configurado com uma configuração de digitalização

O exemplo de `create-repository` a seguir cria um repositório configurado para realizar uma verificação de vulnerabilidade no envio de imagens por push no registro padrão de uma conta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Para obter mais informações, consulte Verificação de Imagens no Guia do Usuário do Amazon ECR.

- Para obter detalhes da API, consulte CreateRepositoryem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateRepositorya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * 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
       }
   }
```

- Para obter detalhes da API, consulte a CreateRepositoryreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a CreateRepositoryReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a CreateRepositoryreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteRepository`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteRepository`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do Amazon ECR

CLI

AWS CLI

Para excluir um repositório

A força de comando do exemplo de `delete-repository` a seguir exclui o repositório especificado no registro padrão de uma conta. O sinalizador `--force` será obrigatório se o repositório contiver imagens.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
    }
```

```
}
```

Para obter mais informações, consulte Excluir um repositório no Guia do usuário do Amazon ECR.

- Para obter detalhes da API, consulte DeleteRepositoryem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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();
    }
```

- Para obter detalhes da API, consulte DeleteRepositorya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples 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")
        }
    }
```

- Para obter detalhes da API, consulte a DeleteRepositoryreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DeleteRepositoryReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a DeleteRepositoryreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeImages`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeImages`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para descrever uma imagem em um repositório

O exemplo de `describe-images` a seguir exibe os detalhes sobre uma imagem no repositório `cluster-autoscaler` com a tag `v1.13.6`.

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Saída:

```
{
```

```
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeImagesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * 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();
    }
```

- Para obter detalhes da API, consulte DescribeImagesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a DescribeImagesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DescribeImagesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a DescribeImagesreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeRepositories`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeRepositories`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como descrever os repositórios em um registro

Este exemplo descreve os repositórios no registro padrão de uma conta.

Comando:

```
aws ecr describe-repositories
```

Saída:

```
{
    "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"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeRepositoriesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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();
    }
```

- Para obter detalhes da API, consulte DescribeRepositoriesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * 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 ""
            }
        }
    }
```

- Para obter detalhes da API, consulte a DescribeRepositoriesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DescribeRepositoriesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
}
```

- Para obter detalhes da API, consulte a DescribeRepositoriesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a DescribeRepositoriesreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetAuthorizationToken`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAuthorizationToken`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para obter um token de autorização para o registro padrão

O comando do exemplo de `get-authorization-token` a seguir obtém um token de autorização para o registro padrão.

```
aws ecr get-authorization-token
```

Saída:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetAuthorizationTokenem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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();
    }
```

- Para obter detalhes da API, consulte GetAuthorizationTokena Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * 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.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a GetAuthorizationTokenreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a GetAuthorizationTokenReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a GetAuthorizationTokenreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **GetRepositoryPolicy** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetRepositoryPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para recuperar a política de repositório de um repositório

O exemplo de `get-repository-policy` a seguir exibe os detalhes sobre a política de repositório para o repositório `cluster-autoscaler`.

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Saída:

```
{
    "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}"
}
```

- Para obter detalhes da API, consulte GetRepositoryPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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;
    }
```

- Para obter detalhes da API, consulte GetRepositoryPolicya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * 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
        }
```

```
    }
```

- Para obter detalhes da API, consulte a GetRepositoryPolicyreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a GetRepositoryPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a GetRepositoryPolicyreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListImages`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListImages`.

CLI

AWS CLI

Para listar as imagens em um repositório

O exemplo de `list-images` a seguir exibe uma lista das imagens presentes no repositório `cluster-autoscaler`.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Saída:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
```

```
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListImagesem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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(())
}
```

- Para obter detalhes da API, consulte a ListImagesreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **PushImageCmd** com um AWS SDK

Os exemplos de código a seguir mostram como usar o PushImageCmd.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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();
    }
```

- Para obter detalhes da API, consulte PushImageCmda Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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)
            }
        }
    }
```

- Para obter detalhes da API, consulte a PushImageCmdreferência da API AWS SDK for Kotlin.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`PutLifeCyclePolicy`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutLifeCyclePolicy`.

CLI

AWS CLI

Para criar uma política de ciclo de vida

O exemplo `put-lifecycle-policy` a seguir cria uma política de ciclo de vida para o repositório especificado no registro padrão de uma conta.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
```

```
    --lifecycle-policy-text "file://policy.json"
```

Conteúdo de `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
               "type": "expire"
           }
       }
   ]
}
```

Saída:

```
{
   "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\"}}]}"
}
```

Para obter mais informações, consulte Políticas de ciclo de vida no Guia do usuário do Amazon ECR.

- Para obter detalhes da API, consulte PutLifeCyclePolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

Exemplo que coloca uma política de data de validade.

```
    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)
        )
```

- Para obter detalhes da API, consulte a PutLifeCyclePolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.
```

- Para obter detalhes da API, consulte a PutLifeCyclePolicyreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SetRepositoryPolicy`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetRepositoryPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para definir a política de repositório para um repositório

O exemplo `set-repository-policy` a seguir anexa uma política de repositório contida em um arquivo ao repositório `cluster-autoscaler`.

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Conteúdo de `my-policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Saída:

```
{
    "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}"
}
```

- Para obter detalhes da API, consulte SetRepositoryPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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();
   }
```

- Para obter detalhes da API, consulte SetRepositoryPolicya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * 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.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a SetRepositoryPolicyreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

Exemplo que concede acesso de download a um perfil do 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)
        )
```

- Para obter detalhes da API, consulte a SetRepositoryPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.
```

- Para obter detalhes da API, consulte a SetRepositoryPolicyreferência da API AWS SDK for SAP ABAP.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`StartLifecyclePolicyPreview`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartLifecyclePolicyPreview`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar uma pré-visualização da política de ciclo de vida

O exemplo de `start-lifecycle-policy-preview` a seguir cria uma pré-visualização prévia da política de ciclo de vida definida por um arquivo JSON para o repositório especificado.

```
aws ecr start-lifecycle-policy-preview \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Conteúdo de `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
```

```
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Saída:

```
{
    "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"
}
```

- Para obter detalhes da API, consulte StartLifecyclePolicyPreviewem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * 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();
    }
```

- Para obter detalhes da API, consulte StartLifecyclePolicyPreviewa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * 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.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a StartLifecyclePolicyPreviewreferência da API AWS SDK for Kotlin.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para o Amazon ECR usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon ECR com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon ECR ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos do Amazon ECR

Conceitos básicos do Amazon ECR

O exemplo de código a seguir mostra como:

- Criar uma imagem do Docker
- Crie um repositório do Amazon ECR.
- Excluir recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/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 "================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository
  - GetCallerIdentity

- GetLoginPassword

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Amazon ECR com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cotas de serviço do Amazon ECR

A tabela a seguir fornece as cotas de serviço padrão do Amazon Elastic Container Registry (Amazon ECR).

Nome	Padrão	Ajustá	Description
Verificações básicas de imagens por 24 horas	Cada região compatível: 100.000	Não	Número máximo de imagens que podem ser verificadas em um período de 24 horas na conta corrente e na região usando a verificação básica. Esse limite inclui verificações push e manuais.
Filtros por regra em uma configuração de replicação	Cada região compatível: 100	Não	O número máximo de filtros por regra em uma configuração de replicação.
Imagem por repositório	Cada região compatível: 100.000	Sim	O número máximo de imagens por repositório.
Partes da camada	Cada região compatível: 4.200	Não	O número máximo de partes da camada. Isso é aplicável somente se você estiver usando as ações da API do Amazon ECR diretamente para iniciar multipart uploads para operações de envio de imagem.

Nome	Padrão	Ajustá	Description
Duração da política de ciclo de vida	Cada região compatível: 30.720	Não	O número máximo de caracteres em uma política de ciclo de vida.
Tamanho máximo da parte da camada	Cada região com suporte: 10	Não	O tamanho máximo (MiB) de uma parte da camada. Isso é aplicável somente se você estiver usando as ações da API do Amazon ECR diretamente para iniciar multipart uploads para operações de envio de imagem.
Tamanho máximo da camada	Cada região compatível: 52.000	Não	O tamanho máximo (MiB) de uma camada.
Tamanho mínimo da parte da camada	Cada região compatível: 5	Não	O tamanho mínimo (MiB) de uma parte da camada. Isso é aplicável somente se você estiver usando as ações da API do Amazon ECR diretamente para iniciar multipart uploads para operações de envio de imagem.
Regras de cache de pull-through por registro	Cada região compatível: 50	Não	O número máximo de regras de cache de pull-through.

Nome	Padrão	Ajustá	Description
Taxa de BatchCheckLayerAvailability solicitações	Cada região compatível: 1.000 por segundo	Sim	O número máximo de BatchCheckLayerAvailability solicitações que você pode fazer por segundo na região atual. Quando uma imagem é enviada para um repositório, cada camada da imagem é conferida para verificar se foi feito upload dela anteriormente. Se o upload já tiver sido feito, a camada da imagem será ignorada.
Taxa de BatchGetImage solicitações	Cada região compatível: 2.000 por segundo	Sim	O número máximo de BatchGetImage solicitações que você pode fazer por segundo na região atual. Quando uma imagem é extraída, a BatchGetImage API é chamada uma vez para recuperar o manifesto da imagem. Se você solicitar um aumento de cota para essa API, analise também seu GetDownloadUrlForLayer uso.

Nome	Padrão	Ajustá	Description
Taxa de CompleteLayerUpload solicitações	Cada região compatível: 100 por segundo	Sim	O número máximo de CompleteLayerUpload solicitações que você pode fazer por segundo na região atual. Quando uma imagem é enviada, a CompleteLayerUpload API é chamada uma vez por cada nova camada de imagem para verificar se o upload foi concluído.
Taxa de GetAuthorizationToken solicitações	Cada região compatível: 500 por segundo	Sim	O número máximo de GetAuthorizationToken solicitações que você pode fazer por segundo na região atual.
Taxa de GetDownloadUrlForLayer solicitações	Cada região compatível: 3.000 por segundo	Sim	O número máximo de GetDownloadUrlForLayer solicitações que você pode fazer por segundo na região atual. Quando uma imagem é extraída, a GetDownloadUrlForLayer API é chamada uma vez por camada de imagem que ainda não está armazenada em cache. Se você solicitar um aumento de cota para essa API, analise também seu BatchGetImage uso.

Nome	Padrão	Ajustá	Description
Taxa de InitiateLayerUpload solicitações	Cada região compatível: 100 por segundo	Sim	O número máximo de InitiateLayerUpload solicitações que você pode fazer por segundo na região atual. Quando uma imagem é enviada, a InitiateLayerUpload API é chamada uma vez por camada de imagem que ainda não foi carregada. O upload de uma camada de imagem ou não é determinado pela ação da BatchCheckLayerAvailability API.
Taxa de PutImage solicitações	Cada região compatível: 10 por segundo	Sim	O número máximo de PutImage solicitações que você pode fazer por segundo na região atual. Quando uma imagem é enviada e todas as novas camadas de imagem são carregadas, a PutImage API é chamada uma vez para criar ou atualizar o manifesto da imagem e as tags associadas à imagem.

Nome	Padrão	Ajustá	Description
Taxa de UploadLayerPart solicitações	Cada região compatível: 500 por segundo	Sim	O número máximo de UploadLayerPart solicitações que você pode fazer por segundo na região atual. Quando uma imagem é enviada, cada nova camada de imagem é carregada em partes e a UploadLayerPart API é chamada uma vez por cada nova parte da camada de imagem.
Taxa de verificações de imagens	Cada região compatível: 1	Não	O número máximo de verificações de imagem por imagem, a cada 24 horas.
Repositórios registrados	Cada região compatível: 100.000	Sim	O número máximo de repositórios que você pode criar nesta conta na região atual.
Regras por política de ciclo de vida	Cada região compatível: 50	Não	O número máximo de regras em uma política de ciclo de vida
Regras por configuração de replicação	Cada região com suporte: 10	Não	O número máximo de regras em uma configuração de replicação.
Tags por imagem	Cada região compatível: 1.000	Não	O número máximo de tags por imagem.

Nome	Padrão	Ajustá	Description
Destinos exclusivos em todas as regras em uma configuração de replicação	Cada região compatível: 25	Não	O número máximo de destinos exclusivos em todas as regras em uma configuração de replicaçã o.

Gerenciando suas cotas de serviço do Amazon ECR no Console de gerenciamento da AWS

O Amazon ECR foi integrado ao Service Quotas, AWS um serviço que permite que você visualize e gerencie suas cotas a partir de um local central. Para obter mais informações, consulte O que é são cotas de serviço? no Manual do usuário do Service Quotas.

O Service Quotas facilita a pesquisa do valor de todas as cotas de serviço do Amazon ECR.

Consultar as cotas de serviço do Amazon ECR (Console de gerenciamento da AWS)

1. Abra o console do Service Quotas em https://console.aws.amazon.com/servicequotas/.
2. No painel de navegação, escolha Serviços da AWS .
3. Na lista de serviços da AWS , procure e selecione Amazon Elastic Container Registry (Amazon ECR).

   Na lista de cotas de serviço, você pode ver o nome da cota de serviço, o valor aplicado (se estiver disponível), a cota AWS padrão e se o valor da cota é ajustável.
4. Para visualizar informações adicionais sobre uma service quota, como descrição, escolha o nome da cota.

Para solicitar um aumento de cota, consulte Solicitação de um aumento de cota no Manual do usuário do Service Quotas.

Criação de um CloudWatch alarme para monitorar as métricas de uso da API

O Amazon ECR fornece métricas de CloudWatch uso que correspondem às cotas de AWS serviço para cada um dos APIs envolvidos com as ações de autenticação de registro, envio de imagem e extração de imagem. No console do Service Quotas, é possível visualizar o uso em um gráfico e configurar alarmes que alertarão você quando o uso se aproximar de uma cota de serviço. Para obter mais informações, consulte Métricas de uso do Amazon ECR.

Use as etapas a seguir para criar um CloudWatch alarme com base em uma das métricas de uso da API Amazon ECR.

Para criar um alarme baseado nas cotas de uso do Amazon ECR (Console de gerenciamento da AWS)

1. Abra o console do Service Quotas em https://console.aws.amazon.com/servicequotas/.
2. No painel de navegação, escolha Serviços da AWS .
3. Na lista de serviços da AWS , procure e selecione Amazon Elastic Container Registry (Amazon ECR).
4. Na lista de cotas de serviço, selecione a cota de uso do Amazon ECR para a qual você deseja criar um alarme.
5. Na seção de alarmes do Amazon CloudWatch Events, escolha Create.
6. Em Alarm threshold (Limitação do alarme), escolha a porcentagem do valor da cota aplicada que você deseja definir como o valor do alarme.
7. Em Alarm name (Nome do alarme), insira um nome para o alarme e selecione Create (Criar).

Solução de problemas do Amazon ECR

Este capítulo ajuda a encontrar informações de diagnóstico para o Amazon ECR e fornece as etapas de solução de problemas e mensagens de erro comuns.

Tópicos

Solução de problemas e comandos do Docker usando o Amazon ECR

Em alguns casos, a execução de um comando do Docker no Amazon ECR pode resultar em uma mensagem de erro. Algumas mensagens de erro comuns e possíveis soluções são explicadas abaixo.

Tópicos

Os logs do Docker não contêm mensagens de erro esperadas

Para começar a depurar qualquer problema relacionado ao Docker, você deve começar ativando a saída de depuração do Docker no daemon do Docker em execução nas instâncias do host. Caso esteja usando imagens extraídas do Amazon ECR nas instâncias de contêiner do Amazon ECS, consulte Configuração da saída detalhada do daemon do Docker no Guia do desenvolvedor do Amazon Elastic Container Service.

Erro: "Filesystem Verification Failed (Falha na verificação do sistema de arquivos)" ou "404: Image Not Found (Imagem não encontrada)" ao extrair uma imagem de um repositório do Amazon ECR

Você pode receber o erro `Filesystem verification failed` ao usar o comando docker pull para extrair uma imagem de um repositório do Amazon ECR com o Docker 1.9 ou versões posteriores. Ou o erro `404: Image not found` ao utilizar versões do Docker anteriores à 1.9.

Alguns motivos possíveis e suas explicações são mostrados abaixo.

O disco local está cheio

Se o disco local em que você está executando docker pull estiver cheio, o hash SHA-1 calculado no arquivo local pode ser diferente do calculado pelo Amazon ECR. Verifique se o disco local tem espaço livre suficiente para armazenar a imagem de Docker que você está extraindo. Você também pode excluir imagens antigas para dar espaço às novas. Use o comando docker images para ver uma lista com todas as imagens de Docker obtidas por download localmente, bem como os tamanhos delas.

O cliente não consegue se conectar ao repositório remoto devido a um erro de rede

As chamadas feitas para um repositório do Amazon ECR exigem uma conexão com a Internet. Verifique suas configurações de rede e se outros aplicativos e ferramentas podem acessar recursos na Internet. Se estiver executando docker pull em uma instância do Amazon EC2 em uma sub-rede privada, verifique se a sub-rede tem uma rota para a Internet. Use um servidor de tradução de endereço de rede (NAT) ou um gateway NAT gerenciado.

Atualmente, as chamadas feitas para um repositório do Amazon ECR também exigem acesso de rede por firewall corporativo para o Amazon Simple Storage Service (Amazon S3). Se sua organização usa software de firewall ou um dispositivo NAT que permite endpoints de serviço, verifique se os endpoints de serviço do Amazon S3 para a sua região atual são permitidos.

Se você usa o Docker atrás de um proxy HTTP, pode configurá-lo com as configurações de proxy apropriadas. Para obter mais informações, consulte proxy HTTP na documentação do Docker.

Erro: "Filesystem Layer Verification Failed (Falha na verificação da camada do sistema de arquivos)" ao extrair imagens do Amazon ECR

Você pode receber o erro `image image-name not found` ao extrair imagens com o comando docker pull. Se você inspecionar os logs do Docker, poderá ver um erro como este:

```
filesystem layer verification failed for digest sha256:2b96f...
```

Esse erro indica que uma ou mais das camadas da sua imagem não foram baixadas. Alguns motivos possíveis e suas explicações são mostrados abaixo.

Você está usando uma versão antiga do Docker

Esse erro pode ocorrer em alguns casos, quando uma versão do Docker anterior à 1.10 é usada. Atualize o cliente do Docker para a versão 1.10 ou posterior.

O cliente encontrou um erro de rede ou de disco

Um disco cheio ou um problema de rede pode impedir que uma ou mais camadas sejam baixadas, como já falamos sobre a mensagem `Filesystem verification failed`. Siga as recomendações acima para que seu sistema de arquivos não fique cheio e para verificar se você habilitou o acesso ao Amazon S3 de dentro da sua rede.

Erros 403 de HTTP ou o erro "no basic auth credentials (não há credenciais de autenticação básica)" ao enviar ao repositório

Algumas vezes, você poderá receber um erro `HTTP 403 (Forbidden)` ou a mensagem de erro no `basic auth credentials` dos comandos docker push ou docker pull, mesmo que tenha feito a autenticação no Docker com o comando aws ecr get-login-password. Veja a seguir algumas causas conhecidas desse problema:

Você fez a autenticação em outra região

As solicitações de autenticação são vinculadas a regiões específicas e não podem ser usadas entre regiões. Por exemplo, se você obtiver um token de autorização de Oeste dos EUA (Oregon), não poderá usá-lo para autenticação nos seus repositórios em Leste dos EUA (Norte da Virgínia). Para resolver o problema, verifique se você recuperou um token de autenticação da mesma região na qual o repositório existe. Para obter mais informações, consulte the section called “Autenticação de registro”.

Você realizou uma autenticação para enviar por push para um repositório ao qual não tem permissões

Você não tem as permissões necessárias para realizar o envio por push para o repositório. Para obter mais informações, consulte Políticas de repositório privado no Amazon ECR.

Seu token expirou

O período de expiração padrão para tokens de autorização obtidos usando a operação `GetAuthorizationToken` é de 12 horas.

Erro no gerenciador de credenciais`wincred`

Algumas versões do Docker para Windows usam um gerenciador de credenciais chamado`wincred`, que não manipula adequadamente o comando de login do Docker produzido por aws ecr get-login-password (para obter mais informações, consulte `CredsStore`falha com repositórios privados). Você pode executar o comando de login do Docker gerado. No entanto, quando você tenta enviar ou extrair imagens, esses comandos falham. Para resolver esse erro, remova o esquema `https://` do argumento de registro no comando de login do Docker gerado a partir de aws ecr get-login-password. Um exemplo de comando de login do Docker sem o esquema HTTPS é mostrado abaixo.

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Solução de problemas de mensagens de erro do Amazon ECR

Em alguns casos, uma chamada de API que você iniciou por meio do console do Amazon ECR ou a AWS CLI sai com uma mensagem de erro. Algumas mensagens de erro comuns e possíveis soluções são explicadas abaixo.

HTTP 429: Muitas solicitações ou ThrottleException

Você pode receber um erro `429: Too Many Requests` ou `ThrottleException` de uma ou mais ações do Amazon ECR ou chamadas de API. Isso indica que você está chamando um único endpoint no Amazon ECR repetidamente em um intervalo curto e que suas solicitações estão sendo limitadas. A suspensão ocorre quando as chamadas para um único endpoint de um único usuário ultrapassam um determinado limite em um período.

Cada operação de API no Amazon ECR tem um controle de utilização de taxas associada a ela. Por exemplo, a suspensão da ação `GetAuthorizationToken` é de 20 transações por segundo

(TPS), com permissão para uma intermitência de até 200 TPS. Em cada região, cada conta recebe um bucket que pode armazenar até 200 créditos `GetAuthorizationToken`. Esses créditos são reabastecidos a uma taxa de 20 por segundo. Se seu bucket tem 200 créditos, você pode alcançar 200 transações de API `GetAuthorizationToken` por segundo e sustentar 20 transações por segundo indefinidamente. Para obter mais informações sobre os limites de taxa do Amazon ECR APIs, consulteCotas de serviço do Amazon ECR.

Para gerenciar os erros de controle de utilização, implemente uma função de novas tentativas com backoff incremental no código. Para obter mais informações, consulte o comportamento de repetição no Guia de referência de ferramentas AWS SDKs e ferramentas. Outra opção é solicitar um aumento do limite de taxas, o que você pode fazer usando o console do Service Quotas. Para obter mais informações, consulte Gerenciando suas cotas de serviço do Amazon ECR no Console de gerenciamento da AWS.

HTTP 403: "O usuário [arn] não está autorizado a executar a [operação]"

Você poderá receber o seguinte erro ao tentar realizar uma ação com o 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: *
```

Isso indica que o usuário não tem as permissões para usar o Amazon ECR ou que essas permissões não estão configuradas corretamente. Se você realizar ações especificamente em um repositório do Amazon ECR, verifique se o usuário recebeu permissões para acessá-lo. Para obter mais informações sobre como criar e verificar permissões do Amazon ECR, consulte Gerenciamento de Identidade e Acesso para o Amazon Elastic Container Registry.

HTTP 404: erro "Repository Does Not Exist (O repositório não existe)"

Se você especificar um repositório do Docker Hub que não existe atualmente, o Docker Hub o criará automaticamente. Com o Amazon ECR, novos repositórios devem ser criados explicitamente para que poder serem usados. Isso impede que novos repositórios sejam criados de maneira acidental (por exemplo, devido a erros de digitação) e também garante que uma política de acesso de segurança apropriada seja atribuída explicitamente a todos os novos repositórios. Para obter mais informações sobre como criar repositórios, consulte Repositórios privados do Amazon ECR.

Erro: não é possível realizar um login interativo em um dispositivo não TTY

Se você receber o erro `Cannot perform an interactive login from a non TTY device`, as etapas de solução de problemas a seguir devem ajudar.

- Verifique se você está usando a AWS CLI versão 2 e se não tem uma versão conflitante da AWS CLI versão 1 em seu sistema. Para obter mais informações, consulte Instalar ou atualizar a versão mais recente da AWS CLI.
- Verifique se você configurou seu AWS CLI com credenciais válidas. Para obter mais informações, consulte Instalar ou atualizar a versão mais recente da AWS CLI.
- Verifique se a sintaxe do seu AWS CLI comando está correta.

Usar o Podman com o Amazon ECR

O uso do Podman com o Amazon ECR permite que as organizações aproveitem a segurança e a simplicidade do Podman ao mesmo tempo em que se beneficiam da escalabilidade e confiabilidade do Amazon ECR para gerenciamento de imagens de contêineres. Ao seguir as etapas e os comandos descritos, os desenvolvedores e administradores podem otimizar os fluxos de trabalho de contêineres, aprimorar a segurança e otimizar a utilização de recursos. À medida que a conteinerização continua a ganhar dinamismo, usar o Podman e o Amazon ECR fornece uma solução robusta e flexível para gerenciar e implantar aplicações em contêineres.

Usar o Podman para se autenticar com o Amazon ECR

Antes de interagir com o Amazon ECR usando o Podman, é necessária a autenticação. Isso pode ser feito executando o comando `aws ecr get-login-password` para recuperar um token de autenticação e, em seguida, usando esse token com o comando `podman login` para autenticar com o Amazon ECR.

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

Uso do assistente de credenciais do Amazon ECR com Podman

O Amazon ECR fornece um assistente de credenciais do Docker que funciona com o Podman. O assistente de credenciais facilita o armazenamento e o uso de credenciais do Docker ao enviar e extrair imagens do Amazon ECR. Para obter as etapas de instalação e configuração, consulte Auxiliar de credenciais do Docker do Amazon ECR.

> **Important**
>
> O Podman suporta apenas parcialmente a docker-creds-helper especificação. O Podman oferece suporte à palavra-chave `credHelpers` na configuração do Docker, mas não oferece suporte à palavra-chave `credsStore`.
>
> Para usar o assistente de credenciais do Amazon ECR com o Podman, configure seu arquivo de configuração do Docker com o formato `credHelpers`:
>
> ```
> {
>     "credHelpers": {
> ```

```
        "public.ecr.aws": "ecr-login",
        "<aws_account_id>.dkr.ecr.<region>.amazonaws.com": "ecr-login"
    }
}
```

A configuração `credsStore` a seguir não é compatível com o Podman:

```
{
    "credsStore": "ecr-login"
}
```

> **Note**
>
> No momento, o assistente de credenciais do Docker do Amazon ECR não oferece suporte à autenticação multifator (MFA).

Extrair imagens do Amazon ECR com o Podman

Após a autenticação com êxito, as imagens de contêineres podem ser extraídas do Amazon ECR usando o comando `podman pull` com o URI completo do repositório do Amazon ECR.

```
podman pull aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Executar contêineres do Amazon ECR com o Podman

Depois que a imagem desejada for extraída, um contêiner poderá ser instanciado usando o comando `podman run`.

```
podman run -d aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Enviar imagens por push para o Amazon ECR com o Podman

Para enviar por push uma imagem local para o Amazon ECR, a imagem deve primeiro ser marcada com o URI do repositório do Amazon ECR usando a `podman tag`. Em seguida, o comando `podman push` pode ser usado para fazer o upload da imagem no 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
```

Histórico do documento

A tabela a seguir descreve as alterações importantes na documentação desde a última versão do Amazon ECR . Também atualizamos a documentação com frequência para abordar os comentários enviados por você.

Alteração	Descrição	Data
A descontinuação da digitalização baseada em CLAIR foi concluída	Em 2 de fevereiro de 2026, o escaneamento de imagens baseado em CLAIR foi descontinuado e todas as contas ECR foram migradas para o escaneamento básico nativo. AWS A documentação foi atualizada para remover o conteúdo específico do CLAIR e refletir que o AWS nativo agora é a única implementação do escaneamento básico.	2 de fevereiro de 2026
Assinatura gerenciada por ECR	O Amazon ECR agora oferece suporte à assinatura gerenciada de imagens de contêineres para aprimorar sua postura de segurança e eliminar a sobrecarga operacional da configuração da assinatura. A assinatura de imagens de contêiner permite que você verifique se as imagens são de fontes confiáveis. Para obter mais informações, consulte Assinatura gerenciada.	21 de novembro de 2025
IPv6 suporte para AWS PrivateLink (VPC endpoints)	Foi adicionado suporte para conectividade de pilha dupla (IPv4 e IPv6) para endpoints Amazon ECR VPC baseados em. AWS PrivateLink Agora você pode criar endpoints VPC de pilha dupla que lidam com o tráfego tanto em endereços IP privados quanto em endereços IP privados. IPv4 IPv6 Para obter mais informações, consulte Endpoints VPC da interface Amazon ECR ()AWS PrivateLink.	21 de novembro de 2025
Recurso de arquivamento ECR	O Amazon ECR adicionou suporte ao arquivamento de imagens para retenção a longo prazo. Para obter mais informações, consulte Arquivamento de uma imagem no Amazon ECR.	19 de novembro de 2025

Alteração	Descrição	Data
Atualizado para incluir suporte para padrões de exclusão de imutabilidade de tags de imagens	O Amazon ECR atualizou os recursos de tag de imagens para incluir padrões de exclusão de imutabilidade de tags de imagem ao criar e atualizar repositórios. Agora você pode especificar tags que podem ser atualizadas mesmo quando a imutabilidade de tags está ativada em um repositório definindo padrões curinga (como `latest`, `v*.beta`, `dev-*`) para excluir tags específicas das regras de imutabilidade e, ao mesmo tempo, manter a imutabilidade de todas as outras tags. Para obter mais informações, consulte Criar um repositório privado do Amazon ECR para armazenar imagens.	23 de julho de 2025
Digitalização de imagem aprimorada atualizada para fornecer informações sobre o uso da imagem	O Amazon ECR atualizou os recursos aprimorados de verificação de imagens para incluir visibilidade de como as imagens são usadas no Amazon EKS e no Amazon ECS. Para obter mais informações, consulte Verificar imagens em busca de vulnerabilidades dos pacotes de sistemas operacionais e de linguagens de programação no Amazon ECR.	16 de junho de 2025
IPv6 apoio	Foi adicionado suporte para fazer solicitações aos registros do Amazon ECR usando endpoints IPv4 somente e de pilha dupla (e). IPv4 IPv6 Para obter mais informações, consulte Fazer solicitações aos registros do Amazon ECR.	30 de abril de 2025
Foi adicionado suporte de registro privado do Amazon ECR para cache de pull-through	O Amazon ECR incluiu suporte à criação de regras de cache de pull-through para o registro privado do Amazon ECR. Para obter mais informações, consulte Sincronizar um registro upstream com um registro privado do Amazon ECR e Função vinculada ao serviço do Amazon ECR para cache de pull-through.	12 de março de 2025

Alteração	Descrição	Data
Suporte adicionado para definir o escopo da política de registro	O Amazon ECR adicionou suporte à definição de escopo de política de registro para seu registro privado. Para obter mais informações, consulte Permissões de registro privado no Amazon ECR e Registro privado do Amazon ECR.	23 de dezembro de 2024
Amazon EC2 Container RegistryPullOnly — Nova política	O Amazon ECR adicionou uma nova política que concede permissões somente de extração para o Amazon ECR.	10 de outubro de 2024
As operações com proxy do cliente Docker/OCI em eventos agora apontam para CloudTrail `ecr.amazonaws.com`	O valor é `ecr.amazonaws.com` substituído `AWS Internal` nos campos User Agent (`userAgent`) e Source IP Address (`sourceIPAddress`) pelos CloudTrail eventos associados aos endpoints Docker/OCI do cliente. Veja exemplos em Exemplo: ação de extração de imagem e Exemplo: ação de envio de imagem.	1.º de julho de 2024
Adicionada a descrição do novo perfil vinculado ao serviço do Amazon ECR para modelos de criação de repositório.	O Amazon ECR usa uma função vinculada ao serviço chamada AWSServiceRoleForECRTemplateque dá permissão para o Amazon ECR realizar ações em seu nome para concluir ações de modelo de criação de repositórios. Para obter mais informações, consulte Perfil vinculado ao serviço do Amazon ECR para modelos de criação de repositório.	20 de junho de 2024
Adicionado o perfil vinculado ao serviço `ECRTemplateServiceRolePolicy` .	Adicionado o perfil vinculado ao serviço `ECRTemplateServiceRolePolicy` . Para obter mais informações, consulte `ECRTemplateServiceRolePolicy` .	20 de junho de 2024
Adicionada a replicação entre regiões e entre contas às regiões da China.	O Amazon ECR adicionou suporte à região da China para filtrar quais repositórios são replicados. Para obter mais informações, consulte Replicação de imagem privada no Amazon ECR.	15 de maio de 2024

Alteração	Descrição	Data
Registro de GitLab contêiner adicionado para verificar as regras de cache	O Amazon ECR adicionou suporte para criar regras de cache pull through para o registro de GitLab contêiner es. Para obter mais informações, consulte Sincroniz ar um registro upstream com um registro privado do Amazon ECR.	8 de maio de 2024
Atualização da política de ciclo de vida do Amazon ECR para adicionar suporte ao uso de curingas	O Amazon ECR adicionou suporte para curingas em uma política de ciclo de vida por meio do uso do parâmetro `tagPatternList` em uma regra de política de ciclo de vida. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.	18 de dezembro de 2023
Modelos de criação de repositório do Amazon ECR	O Amazon ECR adicionou suporte para modelos de criação de repositórios. Para obter mais informações, consulte Modelos para controlar repositórios criados durante uma ação de extração por cache, criação por push ou replicação.	15 de novembro de 2023
O cache de pull-thro ugh do Amazon ECR foi adicionado, compatível com registros upstream autenticados	O Amazon ECR adicionou suporte ao uso de registros upstream que exigem autenticação para suas regras de cache pull through. Para obter mais informações, consulte Sincronizar um registro upstream com um registro privado do Amazon ECR.	15 de novembro de 2023
AWSECRPullThroughC ache_ServiceRolePo licy— Atualização de uma política existente	O Amazon ECR adicionou novas permissões à política `AWSECRPullThroughCache_Serv iceRolePolicy` . Essas permissões permitem que o Amazon ECR recupere o conteúdo criptogra fado de um segredo do Secrets Manager. Isso é necessário ao usar uma regra de cache de pull-thro ugh para armazenar em cache imagens de um registro upstream que requer autenticação.	15 de novembro de 2023

Alteração	Descrição	Data
Assinatura de imagem do Amazon ECR	Amazon ECR e suporte AWS Signer adicional para criar e enviar assinaturas de imagens de contêineres usando o cliente Notary. Para obter mais informações, consulte Assine imagens no Amazon ECR.	6 de junho de 2023
Foi adicionado o registro de contêiner do Kubernetes para extrair as regras de cache	O Amazon ECR incluiu suporte à criação de regras de cache de pull-through para o registro de contêiner do Kubernetes. Para obter mais informações, consulte Sincronizar um registro upstream com um registro privado do Amazon ECR.	1 de junho de 2023
Suporte à duração da verificação aprimorada do Amazon ECR	O Amazon Inspector adicionou suporte para definir a duração pela qual seus repositórios são monitorados quando a verificação aprimorada está habilitada. Para obter mais informações, consulte Alterar a duração da verificação avançada de imagens no Amazon Inspector.	28 de junho de 2022
O Amazon ECR envia métricas de contagem de pull do repositório para a Amazon CloudWatch	O Amazon ECR envia métricas de contagem de pull do repositório para a Amazon. CloudWatch Para obter mais informações, consulte Métricas do repositório do Amazon ECR.	6 de janeiro de 2022
Suporte ampliado a replicação	O Amazon ECR adicionou suporte para filtragem de quais repositórios são replicados. Para obter mais informações, consulte Replicação de imagem privada no Amazon ECR.	21 de setembro de 2021
AWS políticas gerenciadas para o Amazon ECR	O Amazon ECR adicionou a documentação das políticas AWS gerenciadas. Para obter mais informaçõ es, consulte AWS políticas gerenciadas para o Amazon Elastic Container Registry.	24 de junho de 2021

Alteração	Descrição	Data
Replicação entre regiões e entre contas	O Amazon ECR adicionou suporte à definição de configurações de replicação para seu registro privado. Para obter mais informações, consulte Configurações de registro privado no Amazon ECR.	8 de dezembro de 2020
Suporte a artefatos OCI	O Amazon ECR adicionou suporte ao envio e extração de artefatos Open Container Initiative (OCI). Um novo parâmetro `artifactMediaType` foi adicionado à resposta da API `DescribeImages` para indicar o tipo de artefato. Para obter mais informações, consulte Para enviar um chart do Helm por push para um repositório privado do Amazon ECR.	24 de agosto de 2020
Criptografia em repouso	O Amazon ECR adicionou suporte à configuração de criptografia para seus repositórios usando a criptografia do lado do servidor com as chaves gerenciadas do cliente armazenada no AWS Key Management Service (AWS KMS). Para obter mais informações, consulte Criptografia em repouso.	29 de julho de 2020
Imagens multiarquitetura	O Amazon ECR adicionou suporte à criação e ao envio de listas de manifesto do Docker usadas para imagens de multiarquitetura. Para obter mais informações, consulte Como enviar uma imagem de multiarquitetura por push para um repositório do privado do Amazon ECR.	28 de abril de 2020

Alteração	Descrição	Data
Métricas de uso do Amazon ECR	O Amazon ECR adicionou métricas CloudWatch de uso que fornecem visibilidade sobre o uso de recursos da sua conta. Você também pode criar CloudWatch alarmes nos consoles CloudWatch e Service Quotas para receber alertas quando seu uso se aproximar da cota de serviço aplicada. Para obter mais informações, consulte Métricas de uso do Amazon ECR.	28 de fevereiro de 2020
Cotas de serviço do Amazon ECR atualizadas	Atualização das cotas de serviço do Amazon ECR para incluir cotas por API. Para obter mais informações, consulte Cotas de serviço do Amazon ECR.	19 de fevereiro de 2020
Adição do comando `get-login-password`	Adição de suporte para get-login-password, que oferece um método simples e seguro para recuperar um token de autorização. Para obter mais informações, consulte Uso de um token de autorização.	4 de fevereiro de 2020
Verificação de imagens	Adição de suporte à verificação de imagens, o que ajuda na identificação de vulnerabilidades de software em suas imagens de contêiner. O Amazon ECR usa o banco de dados Common Vulnerabilities and Exposures (CVEs) do projeto de código aberto CoreOS Clair e fornece uma lista dos resultados do escaneamento. Para obter mais informações, consulte Verificar imagens quanto a vulnerabilidades do software no Amazon ECR.	24 de outubro de 2019

Alteração	Descrição	Data
Política de VPC endpoint	Adição de suporte à configuração de uma política do IAM nos endpoints da interface da VPC do Amazon ECR. Para obter mais informações, consulte Criar uma política de endpoint para os endpoints da VPC do Amazon ECR.	26 de setembro de 2019
Mutabilidade de tag de imagem	Adição de suporte à configuração de um repositór io para ser imutável a fim de impedir que as tags de imagem sejam substituídas. Para obter mais informações, consulte Impedir que as tags de imagens sejam sobrescritas no Amazon ECR.	25 de julho de 2019
Endpoints da VPC de interface (AWS PrivateLi nk)	Foi adicionado suporte para configurar endpoints VPC de interface desenvolvidos por. AWS PrivateLink Isso permite criar uma conexão privada entre sua VPC e o Amazon ECR, sem necessidade de acesso pela Internet, por meio de uma instância NAT, de uma conexão VPN ou do Direct Connect. Para obter mais informações, consulte Endpoints VPC da interface Amazon ECR ()AWS PrivateLink.	25 de janeiro de 2019
Marcação de recursos	O Amazon ECR adicionou suporte à adição de tags de metadados aos seus repositórios. Para obter mais informações, consulte Marcar um repositório privado no Amazon ECR.	18 de dezembro de 2018
Alteração de nome do Amazon ECR	O Amazon Elastic Container Registry foi renomeado (anteriormente, Amazon EC2 Container Registry).	21 de novembro de 2017

Alteração	Descrição	Data
Políticas de ciclo de vida	As políticas de ciclo de vida do Amazon ECR permitem que você especifique o gerenciamento do ciclo de vida das imagens em um repositório. Para obter mais informações, consulte Automatizar a limpeza de imagens usando políticas de ciclo de vida no Amazon ECR.	11 de outubro de 2017
Suporte do Amazon ECR ao manifesto V2 esquema 2 de imagem do Docker	O Amazon ECR agora suporta o manifesto V2 esquema 2 de imagem do Docker (usado com o Docker versão 1.10 e posteriores). Para obter mais informações, consulte O formato de manifesto de imagem de contêiner é compatível com o Amazon ECR.	27 de janeiro de 2017
Disponibilidade geral do Amazon ECR	O Amazon Elastic Container Registry (Amazon ECR) é um serviço AWS gerenciado de registro Docker seguro, escalável e confiável.	21 de dezembro de 2015

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