aws

Benutzer-Leitfaden

Amazon ECR

API-Version 2015-09-21

Amazon ECR: Benutzer-Leitfaden

Table of Contents

Was ist Amazon Elastic Container Registry?

Amazon Elastic Container Registry (Amazon ECR) ist ein AWS verwalteter Container-Image-Registry-Service, der sicher, skalierbar und zuverlässig ist. Amazon ECR unterstützt private Repositorys mit ressourcenbasierten Berechtigungen mithilfe von IAM. AWS Auf diese Weise können bestimmte Benutzer oder EC2 Amazon-Instances auf Ihre Container-Repositorys und Images zugreifen. Sie können Ihre bevorzugte CLI verwenden, um Docker-Images, Open Container Initiative (OCI)-Images und OCI-kompatible Artefakte zu schieben, zu ziehen und zu verwalten.

> **Note**
>
> Amazon ECR unterstützt auch öffentliche Container-Image-Repositories. Weitere Informationen finden Sie unter Was ist öffentliches Amazon ECR im Öffentliches Amazon ECR Benutzerhandbuch.

Das AWS Container-Services-Team unterhält eine öffentliche Roadmap für. GitHub Sie enthält Informationen darüber, woran die Teams gerade arbeiten, und ermöglicht es allen AWS Kunden, direktes Feedback zu geben. Weitere Informationen erhalten Sie unter AWS -Container-Roadmap.

Konzepte und Komponenten von Amazon ECR

Amazon ECR ist ein vollständig verwalteter Docker-Container-Registrierungsservice, der von bereitgestellt wird. AWS Er ermöglicht Ihnen das sichere und zuverlässige Speichern, Verwalten und Bereitstellen von Docker-Container-Images. Diese Konzepte und Komponenten arbeiten zusammen, um einen sicheren, skalierbaren und zuverlässigen Docker-Container-Registrierungsdienst innerhalb von bereitzustellen AWS, der es Ihnen ermöglicht, Ihre containerisierten Anwendungen effizient zu verwalten und bereitzustellen.

Hier sind einige der wichtigsten Konzepte und Komponenten von Amazon ECR:

Registrierung

Eine Amazon ECR-Registrierung ist ein privates Repository, das für jedes AWS Konto bereitgestellt wird und in dem Sie ein oder mehrere Repositorys erstellen können. Diese Repositorys ermöglichen es Ihnen, Docker-Images, Open Container Initiative (OCI) -Images und andere OCI-kompatible Artefakte in Ihrer Umgebung zu speichern und zu verteilen. AWS Weitere Informationen finden Sie unter Private Amazon-ECR-Registrierung.

Autorisierungstoken

Ihr Client muss sich bei einer privaten Registrierung von Amazon ECR als AWS -Benutzer authentifizieren, bevor er Images übertragen und abrufen kann. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.

Repository

Ein Repository in Amazon ECR ist eine logische Sammlung, in der Sie Ihre Docker-Images, Open Container Initiative (OCI) -Images und andere OCI-kompatible Artefakte speichern können. Innerhalb einer einzigen Amazon ECR-Registrierung können Sie mehrere Repositorys haben, um Ihre Container-Images zu organisieren. Weitere Informationen finden Sie unter Private Repositories von Amazon ECR.

Repository-Richtlinie

Sie können den Zugriff auf Ihre Repositorys und die darin enthaltenen Inhalte mit Repository-Richtlinien steuern. Weitere Informationen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.

Abbild

Sie können Container-Images per Push und Pull an die Repositorys übertragen. Sie können diese Bilder lokal auf Ihrem Entwicklungssystem verwenden, oder Sie können sie in Amazon ECS-Aufgabendefinitionen und Amazon EKS-Pod-Spezifikationen verwenden. Weitere Informationen erhalten Sie unter Verwendung von Amazon ECR-Bildern mit Amazon ECS und Verwendung von Amazon ECR-Bildern mit Amazon EKS.

Lebenszyklus-Richtlinie

Mit den Lebenszyklusrichtlinien von Amazon ECR können Sie den Lebenszyklus Ihrer Images verwalten, indem Sie Regeln für das Löschen und Ablaufen alter oder ungenutzter Images definieren. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

Scannen von Bildern

Amazon ECR bietet eine integrierte Funktion zum Scannen von Bildern, mit deren Hilfe Softwareschwachstellen in Ihren Container-Images identifiziert werden können. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.

Zugriffskontrolle

Amazon ECR verwendet IAM, um den Zugriff auf Ihre Repositorys zu kontrollieren. Sie können IAM-Benutzer, -Gruppen und -Rollen mit bestimmten Berechtigungen zum Push-, Pull- oder

Verwalten von Amazon ECR-Repositorys erstellen. Weitere Informationen finden Sie unter Sicherheit in Amazon Elastic Container Registry.

Konto- und regionsübergreifende Replikation

Amazon ECR unterstützt die Replikation von Bildern über mehrere AWS Konten und Regionen hinweg, um die Verfügbarkeit zu erhöhen und die Latenz zu reduzieren. Weitere Informationen finden Sie unter Replikation privater Images in Amazon ECR.

Verschlüsselung

Amazon ECR unterstützt die serverseitige Verschlüsselung Ihrer Docker-Images im Ruhezustand mithilfe von. AWS KMS Weitere Informationen finden Sie unter Datenschutz bei Amazon ECR.

AWS Command Line Interface Integration

Das AWS CLI bietet Befehle für die Interaktion mit Amazon ECR-Repositorys, z. B. das Erstellen, Auflisten, Übertragen und Abrufen von Bildern.

AWS-Managementkonsole

Amazon ECR kann auch über das AWS-Managementkonsole verwaltet werden und bietet eine benutzerfreundliche Weboberfläche für die Arbeit mit Ihren Repositorys und Bildern.

AWS CloudTrail

Amazon ECR ist in Amazon ECR integriert und ermöglicht es Ihnen AWS CloudTrail, API-Aufrufe an Amazon ECR aus Sicherheits- und Compliance-Gründen zu protokollieren und zu prüfen. Weitere Informationen finden Sie unter Protokollierung von Amazon ECR-Aktionen mit AWS CloudTrail.

Amazon CloudWatch

Amazon ECR bietet Metriken und Protokolle Amazon CloudWatch, mit denen Sie die Leistung und Nutzung Ihrer Amazon ECR-Repositorys überwachen können. Weitere Informationen finden Sie unter Amazon-ECR-Repository-Metriken.

Verwaltetes Signieren

Managed Signing generiert automatisch kryptografische Signaturen, wenn Bilder an Amazon ECR übertragen werden, wodurch das Signieren von Container-Images vereinfacht wird. Weitere Informationen finden Sie unter Verwaltetes Signieren.

Häufige Anwendungsfälle in Amazon ECR

Amazon ECR ist ein vollständig verwalteter Docker-Container-Registrierungsservice, der von angeboten wird. AWS Er bietet ein sicheres und skalierbares Repository für die Speicherung und Verteilung von Docker-Container-Images und ist damit eine wichtige Komponente bei der Bereitstellung containerisierter Anwendungen. Amazon ECR vereinfacht den Prozess der Erstellung, Verteilung und Ausführung von containerisierten Anwendungen für verschiedene AWS Services und lokale Umgebungen.

Hier sind einige wichtige Anwendungsfälle für Amazon ECR:

Speicherung und Verteilung von Container-Images

Amazon ECR dient als zentrales Repository für die Speicherung und Verteilung von Docker-Container-Images innerhalb einer Organisation oder für den öffentlichen Gebrauch. Entwickler können ihre Container-Images auf Amazon ECR übertragen und sie dann aus einer beliebigen AWS Datenverarbeitungsumgebung wie Amazon EC2 oder Amazon EKS abrufen. AWS Fargate Weitere Informationen finden Sie unter Private Repositories von Amazon ECR.

Fortlaufende Integration und fortlaufende Bereitstellung (CI/CD)

Amazon ECR lässt sich nahtlos in AWS CodeBuild AWS CodePipeline, und andere CI/CD Tools integrieren und ermöglicht so das automatisierte Erstellen, Testen und Bereitstellen von containerisierten Anwendungen. Container-Images können als Teil der CI/CD Pipeline automatisch an Amazon ECR übertragen werden, wodurch eine konsistente und zuverlässige Bereitstellung in verschiedenen Umgebungen gewährleistet wird.

Microservices-Architektur

Amazon ECR eignet sich gut für Microservices-Architekturen, bei denen Anwendungen in kleinere, entkoppelte Services aufgeteilt werden, die als Container verpackt sind. Für jeden Microservice kann ein eigenes Container-Image in Amazon ECR gespeichert werden, was die unabhängige Entwicklung, Bereitstellung und Skalierung einzelner Services ermöglicht.

Hybrid- und Multi-Cloud-Bereitstellungen

Amazon ECR unterstützt die Möglichkeit, Container-Images aus anderen Container-Registern wie Docker Hub oder Registern von Drittanbietern abzurufen. Auf diese Weise können Unternehmen ein konsistentes Bereitstellungsmodell für Hybrid- oder Multi-Cloud-Umgebungen beibehalten und Amazon ECR als zentrales Repository für Container-Images verwenden.

Zugriffskontrolle und Sicherheit

Amazon ECR bietet detaillierte Zugriffskontrollmechanismen, mit denen Unternehmen kontrollieren können, wer Container-Images per Push oder Pull aus der Registrierung abrufen kann. Es lässt sich auch AWS Identity and Access Management zur Authentifizierung und Autorisierung integrieren und gewährleistet so einen sicheren Zugriff auf Container-Images. Weitere Informationen finden Sie unter Sicherheit in Amazon Elastic Container Registry.

Scannen nach Sicherheitslücken in Bildern

Amazon ECR bietet automatisches Scannen von Container-Images auf Softwareschwachstellen und mögliche Fehlkonfigurationen und trägt so zur Aufrechterhaltung einer sicheren und konformen Container-Umgebung bei. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.

Privates Container-Register

Für Unternehmen mit strengen Sicherheits- oder Compliance-Anforderungen kann Amazon ECR als private Container-Registry verwendet werden, um sicherzustellen, dass sensible Container-Images nicht öffentlichen Registern zugänglich gemacht werden und nur innerhalb der AWS Unternehmensumgebung zugänglich sind. Weitere Informationen finden Sie unter Private Amazon-ECR-Registrierung.

Weltweit verteilte Anwendungsbereitstellung mit Amazon ECR Replication

Mithilfe der Amazon ECR-Replikationsfunktionen können Sie Ihre containerisierten Webanwendungs-Images in einem primären Repository zentralisieren. Dies ermöglicht eine automatisierte Verteilung über mehrere AWS Regionen, gewährleistet konsistente globale Bereitstellungen mit geringer Latenz weltweit und reduziert den betrieblichen Aufwand. Weitere Informationen finden Sie unter Replikation privater Images in Amazon ECR.

Automatisierte Bereinigung veralteter Container-Images

Amazon ECR-Lebenszyklusrichtlinien ermöglichen die automatische Bereinigung veralteter Container-Images auf der Grundlage definierter Regeln wie Alter, Anzahl oder Tags, wodurch die Speicherkosten optimiert, ein organisiertes Register verwaltet, Sicherheit und Compliance verbessert und Entwicklungsabläufe durch Automatisierung rationalisiert werden. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

Features von Amazon ECR

Amazon ECR bietet die folgenden Features:

- Lebenszyklusrichtlinien helfen bei der Verwaltung des Lebenszyklus der Images in Ihren Repositories. Sie definieren Regeln, die dazu führen, dass nicht verwendete Bilder bereinigt werden. Sie können Regeln testen, bevor Sie sie auf Ihr Repository anwenden. Weitere Informationen erhalten Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.
- Image-Scans helfen bei der Identifizierung von Software-Schwachstellen in Ihren Container-Images. Jedes Repository kann so konfiguriert werden, dass es bei Push gescannt wird. Dadurch wird sichergestellt, dass jedes neue Image, das dem Repository zugeführt wird, gescannt wird. Sie können dann die Ergebnisse des Image-Scans abrufen. Weitere Informationen erhalten Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.
- Die regionen- und kontoübergreifende Replikation macht es Ihnen leichter, Ihre Images dort zu haben, wo Sie sie brauchen. Dies wird als Registrierungseinstellung konfiguriert und gilt für jede Region. Weitere Informationen finden Sie unter Private Registrierungseinstellungen in Amazon ECR.
- Durch Pull-Through-Cache-Regeln können Repositorys in einer Upstream-Registrierung in Ihrer privaten Registrierung von Amazon ECR zwischengespeichert werden. Mithilfe einer Pull-Through-Cache-Regel wendet sich Amazon ECR regelmäßig an die Upstream-Registrierung, um sicherzustellen, dass das zwischengespeicherte Image in Ihrer privaten Registrierung von Amazon ECR auf dem neuesten Stand ist. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung.
- Mit Vorlagen zur Repository-Erstellung können Sie die Einstellungen für Repositorys definieren, die von Amazon ECR in Ihrem Namen während der Aktionen Pull-Through-Cache, Create on Push oder Replikation erstellt wurden. Sie können die Unveränderlichkeit von Tags, die Verschlüsselungskonfiguration, Repository-Richtlinien, Lebenszyklusrichtlinien und Ressourcen-Tags für automatisch erstellte Repositorys angeben. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.
- Managed Signing generiert automatisch kryptografische Signaturen, wenn Bilder an Amazon ECR übertragen werden, wodurch das Signieren von Container-Images vereinfacht wird. Weitere Informationen finden Sie unter Verwaltetes Signieren.

Wie man mit Amazon ECR anfängt

Wenn Sie Amazon Elastic Container Service (Amazon ECS) oder Amazon Elastic Kubernetes Service (Amazon EKS) verwenden, beachten Sie, dass das Setup für diese beiden Services dem Setup für Amazon ECR ähnelt, da Amazon ECR eine Erweiterung beider Services ist.

Wenn Sie das AWS Command Line Interface mit Amazon ECR verwenden, verwenden Sie eine Version von AWS CLI , die die neuesten Amazon ECR-Funktionen unterstützt. Wenn Sie in der keine Unterstützung für eine Amazon ECR-Funktion sehen AWS CLI, führen Sie ein Upgrade auf die neueste Version von durch. AWS CLI Informationen zur Installation der neuesten Version von finden Sie unter Installation oder Aktualisierung auf die neueste Version von AWS CLI im AWS Command Line Interface Benutzerhandbuch. AWS CLI

Informationen zum Pushen eines Container-Images in ein privates Amazon ECR-Repository mithilfe von Docker AWS CLI und finden Sie unter. Ein Bild in Amazon ECR durch seinen Lebenszyklus bewegen

Preise für Amazon ECR

Mit Amazon ECR zahlen Sie für die Datenmenge, die Sie in Ihren Repositorys speichern, für die Datenübertragung aus Ihren Image-Pushes und -Pulls sowie für Image-Aktionen, für die Sie sich entscheiden, wie z. B. das Signieren und Replizieren von Bildern. Weitere Informationen erhalten Sie unter Amazon ECR-Preise.

Ein Bild in Amazon ECR durch seinen Lebenszyklus bewegen

Wenn Sie Amazon ECR zum ersten Mal verwenden, verwenden Sie die folgenden Schritte mit der Docker-CLI und dem, um ein Beispiel-Image AWS CLI zu erstellen, sich bei der Standardregistrierung zu authentifizieren und ein privates Repository zu erstellen. Senden Sie dann ein Bild in das private Repository und ziehen Sie ein Bild aus dem privaten Repository ab. Wenn Sie mit dem Beispielbild fertig sind, löschen Sie das Beispielbild und das Repository.

Informationen zur Verwendung von AWS-Managementkonsole anstelle von finden Sie unterthe section called “Erstellen eines Repositorys zum Speichern von Bildern”. AWS CLI

Weitere Informationen zu den anderen verfügbaren Tools für die Verwaltung Ihrer AWS Ressourcen, einschließlich der verschiedenen AWS SDKs IDE-Toolkits und der PowerShell Windows-Befehlszeilentools, finden Sie unter http://aws.amazon.com/tools/.

Voraussetzungen

Wenn Sie nicht die neueste Version von Docker installiert AWS CLI und einsatzbereit haben, führen Sie die folgenden Schritte aus, um diese beiden Tools zu installieren.

Installieren Sie das AWS CLI

Um das AWS CLI mit Amazon ECR zu verwenden, installieren Sie die neueste AWS CLI Version. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im Benutzerhandbuch von AWS Command Line Interface .

Installieren von Docker

Docker ist auf vielen verschiedenen Betriebssystemen verfügbar, darunter die meisten modernen Linux-Distributionen wie Ubuntu und sogar macOS und Windows. Weitere Informationen zur Installation von Docker unter einem bestimmten Betriebssystem finden Sie im Docker-Installationshandbuch.

Für die Verwendung von Docker wird kein lokales Entwicklungssystem benötigt. Wenn Sie Amazon EC2 bereits verwenden, können Sie eine Amazon Linux 2023-Instance starten und Docker installieren, um loszulegen.

Wenn Sie Docker bereits installiert haben, fahren Sie mit Schritt 1: Erstellen eines Docker-Images fort.

So installieren Sie Docker auf einer EC2 Amazon-Instance mithilfe eines Amazon Linux 2023 AMI

1. Starten Sie eine Instance mit dem neuesten Amazon-Linux-2023-AMI. Weitere Informationen finden Sie unter Starten einer Instance im EC2 Amazon-Benutzerhandbuch.
2. Verbinden Sie sich mit der Instance. Weitere Informationen finden Sie unter Connect to Your Linux Instance im EC2 Amazon-Benutzerhandbuch.
3. Aktualisieren Sie die installierten Pakete und den Cache der Paketverwaltung auf Ihrer Instance.

   ```
   sudo yum update -y
   ```

4. Installieren Sie das neueste Docker-Community Edition-Paket.

   ```
   sudo yum install docker
   ```

5. Starten Sie den Docker-Service.

   ```
   sudo service docker start
   ```

6. Fügen Sie den `ec2-user` zur Gruppe `docker` hinzu, sodass Sie Docker-Befehle ohne Verwendung von `sudo` ausführen können.

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

7. Melden Sie sich ab und wieder an, um die neuen Berechtigungen der Gruppe `docker` zu übernehmen. Sie erreichen dies, indem Sie das aktuelle SSH-Terminalfenster schließen und sich über ein neues Terminalfenster wieder mit Ihrer Instance verbinden. Ihre neue SSH-Sitzung verfügt über die entsprechenden `docker`-Gruppenberechtigungen.
8. Überprüfen Sie, ob der `ec2-user` Docker-Befehle ohne `sudo` ausführen kann.

   ```
   docker info
   ```

   > **Note**
   >
   > In einigen Fällen müssen Sie möglicherweise Ihre Instance neu starten, um den `ec2-user` für den Zugriff auf den Docker-Daemon zu berechtigen. Versuchen Sie, Ihre Instance neu zu starten, wenn die folgende Fehlermeldung angezeigt wird:

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

Schritt 1: Erstellen eines Docker-Images

In diesem Schritt erstellen Sie ein Docker-Image einer einfachen Webanwendung und testen es auf Ihrem lokalen System oder Ihrer EC2 Amazon-Instance.

So erstellen Sie ein Docker-Image einer einfachen Webanwendung

1. Erstellen Sie eine Datei mit dem Namen `Dockerfile`. Eine Docker-Datei ist eine Manifestdatei, die das für Ihr Docker-Image zu verwendende Basis-Image sowie die Inhalte beschreibt, die Sie darauf installieren und ausführen möchten. Weitere Informationen zu Dockerfiles finden Sie unter Dockerfile Reference.

   ```
   touch Dockerfile
   ```

2. Bearbeiten Sie die soeben von Ihnen erstellte `Dockerfile` und fügen Sie die folgenden Inhalte hinzu.

   ```
   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
   ```

Diese Docker-Datei verwendet das öffentliche Amazon-Linux-2-Image, das auf Amazon ECR Public gehostet wird. Die `RUN`-Anweisungen aktualisieren die Caches der Paketverwaltung, installieren einige Softwarepakete für den Webserver und schreiben dann den Inhalt "Hello World!" in das Stammverzeichnis für Dokumente des Webservers. Die `EXPOSE`-Anweisung stellt Port 80 auf dem Container bereit, und die `CMD`-Anweisung startet den Webserver.

3. Erstellen Sie das Docker-Image aus der Dockerfile.

> **Note**
>
> Einige Versionen von Docker erfordern im folgenden Befehl anstelle des unten angegebenen relativen Pfads den vollständigen Pfad zu Ihrer Dockerfile.

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

4. Führen Sie Ihr Container-Image auf.

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

Ausgabe:

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

5. Führen Sie das neu erstellte Image aus. Die Option `-p 80:80` ordnet den bereitgestellten Port 80 auf dem Container dem Port 80 auf dem Hostsystem zu. Weitere Informationen zu docker run finden Sie unter Referenz zu Docker run.

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

> **Note**
>
> Ausgabe vom Apache-Webserver wird im Terminal-Fenster angezeigt. Sie können die Meldung „`Could not reliably determine the fully qualified domain name`“ ignorieren.

6. Öffnen Sie einen Browser und richten Sie ihn auf den Server aus, auf dem Docker ausgeführt und Ihr Container gehostet wird.

   - Wenn Sie eine EC2 Instance verwenden, ist dies der öffentliche DNS-Wert für den Server. Dabei handelt es sich um dieselbe Adresse, die Sie für die Verbindung mit der Instance über SSH verwenden. Vergewissern Sie sich, dass die Sicherheitsgruppe für Ihre Instance eingehenden Datenverkehr auf Port 80 zulässt.
   - Wenn Sie Docker lokal ausführen, richten Sie Ihren Browser auf http://localhost/ aus.
   - Wenn Sie es docker-machine auf einem Windows- oder Mac-Computer verwenden, suchen Sie die IP-Adresse der VirtualBox VM, die Docker hostet, mit dem docker-machine ip Befehl und ersetzen *`machine-name`* Sie es durch den Namen des Docker-Computers, den Sie verwenden.

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

   Sie sollten eine Webseite mit dem Text "Hello, World!" Nachricht sehen.

7. Beenden Sie den Docker-Container, indem Sie Strg+C eingeben.

Schritt 2: Erstellen Sie ein Repository

Da Sie nun ein Image haben, das Sie an Amazon ECR pushen möchten, müssen Sie ein Repository erstellen, das dieses Image enthält. In diesem Beispiel erstellen Sie das Repository `hello-repository`, an das Sie später das `hello-world:latest`-Image per Push übertragen. Führen Sie zum Erstellen eines Repositorys den folgenden Befehl aus:

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

Schritt 3: Authentifizieren Sie sich bei Ihrer Standardregistrierung

Nachdem Sie das installiert und konfiguriert haben AWS CLI, authentifizieren Sie die Docker-CLI bei Ihrer Standardregistrierung. Auf diese Weise kann der docker-Befehl Images mit Amazon ECR pushen und abrufen. Das AWS CLI bietet einen get-login-password Befehl zur Vereinfachung des Authentifizierungsprozesses.

Um Docker bei einer Amazon ECR-Registry mit zu authentifizieren get-login-password, führen Sie den Befehl aus. aws ecr get-login-password Verwenden Sie bei der Übergabe des Authentifizierungs-Tokens an den Befehl docker login den Wert `AWS` für den Benutzernamen und geben Sie die URI der Amazon-ECR-Registrierung an, bei der Sie sich authentifizieren möchten. Wenn Sie sich bei mehreren Registrierungen authentifizieren, müssen Sie den Befehl für jede Registrierung wiederholen.

> **Important**
>
> Bei einem Fehler installieren oder aktualisieren Sie auf die neueste AWS CLI-Version. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface -Benutzerhandbuch.

- 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- Befehl () ECRLoginAWS Tools for Windows PowerShell

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

Schritt 4: Pushen Sie ein Image an Amazon ECR

Jetzt können Sie Ihr Image in das Amazon ECR-Repository pushen, das Sie im vorherigen Abschnitt erstellt haben. Verwenden Sie die docker CLI, um Bilder zu pushen, wenn die folgenden Voraussetzungen erfüllt sind:

- Die Mindestversion von docker ist installiert: 1.7.

- Das Amazon ECR-Autorisierungstoken wurde mit docker login konfiguriert.
- Das Amazon ECR-Repository ist vorhanden und der Benutzer hat Zugriff auf den Push zum Repository.

Wenn diese Voraussetzungen erfüllt sind, können Sie das Image per Push an das neu erstellte Repository in der Standardregistrierung Ihres Kontos übertragen.

So markieren und pushen Sie ein Image zu Amazon ECR

1. Listen Sie Images auf, die Sie lokal gespeichert haben, um das Image zu identifizieren, das mit Tags versehen und gepusht werden soll.

   ```
   docker images
   ```

   Ausgabe:

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

2. Versehen Sie Ihr Image mit Tags, um es in Ihr Repository zu pushen.

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

3. Übertragen Sie das Image per Push.

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

   Ausgabe:

   ```
   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
```

Schritt 5: Ein Image von Amazon ECR pullen

Nachdem Ihr Bild in Ihr Amazon ECR-Repository übertragen wurde, können Sie es von anderen Speicherorten abrufen. Verwenden Sie die docker CLI, um Bilder abzurufen, wenn die folgenden Voraussetzungen erfüllt sind:

- Die Mindestversion von docker ist installiert: 1.7.
- Das Amazon ECR-Autorisierungstoken wurde mit docker login konfiguriert.
- Das Amazon ECR-Repository ist vorhanden und der Benutzer hat Zugriff, um aus dem Repository zu pullen.

Wenn diese Voraussetzungen erfüllt sind, können Sie das Image pullen. Um Ihr Beispiel-Image von Amazon ECR zu beziehen, führen Sie folgenden Befehl aus:

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

Ausgabe:

```
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
```

Schritt 6: Löschen eines Images

Wenn Sie ein Bild in einem Ihrer Repositorys nicht mehr benötigen, können Sie das Bild löschen. Um ein Bild zu löschen, geben Sie das Repository an, in dem es sich befindet, und `imageTag` entweder einen `imageDigest` Oder-Wert für das Bild. Im folgenden Beispiel wird ein Bild im `hello-repository` Repository mit dem Image-Tag `latest` gelöscht. Führen Sie den folgenden Befehl aus, um Ihr Beispielbild aus dem Repository zu löschen:

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

Schritt 7: Löschen eines Repositorys

Wenn Sie kein ganzes Repository mit Bildern mehr benötigen, können Sie das Repository löschen. Im folgenden Beispiel wird das `--force` Flag verwendet, um ein Repository zu löschen, das Bilder enthält. Führen Sie die folgenden Schritte aus, um ein Repository mit allen darin enthaltenen Images zu löschen:

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

Optimierung der Leistung für Amazon ECR

Sie können die folgenden Empfehlungen zu Einstellungen und Strategien verwenden, um die Leistung bei der Verwendung von Amazon ECR zu optimieren.

Verwenden von Docker 1.10 (und neueren Versionen) für simultanes Hochladen der Layer

Docker-Images bestehen aus Ebenen, d. h. aus Zwischenschritten bei der Erstellung des Images. Jede Zeile in einer Docker-Datei führt zur Erstellung eines neuen Layers. Wenn Sie Docker 1.10 und höher verwenden, überträgt Docker standardmäßig so viele Schichten wie möglich gleichzeitig auf Amazon ECR, was zu schnelleren Hochladezeiten führt.

Verwenden eines kleineren Basis-Image

In den von Docker Hub bereitgestellten Standard-Images sind möglicherweise Abhängigkeiten vorhanden, die für Ihre Anwendung nicht benötigt werden. Ziehen Sie in Betracht, ein kleineres Image zu verwenden, das über die Docker-Community bereitgestellt wird. Alternativ können Sie das Scratch-Image von Docker als Basis nutzen und ein eigenes Image erstellen. Weitere Informationen finden Sie unter Ein Basis-Image erstellen in der Docker-Dokumentation.

Platzieren der Abhängigkeiten mit den wenigsten Änderungen an vorderer Stelle in der Docker-Datei

Docker legt die Layer im Zwischenspeicher ab, um die Erstellungszeiten zu verkürzen. Hat sich der Layer seit der letzten Erstellung nicht geändert, verwendet Docker die zwischengespeicherte Version (anstatt den Layer neu zu erstellen). Jedoch basiert jeder Layer auf den vorherigen Layern. Wenn ein Layer geändert wurde, erstellt Docker nicht nur diesen Layer neu, sondern auch alle nachfolgenden Layer.

Um die Zeit zu minimieren, die für die Neuerstellung eines Dockerfiles und das erneute hochladen von Ebenen benötigt wird, sollten Sie die Abhängigkeiten, die sich am wenigsten häufig ändern, am Anfang Ihres Dockerfiles platzieren. Abhängigkeiten mit häufigen Änderungen (z. B. der Quellcode der Anwendung) platzieren Sie hingegen an späterer Position im Stack.

Verketten von Befehlen zur Vermeidung unnötiger Dateispeicherung

Die auf einem Layer erstellten Zwischendateien bleiben ein Bestandteil des Layers, auch wenn sie in einem nachfolgenden Layer gelöscht werden. Betrachten Sie das folgende Beispiel:

```
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
```

In diesem Beispiel enthalten die mit dem ersten und dem zweiten RUN-Befehl erstellten Layer die ursprüngliche .tar.gz-Datei und den vollständigen unkomprimierten Inhalt. Dies trifft zu, obwohl die .tar.gz-Datei mit dem vierten RUN-Befehl gelöscht wird. Diese Befehle können zu einer einzigen RUN-Anweisung verkettet werden, damit diese unnötigen Dateien nicht mehr im letztendlichen Docker-Image enthalten sind:

```
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
```

Verwenden des nächstgelegenen regionalen Endpunkts

Sie können die Latenz beim Abrufen von Images aus Amazon ECR verringern, indem Sie sicherstellen, dass Sie den regionalen Endpunkt verwenden, der dem Ort, an dem Ihre Anwendung ausgeführt wird, am nächsten ist. Wenn Ihre Anwendung auf einer EC2 Amazon-Instance ausgeführt wird, können Sie den folgenden Shell-Code verwenden, um die Region aus der Availability Zone der Instance abzurufen:

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

Die Region kann mithilfe des --region Parameters an AWS CLI Befehle übergeben oder mithilfe des aws configure Befehls als Standardregion für ein Profil festgelegt werden. Sie können die Region auch festlegen, wenn Sie mit dem AWS SDK Anrufe tätigen. Weitere Informationen finden Sie in der SDK-Dokumentation für Ihre Programmiersprache.

Anfragen an Amazon ECR-Registries stellen

Sie können OCI-Images, Docker-Images und OCI-kompatible Artefakte in privaten Amazon ECR-Registern übertragen, abrufen, löschen, anzeigen und verwalten, indem Sie entweder IPv4 Nur-Endpunkte oder Dual-Stack- (und) Endpunkte verwenden. IPv4 IPv6 Für Anfragen aus Netzwerken können Sie entweder Dual-Stack- oder Endpoints verwenden. IPv4 IPv4 Verwenden Sie einen Dual-Stack-Endpunkt, um Anfragen von einem IPv6 Netzwerk aus zu stellen. Weitere Informationen zum Stellen von Anfragen an öffentliche Amazon ECR-Register mithilfe IPv4 von Dual-Stack-Endpunkten finden Sie unter Anfragen an öffentliche Amazon ECR-Registries stellen. Für den Zugriff auf Amazon ECR über IPv6 fallen keine zusätzlichen Gebühren an. Weitere Informationen zu den Preisen finden Sie unter Amazon Elastic Container Registry — Preise.

Amazon ECR-Endpunkte werden durch Attribute gekennzeichnet, die über die Unterstützung von Only-Endpoints oder IPv4 Dual-Stack-Endpunkten hinausgehen. Zu diesen Attributen können gehören:

- Region — Jeder Endpunkt ist spezifisch für eine Region.
- Typ — Die Auswahl des Endpunkts hängt davon ab, ob Sie das AWS SDK oder OCI-kompatible Schnittstellen und Docker-Befehlszeilenschnittstellen verwenden.
- Sicherheit — In ausgewählten Regionen bietet Amazon ECR FIPS-konforme Endgeräte. Weitere Informationen zu einer Liste von FIPS-konformen Amazon ECR-Endpunkten finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Weitere Informationen zu Service-Endpunkten, die von Dual-Stack- IPv4, Docker- und OCI-Client unterstützt werden, der Amazon ECR-API-Aufrufe über die AWS CLI verarbeitet, finden Sie unter Service-Endpunkte. AWS SDKs

Erste Schritte mit dem Stellen von Anfragen über IPv6

Um eine Anfrage an eine Amazon ECR-Registrierung zu stellen IPv6, müssen Sie einen Dual-Stack-Endpunkt verwenden. Bevor Sie auf eine Amazon ECR-Registrierung zugreifen IPv6, überprüfen Sie die folgenden Anforderungen:

- Ihr Client und Ihr Netzwerk müssen dies unterstützen IPv6.
- Amazon ECR unterstützt die folgenden Anforderungstypen über IPv6:

- OCI- und Docker-Client-Anfragen:

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

- AWS API-Anfragen:

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

- Sie müssen alle AWS Identity and Access Management (IAM-) oder Registrierungsrichtlinien aktualisieren, die Quell-IP-Adressfilterung verwenden, um IPv6 Adressbereiche einzubeziehen. Weitere Informationen finden Sie unter Verwendung von IPv6 Adressen in IAM-Richtlinien.
- Wenn Sie diese Option verwenden IPv6, zeigen Serverzugriffsprotokolle `Remote IP` Adressen im IPv6 Format an. Aktualisieren Sie Ihre vorhandenen Tools, Skripts und Software, um diese IPv6 -formatierten IP-Adressen zu analysieren.

> **Note**
>
> Wenn Sie Probleme mit dem Vorhandensein von IPv6 Adressen in Protokolldateien haben, wenden Sie sich an. AWS Support

Testen der IP-Adresskompatibilität

Wenn Sie use Linux/Unix oder Mac OS X verwenden, können Sie testen, ob Sie auf einen Dual-Stack-Endpunkt zugreifen können, IPv6 indem Sie den `curl` Befehl verwenden, wie im folgenden Beispiel gezeigt:

Example

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

Sie erhalten Informationen wie im folgenden Beispiel gezeigt zurück. Wenn Sie über IPv6 die verbundene IP-Adresse verbunden sind, wird dies eine IPv6 Adresse sein.

```
* 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
```

Wenn Sie Microsoft Windows 7 oder Windows 10 verwenden, können Sie testen, ob Sie über IPv4 oder IPv6 mithilfe des `ping` Befehls auf einen Dual-Stack-Endpunkt zugreifen können, wie im folgenden Beispiel gezeigt.

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

Senden von Anfragen mithilfe IPv6 von Dual-Stack-Endpunkten

Sie können Amazon ECR API-Aufrufe über IPv6 Dual-Stack-Endpunkte tätigen. Die Funktionalität und Leistung der Amazon ECR-API-Operationen bleiben konsistent, unabhängig davon, ob Sie IPv4 oder IPv6 verwenden.

Wenn Sie AWS Command Line Interface (AWS CLI) und verwenden AWS SDKs, können Sie IPv6 entweder einen Parameter oder ein Flag verwenden, um zu einem Dual-Stack-Endpunkt zu wechseln, oder indem Sie den Dual-Stack-Endpunkt direkt in Ihrer Konfigurationsdatei angeben, um den standardmäßigen Amazon ECR-Endpunkt zu überschreiben. Sie können Konfigurationsänderungen auch mithilfe eines Befehls vornehmen, der im Standardprofil `use_dualstack_endpoint` auf true gesetzt ist. Weitere Informationen zu `use_dualstack_endpoint` finden Sie unter Dual-Stack- und FIPS-Endpunkte.

Example Vornehmen von Konfigurationsänderungen mithilfe eines Befehls

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

Example Anfragen stellen statt IPv6 verwenden AWS CLI

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

Verwenden von Amazon ECR-Endpunkten über die Docker-CLI

Nachdem Sie sich bei Ihrem Amazon ECR-Repository angemeldet und Ihr Image mit einem Tag versehen haben, können Sie OCI-Images und Docker-Images per Push zu und von Amazon ECR-Registern übertragen und abrufen. Die folgenden Beispiele demonstrieren die Befehle Docker Push und Docker Pull mit beiden Dual-Stack-Endpunkten.

Example Docker-Images mithilfe IPv4 des Endpunkts pushen

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

Example Docker-Images mithilfe eines Dual-Stack-Endpunkts pushen

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

Example Docker-Images mithilfe des Endpunkts abrufen IPv4

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

Example Abrufen von Docker-Images mithilfe eines Dual-Stack-Endpunkts

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

Verwendung von IPv6 Adressen in IAM-Richtlinien

Bevor Sie mithilfe von auf eine Registrierung zugreifen IPv6, stellen Sie sicher, dass Ihre IAM-Benutzer- und Amazon ECR-Registrierungsrichtlinien, die IP-Adressfilterung verwenden, Adressbereiche enthalten IPv6 . Wenn die Richtlinien zur IP-Adressfilterung nicht aktualisiert werden, um IPv6 Adressen zu verarbeiten, verlieren oder erhalten Kunden möglicherweise fälschlicherweise Zugriff auf die Registrierung, wenn sie sie verwenden. IPv6 Weitere Informationen über die Verwaltung von Zugriffsberechtigungen mit IAM finden Sie unter Identity and Access Management für Amazon Elastic Container Registry.

IAM-Richtlinien, die IP-Adressen filtern, verwenden Bedingungsoperatoren für IP-Adressen. Das folgende Beispiel für eine Registrierungsrichtlinie zeigt, wie der `54.240.143.*` Bereich der zulässigen IPv4 Adressen mithilfe von Bedingungsoperatoren für IP-Adressen identifiziert werden kann. Allen IP-Adressen außerhalb dieses Bereichs wird der Zugriff auf die Registrierung verweigert (`exampleregistry`). Da sich alle IPv6 Adressen außerhalb des zulässigen Bereichs befinden, verhindert diese Richtlinie den Zugriff von IPv6 Adressen`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"}
      }
    }
  ]
}
```

Um sowohl IPv4 (`54.240.143.0/24`) als auch IPv6 (`2001:DB8:1234:5678::/64`) Adressbereiche zuzulassen, ändern Sie das Condition-Element der Registrierungsrichtlinie, wie im folgenden Beispiel gezeigt. Sie können dieses `Condition` Blockformat verwenden, um sowohl Ihre IAM-Benutzer- als auch Ihre Registrierungsrichtlinien zu aktualisieren.

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

> **Important**
>
> Vor der Verwendung müssen IPv6 Sie alle relevanten IAM-Benutzer- und Registrierungsrichtlinien aktualisieren, die IP-Adressfilterung verwenden. Es wird nicht empfohlen, die IP-Adressfilterung in Registrierungsrichtlinien zu verwenden.

Sie können Ihre IAM-Benutzerrichtlinien in der IAM-Konsole unter überprüfen. https://console.aws.amazon.com/iam/ Weitere Informationen zu IAM finden Sie im IAM-Benutzerhandbuch.

Private Amazon-ECR-Registrierung

Eine private Amazon-ECR-Registrierung host Ihre Container-Images in einer hochverfügbaren und skalierbaren Architektur. Sie können Ihre private Registrierung verwenden, um private Image-Repositories zu verwalten, die aus Docker- und Open Container Initiative (OCI)-Images und Artefakten bestehen. Jedes AWS Konto verfügt standardmäßig über eine private Amazon ECR-Registrierung. Weitere Informationen über öffentliche Amazon-ECR-Registrierungen finden Sie unter Öffentliche Registrierungen im öffentlichen Benutzerhandbuch von Amazon Elastic Container Registry.

Private Registrierungskonzepte

- Die URL für Ihre private Standardregistrierung lautet `https://aws_account_id.dkr.ecr.region.amazonaws.com`.
- Standardmäßig hat Ihr Konto Lese- und Schreibzugriff auf die Repositories in Ihrer privaten Registrierung. Benutzer benötigen jedoch Berechtigungen, um Amazon ECR aufzurufen und Bilder in APIs und aus Ihren privaten Repositorys zu übertragen oder abzurufen. Amazon ECR bietet mehrere verwaltete Richtlinien zur Steuerung des Benutzerzugriffs auf verschiedenen Ebenen. Weitere Informationen finden Sie unter Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien.
- Sie müssen Ihren Docker-Client bei Ihrer privaten Registrierung authentifizieren, damit Sie die Befehle docker push und docker pull verwenden können, um Images zu den Repositories in dieser Registrierung zu pushen und zu pullen. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.
- Private Repositories können sowohl mit -Benutzerzugriffsrichtlinien als auch mit Repository-Richtlinien kontrolliert werden. Weitere Hinweise zu Repository-Richtlinien finden Sie unter Richtlinien für private Repositorys in Amazon ECR.
- Die Repositorys in Ihrer privaten Registrierung können regionsübergreifend in Ihrer eigenen privaten Registrierung und über separate Konten hinweg AWS repliziert werden, indem Sie die Replikation für Ihre private Registrierung konfigurieren. Weitere Informationen finden Sie unter Replikation privater Images in Amazon ECR.

Authentifizierung bei privaten Registern in Amazon ECR

Sie können das AWS-Managementkonsole, das oder das verwenden AWS CLI, um private Repositorys AWS SDKs zu erstellen und zu verwalten. Sie können mit diesen Methoden auch einige Aktionen für Images (z. B. auflisten oder löschen) ausführen. Diese Clients verwenden AWS Standardauthentifizierungsmethoden. Auch wenn Sie die Amazon ECR-API verwenden können, um Images zu pushen und zu ziehen, werden Sie wahrscheinlich eher die Docker-CLI oder eine sprachspezifische Docker-Bibliothek verwenden.

Die Docker-CLI unterstützt keine nativen IAM-Authentifizierungsmethoden. Es müssen zusätzliche Schritte unternommen werden, damit Amazon ECR die Push- und Pull-Anforderungen von Docker authentifizieren und autorisieren kann.

Die in den folgenden Abschnitten beschriebenen Authentifizierungsmethoden der Registrierung sind verfügbar.

Verwendung des Amazon ECR Credential Helper

Amazon ECR stellt einen Docker Credential Helper zur Verfügung, der das Speichern und Verwenden von Docker Credentials beim Push- und Pull-Images an Amazon ECR erleichtert. Informationen zu Installations- und Konfigurationsschritten finden Sie unter Amazon ECR Docker Credential Helper.

> **Note**
>
> Der ECR Docker Credential Helper unterstützt derzeit keine Multi-Faktor-Authentifizierung (MFA).

Verwendung eines Autorisierungs-Tokens

Der Berechtigungsbereich eines Berechtigungstokens entspricht dem des IAM-Principals, der zum Abrufen des Authentifizierungstokens verwendet wird. Ein Authentifizierungstoken wird für den Zugriff auf jede Amazon ECR-Registrierung verwendet, auf die Ihr IAM-Prinzipal Zugriff hat, und ist 12 Stunden lang gültig. Um ein Autorisierungstoken zu erhalten, müssen Sie mithilfe der GetAuthorizationTokenAPI-Operation ein Base64-kodiertes Autorisierungstoken abrufen, das den Benutzernamen `AWS` und ein codiertes Passwort enthält. Der AWS CLI `get-login-password` Befehl vereinfacht dies, indem er das Autorisierungstoken abruft und dekodiert, das Sie dann an einen Befehl zur Authentifizierung weiterleiten können. docker login

So authentifizieren Sie Docker bei einer privaten Amazon ECR-Registry mit get-login

- Um Docker bei einer Amazon ECR-Registry mit zu authentifizieren get-login-password, führen Sie den Befehl aus. aws ecr get-login-password Verwenden Sie bei der Übergabe des Authentifizierungs-Tokens an den Befehl docker login den Wert `AWS` für den Benutzernamen und geben Sie die URI der Amazon-ECR-Registrierung an, bei der Sie sich authentifizieren möchten. Wenn Sie sich bei mehreren Registrierungen authentifizieren, müssen Sie den Befehl für jede Registrierung wiederholen.

  > **Important**
  >
  > Bei einem Fehler installieren oder aktualisieren Sie auf die neueste AWS CLI-Version. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface -Benutzerhandbuch.

  - 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- Befehl () ECRLoginAWS Tools for Windows PowerShell

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

HTTP-API-Authentifizierung verwenden

Amazon ECR unterstützt die Docker Registry HTTP API. Da es sich bei Amazon ECR jedoch um eine private Registrierung handelt, müssen Sie bei jeder HTTP-Anforderung ein Autorisierungs-Token bereitstellen. Sie können mithilfe der `-H` Option for einen HTTP-Autorisierungsheader hinzufügen curl und das vom get-authorization-token AWS CLI Befehl bereitgestellte Autorisierungstoken übergeben.

So authentifizieren Sie sich mit der Amazon ECR HTTP API

1. Rufen Sie mit dem ein Autorisierungstoken ab AWS CLI und setzen Sie es auf eine Umgebungsvariable.

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

2. Um sich bei der API zu authentifizieren, übergeben Sie die Variable `$TOKEN` der Option `-H` des Befehls curl. Der folgende Befehl listet zum Beispiel die Image-Tags in einem Amazon ECR-Repository auf. Weitere Informationen finden Sie in der Docker Registry HTTP API-Referenzdokumentation.

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

   Die Ausgabe sieht wie folgt aus:

   ```
   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"]}
   ```

Private Registrierungseinstellungen in Amazon ECR

Amazon ECR verwendet private Registrierungseinstellungen, um Features auf der Registrierungsebene zu konfigurieren. Die privaten Registrierungseinstellungen werden für jede Region separat konfiguriert. Sie können private Registrierungseinstellungen verwenden, um die folgenden Features zu konfigurieren.

- Registrierungsberechtigungen — Eine Richtlinie für Registrierungsberechtigungen bietet die Kontrolle über die Replikation und die Zugriffsberechtigungen für den Cache. Weitere Informationen finden Sie unter Private Registrierungsberechtigungen in Amazon ECR.
- Pull-Through-Cache-Regeln — Eine Pull-Through-Cache-Regel wird verwendet, um Bilder aus einer Upstream-Registry in Ihrer privaten Amazon ECR-Registry zwischenzuspeichern. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung.

- Replikationskonfiguration — Die Replikationskonfiguration wird verwendet, um zu steuern, ob Ihre Repositorys zwischen AWS-Regionen oder kopiert werden. AWS-Konten Weitere Informationen finden Sie unter Replikation privater Images in Amazon ECR.
- Vorlagen für die Repository-Erstellung — Eine Vorlage für die Erstellung eines Repositorys wird verwendet, um die Standardeinstellungen zu definieren, die angewendet werden, wenn Amazon ECR in Ihrem Namen neue Repositorys erstellt. Zum Beispiel Repositorys, die durch eine Pull-Through-Cache-Aktion, durch Push erstellen oder durch Replikation erstellt wurden. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.
- Scan-Konfiguration — Standardmäßig ist Ihre Registrierung für grundlegende Scans aktiviert. Sie können die erweiterte Überprüfung aktivieren, die einen automatischen, kontinuierlichen Überprüfungsmodus bereitstellt, der sowohl nach Schwachstellen im Betriebssystem als auch in Programmiersprachenpaketen sucht. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.
- Ausschluss von Pull-Time-Updates — Sie können Ausschlüsse für Pull-Time-Updates konfigurieren, um zu verhindern, dass die letzte Pull-Time für bestimmte Bilder aktualisiert wird, wenn diese abgerufen werden. Dies ist nützlich für Images, die zu CI/CD Testzwecken oder für Zwecke verwendet werden, bei denen Sie nicht möchten, dass die Abrufzeit die Entscheidungen über die Lebenszyklusrichtlinien beeinflusst. Weitere Informationen finden Sie unter Ausschlüsse für Pull-Time-Updates.

Private Registrierungsberechtigungen in Amazon ECR

Amazon ECR verwendet eine Registrierungsrichtlinie, um einem AWS -Prinzipal auf privater Registry-Ebene Berechtigungen zu erteilen.

Der Geltungsbereich wird durch Auswahl der Version der Registrierungsrichtlinie festgelegt. Es gibt zwei Versionen mit unterschiedlichem Geltungsbereich der Registrierungsrichtlinie: Version 1 (V1) und Version 2 (V2). V2 ist der erweiterte Geltungsbereich der Registrierungsrichtlinie, der alle ECR-Berechtigungen umfasst. Die vollständige Liste der API-Aktionen finden Sie im Amazon ECR API-Leitfaden. Die Version V2 ist der standardmäßige Geltungsbereich der Registrierungsrichtlinie. Weitere Informationen zum Anzeigen oder Festlegen des Geltungsbereichs Ihrer Registrierungsrichtlinie finden Sie unterZum erweiterten Geltungsbereich der Registrierungsrichtlinie wechseln. Informationen zu den allgemeinen Einstellungen für Ihre private Amazon ECR-Registrierung finden Sie unterPrivate Registrierungseinstellungen in Amazon ECR.

Die Versionen sind wie folgt detailliert.

- V1 — Für Version 1 erzwingt Amazon ECR nur die folgenden Berechtigungen auf privater Registrierungsebene.
  - `ecr:ReplicateImage` – Erteilt einem anderen Konto, das als Quellregistrierung bezeichnet wird, die Erlaubnis, seine Images in Ihre Registrierung zu replizieren. Dies wird nur für die kontoübergreifende Replikation verwendet.
  - `ecr:BatchImportUpstreamImage` – Erteilt die Berechtigung, das externe Image abzurufen und in Ihre private Registrierung zu importieren.
  - `ecr:CreateRepository` – Gewährt die Berechtigung zum Erstellen eines Repositorys in einer privaten Registrierung. Diese Berechtigung ist erforderlich, wenn das Repository, welches entweder die replizierten oder zwischengespeicherten Images speichert, noch nicht in der privaten Registrierung existiert.
- V2 — Für Version 2 lässt Amazon ECR alle ECR-Aktionen in der Richtlinie zu und setzt die Registrierungsrichtlinie in allen ECR-Anfragen durch.

Sie können die Konsole oder die CLI verwenden, um den Geltungsbereich Ihrer Registrierungsrichtlinie anzuzeigen oder zu ändern.

> **Note**
>
> Es ist zwar möglich, die `ecr:*` Aktion zu einer privaten Registrierungsrichtlinie hinzuzufügen, es wird jedoch als bewährte Methode angesehen, nur die spezifischen Aktionen hinzuzufügen, die für die von Ihnen verwendete Funktion erforderlich sind, anstatt einen Platzhalter zu verwenden.

Themen

Beispiele für Richtlinien für private Registrierungen für Amazon ECR

Die folgenden Beispiele zeigen Richtlinienanweisungen für Registrierungsberechtigungen, die Sie verwenden können, um die Berechtigungen zu kontrollieren, die Benutzer für Ihre Amazon ECR-Registrierung haben.

> **Note**
>
> In jedem Beispiel kann die Replikation trotzdem stattfinden, wenn die `ecr:CreateRepository` Aktion aus Ihrer Registrierungsrichtlinie entfernt wird. Für eine erfolgreiche Replikation müssen Sie jedoch Repositories mit demselben Namen innerhalb Ihres Kontos erstellen.

Beispiel: Erlauben Sie allen IAM-Prinzipalen in einem Quellkonto, alle Repositorys zu replizieren

Die folgende Richtlinie für Registrierungsberechtigungen ermöglicht es allen IAM-Prinzipalen (Benutzern und Rollen) in einem Quellkonto, alle Repositorys zu replizieren.

Beachten Sie Folgendes:

- Wichtig: Wenn Sie in einer Richtlinie eine AWS-Konto ID als Principal angeben, gewähren Sie allen IAM-Benutzern und -Rollen innerhalb dieses Kontos Zugriff, nicht nur dem Root-Benutzer. Dies ermöglicht einen umfassenden Zugriff auf das gesamte Konto.
- Sicherheitsüberlegung: Berechtigungen auf Kontoebene gewähren Zugriff auf alle IAM-Entitäten im angegebenen Konto. Geben Sie für einen restriktiveren Zugriff einzelne IAM-Benutzer oder Rollen an oder verwenden Sie Bedingungsanweisungen, um den Zugriff weiter einzuschränken.

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/*"
            ]
        }
    ]
}
```

Beispiel: Erlauben Sie IAM-Prinzipale von mehreren Konten

Die folgende Richtlinie für Registrierungsberechtigungen besteht aus zwei Aussagen. Jede Anweisung ermöglicht es allen IAM-Prinzipalen (Benutzern und Rollen) in einem Quellkonto, alle Repositorys zu replizieren.

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/*"
            ]
        }
    ]
}
```

Beispiel: Erlaubt allen IAM-Prinzipalen in einem Quellkonto, alle Repositorys mit Präfix zu replizieren. **`prod-`**

Die folgende Richtlinie für Registrierungsberechtigungen ermöglicht es allen IAM-Prinzipalen (Benutzern und Rollen) in einem Quellkonto, alle Repositorys zu replizieren, die mit beginnen. `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-*"
            ]
        }
    ]
```

```
}
```

Zum erweiterten Geltungsbereich der Registrierungsrichtlinie wechseln

> **Important**
>
> Für neue Benutzer werden Ihre Registrierungen automatisch so konfiguriert, dass sie bei der Erstellung die `V2` Registrierungsrichtlinie verwenden. Sie müssen keine Maßnahmen ergreifen. Amazon ECR empfiehlt nicht, zur vorherigen Registrierungsrichtlinie zurückzukehren. `V1`

Sie können die Konsole oder die CLI verwenden, um den Geltungsbereich Ihrer Registrierungsrichtlinie anzuzeigen oder zu ändern.

AWS-Managementkonsole

Gehen Sie wie folgt vor, um Ihre Kontoeinstellungen einzusehen. Informationen zum Anzeigen oder Aktualisieren des Geltungsbereichs der Registrierungsrichtlinie finden Sie im CLI-Verfahren auf dieser Seite.

Aktivieren Sie die erweiterte Registrierungsrichtlinie für Ihre private Registrierung

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/ private-registry/repositories
2. Wählen Sie in der Navigationsleiste die Region aus.
3. Wählen Sie im Navigationsbereich Private Registrierung, Funktion und Einstellungen und anschließend Berechtigungen aus.
4. Sehen Sie sich auf der Seite „Berechtigungen“ unter „Registrierungsrichtlinie“ Ihren Richtlinien-JSON an. Wenn Sie über die V1-Richtlinie verfügen, wird ein Banner mit Anweisungen zur Aktualisierung auf Version 2 angezeigt. Wählen Sie Enable (Aktivieren) aus.

   Es wird ein Banner angezeigt, das darauf hinweist, dass der Geltungsbereich der Registrierungsrichtlinie auf Version V2 aktualisiert wurde.
5. Sie können optional auch Berechtigungen mit der CLI konfigurieren. Weitere Informationen finden Sie unter Private Registrierungseinstellungen in Amazon ECR.

> **Note**
>
> Informationen zum Anzeigen oder Aktualisieren des Geltungsbereichs der Registrierungsrichtlinie finden Sie im CLI-Verfahren auf dieser Seite.

AWS CLI

Amazon ECR generiert die V2-Registrierungsrichtlinie. Gehen Sie wie folgt vor, um den Geltungsbereich der Registrierungsrichtlinie anzuzeigen oder zu aktualisieren. Sie können den Geltungsbereich der Registrierungsrichtlinie in der Konsole nicht anzeigen oder ändern

- Um die Registrierungsrichtlinie abzurufen, die Sie derzeit verwenden.

```
aws ecr get-account-setting --name REGISTRY_POLICY_SCOPE
```

Der Parametername ist ein Pflichtfeld. Wenn Sie den Namen nicht angeben, erhalten Sie die folgende Fehlermeldung:

```
 aws: error: the following arguments are required: --name
```

Sehen Sie sich die Ausgabe für Ihren Registrierungsrichtlinien-Befehl an. In der folgenden Beispielausgabe ist die Version der Registrierungsrichtlinie V1.

```
{
    "name": "REGISTRY_POLICY_SCOPE",
    "value": "V1"
}
```

Sie können die Version Ihrer Registrierungsrichtlinie von V1 zu ändernV2. V1 ist nicht der empfohlene Geltungsbereich der Registrierungsrichtlinie.

```
aws ecr put-account-setting --name REGISTRY_POLICY_SCOPE --value value
```

Verwenden Sie beispielsweise den folgenden Befehl, um auf V2 zu aktualisieren.

```
aws ecr put-account-setting --name REGISTRY_POLICY_SCOPE --value V2
```

Sehen Sie sich die Ausgabe für Ihren Registrierungsrichtlinien-Befehl an. In der folgenden Beispielausgabe wurde die Version der Registrierungsrichtlinie auf V2 aktualisiert.

```
{
    "name": "REGISTRY_POLICY_SCOPE",
    "value": "V2"
}
```

Erteilen von Registrierungsberechtigungen für die kontoübergreifende Replikation in Amazon ECR

Der kontoübergreifende Richtlinientyp wird verwendet, um einem AWS Prinzipal Berechtigungen zu erteilen, sodass die Repositorys von einer Quellregistrierung in Ihre Registrierung repliziert werden können. Standardmäßig haben Sie die Berechtigung, die regionenübergreifende Replikation innerhalb Ihrer eigenen Registrierung zu konfigurieren. Sie müssen die Registrierungsrichtlinie nur konfigurieren, wenn Sie einem anderen Konto die Berechtigung erteilen, Inhalte in Ihre Registrierung zu replizieren.

Eine Registrierungsrichtlinie muss die Berechtigung für die API-Aktion `ecr:ReplicateImage` erteilen. Diese API ist eine interne Amazon ECR-API, die Images zwischen Regionen oder Konten replizieren kann. Sie können auch die Berechtigung für die `ecr:CreateRepository`-Berechtigung erteilen, die es Amazon ECR erlaubt, Repositories in Ihrer Registrierung zu erstellen, wenn diese noch nicht vorhanden sind. Wenn die Berechtigung `ecr:CreateRepository` nicht vorhanden ist, muss ein Repository mit demselben Namen wie das Quell-Repository manuell in Ihrer Registrierung erstellt werden. Wenn dies nicht geschieht, schlägt die Replikation fehl. Alle fehlgeschlagenen Aktionen `CreateRepository` oder `ReplicateImage` API-Aktionen werden in CloudTrail angezeigt.

So konfigurieren Sie eine Berechtigungsrichtlinie für die Replikation (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre Registrierungsrichtlinie konfigurieren möchten.
3. Wählen Sie im Navigationsbereich Private Registrierung, dann Funktionen und Einstellungen und anschließend Berechtigungen aus.
4. Wählen Sie auf der Seite Registrierungsberechtigungen die Option Anweisung generieren aus.

5. Führen Sie die folgenden Schritte aus, um Ihre Richtlinie mit Hilfe des Richtliniengenerators zu definieren.

   a. Wählen Sie als Richtlinientyp die Option Replikation — kontenübergreifend aus.

   b. Geben Sie für Kontoauszug-ID eine eindeutige Kontoauszug-ID ein. Dieses Feld wird als `Sid` für die Registrierungsrichtlinie verwendet.

   c. Geben Sie unter Konten das Konto IDs für jedes Konto ein, dem Sie Berechtigungen erteilen möchten. Wenn Sie mehrere Konten angeben IDs, trennen Sie diese durch ein Komma.

6. Wählen Sie Speichern.

So konfigurieren Sie eine Berechtigungsrichtlinie für die Replikation (AWS CLI)

1. Erstellen Sie eine Datei mit dem Namen `registry_policy.json` und füllen Sie sie mit einer Registrierungsrichtlinie.

   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. Erstellen Sie die Registrierungsrichtlinie mithilfe der Richtliniendatei.

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

3. Rufen Sie die Richtlinie für Ihre Registry zur Bestätigung ab.

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

Erteilen von Registrierungsberechtigungen für den Pull-Through-Cache in Amazon ECR

Private Registrierungsberechtigungen von Amazon ECRs können verwendet werden, um die Berechtigungen einzelner IAM-Entitäten zur Verwendung von Pull-Through-Cache zu nutzen. Wenn eine IAM-Entität mehr Berechtigungen hat, die durch eine IAM-Richtlinie gewährt werden, als die Registrierungsberechtigungsrichtlinie gewährt, hat die IAM-Richtlinie Vorrang.

So erstellen Sie eine Richtlinie für private Registrierungsberechtigungen (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre private Registrierungsberechtigungserklärung konfigurieren möchten.
3. Wählen Sie im Navigationsbereich Private Registrierung, dann Funktionen und Einstellungen und anschließend Berechtigungen aus.
4. Wählen Sie auf der Seite Registrierungsberechtigungen die Option Anweisung generieren aus.
5. Gehen Sie für jede Richtlinienanweisung für Pull-Through-Cache-Berechtigungen, die Sie erstellen möchten, wie folgt vor.

    a. Wählen Sie für Richtlinientyp, Pull-Through-Cache-Richtlinie aus.

    b. Für Anweisungs-ID, geben Sie einen Namen für die Richtlinie zur Pull-Through-Cache-Anweisung an.

    c. Geben Sie für IAM entities (IAM-Entitäten) die Benutzer, Gruppen oder Rollen an, die in die Richtlinie aufgenommen werden sollen.

    d. Wählen Sie unter Cache-Namespace die Pull-Through-Cacheregel aus, der Sie die Richtlinie zuordnen möchten.

e. Für Repository-Namen, geben Sie den Repository-Basisnamen an, für den die Regel angewendet werden soll. Wenn Sie beispielsweise das Amazon-Linux-Repository auf Amazon ECR Public angeben möchten, lautet der Repository-Name `amazonlinux`.

Private Repositories von Amazon ECR

Ein privates Amazon ECR-Repository enthält Ihre Docker-Images, Open Container Initiative (OCI) -Images und OCI-kompatible Artefakte. Sie können Bild-Repositorys erstellen, überwachen und löschen und Berechtigungen festlegen, mit denen gesteuert wird, wer auf sie zugreifen kann, indem Sie Amazon ECR-API-Operationen oder den Abschnitt Repositorys der Amazon ECR-Konsole verwenden. Amazon ECR ist auch in die Docker-CLI integriert, sodass Sie Images aus Ihren Entwicklungsumgebungen in Ihre Repositorys übertragen und abrufen können.

Themen

Private Repository-Konzepte

- Standardmäßig verfügt Ihr Konto über Lese- und Schreibzugriff auf die Repositorys in der Standardregistrierung (*aws_account_id*.dkr.ecr.*region*.amazonaws.com). Benutzer benötigen jedoch Berechtigungen, um Amazon ECR aufzurufen und Bilder in APIs und aus Ihren Repositorys zu übertragen oder abzurufen. Amazon ECR bietet mehrere verwaltete Richtlinien zur Steuerung des Benutzerzugriffs auf verschiedenen Ebenen. Weitere Informationen finden Sie unter Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien.
- Repositories können sowohl mit -Benutzerzugriffsrichtlinien als auch mit individuellen Repository-Richtlinien kontrolliert werden. Weitere Informationen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.
- Repository-Namen unterstützen Namespaces, sodass ähnliche Repositorys gruppiert werden können. Wenn zum Beispiel mehrere Teams dieselbe Registriy verwenden, kann Team A den Namespace `team-a` und Team B den Namespace `team-b` verwenden. Auf diese Weise hat jedes Team sein eigenes Image mit dem Namen `web-app`, wobei jedem Image der Namespace des Teams vorangestellt ist. Mit dieser Konfiguration können diese Images in jedem Team gleichzeitig

verwendet werden, ohne dass es zu Störungen kommt. Das Image von Team A ist `team-a/web-app` und das Image von Team B ist `team-b/web-app`.

- Ihre Images können in andere Repositories repliziert werden, und zwar regionenübergreifend in Ihrer eigenen Registrierung und über Konten hinweg. Sie können dies tun, indem Sie eine Replikationskonfiguration in Ihren Registrierungseinstellungen angeben. Weitere Informationen finden Sie unter Private Registrierungseinstellungen in Amazon ECR.

Erstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern

> **Important**
>
> Die zweischichtige serverseitige Verschlüsselung mit AWS KMS (DSSE-KMS) ist nur in den Regionen verfügbar. AWS GovCloud (US)

Erstellen Sie ein privates Amazon ECR-Repository und verwenden Sie das Repository dann zum Speichern Ihrer Container-Images. Führen Sie die folgenden Schritte aus, um ein privates Repository mit der AWS-Managementkonsole zu erstellen.

So erstellen Sie ein Repository (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Ihr Repository erstellt werden soll.
3. Wählen Sie Private Repositorys und dann Repository erstellen aus.
4. Geben Sie unter Repository-Name einen eindeutigen Namen für Ihr Repository ein. Der Name des Repository kann allein angegeben werden (z. B. `nginx-web-app`). Alternativ kann ihm ein Namespace vorangestellt werden, um das Repository einer Kategorie zuzuordnen (zum Beispiel `project-a/nginx-web-app`).

   > **Note**
   >
   > Der Repository-Name darf maximal 256 Zeichen enthalten. Der Name muss mit einem Buchstaben beginnen und darf nur Kleinbuchstaben, Zahlen, Bindestriche, Unterstriche,

Punkte und Schrägstriche enthalten. Die Verwendung eines doppelten Bindestrichs, Unterstrichs oder Schrägstrichs wird nicht unterstützt.

5. Wählen Sie für die Unveränderlichkeit von Image-Tags eine der folgenden Einstellungen für die Tag-Veränderlichkeit für das Repository aus.

   - Veränderbar — Wählen Sie diese Option, wenn Sie möchten, dass Bild-Tags überschrieben werden. Empfohlen für Repositorys, die Pull-Through-Cache-Aktionen verwenden, um sicherzustellen, dass Amazon ECR zwischengespeicherte Bilder aktualisieren kann. Um Tag-Updates für einige veränderbare Tags zu deaktivieren, geben Sie außerdem Tag-Namen ein oder verwenden Sie Platzhalter (*), um mehrere ähnliche Tags im Textfeld zum Ausschluss veränderbarer Tags abzugleichen.
   - Unveränderlich — Wählen Sie diese Option, wenn Sie verhindern möchten, dass Bild-Tags überschrieben werden. Sie gilt für alle Tags und Ausschlüsse im Repository, wenn Sie ein Bild mit vorhandenem Tag übertragen. Amazon ECR gibt eine zurück, `ImageTagAlreadyExistsException` wenn Sie versuchen, ein Bild mit einem vorhandenen Tag zu pushen. Um Tag-Updates für einige unveränderliche Tags zu aktivieren, geben Sie außerdem Tagnamen ein oder verwenden Sie Platzhalter (*), um mehrere ähnliche Tags im Textfeld zum Ausschluss unveränderlicher Tags abzugleichen.

   > **Note**
   >
   > Einstellungen für die Veränderbarkeit einzelner Tags werden nicht unterstützt.

6. Wählen Sie für die Verschlüsselungskonfiguration zwischen AES-256 oder AWS KMS Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

   a. Wenn diese Option ausgewählt AWS KMS ist, wählen Sie zwischen Single-Layer-Verschlüsselung und Dual-Layer-Verschlüsselung. Für die Nutzung der Dual-Layer-Verschlüsselung AWS KMS fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter Amazon ECR Service Pricing.

   b. Standardmäßig wird der AWS verwaltete Schlüssel mit dem Alias `aws/ecr` ausgewählt. Dieser Schlüssel wird in Ihrem Konto erstellt, wenn Sie zum ersten Mal ein Repository mit aktivierter AWS KMS Verschlüsselung erstellen. Wählen Sie Vom Kunden verwalteter Schlüssel (erweitert), um Ihren eigenen AWS KMS Schlüssel auszuwählen. Der AWS KMS Schlüssel muss sich in derselben Region wie der Cluster befinden. Wählen Sie AWS

KMS Schlüssel erstellen aus, um zur AWS KMS Konsole zu navigieren und Ihren eigenen Schlüssel zu erstellen.

7. Bei den Einstellungen für das Scannen von Bildern können Sie zwar die Scaneinstellungen für grundlegende Scans auf Repository-Ebene angeben, es hat sich jedoch bewährt, die Scankonfiguration auf der Ebene der privaten Registrierung festzulegen. Wenn Sie die Scaneinstellungen auf der Ebene der privaten Registrierung konfigurieren, können Sie zwischen erweitertem Scannen und einfachem Scannen wählen. Außerdem können Sie Filter definieren, um festzulegen, welche Repositorys gescannt werden sollen.
8. Wählen Sie Erstellen aus.

So erstellen Sie ein Repository (AWS CLI)

1. Sie können AWS CLI mit dem aws ecr create-repository Befehl ein Repository erstellen.

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

2. Wenn Sie eine Vorlage für die Erstellung eines Repositorys definiert haben, können Sie ein Repository erstellen, indem Sie Ihr Image mit vertrauten Amazon ECR-Push-Befehlen mit Ihrem gewünschten Repository-Namen übertragen. Amazon ECR erstellt das Repository automatisch für Sie anhand der vordefinierten Einstellungen Ihrer Repository-Erstellungsvorlage. Wenn Sie noch keine Vorlage für die Erstellung eines Repositorys definiert haben, schlägt Ihre Anfrage an Ihr nicht vorhandenes Image-Repository fehl.

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

Nächste Schritte

Um die Schritte zum Übertragen eines Images in Ihr Repository anzuzeigen, wählen Sie das Repository aus und wählen Sie „Push-Befehle anzeigen“. Weitere Informationen zum Pushen eines Images in Ihr Repository finden Sie unter Ein Bild in ein privates Amazon ECR-Repository übertragen.

Inhalt und Details eines privaten Repositorys in Amazon ECR anzeigen

Nachdem Sie ein privates Repository erstellt haben, können Sie Details zum Repository im folgenden AWS-Managementkonsole Verzeichnis einsehen:

- Welche Images sind in einem Repository gespeichert
- Details zu jedem im Repository gespeicherten Bild, einschließlich Größe und SHA-Digest für jedes Bild
- Die für den Inhalt des Repositorys angegebene Scan-Häufigkeit
- Ob dem Repository eine aktive Pull-Through-Cache-Regel zugeordnet ist
- Die Verschlüsselungseinstellung für das Repository

> **Note**
>
> Seit der Docker-Version 1.9 komprimiert der Docker-Client die Image-Ebenen, bevor er sie in eine V2-Docker-Registrierung überträgt. Die Ausgabe des Befehls docker images zeigt die unkomprimierte Imagegröße an. Beachten Sie daher, dass Docker möglicherweise ein größeres Image als das in der AWS-Managementkonsole angezeigte Image zurückgibt.

So zeigen Sie Repository-Informationen (AWS-Managementkonsole) an

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der das anzuzeigende Repository enthalten ist.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys Privat und anschließend das anzuzeigende Repository aus.
5. Auf der Repository-Detailseite ist die Konsole standardmäßig auf die Images-Ansicht eingestellt. Verwenden Sie das Navigationsmenü, um weitere Informationen über das Repository anzuzeigen.
   - Klicken Sie auf Übersicht, um die Repository-Details anzuzeigen und Zähldaten für das Repository abzurufen.

- Wählen Sie Images, um Informationen zu den Image-Registerkarten im Repository anzuzeigen. Um weitere Informationen über das Image anzuzeigen, wählen Sie die Image-Registerkarte aus. Weitere Informationen finden Sie unter Bilddetails in Amazon ECR anzeigen.

  Wenn es nicht gekennzeichnete Images gibt, die Sie löschen möchten, können Sie das Feld links neben den zu löschenden Repositories markieren und Löschen wählen. Weitere Informationen finden Sie unter Löschen eines Bilds in Amazon ECR.
- Wählen Sie die Registerkarte Berechtigungen, um die Repository-Richtlinien anzuzeigen, die auf das Repository angewendet werden. Weitere Informationen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.
- Wählen Sie die Registerkarte Lebenszyklus-Richtlinie, um die Lebenszyklus-Richtlinienregeln anzuzeigen, die auf das Repository angewendet werden. Der Verlauf der Lebenszyklus-Ereignisse wird hier ebenfalls angezeigt. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.
- Wählen Sie Tags, um die Metadaten-Tags anzuzeigen, die auf das Repository angewendet werden.

Löschen eines privaten Repositorys in Amazon ECR

Wenn Sie ein Repository nicht mehr verwenden möchten, können Sie es löschen. Wenn Sie ein Repository in der löschen AWS-Managementkonsole, werden auch alle im Repository enthaltenen Bilder gelöscht. Dies kann nicht rückgängig gemacht werden.

> **Important**
>
> Bilder in den gelöschten Repositorys werden ebenfalls gelöscht. Dieser Vorgang kann nicht rückgängig gemacht werden.

So löschen Sie ein Repository (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der sich das zu löschende Repository befindet.

3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys die Registerkarte Privatund wählen anschließend das zu löschende Repository aus und klicken Sie auf Löschen.
5. Vergewissern Sie sich im ***repository_name*** Fenster Löschen, dass die ausgewählten Repositorys gelöscht werden sollen, und wählen Sie Löschen.

Richtlinien für private Repositorys in Amazon ECR

Amazon ECR verwendet ressourcenbasierte Berechtigungen, um den Zugriff auf Repositories zu kontrollieren. Mit ressourcenbasierten Berechtigungen können Sie angeben, welche Benutzer oder Rollen Zugriff auf ein Repository haben und welche Aktionen sie im Repository ausführen können. Standardmäßig hat nur das AWS Konto, das das Repository erstellt hat, Zugriff auf das Repository. Sie können eine Repository-Richtlinie anwenden, die zusätzlichen Zugriff auf Ihr Repository ermöglicht.

Themen

Repository-Richtlinien im Vergleich zu IAM-Richtlinien

Amazon ECR Repository-Richtlinien sind eine Untergruppe von IAM-Richtlinien, die für die Kontrolle des Zugriffs auf einzelne Amazon ECR-Repositorys ausgelegt sind und speziell dafür verwendet werden. IAM-Richtlinien werden im Allgemeinen verwendet, um Berechtigungen für den gesamten Amazon ECR-Service anzuwenden, können aber auch verwendet werden, um den Zugriff auf bestimmte Ressourcen zu steuern.

Sowohl Amazon-ECR-Repository-Richtlinien als auch IAM-Richtlinien werden verwendet, um zu bestimmen, welche Aktionen ein bestimmter Benutzer oder eine bestimmte Rolle in einem Repository ausführen darf. Wenn ein Benutzer oder eine Rolle eine Aktion über eine Repository-Richtlinie ausführen darf, aber über eine IAM-Richtlinie nicht dazu berechtigt ist (oder umgekehrt), wird die Aktion verweigert. Ein Benutzer oder eine Rolle muss nur entweder über eine Repository-Richtlinie oder eine IAM-Richtlinie die Berechtigung für eine Aktion erhalten, nicht aber über beide, damit die Aktion erlaubt ist.

> **Important**
>
> Amazon ECR erfordert, dass Benutzer über eine IAM-Richtlinie die Berechtigung haben, die `ecr:GetAuthorizationToken` API aufzurufen, bevor sie sich bei einer Registrierung authentifizieren und Images aus einem Amazon ECR-Repository pushen oder pullen können. Amazon ECR bietet mehrere verwaltete IAM-Richtlinien zur Steuerung des Benutzerzugriffs auf unterschiedlichen Ebenen. Weitere Informationen finden Sie unter Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien.

Sie können eine der beiden Richtlinientypen für die Zugriffssteuerung Ihrer Repositorys verwenden, wie in den folgenden Beispielen dargestellt.

Dieses Beispiel zeigt eine Amazon-ECR-Repository-Richtlinie, die es einem bestimmten Benutzer ermöglicht, das Repository und die Images innerhalb des Repositorys zu beschreiben.

JSON

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

Dieses Beispiel zeigt eine IAM-Richtlinie, die das gleiche Ziel wie oben erreicht, indem die Richtlinie auf ein Repository (angegeben durch den vollständigen ARN des Repository) unter Verwendung des Ressourcenparameters beschränkt wird. Weitere Informationen zum Format von Amazon-Ressourcenname (ARN) finden Sie unter Ressourcen.

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"]
        }
    ]
}
```

Beispiele für Richtlinien für private Repositorien in Amazon ECR

> **Important**
>
> Die Beispiele für Repository-Richtlinien auf dieser Seite sollen auf private Repositorys von Amazon ECR angewendet werden. Sie funktionieren nicht richtig, wenn sie direkt mit einem IAM-Prinzipal verwendet werden, es sei denn, sie werden dahingehend geändert, dass das Amazon-ECR-Repository als Ressource angegeben wird. Weitere Hinweise zur Einrichtung von Repository-Richtlinien finden Sie unter Festlegung einer Richtlinienerklärung für private Repositorien in Amazon ECR.

Amazon ECR Repository-Richtlinien sind eine Untergruppe von IAM-Richtlinien, die für die Kontrolle des Zugriffs auf einzelne Amazon ECR-Repositorys ausgelegt sind und speziell dafür verwendet werden. IAM-Richtlinien werden im Allgemeinen verwendet, um Berechtigungen für den gesamten Amazon ECR-Service anzuwenden, können aber auch verwendet werden, um den Zugriff auf bestimmte Ressourcen zu steuern. Weitere Informationen finden Sie unter Repository-Richtlinien im Vergleich zu IAM-Richtlinien.

Die folgenden Beispiele für Repository-Richtlinien zeigen Berechtigungsanweisungen, die Sie verwenden können, um den Zugriff auf Ihre privaten Repositorys von Amazon ECR zu kontrollieren.

> **Important**
>
> Amazon ECR setzt voraus, dass Benutzer über eine IAM-Richtlinie die Erlaubnis haben, die `ecr:GetAuthorizationToken`-API aufzurufen, bevor sie sich bei einer Registrierung authentifizieren und Images aus einem Amazon ECR-Repository pushen oder pullen können. Amazon ECR bietet mehrere verwaltete IAM-Richtlinien zur Steuerung des Benutzerzugriffs auf unterschiedlichen Ebenen. Weitere Informationen finden Sie unter Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien.

Beispiel: Eine oder mehrere -Benutzer zulassen

Die folgende Repository-Richtlinie erlaubt es einem oder mehreren -Benutzern, Images in ein Repository zu pushen und von dort zu pullen.

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": "*"
        }
    ]
}
```

Beispiel: Ein anderes Konto erlauben

Die folgende Repository-Richtlinie gewährt einem angegebenen Konto die Berechtigung zur Push-Übertragung von Images.

> **Important**
>
> Für das Konto, dem Sie Berechtigungen erteilen, muss die Region, in der Sie die Repository-Richtlinie erstellen, aktiviert sein, sonst tritt ein Fehler auf.

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": "*"
        }
    ]
}
```

Die folgende Repository-Richtlinie ermöglicht es einigen Benutzern, Bilder abzurufen (*pull-user-1*und*pull-user-2*) und gleichzeitig vollen Zugriff auf andere (*admin-user*) zu gewähren.

> **Note**
>
> Bei komplizierteren Repository-Richtlinien, die derzeit nicht in der unterstützt werden AWS-Managementkonsole, können Sie die Richtlinie mit dem set-repository-policy AWS CLI Befehl anwenden.

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": "*"
        }
```

```
    ]
}
```

Beispiel: Allen verweigern

Die folgende Repository-Richtlinie verweigert allen Benutzern in allen Konten die Möglichkeit, Images zu pullen.

JSON

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

Beispiel: Beschränkung des Zugriffs auf bestimmte IP-Adressen

Im folgenden Beispiel wird jedem Benutzer die Berechtigung zum Ausführen von Amazon-ECR-Vorgängen verweigert, wenn es aus einem bestimmten Adressbereich auf ein Repository angewendet wird.

Die Bedingung in dieser Anweisung identifiziert den `54.240.143.*` Bereich der zulässigen IP-Adressen des Internetprotokolls, Version 4 (IPv4).

Der `Condition` Block verwendet die `NotIpAddress` Bedingungen und den `aws:SourceIp` Bedingungsschlüssel, bei dem es sich um einen AWS-weiten Bedingungsschlüssel handelt. Weitere Informationen über diese Bedingungsschlüssel finden Sie unter Globale AWS - Bedingungskontextschlüssel. Die `aws:sourceIp` IPv4 Werte verwenden die Standard-CIDR-

Notation. Weitere Informationen finden Sie unter IP-Adressen-Bedingungsoperatoren im IAM-Benutzerhandbuch.

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"
                }
            }
        }
    ]
}
```

Beispiel: Einen AWS Dienst zulassen

Die folgende Repository-Richtlinie ermöglicht den AWS CodeBuild Zugriff auf die Amazon ECR-API-Aktionen, die für die Integration mit diesem Service erforderlich sind. Wenn Sie das folgende Beispiel verwenden, sollten Sie die Bedingungsschlüssel `aws:SourceArn` und `aws:SourceAccount` verwenden, um zu ermitteln, welche Ressourcen diese Berechtigungen übernehmen können. Weitere Informationen finden Sie unter Amazon ECR sample for CodeBuild im AWS CodeBuild Benutzerhandbuch.

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"
               }
            }
         }
      ]
}
```

Festlegung einer Richtlinienerklärung für private Repositorien in Amazon ECR

Sie können einem Repository im eine Erklärung zur Zugriffsrichtlinie hinzufügen, AWS-Managementkonsole indem Sie die folgenden Schritte ausführen. Pro Repository können mehrere Richtlinienanweisungen hinzugefügt werden. Beispiele für Richtlinien finden Sie unter Beispiele für Richtlinien für private Repositorien in Amazon ECR.

> **Important**
>
> Amazon ECR erfordert, dass Benutzer über eine IAM-Richtlinie die Erlaubnis haben, die`ecr:GetAuthorizationToken`-API aufzurufen, bevor sie sich bei einer Registrierung authentifizieren und Images aus einem Amazon ECR-Repository pushen oder pullen können. Amazon ECR bietet mehrere verwaltete IAM-Richtlinien zur Steuerung des Benutzerzugriffs auf unterschiedlichen Ebenen. Weitere Informationen finden Sie unter Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien.

So legen Sie eine Repository-Richtlinienanweisung fest

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der das Repository enthalten ist, für das eine Richtlinienanweisung festgelegt werden soll.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das Repository aus, für das Sie eine Richtlinienanweisung festlegen möchten, um den Inhalt des Repositorys anzuzeigen.
5. Wählen Sie in der Listenansicht des Repository-Images im Navigationsbereich Berechtigungen, Bearbeiten.

   > **Note**
   >
   > Wenn Sie die Option Berechtigungen im Navigationsbereich nicht sehen, vergewissern Sie sich, dass Sie sich in der Listenansicht des Repository-Images befinden.

6. Wählen Sie auf der Seite Berechtigungen bearbeiten die Option Anweisung hinzufügen aus.
7. Geben Sie für Anweisungsname einen Namen für die Anweisung ein.
8. Wählen Sie für Effect aus, ob die Richtlinienanweisung zu einer Zugriffserlaubnis oder einer expliziten Zugriffsverweigerung führt.
9. Wählen Sie für Principal den Bereich aus, für den die Richtlinienanweisung angewendet werden soll. Weitere Informationen finden Sie unter AWS JSON-Richtlinienelemente: Principal im IAM-Benutzerhandbuch.

   - Sie können die Erklärung auf alle authentifizierten AWS Benutzer anwenden, indem Sie das Kontrollkästchen Jeder (*) aktivieren.
   - Geben Sie für Service principal den Prinzipalnamen des Services (z. B. `ecs.amazonaws.com`) an, um die Anweisung auf einen bestimmten Service anzuwenden.
   - Geben Sie für AWS AWS Konto IDs eine Kontonummer an (z. B.`111122223333`), um die Abrechnung für alle Benutzer eines bestimmten AWS Kontos zu verwenden. Mehrere Konten können mithilfe einer durch Komma getrennten Liste angegeben werden.

> **Important**
>
> Für das Konto, dem Sie Berechtigungen erteilen, muss die Region, in der Sie die Repository-Richtlinie erstellen, aktiviert sein, sonst tritt ein Fehler auf.

- Wählen Sie für IAM-Entitäten die Rollen oder Benutzer in Ihrem AWS Konto aus, auf die die Abrechnung angewendet werden soll.

> **Note**
>
> Bei komplizierteren Repository-Richtlinien, die derzeit nicht in der unterstützt werden AWS-Managementkonsole, können Sie die Richtlinie mit dem set-repository-policy AWS CLI Befehl anwenden.

10. Wählen Sie für Aktionen aus der Liste der einzelnen API-Vorgänge den Bereich der Amazon ECR-API-Vorgänge aus, für den die Richtlinienanweisung gelten soll.
11. Wenn Sie fertig sind, klicken Sie auf Save, um die Richtlinie zu speichern.
12. Wiederholen Sie den vorherigen Schritt für jede hinzuzufügende Repository-Richtlinie.

Kennzeichnen eines privaten Repositorys in Amazon ECR

Um Ihnen bei der Verwaltung Ihrer Amazon ECR-Repositorys zu helfen, können Sie neuen oder bestehenden Amazon ECR-Repositorys mithilfe von Ressourcen-Tags Ihre eigenen Metadaten zuweisen. AWS Sie können zum Beispiel eine Reihe von Tags für die Amazon-ECR-Repositories Ihres Kontos definieren, mit denen Sie den Besitzer jedes Repositories verfolgen können.

Grundlagen zu Tags (Markierungen)

Tags haben für Amazon ECR keine semantische Bedeutung und werden ausschließlich als Zeichenkette interpretiert. Tags werden nicht automatisch Ihren Ressourcen zugewiesen. Sie können Tag (Markierung)-Schlüssel und -Werte bearbeiten und Tags (Markierungen) jederzeit von einer Ressource entfernen. Sie können den Wert eines Tags (Markierung) zwar auf eine leere Zeichenfolge, jedoch nicht null festlegen. Wenn Sie ein Tag (Markierung) mit demselben Schlüssel wie ein vorhandener Tag (Markierung) für die Ressource hinzufügen, wird der alte Wert mit dem neuen überschrieben. Wenn Sie eine Ressource löschen, werden alle Tags (Markierungen) der Ressource ebenfalls gelöscht.

Sie können mithilfe der Amazon ECR-Konsole, der und der AWS CLI Amazon ECR-API mit Tags arbeiten.

Mithilfe von AWS Identity and Access Management (IAM) können Sie steuern, welche Benutzer in Ihrem AWS Konto berechtigt sind, Tags zu erstellen, zu bearbeiten oder zu löschen. Informationen zu Tags in IAM-Richtlinien finden Sie unter. the section called “Verwenden Tag-basierter Zugriffskontrolle”

Markieren von Ressourcen für die Fakturierung

Die Tags, die Sie Ihren Amazon ECR-Repositories hinzufügen, sind hilfreich bei der Überprüfung der Kostenzuweisung, nachdem Sie sie in Ihrem Kosten- und Nutzungsbericht aktiviert haben. Weitere Informationen finden Sie unter Amazon ECR-Nutzungsberichte.

Um die Kosten kombinierter Ressourcen anzuzeigen, können Sie Ihre Fakturierungsinformationen nach Ressourcen mit gleichen Tag (Markierung)-Schlüsselwerten strukturieren. Beispielsweise können Sie mehrere Ressourcen mit einem bestimmten Anwendungsnamen markieren und dann Ihre Fakturierungsinformationen so organisieren, dass Sie die Gesamtkosten dieser Anwendung über mehrere Services hinweg sehen können. Weitere Informationen zum Einrichten eines Kostenverteilungsberichts mit Tags finden Sie unter Der monatliche Kostenverteilungsbericht im AWS Billing Benutzerhandbuch.

> **Note**
>
> Wenn Sie die Berichterstellung gerade erst aktiviert haben, werden die Daten für den aktuellen Monat nach 24 Stunden bereitgestellt.

Hinzufügen von Tags zu einem privaten Repository in Amazon ECR

Sie können Tags zu einem privaten Repository hinzufügen.

Informationen zu Namen und bewährten Methoden für Tags finden Sie unter Beschränkungen und Anforderungen für die Benennung von Tags und Bewährte Methoden im Tagging AWS Resources User Guide.

Hinzufügen von Tags zu einem Repository ()AWS-Managementkonsole

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.

2. Wählen Sie die zu verwendende Region in der Navigationsleiste aus.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Markieren Sie auf der Seite Repositorys das Kontrollkästchen neben dem Repository, das Sie taggen möchten.
5. Wählen Sie im Menü Aktion die Option Repository-Tags aus.
6. Wählen Sie auf der Seite Repository-Tags nacheinander Tags hinzufügen, Tag hinzufügen aus.
7. Geben Sie auf der Seite Repository-Tags bearbeiten den Schlüssel und Wert für jedes Tag an und klicken Sie auf Speichern.

Hinzufügen von Tags zu einem Repository (AWS CLI oder einer API)

Sie können ein oder mehrere Tags hinzufügen oder überschreiben, indem Sie die AWS CLI oder eine API verwenden.

- AWS CLI - Tag-Ressource
- API-Aktion - TagResource

Die folgenden Beispiele zeigen, wie Sie Tags mit dem hinzufügen AWS CLI.

Beispiel 1: Kennzeichnen Sie ein Repository

Der folgende Befehl kennzeichnet ein Repository.

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

Beispiel 2: Kennzeichnen Sie ein Repository mit mehreren Tags

Der folgende Befehl fügt einem Repository drei Tags hinzu.

```
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
```

Beispiel 3: Auflisten der Tags für ein Repository

Der folgende Befehl listet die mit einem Repository verknüpften Tags auf.

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

Beispiel 4: Erstellen Sie ein Repository und fügen Sie ein Tag hinzu

Der folgende Befehl erstellt ein Repository mit dem Namen `test-repo` und fügt ein Tag mit dem Schlüssel `team` und dem Wert `devs` hinzu.

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

Löschen von Tags aus einem privaten Repository in Amazon ECR

Sie können Tags aus einem privaten Repository löschen.

Um ein Tag aus einem privaten Repository zu löschen (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie die zu verwendende Region in der Navigationsleiste aus.
3. Markieren Sie auf der Seite Repositorys das Kontrollkästchen neben dem Repository, aus dem Sie ein Tag entfernen möchten.
4. Wählen Sie im Menü Aktion die Option Repository-Tags aus.
5. Wählen Sie auf der Seite Repository-Tags Bearbeiten aus.
6. Wählen Sie auf der Seite Repository-Tags bearbeiten für jedes Tag, das Sie löschen möchten, Entfernen und klicken Sie auf Speichern.

Um ein Tag aus einem privaten Repository zu löschen ()AWS CLI

Sie können ein oder mehrere Tags mithilfe der AWS CLI oder einer API löschen.

- AWS CLI - Untag-Ressource
- API-Aktion - UntagResource

Das folgende Beispiel zeigt, wie Sie ein Tag mit dem aus einem Repository löschen AWS CLI.

```
aws ecr untag-resource \
```

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

Private Bilder in Amazon ECR

Amazon ECR speichert Docker-Images, Open Container Initiative (OCI) -Images und OCI-kompatible Artefakte in privaten Repositorys. Sie können die Docker-Befehlszeilenschnittstelle (CLI) oder Ihren bevorzugten Client verwenden, um Images in Ihre Repositories zu pushen und von dort abzuziehen.

Mit der Amazon ECR-Unterstützung für OCI v1.1 können Sie Referenzartefakte speichern und verwalten, die von der OCI Referrers API definiert werden. Zu den Artefakten gehören Signaturen, Software-Stücklisten (SBoMs), Helmdiagramme, Scanergebnisse und Bescheinigungen. Ein Satz von Artefakten für ein Container-Image wird zusammen mit diesem Container übertragen und als separates Bild gespeichert, das als für Ihr Repository verbrauchtes Bild gilt.

Die Löschen von Signaturen und anderen Artefakten aus einem privaten Amazon ECR-Repository Seiten Bilder in Amazon ECR signieren und enthalten Beispiele für die Verwendung signaturbezogener Artefakte. Weitere Informationen zum Signieren von Container-Images finden Sie unter Signieren von Container-Images im AWS Signer Developer Guide.

Themen

Ein Bild in ein privates Amazon ECR-Repository übertragen

Sie können Ihre Docker-Images, Manifestlisten und Open Container Initiative (OCI)-Images sowie kompatible Artefakte in Ihre privaten Repositories verschieben.

Amazon ECR bietet auch eine Möglichkeit, Ihre Bilder in andere Repositorys zu replizieren. Indem Sie in Ihren privaten Registrierungseinstellungen eine Replikationskonfiguration angeben, können Sie regionsübergreifend in Ihrer eigenen Registrierung und über verschiedene Konten hinweg replizieren. Weitere Informationen finden Sie unter Private Registrierungseinstellungen in Amazon ECR.

> **Note**
>
> Wenn Sie ein Bild übertragen, das derzeit archiviert ist, wird dieses Bild automatisch wiederhergestellt und aus dem Archiv entfernt. Weitere Informationen zum Archivieren und Wiederherstellen von Bildern finden Sie unterArchivieren eines Bilds in Amazon ECR.

Themen

IAM-Berechtigungen für das Pushen eines Images in ein privates Amazon ECR-Repository

Benutzer benötigen IAM-Berechtigungen, um Bilder in private Amazon ECR-Repositorys zu übertragen. Gemäß der bewährten Methode der Gewährung der geringsten Rechte können Sie Zugriff auf ein bestimmtes Repository gewähren. Sie können auch Zugriff auf alle Repositorys gewähren.

Ein Benutzer muss sich bei jedem Amazon ECR-Register, in das er Images pushen möchte, authentifizieren, indem er ein Autorisierungs-Token anfordert. Amazon ECR bietet mehrere AWS verwaltete Richtlinien zur Steuerung des Benutzerzugriffs auf unterschiedlichen Ebenen. Weitere Informationen finden Sie unter AWS verwaltete Richtlinien für Amazon Elastic Container Registry.

Sie können auch Ihre eigenen IAM-Richtlinien erstellen. Die folgende IAM-Richtlinie gewährt die erforderlichen Berechtigungen für die Übertragung eines Images in ein bestimmtes Repository. Um die Berechtigungen für ein bestimmtes Repository einzuschränken, verwenden Sie den vollständigen Amazon-Ressourcennamen (ARN) des Repositorys.

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": "*"
        }
    ]
}
```

Die folgende IAM-Richtlinie gewährt die erforderlichen Berechtigungen für die Übertragung eines Images an alle Repositorys.

JSON

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

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

Ein Docker-Image in ein privates Amazon ECR-Repository übertragen

Sie können Ihre Container-Images mit dem Befehl docker push in ein Amazon ECR-Repository pushen.

Amazon ECR unterstützt auch die Erstellung und Übertragung von Docker-Manifestlisten, die für Images mit mehreren Architekturen verwendet werden. Weitere Informationen finden Sie unter Übertragung eines Images mit mehreren Architekturen in ein privates Amazon ECR-Repository.

So pushen Sie ein Docker-Image in ein Amazon ECR-Repository

Das Amazon ECR-Repository muss vorhanden sein, bevor Sie das Image übertragen, oder Sie müssen eine Vorlage für die Repository-Erstellung definiert haben. Weitere Informationen erhalten Sie unter Erstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern und Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.

1. Authentifizieren Sie Ihren Docker-Client bei der Amazon-ECR-Registrierung, in die Sie Ihr Image übertragen möchten. Für jede verwendete Registrierung muss ein Autorisierungs-Token erhalten werden, und die Token sind 12 Stunden lang gültig. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.

   Um Docker bei einer Amazon-ECR-Registrierung zu authentifizieren, führen Sie den Befehl aws ecr get-login-password aus. Verwenden Sie bei der Übergabe des Authentifizierungs-Tokens an den Befehl docker login den Wert `AWS` für den Benutzernamen und geben Sie die URI der Amazon-ECR-Registrierung an, bei der Sie sich authentifizieren möchten. Wenn Sie sich

bei mehreren Registrierungen authentifizieren, müssen Sie den Befehl für jede Registrierung wiederholen.

> **Important**
>
> Bei einem Fehler installieren oder aktualisieren Sie auf die neueste AWS CLI-Version. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface -Benutzerhandbuch.

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

2. Wenn Ihr Image-Repository noch nicht in der Registry vorhanden ist, in die Sie Daten übertragen möchten, und Sie eine Vorlage für die Repository-Erstellung definiert haben, können Sie Ihr Image mit dem Präfix Ihrer Repository-Erstellungsvorlage und Ihrem gewünschten Repository-Namen pushen. ECR erstellt das Repository automatisch für Sie unter Verwendung der vordefinierten Einstellungen Ihrer Repository-Erstellungsvorlage.

   Wenn Sie keine passende Vorlage für die Erstellung eines Repositorys definiert haben, müssen Sie ein Repository erstellen. Für weitere Informationen siehe Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden oder Erstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern.

3. Identifizieren Sie das zu pushende lokale Image. Führen Sie den Befehl docker images aus, um die Container-Images auf Ihrem System aufzulisten.

   ```
   docker images
   ```

   Sie können ein Bild anhand des *repository:tag* Werts oder der Bild-ID in der resultierenden Befehlsausgabe identifizieren.

4. Markieren Sie Ihr Image mit der zu verwendenden Kombination aus Amazon ECR-Registrierung, Repository und optionalem Image-Tag-Namen. Das Registrierungsformat ist *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com`. Der Repository-Name sollte mit dem Repository übereinstimmen, das Sie für Ihr Image erstellt haben. Wenn Sie das Image-Tag weglassen, nehmen wir an, dass das Tag `latest` ist.

Im folgenden Beispiel wird ein lokales Bild mit der ID *e9ae3c220b23* as gekennzeichnet*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. Pushen Sie das Image mit dem Befehl docker push:

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

6. (Optional) Wenden Sie zusätzliche Tags auf Ihr Image an und übertragen Sie diese Tags an Amazon ECR, indem Sie die Schritte Step 4 und Step 5 wiederholen.

Übertragung eines Images mit mehreren Architekturen in ein privates Amazon ECR-Repository

Sie können Images mit mehreren Architekturen in ein Amazon ECR-Repository übertragen, indem Sie Docker-Manifestlisten erstellen und per Push übertragen. Eine Manifestliste ist eine Liste von Images, die durch Angabe eines oder mehrerer Image-Namen erstellt wird. In den meisten Fällen wird die Manifestliste aus Images erstellt, die dieselbe Funktion erfüllen, aber für unterschiedliche Betriebssysteme oder Architekturen bestimmt sind. Die Manifestliste ist nicht erforderlich. Weitere Informationen finden Sie unter Docker-Manifest.

Eine Manifestliste kann in einer Amazon ECS-Aufgabendefinition oder Amazon EKS-Pod-Spezifikation wie andere Amazon ECR-Images abgerufen oder referenziert werden.

Voraussetzungen

- Aktivieren Sie in Ihrer Docker-CLI experimentelle Funktionen. Informationen zu experimentellen Funktionen finden Sie unter Experimentelle Funktionen in der Docker-Dokumentation.
- Das Amazon ECR-Repository muss vorhanden sein, bevor Sie das Image pushen. Weitere Informationen finden Sie unter the section called “Erstellen eines Repositorys zum Speichern von Bildern”.
- Bilder müssen in Ihr Repository übertragen werden, bevor Sie das Docker-Manifest erstellen. Informationen über das Pushen eines Images finden Sie unter Ein Docker-Image in ein privates Amazon ECR-Repository übertragen.

So pushen Sie ein Multi-Architektur-Docker-Image in ein Amazon ECR-Repository

1. Authentifizieren Sie Ihren Docker-Client bei der Amazon-ECR-Registrierung, in die Sie Ihr Image übertragen möchten. Für jede verwendete Registrierung muss ein Autorisierungs-Token erhalten werden, und die Token sind 12 Stunden lang gültig. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.

   Um Docker bei einer Amazon-ECR-Registrierung zu authentifizieren, führen Sie den Befehl aws ecr get-login-password aus. Verwenden Sie bei der Übergabe des Authentifizierungs-Tokens an den Befehl docker login den Wert `AWS` für den Benutzernamen und geben Sie die URI der Amazon-ECR-Registrierung an, bei der Sie sich authentifizieren möchten. Wenn Sie sich bei mehreren Registrierungen authentifizieren, müssen Sie den Befehl für jede Registrierung wiederholen.

   > **Important**
   >
   > Bei einem Fehler installieren oder aktualisieren Sie auf die neueste AWS CLI-Version. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface -Benutzerhandbuch.

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

2. Listen Sie die Images in Ihrem Repository auf und bestätigen Sie die Image-Tags.

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

3. Erstellen Sie die Docker-Manifestliste. Mit dem Befehl `manifest create` wird überprüft, ob sich die referenzierten Images bereits in Ihrem Repository befinden, und das Manifest lokal erstellt.

   ```
   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. (Optional) Überprüfen Sie die Docker-Manifestliste. Auf diese Weise können Sie die Größe und den Digest für jedes Image-Manifest bestätigen, auf das in der Manifestliste verwiesen wird.

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

5. Pushen Sie die Docker-Manifestliste in Ihr Amazon ECR-Repository.

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

Übertragung eines Helm-Diagramms in ein privates Amazon ECR-Repository

Sie können Artefakte der Open Container Initiative (OCI) in ein Amazon ECR-Repository übertragen. Um ein Beispiel für diese Funktionalität zu sehen, verwenden Sie die folgenden Schritte, um ein Helm-Diagramm an Amazon ECR zu übertragen.

Informationen zur Verwendung Ihrer von Amazon ECR gehosteten Helm-Charts mit Amazon EKS finden Sie unterInstallation eines Helm-Diagramms auf einem Amazon EKS-Cluster.

So pushen Sie ein Helm-Diagramm in ein Amazon ECR-Repository

1. Installieren Sie die neueste Version des Helm-Clients. Diese Schritte wurden mit Helm Version `3.18.6` geschrieben. Verwenden Sie Helm Version 3.9 oder höher, um die Kompatibilität mit Kubernetes-Versionen zu gewährleisten, die von Amazon EKS unterstützt werden. Weitere Informationen finden Sie unter Installation von Helm.
2. Verwenden Sie die folgenden Schritte, um ein Helm-Testdiagramm zu erstellen. Weitere Informationen finden Sie unter Helm Docs - Erste Schritte.

   a. Erstellen Sie ein Helm-Diagramm mit dem Namen `helm-test-chart` und löschen Sie den Inhalt des Verzeichnisses `templates`.

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

   b. Erstellen Sie eine `ConfigMap` im Ordner. `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. Verpacken Sie die Karte. Die Ausgabe enthält den Dateinamen des verpackten Diagramms, den Sie beim Pushen des Helm-Diagramms verwenden.

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

Ausgabe

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

4. Erstellen Sie ein Repository, um Ihr Helm-Diagramm zu speichern. Der Name Ihres Repositorys muss dem Namen entsprechen, den Sie bei der Erstellung des Helm-Charts in Schritt 2 verwendet haben. Weitere Informationen finden Sie unter Erstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern.

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

5. Authentifizieren Sie Ihren Helm-Client bei dem Amazon-ECR-Registrierung, in das Sie Ihr Helm-Diagramm verschieben möchten. Für jede verwendete Registrierung muss ein Autorisierungs-Token erhalten werden, und die Token sind 12 Stunden lang gültig. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in 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. Drücken Sie die Steuerkarte mit dem Befehl helm push. Die Ausgabe sollte den Amazon ECR-Repository-URI und den SHA-Digest enthalten.

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

7. Beschreiben Sie Ihr Helm-Diagramm.

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

Überprüfen Sie in der Ausgabe, ob der Parameter `artifactMediaType` den richtigen Artefakttyp angibt.

```
{
    "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. (Optional) Installieren Sie für weitere Schritte den Helm `ConfigMap` und beginnen Sie mit Amazon EKS. Weitere Informationen finden Sie unter Installation eines Helm-Diagramms auf einem Amazon EKS-Cluster.

Löschen von Signaturen und anderen Artefakten aus einem privaten Amazon ECR-Repository

Sie können den ORAS-Client verwenden, um Signaturen und andere Referenztyp-Artefakte aus einem privaten Amazon ECR-Repository aufzulisten und zu löschen. Das Löschen von Signaturen und anderen Referenzartefakten ähnelt dem Löschen eines Bilds (siehe Löschen eines Bilds in Amazon ECR). Gehen Sie wie folgt vor, um Artefakte aufzulisten und Signaturen zu löschen:

So verwalten Sie Bildartefakte mit der ORAS-CLI

1. Installieren und konfigurieren Sie den ORAS-Client.

   Informationen zur Installation und Konfiguration des ORAS-Clients finden Sie in der ORAS-Dokumentation unter Installation.

2. Um verfügbare Artefakte für ein Amazon ECR-Image aufzulisten, verwenden Sie `oras discover`, gefolgt von einem Image-Namen:

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

   Die Ausgabe sollte in etwa so aussehen:

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

3. Um eine Signatur mithilfe der ORAS-CLI zu löschen, führen Sie anhand des vorherigen Beispiels den folgenden Befehl aus:

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

   Die Ausgabe sollte in etwa so aussehen:

   ```
   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. Drücken Sie `y`. Das Artefakt sollte gelöscht werden.

Um Probleme beim Löschen von Artefakten zu beheben

Sollte das Löschen einer Signatur, wie z. B. das gerade gezeigte, fehlschlagen, wird eine Ausgabe ähnlich der folgenden angezeigt.

```
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]
```

Dieser Fehler kann auftreten, wenn ein Bild gelöscht wird, das vor dem Start von OCI 1.1 übertragen wurde. Wie in dem Fehler angegeben, müssen Sie das Manifest löschen, das auf das Bild verweist, bevor Sie das Bild wie folgt löschen können:

1. Geben Sie Folgendes ein, um das Manifest zu löschen, das der Signatur zugeordnet ist, die Sie löschen möchten:

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

   Die Ausgabe sollte in etwa so aussehen:

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

2. Drücken Sie y. Das Manifest sollte gelöscht werden.
3. Wenn das Manifest weg ist, können Sie die Signatur löschen:

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

   Die Ausgabe sollte in etwa so aussehen. Drücken Sie 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. Um zu sehen, dass die Signatur gelöscht wurde, geben Sie Folgendes ein:

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

Die Ausgabe sollte in etwa so aussehen:

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

Bilddetails in Amazon ECR anzeigen

Nachdem Sie ein Bild in Ihr Repository übertragen haben, können Sie Informationen dazu anzeigen. Die Details sind im Folgenden aufgeführt:

- Image-URI
- Image-Tags
- Artifact Medientyp
- Typ des Image-Manifests
- Status des Scannens
- Die Größe des Images in MB
- Wann das Image per Push zum Repository übertragen wurde
- Der Replikationsstatus

So zeigen Sie Image-Details an (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie in der Navigationsleiste die Region aus, die das Repository mit Ihrem Image enthält.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das anzuzeigende Repository aus.
5. Wählen Sie auf der *repository_name* Seite Repositories: das Bild aus, dessen Details Sie anzeigen möchten.

Abrufen eines Images aus einem privaten Amazon ECR-Repository in Ihre lokale Umgebung

Wenn Sie ein Docker-Image ausführen möchten, das in Amazon ECR verfügbar ist, können Sie es mit dem Befehl docker pull in Ihre lokale Umgebung ziehen. Sie können dies entweder von Ihrer Standardregistrierung oder von einer Registrierung aus tun, die mit einem anderen AWS Konto verknüpft ist.

Um ein Amazon-ECR-Image in einer Amazon-ECS-Aufgabendefinition zu verwenden, siehe Verwendung von Amazon ECR-Bildern mit Amazon ECS.

> **Important**
>
> Sie können ein archiviertes Bild nicht abrufen. Archivierte Bilder müssen wiederhergestellt werden, bevor sie abgerufen werden können. Weitere Informationen zum Archivieren und Wiederherstellen von Bildern finden Sie unterArchivieren eines Bilds in Amazon ECR.

> **Important**
>
> Amazon ECR setzt voraus, dass Benutzer über eine IAM-Richtlinie die Erlaubnis haben, die `ecr:GetAuthorizationToken`-API aufzurufen, bevor sie sich bei einer Registrierung authentifizieren und Images aus einem Amazon ECR-Repository pushen oder pullen können. Amazon ECR bietet mehrere AWS verwaltete Richtlinien zur Steuerung des Benutzerzugriffs auf unterschiedlichen Ebenen. Informationen zu den AWS verwalteten Richtlinien für Amazon ECR finden Sie unterAWS verwaltete Richtlinien für Amazon Elastic Container Registry.

So pullen Sie ein Docker-Image aus einem Amazon ECR-Repository

1. Authentifizieren Sie Ihren Docker-Client bei der Amazon-ECR-Registrierung, aus der Sie Ihr Image abrufen möchten. Für jede verwendete Registrierung muss ein Autorisierungs-Token erhalten werden, und die Token sind 12 Stunden lang gültig. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.
2. (Optional) Identifizieren Sie das abzurufende Image.
   - Sie können die Repositorys in einer Registrierung mit dem Befehl aws ecr describe-repositories auflisten:

     ```
     aws ecr describe-repositories
     ```

     Die obige Beispielregistrierung hat ein Repository namens `amazonlinux`.
   - Sie können die Images in einem Repository mit dem Befehl aws ecr describe-images beschreiben:

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

     Das obige Beispie-Repsotoryl zeigt ein als `latest` und `2016.09` markiertes Image, mit dem Image-Digest `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807`.
3. Rufen Sie das Image mit dem Befehl docker pull ab. Das Image-Namensformat sollte *registry*/*repository* [:*tag*] für den Abruf per Tag, oder *registry*/*repository*[@*digest*] für den Abruf per Digest sein.

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

   > **Important**
   >
   > Wenn Sie eine Fehlermeldung *repository-url* `not found: does not exist or no pull access` erhalten, müssen Sie Ihren Docker-Client möglicherweise mit Amazon ECR authentifizieren. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.

Das Amazon Linux-Container-Image abrufen

Das Amazon Linux Container-Image wird aus denselben Softwarekomponenten erstellt, die auch im Amazon Linux AMI enthalten sind. Das Amazon Linux-Container-Image kann in jeder Umgebung als Basis-Image für Docker-Workloads verwendet werden. Wenn Sie das Amazon Linux AMI für Anwendungen in Amazon verwenden EC2, können Sie Ihre Anwendungen mit dem Amazon Linux-Container-Image containerisieren.

Sie können das Amazon Linux-Container-Image in Ihrer lokalen Entwicklungsumgebung AWS verwenden und dann Ihre Anwendung auf Amazon ECS übertragen. Weitere Informationen finden Sie unter Verwendung von Amazon ECR-Bildern mit Amazon ECS.

Das Amazon Linux Container-Image ist auf Amazon ECR Öffentlich und auf Docker Hub verfügbar. Unterstützung für das Amazon Linux-Container-Image finden Sie in den AWS Entwicklerforen.

So pullen Sie das Amazon Linux-Container-Image von Amazon ECR Öffentlich

1. Authentifizieren Sie Ihren Docker-Client bei der Amazon-Linux-Public-Registrierung. Authentifizierungs-Token sind 12 Stunden lang gültig. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.

   > **Note**
   >
   > Die ecr-public-Befehle sind in der AWS CLI ab Version `1.18.1.187` verfügbar. Wir empfehlen jedoch, die neueste Version der AWS CLI zu verwenden. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface -Benutzerhandbuch.

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

   Die Ausgabe sieht wie folgt aus:

   ```
   Login succeeded
   ```

2. Rufen Sie das Amazon-Linux-Container-Image mit dem Befehl docker pull ab. Um das Amazon Linux-Container-Image in der Amazon ECR Public Gallery anzuzeigen, siehe Amazon ECR Public Gallery – amazonlinux.

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

3. (Optional) Führen Sie den Container lokal aus.

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

So pullen Sie das Amazon Linux-Container-Image aus Docker Hub

1. Rufen Sie das Amazon-Linux-Container-Image mit dem Befehl docker pull ab.

```
docker pull amazonlinux
```

2. (Optional) Führen Sie den Container lokal aus.

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

Löschen eines Bilds in Amazon ECR

Wenn Sie ein Image nicht mehr verwenden möchten, können Sie es aus Ihrem Repository löschen. Wenn Sie mit einem Repository fertig sind, können Sie das gesamte Repository und alle darin enthaltenen Images löschen. Weitere Informationen finden Sie unter Löschen eines privaten Repositorys in Amazon ECR.

Als Alternative zum manuellen Löschen von Images können Sie Repository-Lebenszyklusrichtlinien erstellen, die eine bessere Kontrolle über die Verwaltung des Lebenszyklus von Images in Ihren Repositories ermöglichen. Lebenszyklusrichtlinien automatisieren diesen Prozess für Sie. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

> **Note**
>
> Wenn Ihr Repository eine Mischung aus Bildern enthält, von denen einige übertragen wurden, bevor Amazon ECR OCI v1.1 unterstützte, weisen bei einigen Signaturen Bildindizes oder Manifestlisten darauf hin. Wenn Sie ein Image vor OCI v1.1 löschen, müssen Sie daher möglicherweise die Manifestliste, die auf das Bild verweist, manuell löschen, um das Artefakt zu löschen.

So löschen Sie ein Image (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der das zu löschende Image enthalten ist.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositories das Repository aus, das das zu löschende Image enthält.
5. Wählen Sie auf der ***repository_name*** Seite Repositories: das Kästchen links neben dem Bild aus, das Sie löschen möchten, und wählen Sie Löschen aus.
6. Überprüfen Sie im Dialogfeld Delete image(s), ob die ausgewählten Images wirklich gelöscht werden sollen, und wählen Sie dann Delete.

So löschen Sie ein Image (AWS CLI)

1. Listen Sie die Images in Ihrem Repository auf. Markierte Images haben sowohl einen Image-Digest als auch eine Liste der zugehörigen Tags. Nur unmarkierte Images enthalten einen Image-Digest.

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

2. (Optional) Löschen Sie unerwünschte Tags für das Image, indem Sie das Tag angeben, die mit dem zu löschenden Image verbunden ist. Wenn das letzte Tag von einem Image gelöscht wird, wird auch das Image gelöscht.

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

3. Löschen Sie ein markiertes oder unmarkiertes Image, indem Sie den Image-Digest angeben. Wenn Sie ein Image löschen, indem Sie auf seinen Digest verweisen, werden das Image und alle seine Tags gelöscht.

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

Um mehrere Images zu löschen, können Sie in der Anfrage mehrere Image-Tags oder Image-Digests angeben.

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

Archivieren eines Bilds in Amazon ECR

Was ist die ECR-Archivspeicherklasse?

Die Amazon ECR Archival Storage Class ist eine neue Speicherklasse, die kostengünstigen Langzeitspeicher für Container-Images bietet. Amazon ECR bietet zwei Speicherklassen:

- ECR-Standardspeicherklasse — Die Standardspeicherklasse für aktive Bilder, auf die regelmäßig zugegriffen wird.
- ECR-Archivierungsspeicherklasse — Eine kostengünstige Speicherklasse für Bilder, auf die selten zugegriffen wird, die aber aus Compliance-Gründen oder zur langfristigen Referenz aufbewahrt werden müssen. Die Archivspeicherklasse bietet im Vergleich zur Standardspeicherklasse für die langfristige Aufbewahrung von Bildern Kosteneinsparungen bei großen Bildmengen. Detaillierte Preisinformationen finden Sie unter Amazon ECR-Preise.

Um Bilder zu archivieren, haben Sie zwei Möglichkeiten. Erstens können Sie Lebenszyklusregeln so konfigurieren, dass Bilder automatisch archiviert werden, basierend auf:

- Zeit, seit das Bild übertragen wurde
- Zeit seit dem letzten Abruf des Bilds
- Anzahl der Bilder im Repository

Sie können auch Einstellungen so konfigurieren, dass Bilder dauerhaft gelöscht werden, nachdem sie für einen bestimmten Zeitraum archiviert wurden. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

Sie können Bilder auch mit der Amazon ECR-Konsole oder AWS CLI archivieren. Weitere Informationen finden Sie unter Archivieren eines Images.

Wenn Sie ein archiviertes Bild erneut verwenden müssen, können Sie es auf die ECR Standard-Speicherklasse zurücksetzen. Sie können davon ausgehen, dass ECR das Image innerhalb von 20 Minuten wiederherstellt. Wiederhergestellte Bilder verhalten sich wie neu übertragene Bilder und können sofort verwendet werden, wenn die Wiederherstellung abgeschlossen ist. Wiederhergestellte Images unterliegen Richtlinien für das Scannen, die Replikation und den Lebenszyklus des Repositorys. Weitere Informationen finden Sie unter Ein Bild wiederherstellen.

Archivieren eines Images

Sie können Bilder manuell über die Amazon ECR-Konsole oder AWS CLI automatisch mithilfe von Lebenszyklusrichtlinien archivieren. Wenn ein Bild archiviert wird:

- Das Bild wird in die Archivspeicherklasse verschoben.
- Archivierte Bilder können nicht abgerufen werden. Anfragen zum Abrufen des archivierten Bilds schlagen mit einem 404-Fehler fehl.
- Das Bild kann zwar nicht abgerufen werden, kann aber dennoch mit dem describe-images Befehl beschrieben oder mit dem list-images Befehl aufgelistet werden. Der Bildstatus wird als angezeigt`ARCHIVED`.
- Archivierte Bilder haben eine Mindestspeicherdauer von 90 Tagen. Sie können keine Lebenszyklusrichtlinien konfigurieren, mit denen Bilder gelöscht werden, die sich seit weniger als 90 Tagen im Archiv befinden. Wenn Sie Bilder löschen müssen, die seit weniger als 90 Tagen archiviert wurden, müssen Sie die batch-delete-image API verwenden. Die Mindestspeicherdauer von 90 Tagen wird Ihnen jedoch in Rechnung gestellt.
- Das Bild wird in der Repository-Ansicht auf der Registerkarte Archivierte Bilder angezeigt (diese Registerkarte wird nur angezeigt, wenn mindestens ein Bild im Repository archiviert ist).
- Das Bild kann als aktives Bild wiederhergestellt werden, indem es manuell ausgewählt wird, um es wiederherzustellen, oder indem das Bild erneut in das Repository übertragen wird.
- Das Bild wird gelöscht, wenn das Repository über Lebenszyklusrichtlinien verfügt, die das Bild anhand von Kriterien wie der Archivierungszeit löschen.

AWS-Managementkonsole

Um ein Bild zu archivieren

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.

2. Wählen Sie in der Navigationsleiste die Region aus, die das Repository mit dem Bild enthält, das Sie archivieren möchten.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das Repository aus, das das Bild enthält, das Sie archivieren möchten.
5. Wählen Sie das Bild aus, das Sie archivieren möchten. Sie werden Bilddetails sehen.
6. Um das Bild zu archivieren, klicken Sie auf Archivieren und wählen Sie Bestätigen, wenn Sie dazu aufgefordert werden.
7. Wenn dies das erste archivierte Bild im Repository ist, wird ein neuer Tab Archivierte Bilder mit dem neu archivierten Bild angezeigt. Wenn es andere archivierte Bilder gibt, wird dieses Bild zu dieser Registerkarte hinzugefügt.

AWS CLI

Um ein Bild zu archivieren

- Verwenden Sie den update-image-storage-class Befehl, um ein Bild zu archivieren, indem Sie seine Speicherklasse wie folgt aktualisieren`ARCHIVE`:

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

Um ein Bild mithilfe von Lebenszyklusrichtlinien zu archivieren

- Sie können Archivierungsregeln für Ihre Repositorys mithilfe von Lebenszyklusrichtlinien konfigurieren, um Bilder automatisch zu archivieren. Mithilfe von Lebenszyklusrichtlinien können Sie Bilder anhand von Kriterien wie den folgenden automatisch archivieren:
  - Zeit, seit das Bild übertragen wurde
  - Zeit seit dem letzten Abruf des Bilds
  - Maximale Anzahl von Bildern, die aktiv bleiben sollen

Sie können Lebenszyklusrichtlinien auch so konfigurieren, dass Bilder dauerhaft gelöscht werden, nachdem sie für einen bestimmten Zeitraum archiviert wurden. Weitere Informationen und Beispiele für Lebenszyklusrichtlinien mit Archivierungsaktionen finden Sie unterAutomatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

> **Note**
>
> Archivierte Bilder haben eine Mindestspeicherdauer von 90 Tagen. Sie können keine Lebenszyklusrichtlinien konfigurieren, mit denen Bilder gelöscht werden, die sich seit weniger als 90 Tagen im Archiv befinden. Wenn Sie Bilder löschen müssen, die seit weniger als 90 Tagen archiviert wurden, müssen Sie die batch-delete-image API verwenden. Die Mindestspeicherdauer von 90 Tagen wird Ihnen jedoch in Rechnung gestellt.

Wenn Sie Bilder mit dem describe-images Befehl beschreiben, haben archivierte Bilder den Wert `image-status` von`ARCHIVED`. Sie können Bilder filtern`image-status`, um nur archivierte Bilder oder nur aktive Bilder anzuzeigen.

Ein Bild wiederherstellen

Wenn Sie ein archiviertes Bild wiederherstellen, wird es von der Speicherklasse ECR Archive zurück in die Speicherklasse ECR Standard verschoben. Wiederhergestellte Bilder werden zu den Standardspeicherpreisen berechnet. Bei der Wiederherstellung werden ähnliche Aktionen ausgeführt wie bei der Erstellung eines neuen Images:

- Das Image kann abgerufen werden, wenn die Wiederherstellung abgeschlossen ist. Die Wiederherstellung dauert in der Regel bis zu 20 Minuten, kann aber auch schneller abgeschlossen werden.
- Wenn Scan on Push für das Repository aktiviert ist, wird das wiederhergestellte Bild gescannt. Beachten Sie, dass frühere Scanergebnisse aus der Zeit vor der Archivierung des Bilds nicht verfügbar sein werden.
- Wenn die Replikation für das Repository konfiguriert ist, wird das wiederhergestellte Image repliziert, sofern die Replikation zum Zeitpunkt der Wiederherstellung aktiviert war.
- Das wiederhergestellte Bild wird in der Liste der aktiven Bilder angezeigt.

Das Wiederherstellen eines Bilds dauert in der Regel bis zu 20 Minuten, kann aber auch schneller abgeschlossen werden. Während des Wiederherstellungsvorgangs verbleibt das Image im archivierten Zustand und kann erst abgerufen werden, wenn die Wiederherstellung abgeschlossen ist.

AWS-Managementkonsole

So stellen Sie ein archiviertes Image wieder her

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie in der Navigationsleiste die Region aus, die das Repository mit dem archivierten Bild enthält, das Sie wiederherstellen möchten.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das Repository aus, das das archivierte Bild enthält.
5. Wählen Sie den Tab Archivierte Bilder.
6. Wählen Sie das archivierte Bild aus, das Sie wiederherstellen möchten.
7. Wählen Sie Wiederherstellen und bestätigen Sie die Wiederherstellungsaktion.
8. Warten Sie, bis die Wiederherstellung abgeschlossen ist. Das Bild wird in der Liste der aktiven Bilder angezeigt, sobald die Wiederherstellung abgeschlossen ist.

AWS CLI

Um ein archiviertes Bild wiederherzustellen

- Verwenden Sie den update-image-storage-class Befehl, um ein archiviertes Bild wiederherzustellen, indem Sie seine Speicherklasse wie folgt aktualisierenSTANDARD:

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

Wenn Sie Bilder mit dem describe-images Befehl beschreiben, haben Bilder, die wiederhergestellt werden, den Wert `image-status` von`ACTIVATING`. Sie können Bilder nach `image-status` dem Wert filtern`ACTIVATING`, um Bilder anzuzeigen, die gerade wiederhergestellt werden.

Eine alternative Methode zum Wiederherstellen eines archivierten Bilds besteht darin, das Bild erneut in das Repository zu übertragen. Wenn Sie ein Bild übertragen, das derzeit archiviert ist, wird dieses Bild sofort wiederhergestellt und aus dem Archiv entfernt.

Ein Bild in Amazon ECR neu taggen

Bei Docker Image Manifest V2 Schema 2-Images können Sie mit der Option `--image-tag` des Befehls put-image ein vorhandenes Image erneut markieren. Eine erneute Markierung ist möglich, ohne das Image per Push oder Pull mit Docker zu übertragen. Bei umfangreichen Images lassen sich so die benötigte Netzwerkbandbreite und der Zeitaufwand, der zum erneuten Markieren eines Image nötig ist, ganz erheblich reduzieren.

So markieren Sie ein Image neu (AWS CLI)

Um ein Bild erneut zu taggen mit dem AWS CLI

1. Verwenden Sie den batch-get-image-Befehl, um das Image-Manifest für das Image abzurufen, um es neu zu markieren und in eine Datei zu schreiben. In diesem Beispiel wird das Manifest für ein Bild mit dem Tag,*`latest`*, im Repository,*`amazonlinux`*, in eine Umgebungsvariable mit dem Namen *`MANIFEST`* geschrieben.

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

2. Verwenden Sie die Option `--image-tag` des Befehls put-image, um das Image-Manifest in Amazon ECR mit einem neuen Tag zu versehen. In diesem Beispiel ist das Bild als gekennzeichnet*`2017.03`*.

   > **Note**
   >
   > Wenn die `--image-tag` Option in Ihrer Version von nicht verfügbar ist AWS CLI, führen Sie ein Upgrade auf die neueste Version durch. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface - Benutzerhandbuch.

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

3. Vergewissern Sie sich, dass Ihr neues Image-Tag mit Ihrem Image verbunden ist. In der nachfolgenden Ausgabe hat das Image die Tags `latest` und `2017.03`.

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

   Die Ausgabe sieht wie folgt aus:

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

So markieren Sie ein Image neu (AWS Tools for Windows PowerShell)

Um ein Bild erneut zu taggen mit dem AWS Tools for Windows PowerShell

1. Verwenden Sie die Get-ECRImageBatchcmdlet, um die Beschreibung des Bilds abzurufen, um es erneut zu taggen und in eine Umgebungsvariable zu schreiben. In diesem Beispiel wird ein Bild mit dem Tag, *`latest`*, im Repository,*`amazonlinux`*, in die Umgebungsvariable, *`$Image`* geschrieben.

> **Note**
>
> Falls das nicht auf Ihrem System Get-ECRImageBatch cmdlet verfügbar ist, finden Sie weitere Informationen unter Einrichten von AWS Tools for Windows PowerShell im AWS -Tools für PowerShell Benutzerhandbuch.

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

2. Schreiben Sie das Manifest des Images in die *$Manifest* Umgebungsvariable.

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

3. Verwenden Sie die `-ImageTag` Option von Write-ECRImage cmdlet, um das Image-Manifest mit einem neuen Tag in Amazon ECR zu speichern. In diesem Beispiel ist das Bild gekennzeichnet als*2017.09*.

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

4. Vergewissern Sie sich, dass Ihr neues Image-Tag mit Ihrem Image verbunden ist. In der nachfolgenden Ausgabe hat das Image die Tags `latest` und `2017.09`.

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

Die Ausgabe sieht wie folgt aus:

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

Verhindern, dass Bild-Tags in Amazon ECR überschrieben werden

Sie können verhindern, dass Bild-Tags überschrieben werden, indem Sie die Tag-Unveränderlichkeit in einem Repository aktivieren. Nachdem die Unveränderlichkeit von Tags aktiviert wurde, wird

der `ImageTagAlreadyExistsException` Fehler zurückgegeben, wenn Sie ein Bild mit einem Tag übertragen, das sich bereits im Repository befindet. Die Unveränderlichkeit von Tags wirkt sich auf alle Tags aus. Sie können einige Tags nicht unveränderlich machen, während andere nicht unveränderlich sind.

Sie können die AWS CLI Tools AWS-Managementkonsole und verwenden, um die Veränderbarkeit von Image-Tags für ein neues Repository oder für ein vorhandenes Repository festzulegen. Informationen zum Erstellen eines Repositorys mithilfe von Konsolenschritten finden Sie unterErstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern.

Einstellung der Veränderbarkeit von Bild-Tags ()AWS-Managementkonsole

Um die Veränderbarkeit von Bild-Tags festzulegen

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der das zu bearbeitende Repository enthalten ist.
3. Wählen Sie im Navigationsbereich unter Private Registrierung die Option Repositories aus.

   Wenn Repositorys nicht angezeigt werden, wählen Sie Private Registrierung aus, um das Menü zu erweitern, und wählen Sie dann Repositorys aus.
4. Wählen Sie auf der Seite Private Repositorys das Optionsfeld vor dem Namen des Repositorys aus, für das Sie die Veränderbarkeitseinstellungen für Image-Tags festlegen möchten.
5. Wählen Sie Aktionen und dann unter Bearbeiten die Option Repository aus.
6. Wählen Sie für die Unveränderlichkeit von Image-Tags eine der folgenden Tag-Veränderbarkeitseinstellungen für das Repository aus.

   - Veränderbar — Wählen Sie diese Option, wenn Sie möchten, dass Bild-Tags überschrieben werden. Empfohlen für Repositorys, die Pull-Through-Cache-Aktionen verwenden, um sicherzustellen, dass Amazon ECR zwischengespeicherte Bilder aktualisieren kann. Um Tag-Updates für einige veränderbare Tags zu deaktivieren, geben Sie außerdem Tag-Namen ein oder verwenden Sie Platzhalter (*), um mehrere ähnliche Tags im Textfeld zum Ausschluss veränderbarer Tags abzugleichen.
   - Unveränderlich — Wählen Sie diese Option, wenn Sie verhindern möchten, dass Bild-Tags überschrieben werden. Sie gilt für alle Tags und Ausschlüsse im Repository, wenn Sie ein Bild mit vorhandenem Tag pushen. Amazon ECR gibt eine zurück, `ImageTagAlreadyExistsException` wenn Sie versuchen, ein Bild mit einem vorhandenen

Tag zu pushen. Um Tag-Updates für einige unveränderliche Tags zu aktivieren, geben Sie außerdem Tagnamen ein oder verwenden Sie Platzhalter (*), um mehrere ähnliche Tags im Textfeld zum Ausschluss unveränderlicher Tags abzugleichen.

7. Obwohl Sie bei Image-Scaneinstellungen die Scaneinstellungen auf Repository-Ebene für das grundlegende Scannen angeben können, empfiehlt es sich, die Scankonfiguration auf privater Registrierungsebene anzugeben. Geben Sie die Scaneinstellungen in der privaten Registrierung an, um entweder das erweiterte Scannen oder das grundlegende Scannen zu aktivieren sowie Filter zu definieren, um anzugeben, welche Repositorys gescannt werden. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.
8. Für Verschlüsselungseinstellungen ist dies ein Nur-Ansichtsfeld, da die Verschlüsselungseinstellungen für ein Repository nicht geändert werden können, sobald das Repository erstellt wurde.
9. Wählen Sie Speichern aus, um die Repository-Einstellungen zu aktualisieren.

Einstellung der Veränderbarkeit von Bild-Tags ()AWS CLI

So erstellen Sie ein Repository, das für unveränderlichen Tags konfiguriert ist

Verwenden Sie einen der folgenden Befehle, um ein neues Image-Repository mit unveränderlichen Tags zu erstellen.

- create-repository (AWS CLI) mit Veränderbarkeit von Image-Tags

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

- create-repository (AWS CLI) mit Filtern zum Ausschluss der Veränderbarkeit von Bild-Tags

```
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) mit Veränderbarkeit des Bild-Tags

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

- Neu- ECRRepository (AWS Tools for Windows PowerShell) mit Filtern zum Ausschluss der Veränderbarkeit von Bild-Tags

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

Um die Mutabilitätseinstellungen für Bild-Tags für ein Repository zu aktualisieren

Verwenden Sie einen der folgenden Befehle, um die Einstellungen zur Veränderlichkeit von Image Tags für ein vorhandenes Repository zu aktualisieren.

- put-image-tag-mutability(AWS CLI) mit Bild-Tag-Veränderlichkeit

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

- put-image-tag-mutability(AWS CLI) mit Filtern zum Ausschluss der Veränderlichkeit von Bild-Tags

```
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- ECRImage TagMutability (AWS Tools for Windows PowerShell) mit Bildtag-Veränderlichkeit

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

- Write- ECRImage TagMutability (AWS Tools for Windows PowerShell) mit Filtern zum Ausschluss der Veränderlichkeit von Bild-Tags

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

Unterstützung des Container-Image-Manifestformats in Amazon ECR

Amazon ECR unterstützt die folgenden Container-Image-Manifestformate:

- Docker Image Manifest V2 Schema 1 (mit Docker-Version 1.9 und älter)
- Docker Image Manifest V2 Schema 2 (mit Docker-Version 1.10 und neuer)
- Spezifikationen der Open Container Initiative (OCI) (v1.0 und v1.1)

Die Unterstützung für Docker Image-Manifest V2 Schema 2 bietet folgende Funktionalität:

- Die Möglichkeit, mehrere Tags für ein einzelnes Image zu verwenden.
- Windows-Container-Images können gespeichert werden.

Amazon ECR-Image-Manifest-Konvertierung

Wenn Sie Images zu und von Amazon ECR schieben und ziehen, kommuniziert Ihr Container-Engine-Client (z. B. Docker) mit der Registrierung, um ein Manifestformat zu vereinbaren, das vom Client und der Registrierung verstanden wird und für das Image verwendet werden soll.

Wenn Sie ein Image mit Docker Version 1.9 oder früher an Amazon ECR pushen, wird das Image-Manifest-Format als Docker Image Manifest V2 Schema 1 gespeichert. Wenn Sie ein Image auf Amazon ECR mit Docker Version 1.10 oder höher pushen, wird das Image-Manifest-Format als Docker Image Manifest V2 Schema 2 gespeichert.

Wenn Sie ein Image per Tag aus Amazon ECR abrufen, gibt Amazon ECR das Image-Manifest-Format zurück, das im Repository gespeichert ist. Das Format wird nur zurückgegeben, wenn es vom Client verarbeitet werden kann. Wenn das Format des gespeicherten Image-Manifests vom Client nicht verstanden wird, konvertiert Amazon ECR das Image-Manifest in ein Format, das verstanden wird. Wenn zum Beispiel ein Docker 1.9-Client ein Image-Manifest anfordert, das als Docker Image Manifest V2 Schema 2 gespeichert ist, gibt Amazon ECR das Manifest im Format Docker Image Manifest V2 Schema 1 zurück. Die folgende Tabelle beschreibt die verfügbaren Konvertierungen, die von Amazon ECR unterstützt werden, wenn ein Image nach Tag abgerufen wird:

Vom Client angefordertes Schema	Push-Übertragung an ECR als V2, Schema 1	Push-Übertragung an ECR als V2, Schema 2	Push-Übertragung an ECR als OCI
V2, Schema 1	Keine Konvertierung erforderlich	Konvertiert in V2, Schema 1	Keine Konvertierung verfügbar
V2, Schema 2	Keine Konvertierung verfügbar, Client greift auf V2, Schema 1 zurück	Keine Konvertierung erforderlich	Konvertiert in V2, Schema 2
OCI	Keine Konvertierung verfügbar	Konvertiert in OCI	Keine Konvertierung erforderlich

Important

Wenn Sie ein Image per Digest abrufen, ist keine Konvertierung verfügbar. Ihr Client muss das Imagemanifestformat verstehen, das in Amazon ECR gespeichert ist. Falls Sie die "by digest"-Anforderung für ein Image im Format Docker Image Manifest V2 Schema 2 mit einem Docker 1.9-Client (oder einer älteren Version) ausführen, schlägt das Abrufen fehl. Weitere Informationen finden Sie unter Registrierungskompatibilität in der Docker-Dokumentation. Wenn Sie in diesem Beispiel das gleiche Image per Tag anfordern, übersetzt Amazon ECR das Image-Manifest in ein Format, das der Client versteht. Das Abrufen des Images war erfolgreich.

Verwendung von Amazon ECR-Bildern mit Amazon ECS

Sie können Ihre privaten Amazon-ECR-Repositorys verwenden, um Container-Images und Artefakte zu hosten, aus denen Ihre Amazon-ECR-Aufgaben möglicherweise abrufen. Damit dies funktioniert, muss der Amazon ECS- oder Fargate-Container-Agent über die erforderlichen Berechtigungen verfügen, um `ecr:BatchGetImageecr:GetDownloadUrlForLayer`, und `ecr:GetAuthorizationToken` APIs zu erstellen.

Erforderliche IAM-Berechtigungen

Die folgende Tabelle zeigt die zu verwendende IAM-Rolle für jeden Starttyp, die die erforderlichen Berechtigungen für Ihre Aufgaben zum Abrufen aus einem privaten Amazon-ECR-Repository bereitstellt. Amazon ECS stellt verwaltete IAM-Richtlinien bereit, die die erforderlichen Berechtigungen enthalten.

Starttyp	IAM role (IAM-Rolle)	AWS verwaltete IAM-Richtlinie
Amazon ECS auf EC2 Amazon-Instances	Verwenden Sie die Container-Instance-IAM-Rolle, die der EC2 Amazon-Instance zugeordnet ist, die in Ihrem Amazon ECS-Cluster registriert ist. Weitere Informationen finden Sie unter IAM-Rolle der Container-Instance im Entwicklerhandbuch für Amazon Elastic Container Service.	`AmazonEC2ContainerServiceforEC2Role` Weitere Informationen finden Sie unter AmazonEC2ContainerServiceforEC2Role im Entwicklerhandbuch für Amazon Elastic Container Service
Amazon ECS auf Fargate	Verwenden Sie die IAM-Rolle zur Aufgabenausführung, auf die Sie in Ihrer Amazon-ECS-Aufgabendefinition verweisen. Weitere Informationen finden Sie unter IAM-Rolle für die Aufgabenausführung im Entwicklerhandbuch für Amazon Elastic Container Service.	`AmazonECSTaskExecutionRolePolicy` Weitere Informationen finden Sie unter AmazonECSTaskExecutionRolePolicy im Entwicklerhandbuch für Amazon Elastic Container Service.
Amazon ECS auf externen Instances	Verwenden Sie die IAM-Rolle der Container-Instance, die dem On-Premises Server oder der virtuellen Maschine (VM) zugeordnet ist, die in Ihrem	`AmazonEC2ContainerServiceforEC2Role` Weitere Informationen finden Sie unter AmazonEC2ContainerServiceforEC2Role

Starttyp	IAM role (IAM-Rolle)	AWS verwaltete IAM-Richtlinie
	Amazon_ECS-Cluster registriert ist. Weitere Informationen finden Sie unter Amazon ECS-Rolle der Container-Instance im Entwicklerhandbuch für Amazon Elastic Container Service.	im Entwicklerhandbuch für Amazon Elastic Container Service.

Important

Die AWS verwalteten IAM-Richtlinien enthalten zusätzliche Berechtigungen, die Sie für Ihre Nutzung möglicherweise nicht benötigen. In diesem Fall sind dies die erforderlichen Mindestberechtigungen für den Abruf aus einem privaten Amazon-ECR-Repository.

JSON

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

Angeben eines Amazon-ECR-Images in einer Amazon-ECS-Aufgabendefinition

Beim Erstellen einer Amazon-ECR-Aufgabendefinition können Sie ein Container-Image angeben, das in einem privaten Amazon-ECR-Repository gehostet wird. Stellen Sie in der Aufgabendefinition sicher, dass Sie die vollständige `registry/repository:tag`-Benennung für Ihre Amazon-ECR-Images verwenden. Beispiel, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-repository`*`:`*`latest`*.

Der folgende Ausschnitt aus der Aufgabendefinition zeigt die Syntax, die Sie verwenden würden, um ein in Amazon ECR gehostetes Container-Image in Ihrer Amazon ECS-Aufgabendefinition anzugeben.

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

Verwendung von Amazon ECR-Bildern mit Amazon EKS

Sie können Ihre Amazon ECR-Images mit Amazon EKS verwenden.

Wenn Sie ein Image von Amazon ECR referenzieren, müssen Sie den vollständigen `registry/repository:tag`-Namen für das Image verwenden. Beispiel, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-repository`*`:`*`latest`*.

Erforderliche IAM-Berechtigungen

Wenn Sie Amazon EKS-Workloads auf verwalteten Knoten, selbstverwalteten Knoten oder hosten AWS Fargate, überprüfen Sie Folgendes:

- Amazon EKS-Workloads, die auf verwalteten oder selbstverwalteten Knoten gehostet werden: Die Amazon EKS-Worker-Knoten-IAM-Rolle (`NodeInstanceRole`) ist erforderlich. Die Amazon EKS-Worker-Knoten-IAM-Rolle muss die folgenden IAM-Richtlinienberechtigungen für Amazon ECR enthalten.

  JSON

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

> **Note**
>
> Wenn Sie Ihre Cluster `eksctl` - und Worker-Knotengruppen mithilfe der CloudFormation Vorlagen in Getting Started with Amazon EKS erstellt haben, werden diese IAM-Berechtigungen standardmäßig auf Ihre Worker-Knoten-IAM-Rolle angewendet.

- Amazon EKS-Workloads, gehostet auf AWS Fargate: Verwenden Sie die Fargate-Pod-Ausführungsrolle, die Ihren Pods die Erlaubnis gibt, Bilder aus privaten Amazon ECR-Repositorys abzurufen. Weitere Informationen finden Sie unter Erstellen einer Fargate-Pod-Ausführungsrolle.

Installation eines Helm-Diagramms auf einem Amazon EKS-Cluster

In Amazon ECR gehostete Helm-Diagramme können auf Ihren Amazon EKS-Clustern installiert werden.

Voraussetzungen

- Installieren Sie die neueste Version des Helm-Clients. Diese Schritte wurden mit Helm Version `3.9.0` geschrieben. Weitere Informationen finden Sie unter Installation von Helm.
- Sie haben mindestens Version `1.23.9` oder `2.6.3` von AWS CLI auf Ihrem Computer installiert. Weitere Informationen finden Sie unter Installieren oder Aktualisierung auf die neueste Version von AWS CLI.
- Sie haben ein Helm-Diagramm in Ihr Amazon ECR-Repository übertragen. Weitere Informationen finden Sie unter Übertragung eines Helm-Diagramms in ein privates Amazon ECR-Repository.
- Sie haben `kubectl` für die Arbeit mit Amazon EKS konfiguriert. Weitere Informationen finden Sie unter Erstellen Sie ein `kubeconfig` für Amazon EKS im Amazon EKS Benutzerhandbuch. Wenn die folgenden Befehle für Ihren Cluster erfolgreich sind, sind Sie richtig konfiguriert.

```
kubectl get svc
```

So installieren Sie ein Helm-Diagramm auf einem Amazon EKS-Cluster

1. Authentifizieren Sie Ihren Helm-Client bei dem Amazon ECR-Registry, in dem Ihr Helm-Diagramm gehostet wird. Für jede verwendete Registrierung muss ein Autorisierungs-Token erhalten werden, und die Token sind 12 Stunden lang gültig. Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in 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. Installieren Sie das Diagramm. *`helm-test-chart`*Ersetzen Sie es durch Ihr Repository und *`0.1.0`* durch das Tag Ihres Helm-Diagramms.

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

   Die Ausgabe sollte in etwa so aussehen:

   ```
   NAME: ecr-chart-demo
   LAST DEPLOYED: Tue May 31 17:38:56 2022
   NAMESPACE: default
   STATUS: deployed
   REVISION: 1
   ```

```
TEST SUITE: None
```

3. Überprüfen Sie die Installation der Karte.

```
helm list -n default
```

Beispielausgabe:

```
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. (Optional) Siehe installiertes Helm-Diagramm `ConfigMap`.

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

5. Wenn Sie fertig sind, können Sie das Chart-Release aus Ihrem Cluster entfernen.

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

Bilder in Amazon ECR signieren

Amazon ECR lässt sich integrieren und bietet Ihnen zwei Möglichkeiten, Ihre Container-Images zu signieren: verwaltetes Signieren (automatisch, empfohlen) und manuelles Signieren (clientseitig). AWS Signer Sie können sowohl Ihre Container-Images als auch die Signaturen in Ihren privaten Repositorys speichern.

Wählen Sie eine Signaturmethode

Amazon ECR unterstützt zwei Methoden zum Signieren von Container-Images:

Verwaltetes Signieren (empfohlen)

Managed Signing generiert automatisch kryptografische Signaturen, wenn Bilder an Amazon ECR übertragen werden. Diese Methode vereinfacht die Einrichtung. Verwaltetes Signieren ist der empfohlene Ansatz für die meisten Benutzer. Weitere Informationen finden Sie unter Verwaltetes Signieren.

Manuelles Signieren

Beim manuellen Signieren werden die Notation CLI und AWS Signer das Plugin verwendet, um Bilder zu signieren, bevor sie an Amazon ECR übertragen werden. Diese Methode bietet mehr Kontrolle über den Signaturprozess und ist nützlich, wenn Sie Bilder außerhalb des Push-Workflows signieren müssen oder eine detaillierte Kontrolle über die Signaturvorgänge benötigen. Weitere Informationen finden Sie unter Manuelles Signieren.

Überlegungen

Folgendes sollte bei der Verwendung von Amazon ECR Image Signing berücksichtigt werden:

- In Ihrem Repository gespeicherte Signaturen werden auf die Service Quota für die maximale Anzahl von Images pro Repository angerechnet. Jede Signatur wird als ein Artefakt auf die Bilder pro Repository-Kontingent angerechnet. Weitere Informationen finden Sie unter Amazon ECR Service Quotas.
- Wenn Referenzartefakte in einem Repository vorhanden sind, bereinigen die Amazon ECR-Lebenszyklusrichtlinien diese Artefakte innerhalb von 24 Stunden nach dem Löschen des betreffenden Bilds automatisch.

Verwaltetes Signieren

Die von Amazon ECR verwaltete Signatur signiert Ihre Container-Images automatisch, indem kryptografische Signaturen mithilfe von AWS Signer generiert werden, wenn Bilder an Amazon ECR übertragen werden. Dadurch entfällt die Notwendigkeit, clientseitige Tools zu installieren und zu konfigurieren, und Sie können das Signieren als Registrierungskonfiguration zentral steuern.

Voraussetzungen

Um verwaltetes Signieren zu konfigurieren, erstellen Sie mit Amazon ECR eine Signaturkonfiguration, die auf ein oder mehrere Unterzeichner-Signaturprofile und optional auf Repository-Filter verweist, die einschränken, welche Repositorys ihre Bilder signieren sollen. Nach der Konfiguration signiert Amazon ECR Managed Signing Bilder automatisch, wenn sie übertragen werden. Dabei wird die Identität der Entität verwendet, die das Bild überträgt.

Bevor Sie die verwaltete Signatur konfigurieren können, müssen Sie über Folgendes verfügen:

- Ein Signaturprofil für Unterzeichner — Erstellen Sie mindestens ein Signaturprofil für Unterzeichner. Ein AWS Signaturprofil ist eine einzigartige Unterzeichnerressource, mit der Sie Signaturvorgänge in Amazon ECR ausführen können. Mit Signaturprofilen können Sie Codeartefakte wie Container-Images und AWS Lambda Bereitstellungspakete signieren und verifizieren. Jedes Signaturprofil gibt die Signaturplattform an, für die signiert werden soll, eine Plattform-ID und andere plattformspezifische Informationen. Ein ARN für ein Signaturprofil sieht beispielsweise so aus:`arn:`*`partition`*`:signer:`*`region`*`:`*`account-id`*`:/signing-profiles/`*`profile-name`*.
- IAM-Berechtigungen — Der IAM-Principal, der das Image überträgt, muss über die erforderlichen IAM-Berechtigungen für den Zugriff auf das entsprechende Signaturprofil des Unterzeichners und das entsprechende ECR-Repository verfügen. Sie müssen die identitätsbasierte Richtlinie für den IAM-Prinzipal so ändern, dass sie Berechtigungen sowohl für ECR-Repository-Operationen als auch für Signaturvorgänge des Unterzeichners umfasst. Die folgende Beispielrichtlinie zeigt die erforderlichen Berechtigungen:

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

Mit Amazon ECR Managed Signing können Sie mehrere Signaturregeln (bis zu 10 pro Registrierung) erstellen, um stärkere Sicherheitsgrenzen zu schaffen. Sie könnten beispielsweise mehrere Build-Pipelines ausführen und einschränken wollen, welche Repositorys jede Pipeline signieren kann. In jeder Regel konfigurieren Sie ein Signaturprofil und geben Repository-Namensfilter an. Wenn ein neues Bild übertragen wird, passt Amazon ECR an, welche Signaturregel und welches Signaturprofil das Bild signieren können. Wenn es mehrere Übereinstimmungen gibt, generiert Amazon ECR mehrere Signaturen.

> **Note**
>
> Wenn Sie Signaturen manuell verifizieren, müssen Sie trotzdem die Notation CLI installieren.

> **Note**
>
> Die von Amazon ECR verwaltete Signatur ist in allen AWS Regionen verfügbar, in denen das Signieren von Container-Images mit AWS Signer verfügbar ist.

Erste Schritte

Gehen Sie wie folgt vor, um die verwaltete Signatur zu konfigurieren. Sie stellen Amazon ECR einen Verweis auf ein Signaturprofil eines Unterzeichners und optional Filter zur Verfügung, die einschränken, welche Repositorys ihre Bilder signieren sollen.

AWS-Managementkonsole

Verwenden Sie die folgenden Schritte, um das verwaltete Signieren mithilfe von zu konfigurieren. AWS-Managementkonsole

1. Öffnen Sie die Amazon ECR-Konsole. Wählen Sie im linken Navigationsbereich Private Registrierung, Funktionen und Einstellungen, Verwaltete Signatur aus.
2. Wählen Sie auf der Seite Signaturregeln die Option Regel erstellen aus.
3. Wählen Sie auf der Seite AWS Signaturprofil unter Unterzeichnerprofil auswählen die Option Neues AWS Unterzeichnerprofil erstellen aus, geben Sie einen Profilnamen ein und ändern Sie optional die Gültigkeitsdauer der Signatur. Wählen Sie dann Weiter aus.
4. Geben Sie auf der Seite Filter unter Repositorys auswählen einen Repository-Namensfilter ein. Wählen Sie dann Weiter aus.
5. Überprüfen Sie auf der Seite Überprüfen und erstellen die von Ihnen eingegebenen Filter für das AWS Unterzeichnerprofil und den Namen des Repositorys. Wenn alles korrekt aussieht, wählen Sie Speichern aus.

AWS CLI

Verwenden Sie die folgenden AWS CLI Befehle, um die verwaltete Signatur zu konfigurieren.

- Erstellen Sie eine Signaturregel

  Erstellen Sie eine Signaturkonfiguration mit Ihrem Signierprofil ARN. Erstellen Sie eine JSON-Datei mit folgendem Inhalt:

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

```
                    "filter": "test*",
                    "filterType": "WILDCARD_MATCH"
                }
            ]
        }
    ]
}
```

Führen Sie anschließend den folgenden Befehl aus:

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

Sie sollten die API-Antwort sehen, die die Signaturkonfiguration enthält.

- Sehen Sie sich Ihre Signaturkonfiguration an

  Rufen Sie Ihre Signaturkonfiguration ab:

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

  Sie sollten die API-Antwort sehen, die die Signaturkonfiguration enthält.

- Überprüfen Sie den Status der Bildsignierung

  Schieben Sie ein Bild in Ihr Repository. Zum Beispiel:

  ```
  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
  ```

  Verwenden Sie nach dem Push Ihr Image-Tag, um den Signaturstatus zu überprüfen:

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

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

Wenn der Repository-Name mit dem in der Signaturkonfiguration definierten Repository-Filter übereinstimmt, sollte in der API-Antwort der Signaturstatus angezeigt werden. Wenn der Status erfolgreich ist, solltest du sehen, dass eine Signatur an dein Repository gesendet wird.

- Löschen Sie Ihre Signaturkonfiguration

  Löschen Sie Ihre Signaturkonfiguration:

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

  Sie sollten die API-Antwort sehen, die die gelöschte Signaturkonfiguration enthält.

Überlegungen

Die folgenden Einschränkungen und Funktionen gelten für verwaltetes Signieren:

- Regionsübergreifendes Signieren wird nicht unterstützt — Signaturprofile müssen sich in derselben Region wie Ihre Amazon ECR-Registrierung befinden. Sie können kein Signaturprofil aus einer Region verwenden, um Bilder in einer Registry in einer anderen Region zu signieren.
- Kontoübergreifendes Signieren wird unterstützt — Signaturprofile können sich auf anderen Konten als Ihrer Amazon ECR-Registrierung befinden. Auf diese Weise können Unternehmen Signierprofile zentral verwalten und gleichzeitig Entwicklern mit anderen Konten ermöglichen, sie zu verwenden. Weitere Informationen finden Sie im AWS Signer Entwicklerhandbuch unter Kontenübergreifendes Signieren für Unterzeichner einrichten.
- Signaturen können nicht signiert werden — Sie können Signaturen nicht selbst signieren. Nur Container-Images können signiert werden.

Überprüfung der Signatur

Nachdem Sie Ihre Container-Images signiert haben, können Sie die Signaturen überprüfen, um sicherzustellen, dass die Bilder nicht manipuliert wurden und aus einer vertrauenswürdigen Quelle stammen. Amazon ECR unterstützt mehrere Methoden zur Überprüfung von Signaturen:

Verwaltete Überprüfung mit Amazon EKS

Amazon EKS bietet eine native Integration für die automatische Signaturüberprüfung. Wenn Sie die Signaturüberprüfung in Ihren Amazon EKS-Clustern konfigurieren, überprüft der Service automatisch Bildsignaturen, bevor Container ausgeführt werden können. Weitere Informationen zur Konfiguration der Signaturüberprüfung finden Sie unter Validieren von Container-Image-Signaturen während der Bereitstellung im Amazon EKS-Benutzerhandbuch.

Lambda-Zugangscontroller für Amazon ECS

Amazon ECS bietet Service Lifecycle-Hooks, mit denen Sie bei Servicebereitstellungen benutzerdefinierte Logik ausführen können. Diese Hooks können an bestimmten Punkten des Bereitstellungsprozesses AWS Lambda Funktionen auslösen, sodass Sie Container-Image-Signaturen validieren können, bevor Dienste gestartet werden können. Weitere Informationen finden Sie unter Verifizieren von Container-Image-Signaturen für Amazon ECS im AWS Signer Entwicklerhandbuch.

Manuelle Überprüfung mit Notation CLI

Sie können Signaturen manuell mit der Notation CLI überprüfen. Für diese Methode müssen Sie die Notation CLI auf Ihrem lokalen Computer oder in Ihrer Überprüfungsumgebung installieren und konfigurieren. Eine ausführliche Anleitung zur Verifizierung eines Images mit der Notation CLI finden Sie unter Lokales Überprüfen eines Images nach dem Signieren im AWS Signer Developer Guide.

Konfigurieren Sie die Authentifizierung für den Notation-Client

Wenn Sie manuelles Signieren verwenden oder Signaturen manuell mit der Notation CLI verifizieren, müssen Sie den Notation-Client so konfigurieren, dass er sich bei Amazon ECR authentifizieren kann. Wenn Sie Docker auf demselben Host installiert haben, auf dem Sie den Notation-Client installieren, verwendet Notation dieselbe Authentifizierungsmethode, die Sie für den Docker-Client verwenden. Der Docker `login` und die `logout` Befehle ermöglichen es der Notation `sign` und den `verify` Befehlen, dieselben Anmeldeinformationen zu verwenden, und Sie müssen Notation nicht separat authentifizieren. Weitere Informationen zur Konfiguration Ihres Notation-Clients für die Authentifizierung finden Sie unter Authentifizieren mit OCI-konformen Registern in der Notary Project-Dokumentation.

Wenn Sie Docker oder ein anderes Tool, das Docker-Anmeldeinformationen verwendet, nicht verwenden, empfehlen wir, das Amazon ECR Docker Credential Helper als Speicher für

Anmeldeinformation zu verwenden. Weitere Informationen zur Installation und Konfiguration des Amazon ECR Hilfsprogramm für Anmeldeinformationen finden Sie unter ECR Docker Credential Helper.

Manuelles Signieren

Beim manuellen Signieren werden die Notation CLI und AWS Signer das Plugin verwendet, um Bilder zu signieren, bevor sie an Amazon ECR übertragen werden. Diese Methode bietet mehr Kontrolle über den Signaturprozess und ist nützlich, wenn Sie Bilder außerhalb des Push-Workflows signieren müssen oder eine detaillierte Kontrolle über die Signaturvorgänge benötigen.

Ausführliche Anweisungen zum Signieren von Container-Images mit der Notation CLI und AWS Signer finden Sie unter Signieren von Container-Images in Signer und den verwandten Themen im AWS Signer Developer Guide.

Voraussetzungen

Bevor Sie beginnen, müssen die folgenden Voraussetzungen erfüllt sein.

- Installieren und konfigurieren Sie die neuste Version von AWS CLI. Weitere Informationen finden Sie unter Installieren oder Aktualisieren auf die neueste Version von AWS CLI im AWS Command Line Interface -Benutzerhandbuch.
- Installieren Sie die Notation CLI und das AWS Signer Plugin für Notation. Weitere Informationen finden Sie im AWS Signer -Entwicklerhandbuch unter Voraussetzungen für das Signieren von Container-Images.
- Speichern Sie ein Container-Image in einem privaten Amazon-ECR-Repository, um es zu signieren. Weitere Informationen finden Sie unter Ein Bild in ein privates Amazon ECR-Repository übertragen.

Bilder auf Softwareschwachstellen in Amazon ECR scannen

Amazon ECR Image Scanning hilft dabei, Softwareschwachstellen in Ihren Container-Images zu identifizieren. Die folgenden Scantypen werden angeboten.

Important

Wenn Sie zwischen den Versionen Erweitertes Scannen, Standard-Scannen und Verbessertes Standardscannen wechseln, sind zuvor eingerichtete Scans nicht mehr verfügbar. Sie müssen Ihre Scans erneut einrichten. Wenn Sie jedoch zu Ihrer vorherigen Scan-Version zurückkehren, sind die etablierten Scans verfügbar.

Note

Archivierte Bilder können nicht gescannt werden. Archivierte Bilder müssen wiederhergestellt werden, bevor sie gescannt werden können. Weitere Informationen zum Archivieren und Wiederherstellen von Bildern finden Sie unterArchivieren eines Bilds in Amazon ECR.

- Verbessertes Scannen — Amazon ECR ist in Amazon Inspector integriert, um ein automatisiertes, kontinuierliches Scannen Ihrer Repositorys zu ermöglichen. Ihre Container-Images werden auf Schwachstellen in Betriebssystemen und Programmiersprachenpaketen gescannt. Sobald neue Sicherheitslücken auftauchen, werden die Scanergebnisse aktualisiert und Amazon Inspector gibt ein Ereignis aus, um Sie EventBridge zu benachrichtigen. Verbessertes Scannen bietet Folgendes:
  - Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen
  - Zwei Scanfrequenzen: Scan on Push und kontinuierlicher Scan
- Grundlegendes Scannen — Amazon ECR bietet zwei Versionen des grundlegenden Scannens, die die Datenbank Common Vulnerabilities and Exposures (CVEs) verwenden:.
  - AWS systemeigenes Standardscannen — Verwendet AWS native Technologie, die jetzt allgemein verfügbar ist und empfohlen wird. Alle neuen Kunden-Registrys sind standardmäßig für diese verbesserte Version aktiviert.
  - Clair Basic Scanning — Verwendet das Open-Source-Projekt Clair. Clair ist jetzt veraltet. Details dazu finden Sie unter Clair: Missbilligung.

Beim einfachen Scannen konfigurieren Sie Ihre Repositorys so, dass bei Push gescannt wird, oder Sie können manuelle Scans durchführen und Amazon ECR stellt eine Liste der Scanergebnisse bereit. Das einfache Scannen bietet Folgendes:

- Betriebssystem-Scans
- Zwei Scanfrequenzen: Manuell und Scan bei Druck

> **Important**
>
> Die neue Version von Amazon ECR Basic Scanning verwendet nicht die `imageScanStatus` Attribute `imageScanFindingsSummary` und aus der `DescribeImages` API-Antwort, um Scanergebnisse zurückzugeben. Verwenden Sie stattdessen die `DescribeImageScanFindings` API. Weitere Informationen finden Sie unter `DescribeImageScanFindings`.

Filter zur Auswahl der Repositorys, die in Amazon ECR gescannt werden

Wenn Sie das Scannen von Bildern für Ihre private Registrierung konfigurieren, können Sie mithilfe von Filtern auswählen, welche Repositorys gescannt werden.

Wenn einfaches Scannen verwendet wird, können Sie Scan-bei-Push-Filter angeben, um anzugeben, welche Repositorys für einen Image-Scan eingestellt sind, wenn neue Images gepusht werden. Für alle Repositorien, die nicht mit einem Standardfilter zum Scannen auf Push-Verbindung übereinstimmen, wird die manuelle Scanfrequenz festgelegt. Das bedeutet, dass Sie den Scan manuell auslösen müssen, um einen Scan durchzuführen.

Wenn erweitertes Scannen verwendet wird, können Sie separate Filter für den Scan bei Push und kontinuierliches Scannen angeben. Für alle Repositorys, die nicht mit einem erweiterten Scanfilter übereinstimmen, wird das Scannen deaktiviert. Wenn Sie den erweiterten Scan verwenden und separate Filter für den Scan bei Push und für das kontinuierliche Scannen angeben, bei denen mehrere Filter mit demselben Repository übereinstimmen, erzwingt und zieht Amazon ECR den kontinuierlichen Scan-Filter dem Scan bei Push-Filter für dieses Repository vor.

Platzhalter filtern

Wenn ein Filter angegeben wird, stimmt ein Filter ohne Platzhalter mit allen Repository-Namen überein, die den Filter enthalten. Ein Filter mit einem Platzhalter (*) stimmt mit jedem Repository-Namen überein, bei dem der Platzhalter null oder mehr Zeichen im Repository-Namen ersetzt.

Die folgende Tabelle enthält Beispiele, in denen Repository-Namen auf der horizontalen Achse ausgedrückt werden und Beispielfilter auf der vertikalen Achse angegeben werden.

	Prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
Prod	Ja	Ja	Ja	Ja	Ja
*Prod	Ja	Ja	Nein	Nein	Nein
Prod*	Ja	Nein	Ja	Nein	Ja
Prod	Ja	Ja	Ja	Ja	Ja
prod*repo	Nein	Nein	Ja	Nein	Ja

Bilder auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR scannen

Das erweiterte Scannen von Amazon ECRs ist eine Integration mit Amazon Inspector, die Schwachstellensuche für Ihre Container-Images ermöglicht. Ihre Container-Images werden auf Schwachstellen in Betriebssystemen und Programmiersprachenpaketen gescannt. Sie können die Scanergebnisse sowohl bei Amazon ECR als auch mit Amazon Inspector direkt einsehen. Weitere Informationen zu Amazon Inspector finden Sie unter Scannen von Container-Images mit Amazon Inspector im Benutzerhandbuch für Amazon Inspector.

Beim erweiterten Scannen können Sie auswählen, welche Repositorys für automatisches, kontinuierliches Scannen und welche für Scan bei Push konfiguriert sind. Dies geschieht durch Festlegen von Scanfiltern.

Überlegungen für das erweiterte Scannen

Beachten Sie Folgendes, bevor Sie Amazon ECR Enhanced Scanning aktivieren.

- Für die Nutzung dieses Features fallen für Amazon ECR keine zusätzlichen Kosten an. Amazon Inspector berechnet jedoch Kosten für das Scannen Ihrer Bilder. Diese Funktion ist in Regionen verfügbar, in denen Amazon Inspector unterstützt wird. Weitere Informationen finden Sie unter:
  - Amazon Inspector Inspector-Preise — Amazon Inspector Inspector-Preise.
  - Von Amazon Inspector unterstützte Regionen — Regionen und Endpunkte.
- Das erweiterte Scannen von Amazon ECR zeigt, wie Bilder auf Amazon EKS und Amazon ECS verwendet werden. Sie können sehen, wann Bilder zuletzt verwendet wurden, und feststellen, wie viele Cluster jedes Bild verwenden. Diese Informationen helfen Ihnen bei der Priorisierung der Behebung von Sicherheitslücken für aktiv genutzte Images. Sie können schnell feststellen, welche Cluster von neu entdeckten Sicherheitslücken betroffen sein könnten. Weitere Informationen zum Anfordern dieser Informationen und zum Anzeigen der Antwort finden Sie unter `DescribeImageScanFindings`.
- Amazon Inspector unterstützt das Scannen nach bestimmten Betriebssystemen. Eine vollständige Liste finden Sie unter Unterstützte Betriebssysteme – Amazon-ECR-Scan im Benutzerhandbuch für Amazon Inspector.
- Amazon Inspector verwendet eine servicegebundene IAM-Rolle, die die erforderlichen Berechtigungen bereitstellt, um erweitertes Scannen für Ihre Repositorys bereitzustellen. Die servicegebundene IAM-Rolle wird automatisch von Amazon Inspector erstellt, wenn erweitertes Scannen für Ihre private Registrierung aktiviert ist. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für Amazon Inspector im Benutzerhandbuch für Amazon Inspector.
- Wenn Sie das erweiterte Scannen für Ihre private Registrierung zunächst aktivieren, erkennt Amazon Inspector nur Bilder, die in den letzten 14 Tagen an Amazon ECR übertragen wurden, basierend auf dem Image-Push-Zeitstempel. Ältere Bilder haben den `SCAN_ELIGIBILITY_EXPIRED`-Scanstatus. Wenn Sie möchten, dass diese Bilder von Amazon Inspector gescannt werden, sollten Sie sie erneut in Ihr Repository verschieben.
- Wenn das erweiterte Scannen für Ihre private Amazon ECR-Registrierung aktiviert ist, werden alle Repositorys, die mit den Scanfiltern übereinstimmen, nur mit erweitertem Scannen gescannt. Alle Repositorys, die keinem Filter entsprechen, haben eine `Off`-Scanfrequenz und werden nicht gescannt. Manuelle Scans mit erweitertem Scannen werden nicht unterstützt. Weitere Informationen finden Sie unter Filter zur Auswahl der Repositorys, die in Amazon ECR gescannt werden.

- Wenn Sie separate Filter für den Scan bei Push und das kontinuierliche Scannen angeben, bei denen mehrere Filter mit demselben Repository übereinstimmen, erzwingt und zieht Amazon ECR den kontinuierlichen Scan-Filter dem Scan bei Push-Filter für dieses Repository vor.
- Wenn erweitertes Scannen aktiviert ist, sendet Amazon ECR ein Ereignis an EventBridge die Änderung der Scan-Frequenz für ein Repository. Amazon Inspector gibt Ereignisse aus, EventBridge wenn ein erster Scan abgeschlossen ist und wenn ein Bildscan-Ergebnis erstellt, aktualisiert oder geschlossen wird.

Änderung der erweiterten Scandauer für Bilder in Amazon Inspector

Nach der Aktivierung des erweiterten Scannens scannt Amazon ECR kontinuierlich neu übertragene Bilder für die konfigurierte Dauer. Standardmäßig überwacht Amazon Inspector Ihre Repositorys, bis Bilder gelöscht oder erweitertes Scannen deaktiviert ist. Sie können in der Amazon Inspector Inspector-Konsole sowohl die Dauer des Push-Datums (bis zu Lifetime) als auch die Dauer des erneuten Scans entsprechend den Anforderungen Ihrer Umgebung konfigurieren. Wenn die Scandauer für ein Repository abgelaufen ist, wird der Scanstatus als angezeigt. `SCAN_ELIGIBILITY_EXPIRED` Weitere Informationen zur Konfiguration der Einstellungen für die Dauer des erneuten Scans für Amazon ECR in Amazon Inspector finden Sie unter Konfiguration der Dauer des erneuten Scans von Amazon ECR im Amazon Inspector Inspector-Benutzerhandbuch.

Für erweitertes Scannen in Amazon ECR sind IAM-Berechtigungen erforderlich

Amazon ECR Enhanced Scanning erfordert eine mit dem Amazon Inspector Service verknüpfte IAM-Rolle und dass der IAM-Principal, der erweitertes Scannen aktiviert und verwendet, berechtigt ist, den für das Scannen erforderlichen Amazon Inspector APIs aufzurufen. Die mit Amazon Inspector servicegebundene IAM-Rolle wird automatisch von Amazon Inspector erstellt, wenn erweitertes Scannen für Ihre private Registrierung aktiviert ist. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für Amazon Inspector im Benutzerhandbuch für Amazon Inspector.

Mit der folgenden IAM-Richtlinie werden die erforderlichen Berechtigungen zum Aktivieren und Verwenden des erweiterten Scans erteilt. Es enthält die Berechtigung, die Amazon Inspector zum Erstellen der dienstgebundenen IAM-Rolle benötigt, sowie die Amazon-Inspector-API-Berechtigungen, die erforderlich sind, um das erweiterte Scannen zu aktivieren und zu deaktivieren und die Scanergebnisse abzurufen.

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"
                    ]
                }
            }
        }
    ]
}
```

Konfiguration des erweiterten Scannens für Bilder in Amazon ECR

Konfigurieren Sie das erweiterte Scannen pro Region für Ihre private Registrierung.

Stellen Sie sicher, dass Sie über die richtigen IAM-Berechtigungen verfügen, um erweitertes Scannen zu konfigurieren. Weitere Informationen finden Sie unter Für erweitertes Scannen in Amazon ECR sind IAM-Berechtigungen erforderlich.

AWS-Managementkonsole

Um das erweiterte Scannen für Ihre private Registrierung zu aktivieren

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositories.
2. Wählen Sie in der Navigationsleiste die Region aus, für die die Scankonfiguration festgelegt werden soll.
3. Wählen Sie im Navigationsbereich Private Registrierung und dann Einstellungen aus.
4. Wählen Sie auf der Seite Scankonfiguration für Scantyp die Option Erweitertes Scannen.

   Wenn Erweitertes Scannen ausgewählt ist, werden standardmäßig alle Ihre Repositorys kontinuierlich gescannt.
5. Um bestimmte Repositorys für den kontinuierlichen Scan auszuwählen, deaktivieren Sie das Kontrollkästchen Alle Repositorys kontinuierlich scannen und definieren Sie dann Ihre Filter:

   > **Important**
   >
   > Filter ohne Platzhalter stimmen mit allen Repository-Namen überein, die den Filter enthalten. Filter mit Platzhaltern (*) stimmen mit einem Repository-Namen überein, bei dem der Platzhalter null oder mehr Zeichen im Repository-Namen ersetzt. Beispiele für das Verhalten von Filtern finden Sie unter. the section called “Platzhalter filtern”

   a. Geben Sie einen Filter ein, der auf Repository-Namen basiert, und wählen Sie dann Filter hinzufügen.
   b. Entscheiden Sie, welche Repositorys gescannt werden sollen, wenn ein Bild übertragen wird:

      - Um alle Repositorys per Push zu scannen, wählen Sie Alle Repositorys bei Push scannen aus.
      - Um bestimmte Repositorys auszuwählen, die bei Push gescannt werden sollen, geben Sie einen Filter ein, der auf den Repository-Namen basiert, und wählen Sie dann Filter hinzufügen.
6. Wählen Sie Speichern.

7. Wiederholen Sie diese Schritte in jeder Region, in der Sie das erweiterte Scannen aktivieren möchten.

AWS CLI

Verwenden Sie den folgenden AWS CLI Befehl, um das erweiterte Scannen für Ihre private Registrierung mithilfe von zu aktivieren. AWS CLI Sie können Scanfilter mithilfe des `rules`-Objekts angeben.

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

  Das folgende Beispiel aktiviert erweitertes Scannen für Ihre private Registrierung. Wenn keine `rules` angegeben werden, stellt Amazon ECR die Scankonfiguration standardmäßig auf kontinuierliches Scannen für alle Repositorys ein.

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

  Das folgende Beispiel aktiviert erweitertes Scannen für Ihre private Registrierung und gibt einen Scanfilter an. Der Scanfilter im Beispiel aktiviert kontinuierliches Scannen für alle Repositorys mit `prod` im Namen.

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

  Das folgende Beispiel aktiviert das erweiterte Scannen für Ihre private Registrierung und gibt mehrere Scanfilter an. Die Scanfilter im Beispiel ermöglichen das kontinuierliche Scannen für alle Repositorys mit `prod` im Namen und den Scan bei Push für alle anderen Repositorys.

  ```
  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 Ereignisse, die zum erweiterten Scannen in Amazon ECR gesendet wurden

Wenn erweitertes Scannen aktiviert ist, sendet Amazon ECR ein Ereignis an EventBridge die Änderung der Scan-Frequenz für ein Repository. Amazon Inspector sendet Ereignisse, EventBridge wenn ein erster Scan abgeschlossen ist und wenn ein Bildscan-Ergebnis erstellt, aktualisiert oder geschlossen wird.

Ereignis für eine Frequenzänderung des Repository-Scans

Wenn das erweiterte Scannen für Ihre Registrierung aktiviert ist, wird das folgende Ereignis von Amazon ECR gesendet, wenn es eine Änderung an einer Ressource gibt, für die das erweiterte Scannen aktiviert ist. Dazu gehören neue Repositorys, die Untersuchungshäufigkeit für ein Repository, das geändert wird, oder wenn Images in Repositorys mit aktiviertem erweiterten Scannen erstellt oder gelöscht werden. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.

```
{
 "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"
 }
}
```

Ereignis für einen ersten Image-Scan (erweitertes Scannen)

Wenn der erweiterte Scan für Ihre Registrierung aktiviert ist, wird das folgende Ereignis von Amazon Inspector gesendet, wenn der erste Image-Scan abgeschlossen ist. Der `finding-severity-counts`-Parameter gibt nur einen Wert für einen Schweregrad zurück, wenn ein solcher vorhanden ist. Wenn das Image beispielsweise keine Ergebnisse auf `CRITICAL`-Ebene enthält, wird keine kritische Zählung zurückgegeben. Weitere Informationen finden Sie unter Bilder auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR scannen.

Ereignismuster:

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

Beispielausgabe:

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

Ereignis für ein Update der Image-Scanergebnisse (erweitertes Scannen)

Wenn das erweiterte Scannen für Ihre Registrierung aktiviert ist, wird das folgende Ereignis von Amazon Inspector gesendet, wenn das Image-Scanergebnis erstellt, aktualisiert oder geschlossen wird. Weitere Informationen finden Sie unter Bilder auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR scannen.

Ereignismuster:

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

Beispielausgabe:

```
{
    "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"
    }
}
```

Abrufen der Ergebnisse für erweiterte Scans in Amazon ECR

Sie können die Scanergebnisse für den letzten abgeschlossenen erweiterten Bildscan abrufen und die Ergebnisse dann in Amazon Inspector öffnen, um weitere Details zu sehen. Die entdeckten Software-Sicherheitslücken sind auf der Grundlage der Datenbank Common Vulnerabilities and Exposures (CVEs) nach Schweregrad aufgelistet.

Details zur Problembehandlung bei einigen häufig auftretenden Problemen mit dem Scannen von Images finden Sie unter Problembehandlung beim Scannen von Bildern in Amazon ECR.

AWS-Managementkonsole

Führen Sie die folgenden Schritte aus, um Image-Scanergebnisse mithilfe der abzurufen AWS-Managementkonsole.

Um die Ergebnisse des Bildscans abzurufen

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihr Repository befindet.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das Repository mit dem Image aus, für das die Scanergebnisse abgerufen werden sollen.
5. Wählen Sie auf der Seite Bilder in der Spalte Image-Tag das Image-Tag aus, um die Scanergebnisse abzurufen.

6. Um weitere Details in der Amazon Inspector Inspector-Konsole anzuzeigen, wählen Sie den Namen der Sicherheitslücke in der Spalte Name aus.

AWS CLI

Verwenden Sie den folgenden AWS CLI Befehl, um die Ergebnisse des Bildscans mithilfe von abzurufen AWS CLI. Sie können ein Image mit der `imageTag` oder `imageDigest` angeben. Beides ist mit dem CLI-Befehl list-images möglich.

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

  Im folgenden Beispiel wird ein Image-Tag verwendet.

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

  Im folgenden Beispiel wird ein Image-Digest verwendet.

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

Bilder auf Betriebssystemschwachstellen in Amazon ECR scannen

Amazon ECR bietet zwei Versionen des grundlegenden Scannens, die die Common Vulnerabilities and Exposures (CVEs) -Datenbank verwenden:

- AWS systemeigenes Standardscannen — Verwendet AWS native Technologie, die jetzt allgemein verfügbar ist und empfohlen wird. Dieses verbesserte Standardscannen wurde entwickelt, um Kunden bessere Scanergebnisse und die Erkennung von Sicherheitslücken in einer Vielzahl gängiger Betriebssysteme zu bieten. Auf diese Weise können Kunden die Sicherheit ihrer Container-Images weiter erhöhen. Alle neuen Kunden-Registrys sind standardmäßig für diese verbesserte Version aktiviert.

- Clair Basic Scanning — Die vorherige Version des Basic-Scannings, die das Open-Source-Projekt Clair verwendet (siehe Clair weiter). GitHub Clair ist jetzt veraltet und wird ab dem 2. Februar 2026 nicht mehr unterstützt.

Sowohl das AWS native Scannen als auch das Standardscannen von Clair werden in allen Regionen unterstützt, die unter AWS Dienste nach Regionen aufgeführt sind, mit der Ausnahme, dass Clair nicht für Regionen unterstützt wird, die nach September 2024 hinzugefügt wurden. Weitere Informationen finden Sie unter Clair: Missbilligung.

Amazon ECR verwendet den Schweregrad für ein CVE aus der vorgelagerten Distributionsquelle, falls verfügbar. Andernfalls wird der Common Vulnerability Scoring System (CVSS)-Wert verwendet. Das CVSS-Ergebnis kann verwendet werden, um den Schweregrad der NVD-Schwachstellenbewertung zu erhalten. Weitere Informationen finden Sie unter NVD Vulnerability Severity Ratings.

Beide Versionen von Amazon ECR Basic Scanning unterstützen Filter, mit denen Sie angeben können, welche Repositorys bei Push gescannt werden sollen. Für alle Repositorys, die nicht mit einem Push-Scan-On-Push-Filter übereinstimmen, ist die manuelle Scan-Frequenz festgelegt, was bedeutet, dass Sie den Scan manuell starten müssen. Ein Bild kann einmal alle 24 Stunden gescannt werden. Die 24 Stunden beinhalten den ersten Scan per Push, sofern konfiguriert, und alle manuellen Scans. Mit dem einfachen Scannen können Sie bis zu 100.000 Bilder pro 24 Stunden in einer bestimmten Registrierung scannen. Das Limit von 100.000 gilt sowohl für den ersten Scan bei Push-Scans als auch für manuelle Scans, und zwar sowohl für Clair als auch für die verbesserte Version des Standardscans.

Für jedes Image kann das Ergebnis des letzten abgeschlossenen Image-Scans abgerufen werden. Wenn ein Bildscan abgeschlossen ist, sendet Amazon ECR ein Ereignis an Amazon EventBridge. Weitere Informationen finden Sie unter Amazon ECR-Ereignisse und EventBridge.

Clair: Missbilligung

Clair in Amazon ECR ist veraltet. Clair wird noch bis zum 2. Februar 2026 genutzt werden können. Wir empfehlen Ihnen jedoch dringend, Ihre Nutzung von Clair so bald wie möglich auf AWS systemeigenes Standardscannen umzustellen. Folgendes sollten Sie über Clair Deprecation wissen:

- Clair wird in neuen Regionen nicht unterstützt, sobald sie hinzugefügt werden, und wird ab dem 2. Februar 2026 in keiner Region mehr unterstützt.

- Ab dem 2. Februar 2026 können Sie keine Clair-Scans mehr durchführen, und alle Scans, die Sie zuvor durchgeführt haben, sind nach diesem Datum nicht mehr verfügbar. Sie müssen einen neuen Scan Ihrer Bilder auslösen, um die Scanergebnisse zu regenerieren, nachdem Sie zur neuen Version gewechselt haben.
- Vor dem 2. Februar 2026 können Sie zwischen Clair und nativem Basic Scanning hin und her wechseln.
- Wenn Sie Clair derzeit eingerichtet haben, werden Sie ab dem 2. Februar 2026 automatisch auf systemeigenes Standardscannen umgestellt, falls Sie dies nicht vorher tun.

AWS Das native Standardscannen bietet im Vergleich zum Clair-Scannen die folgenden zusätzlichen Funktionen:

- Wenn der native Standardscan Ressourcen scannt, werden mehr als 50 Datenfeeds abgerufen, um Ergebnisse für häufig auftretende Sicherheitslücken und Sicherheitslücken zu generieren (CVEs). Zu diesen Quellen gehören beispielsweise Sicherheitsempfehlungen von Anbietern, Datenfeeds und Feeds mit Bedrohungsinformationen sowie die National Vulnerability Database (NVD) und MITRE.
- Beim systemeigenen Standard-Scanning werden Sicherheitslückendaten aus Quell-Feeds mindestens einmal täglich aktualisiert.
- Scanergebnisse und die Erkennung von Sicherheitslücken sind für eine Vielzahl gängiger Betriebssysteme verfügbar (siehe unten).

Informationen zur Umstellung auf das verbesserte Standard-Scannen finden Sie unterUmstellung auf das verbesserte grundlegende Scannen von Bildern in Amazon ECR.

Betriebssystemunterstützung für einfaches Scannen und verbessertes Standardscannen

Aus Sicherheitsgründen und zur Gewährleistung eines kontinuierlichen Schutzes empfehlen wir, weiterhin unterstützte Versionen eines Betriebssystems zu verwenden. Gemäß den Herstellerrichtlinien werden ausgelaufene Betriebssysteme nicht mehr mit Patches aktualisiert, und in vielen Fällen werden keine neuen Sicherheitsempfehlungen mehr für sie veröffentlicht. Darüber hinaus entfernen einige Anbieter bestehende Sicherheitsempfehlungen und Sicherheitswarnungen aus ihren Feeds, wenn für ein betroffenes Betriebssystem der Standardsupport ausläuft. Wenn eine Distribution den Support durch ihren Anbieter verliert, unterstützt Amazon ECR das Scannen nach Sicherheitslücken möglicherweise nicht mehr. Alle Ergebnisse, die Amazon ECR für ein eingestelltes

Betriebssystem generiert, sollten nur zu Informationszwecken verwendet werden. Im Folgenden sind die aktuell unterstützten Betriebssysteme und Versionen aufgeführt.

Betriebssystem	Version	AWS systemeigener Grundkurs	Clair einfach
Alpines Linux (Alpin)	3.19	Ja	Ja
Alpines Linux (Alpin)	3,20	Ja	Ja
Alpines Linux (Alpin)	3.21	Ja	Nein
Alpines Linux (Alpin)	3.22	Ja	Nein
Alpines Linux (Alpin)	3.23	Ja	Nein
AlmaLinux	8	Ja	Nein
AlmaLinux	9	Ja	Nein
AlmaLinux	10	Ja	Nein
Amazon Linux (2AL2)	AL2	Ja	Ja
Amazon Linux 2023 (AL2023)	AL2023	Ja	Ja
Debian-Server (Bullseye)	11	Ja	Ja

Betriebssystem	Version	AWS systemeigener Grundkurs	Clair einfach
Debian-Server (Bücherwurm)	12	Ja	Ja
Debian-Server (Trixie)	13	Ja	Nein
Fedora	41	Ja	Nein
openSUSE Leap	15.6	Ja	Nein
Oracle Linux (Oracle)	8	Ja	Ja
Oracle Linux (Oracle)	9	Ja	Ja
Photon OS	4	Ja	Nein
Photon Betriebss ystem	5	Ja	Nein
Red Hat Enterprise Linux (RHEL)	8	Ja	Ja
Red Hat Enterprise Linux (RHEL)	9	Ja	Ja
Red Hat Enterprise Linux (RHEL)	10	Ja	Nein
Rocky Linux	8	Ja	Nein

Betriebssystem	Version	AWS systemeigener Grundkurs	Clair einfach
Rocky Linux	9	Ja	Nein
SUSE Linux Enterprise Server (SLES)	15.6	Ja	Nein
Ubuntu (Xenial)	16,04 (ESM)	Ja	Ja
Ubuntu (Bionisch)	18.04 (ESM)	Ja	Ja
Ubuntu (fokal)	20.04 (LTS)	Ja	Ja
Ubuntu (Jammy)	22,04 (LTS)	Ja	Ja
Ubuntu (Edles Numbat)	24.04	Ja	Nein
Ubuntu (Oracular Oriole)	24,10	Ja	Nein

Konfiguration des grundlegenden Scannens für Bilder in Amazon ECR

Standardmäßig aktiviert Amazon ECR das grundlegende Scannen für alle privaten Register. Daher ist es nicht erforderlich, das grundlegende Scannen zu aktivieren, sofern Sie die Scaneinstellungen in Ihrer privaten Registrierung nicht geändert haben. Beim einfachen Scannen wird das Open-Source-Projekt Clair verwendet.

Sie können die folgenden Schritte verwenden, um einen oder mehrere Scan-On-Push-Filter zu definieren.

So aktivieren Sie das grundlegende Scannen für Ihre private Registrierung

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/ private-registry/repositories
2. Wählen Sie in der Navigationsleiste die Region aus, für die die Scankonfiguration festgelegt werden soll.
3. Wählen Sie im Navigationsbereich Private Registry, Scanning aus.
4. Wählen Sie auf der Seite Scan-Konfiguration für Scan-Typ Einfaches Scannen aus.
5. Standardmäßig sind alle Ihre Repositorys auf manuelles Scannen eingestellt. Sie können optional Scan bei Push konfigurieren, indem Sie Scan bei Push-Filter angeben. Sie können den Scan bei Push für alle Repositorys oder einzelne Repositorys einstellen. Weitere Informationen finden Sie unter Filter zur Auswahl der Repositorys, die in Amazon ECR gescannt werden.

> **Note**
>
> Wenn Scan on Push für ein Repository aktiviert ist, werden auch Bilder gescannt, die nach der Archivierung wiederhergestellt werden. Aus dem wiederhergestellten Bild sind keine alten Scans mehr verfügbar.

Umstellung auf das verbesserte grundlegende Scannen von Bildern in Amazon ECR

Amazon ECR bietet erweiterte Funktionen zum Scannen von Container-Images durch eine verbesserte Version des Basis-Scannens, die AWS native Technologie verwendet. Diese Funktion hilft Ihnen dabei, Softwareschwachstellen in Ihren Container-Images zu identifizieren. Das folgende Verfahren hilft Ihnen dabei, zu dieser verbesserten Version des Standard-Scannens zu wechseln, wenn Sie eine frühere Version des einfachen Scannens verwenden, bei der `CLAIR` Technologie zum Einsatz kommt.

> **Important**
>
> Für neue Benutzer werden Ihre Registrierungen bei der Erstellung automatisch so konfiguriert, dass sie die `AWS_NATIVE` Scantechnologie verwenden. Sie müssen keine Maßnahmen ergreifen. Amazon ECR empfiehlt nicht, zur vorherigen Scantechnologie zurückzukehren `CLAIR`, die veraltet ist. Clair: MissbilligungEinzelheiten finden Sie unter

AWS-Managementkonsole

Um das verbesserte Standard-Scannen für Ihre private Registrierung zu aktivieren

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/ private-registry/repositories
2. Wählen Sie in der Navigationsleiste die Region aus, für die die Scankonfiguration festgelegt werden soll.
3. Wählen Sie im Navigationsbereich Private Registry, Features & Settings, Scanning aus.
4. Wählen Sie auf der Konfigurationsseite für das Scannen die Option Anmelden (empfohlen) aus, um eine verbesserte Version des Standardscans auszuwählen.
5. Standardmäßig sind alle Ihre Repositorys auf manuelles Scannen eingestellt. Sie können optional Scan bei Push konfigurieren, indem Sie Scan bei Push-Filter angeben. Sie können den Scan bei Push für alle Repositorys oder einzelne Repositorys einstellen. Weitere Informationen finden Sie unter Filter zur Auswahl der Repositorys, die in Amazon ECR gescannt werden.

AWS CLI

Amazon ECR hat das grundlegende Scannen für alle privaten Register aktiviert. Verwenden Sie die folgenden Befehle, um Ihren aktuellen Standardscantyp anzuzeigen und Ihren Basis-Scantyp zu ändern.

- Um die Version des Basisscans abzurufen, die Sie derzeit verwenden.

```
aws ecr get-account-setting --name BASIC_SCAN_TYPE_VERSION
```

Der Parametername ist ein Pflichtfeld. Wenn Sie den Namen nicht angeben, erhalten Sie die folgende Fehlermeldung:

```
aws: error: the following arguments are required: --name
```

Um Ihre Version für den grundlegenden Scantyp von `CLAIR` zu zu zu ändern`AWS_NATIVE`. Sobald Sie Ihre Version für den Standard-Scantyp von `CLAIR` zu geändert haben, ist `AWS_NATIVE` es nicht empfehlenswert, zu `CLAIR` zurückzukehren.

```
aws ecr put-account-setting --name BASIC_SCAN_TYPE_VERSION --value value
```

Manuelles Scannen eines Images auf Betriebssystemschwachstellen in Amazon ECR

Wenn Ihre Repositorys nicht für das Scannen per Push konfiguriert sind, können Sie Image-Scans manuell starten. Ein Bild kann einmal alle 24 Stunden gescannt werden. Die 24 Stunden beinhalten den ersten Scan per Push, sofern konfiguriert, und alle manuellen Scans.

Details zur Problembehandlung bei einigen häufig auftretenden Problemen mit dem Scannen von Images finden Sie unter Problembehandlung beim Scannen von Bildern in Amazon ECR.

AWS-Managementkonsole

Führen Sie die folgenden Schritte aus, um einen manuellen Image Scan mit der AWS-Managementkonsole zu starten.

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/ private-registry/repositories
2. Wählen Sie in der Navigationsleiste die Region aus, in der Ihr Repository erstellt werden soll.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das Repository aus, das das zu scannende Image enthält.
5. Wählen Sie auf der Seite Images das zu scannende Image aus und klicken Sie dann auf Scan (Scannen).

AWS CLI

- start-image-scan (AWS CLI)

  Im folgenden Beispiel wird ein Image-Tag verwendet.

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

  Im folgenden Beispiel wird ein Image-Digest verwendet.

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

AWS Tools for Windows PowerShell

- Holen ECRImage ScanFindingAWS Tools for Windows PowerShell Sie sich- ()

  Im folgenden Beispiel wird ein Image-Tag verwendet.

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

  Im folgenden Beispiel wird ein Image-Digest verwendet.

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

Abrufen der Ergebnisse für grundlegende Scans in Amazon ECR

Sie können die Scanergebnisse für den letzten abgeschlossenen einfachen Bildscan abrufen. Die entdeckten Softwareschwachstellen sind anhand der Datenbank Common Vulnerabilities and Exposures (CVEs) nach Schweregrad aufgelistet.

Details zur Problembehandlung bei einigen häufig auftretenden Problemen mit dem Scannen von Images finden Sie unter Problembehandlung beim Scannen von Bildern in Amazon ECR.

AWS-Managementkonsole

Führen Sie die folgenden Schritte aus, um Image-Scanergebnisse mithilfe der abzurufen AWS-Managementkonsole.

Um die Ergebnisse des Bildscans abzurufen

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/ private-registry/repositories
2. Wählen Sie in der Navigationsleiste die Region aus, in der Ihr Repository erstellt werden soll.
3. Wählen Sie im linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys das Repository mit dem Image aus, für das die Scanergebnisse abgerufen werden sollen.
5. Wählen Sie auf der Seite Bilder in der Spalte Image-Tag das Image-Tag aus, um die Scanergebnisse abzurufen.

AWS CLI

Verwenden Sie den folgenden AWS CLI Befehl, um Bildscanergebnisse mithilfe von abzurufen AWS CLI. Sie können ein Image mit der `imageTag` oder `imageDigest` angeben. Beides ist mit dem CLI-Befehl list-images möglich.

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

  Im folgenden Beispiel wird ein Image-Tag verwendet.

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

  Im folgenden Beispiel wird ein Image-Digest verwendet.

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

AWS Tools for Windows PowerShell

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

  Im folgenden Beispiel wird ein Image-Tag verwendet.

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

  Im folgenden Beispiel wird ein Image-Digest verwendet.

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

Problembehandlung beim Scannen von Bildern in Amazon ECR

Im Folgenden finden Sie häufige Fehler beim Scannen von Images. Sie können Fehler wie diesen in der Amazon ECR-Konsole anzeigen, indem Sie die Bilddetails anzeigen oder über die API oder AWS CLI die `DescribeImageScanFindings` API.

UnsupportedImageError

Sie erhalten möglicherweise die Fehlermeldung `UnsupportedImageError`, wenn Sie versuchen, ein Image einfach zu scannen, das mit einem Betriebssystem erstellt wurde, für das Amazon ECR das Scannen von einfachen Images nicht unterstützt. Amazon ECR unterstützt das Scannen von Paketen auf Schwachstellen für die wichtigsten Versionen von Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine und RHEL Linux Distributionen. Sobald eine Distribution nicht mehr von ihrem Hersteller unterstützt wird, kann es sein, dass Amazon ECR die Überprüfung auf Scwachstellen nicht mehr unterstützt. Amazon ECR unterstützt nicht das Scannen von Images, die aus dem Docker-Scratch-Image erstellt wurden.

> **Important**
>
> Bei Verwendung des erweiterten Scannens unterstützt Amazon Inspector das Scannen nach bestimmten Betriebssystem- und Medientypen. Eine vollständige Liste finden Sie unter Unterstützte Betriebssysteme und Medientypen im Benutzerhandbuch für Amazon Inspector.

Als Schweregrad wird `UNDEFINED` zurückgegeben

Möglicherweise erhalten Sie ein Scanergebnis mit dem Schweregrad `UNDEFINED`. Im Folgenden sind die häufigsten Ursachen dafür:

- Der Schwachstelle wurde von der CVE-Quelle keine Priorität zugewiesen.
- Der Schwachstelle wurde eine Priorität zugewiesen, die Amazon ECR nicht erkannt hat.

Um den Schweregrad und die Beschreibung einer Schwachstelle zu ermitteln, können Sie die CVE direkt von der Quelle aus anzeigen.

Verstehen des Scanstatus **`SCAN_ELIGIBILITY_EXPIRED`**

Wenn das erweiterte Scannen mit Amazon Inspector für Ihre private Registrierung aktiviert ist und Sie Ihre Scan-Schwachstellen anzeigen, wird möglicherweise der Scanstatus `SCAN_ELIGIBILITY_EXPIRED` angezeigt. Im Folgenden sind die häufigsten Ursachen dafür.

- Wenn Sie das erweiterte Scannen für Ihre private Registrierung zum ersten Mal aktivieren, erkennt Amazon Inspector anhand des Image-Push-Zeitstempels nur Bilder, die in den letzten 30 Tagen an Amazon ECR übertragen wurden. Ältere Bilder haben den `SCAN_ELIGIBILITY_EXPIRED`-

Scanstatus. Wenn Sie möchten, dass diese Bilder von Amazon Inspector gescannt werden, sollten Sie sie erneut in Ihr Repository verschieben.

- Wenn die Dauer des erneuten ECR-Scans in der Amazon-Inspector-Konsole geändert wird und diese Zeit verstrichen ist, wird der Scanstatus des Images auf `inactive` mit einem Ursachencode von `expired` geändert, und alle zugehörigen Ergebnisse für das Image sollen geschlossen werden. Dies führt dazu, dass die Amazon ECR-Konsole den Scanstatus als `SCAN_ELIGIBILITY_EXPIRED` auflistet.

Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung

Mithilfe von Pull-Through-Cache-Regeln können Sie den Inhalt einer Upstream-Registrierung mit Ihrer privaten Amazon ECR-Registrierung synchronisieren.

Amazon ECR unterstützt derzeit die Erstellung von Pull-Through-Cache-Regeln für die folgenden Upstream-Registries:

- Amazon ECR Public, Kubernetes Container Image Registry und Quay (erfordert keine Authentifizierung)
- Docker Hub, Microsoft Azure Container Registry, GitHub Container Registry und GitLab Container Registry (erfordert AWS Secrets Manager geheime Authentifizierung)
- Amazon ECR (erfordert Authentifizierung mit AWS IAM-Rolle)

Für GitLab Container Registry unterstützt Amazon ECR den Pull-Through-Cache nur mit dem SaaS-Angebot (Software GitLab as a Service). Weitere Informationen zur Nutzung GitLab des SaaS-Angebots finden Sie unter GitLab.com.

Bei Upstream-Registrierungen, die eine Authentifizierung mit Geheimnissen erfordern (wie Docker Hub), müssen Sie Ihre Anmeldeinformationen geheim speichern. AWS Secrets Manager Sie können die Amazon ECR-Konsole verwenden, um Secrets Manager für jede authentifizierte Upstream-Registrierung zu erstellen. Weitere Informationen zum Erstellen eines Secrets Manager Manager-Geheimnisses mit der Secrets Manager-Konsole finden Sie unterSpeichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim.

Für Amazon ECR müssen Sie eine IAM-Rolle erstellen, wenn die Amazon ECR-Upstream- und Downstream-Registers zu unterschiedlichen Konten gehören. AWS Weitere Informationen zum Erstellen einer IAM-Rolle finden Sie unter IAM-Richtlinien sind für den kontenübergreifenden ECR-zu-ECR-Pull-Through-Cache erforderlich.

Nachdem Sie eine Pull-Through-Cache-Regel für die Upstream-Registrierung erstellt haben, rufen Sie mithilfe Ihrer privaten Amazon ECR-Registry-URI ein Bild aus dieser Upstream-Registrierung ab. Amazon ECR erstellt dann ein Repository und zwischenspeichert dieses Image in Ihrer privaten Registrierung. Bei nachfolgenden Pull-Requests des zwischengespeicherten Images mit einem bestimmten Tag sucht Amazon ECR in der Upstream-Registrierung nach einer neuen Version

des Images mit diesem spezifischen Tag und versucht, das Image in Ihrer privaten Registrierung mindestens einmal alle 24 Stunden zu aktualisieren.

Vorlagen zur Erstellung eines Repositor

Amazon ECR hat Unterstützung für Vorlagen zur Erstellung von Repositorys hinzugefügt, sodass Sie mithilfe von Pull-Through-Cache-Regeln die Anfangskonfigurationen für neue Repositorys festlegen können, die von Amazon ECR in Ihrem Namen erstellt wurden. Jede Vorlage enthält ein Präfix für den Repository-Namespace, das verwendet wird, um neue Repositorys einer bestimmten Vorlage zuzuordnen. In Vorlagen kann die Konfiguration für alle Repository-Einstellungen festgelegt werden, einschließlich ressourcenbasierter Zugriffsrichtlinien, Unveränderlichkeit von Tags, Verschlüsselung und Lebenszyklusrichtlinien. Die Einstellungen in einer Repository-Erstellungsvorlage werden nur bei der Repository-Erstellung angewendet und haben keine Auswirkung auf bestehende Repositorys oder Repositorys, die mit einer anderen Methode erstellt wurden. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.

Überlegungen zur Verwendung von Pull-Through-Cache-Regeln

Beachten Sie Folgendes, wenn Sie Amazon ECR Pull-Through-Cache-Regeln verwenden.

- Die Erstellung von Pull-Through-Cache-Regeln wird in den folgenden Regionen nicht unterstützt.
  - China (Peking) (`cn-north-1`)
  - China (Ningxia) (`cn-northwest-1`)
  - AWS GovCloud (US-Ost) () `us-gov-east-1`
  - AWS GovCloud (US-West) () `us-gov-west-1`
- AWS Lambda unterstützt das Abrufen von Container-Images aus Amazon ECR mithilfe einer Pull-Through-Cache-Regel nicht.
- Beim Abrufen von Images mit dem Pull-Through-Cache werden die Amazon-ECR-FIPS-Service-Endpunkte beim ersten Abrufen eines Images nicht unterstützt. Die Verwendung der Amazon-ECR-FIPS-Service-Endpunkte funktioniert jedoch bei nachfolgenden Abrufen.
- Wenn ein zwischengespeichertes Bild über die private Registrierungs-URI von Amazon ECR abgerufen wird, werden die Image-Pulls durch AWS IP-Adressen initiiert. Dadurch wird sichergestellt, dass der Image-Abruf nicht auf die von der Upstream-Registrierung implementierten Abruf-Ratenkontingente angerechnet wird.

- Wenn ein zwischengespeichertes Image durch die URI der privaten Registrierung von Amazon ECR abgerufen wird, überprüft Amazon ECR das Upstream-Repository mindestens einmal alle 24 Stunden, um sicherzustellen, dass das zwischengespeicherte Image die neueste Version ist. Wenn sich in der Upstream-Registrierung ein neueres Image befindet, versucht Amazon ECR, das zwischengespeicherte Image zu aktualisieren. Dieser Timer basiert auf dem letzten Abruf des zwischengespeicherten Images.
- Wenn Amazon ECR das Image aus der Upstream-Registrierung aus irgendeinem Grund nicht aktualisieren kann und das Image abgerufen wird, wird trotzdem das letzte zwischengespeicherte Image abgerufen.
- Bei der Erstellung des Secrets-Manager-Secrets, das die Anmeldeinformationen für die Upstream-Registrierung enthält, muss der geheime Name das Präfix `ecr-pullthroughcache/` verwenden. Das Secret muss sich außerdem in demselben Konto und derselben Region befinden, in der die Pull-Through-Cache-Regel erstellt wurde.
- Wenn ein Image mit mehreren Architekturen mithilfe einer Pull-Through-Cache-Regel abgerufen wird, werden die Manifestliste und jedes in der Manifestliste referenzierte Image in das Amazon-ECR-Repository abgerufen. Wenn Sie nur eine bestimmte Architektur abrufen möchten, können Sie das Image mithilfe des mit der Architektur verknüpften Image-Digests oder -Tags anstelle des mit der Manifestliste verknüpften Tags abrufen.
- Amazon ECR verwendet eine serviceverknüpfte IAM-Rolle, die die Berechtigungen bereitstellt, die Amazon ECR benötigt, um das Repository für Sie zu erstellen, den Wert des Secrets-Manager-Secrets für die Authentifizierung abzurufen und das zwischengespeicherte Image in Ihrem Namen zu übertragen. Die serviceverknüpfte IAM-Rolle wird beim Erstellen einer Pull-Through-Cache-Regel erstellt. Weitere Informationen finden Sie unter Serviceverknüpfte Amazon-ECR-Rolle für Pull-Through-Cache.
- Standardmäßig verfügen der IAM-Prinzipal, der das zwischengespeicherte Image abruft, über die Berechtigungen, die ihnen durch ihre IAM-Richtlinie erteilt wurde. Sie können die Berechtigungsrichtlinie für die private Registrierung von Amazon ECR verwenden, um die Berechtigungen einer IAM-Entität weiter einzugrenzen. Weitere Informationen finden Sie unter Verwenden von Registrierungsberechtigungen.
- Amazon-ECR-Repositorys, die mit dem Pull-Through-Cache-Workflow erstellt wurden, werden wie jedes andere Amazon-ECR-Repository behandelt. Alle Repository-Features wie Replikation und Image-Scan werden unterstützt.
- Wenn Amazon ECR in Ihrem Namen mithilfe einer Pull-Through-Cache-Aktion ein neues Repository erstellt, werden die folgenden Standardeinstellungen auf das Repository angewendet, sofern es keine passende Repository-Erstellungsvorlage gibt. Sie können eine Repository-

Erstellungsvorlage verwenden, um die Einstellungen zu definieren, die auf Repositorys angewendet werden, die von Amazon ECR in Ihrem Namen erstellt wurden. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.

- Unveränderlichkeit von Tags — Die Unveränderlichkeit von Tags gibt an, ob Image-Tags überschrieben werden können. Standardmäßig sind Bild-Tags veränderbar (können überschrieben werden). Sie können das Tag-Verhalten ändern, indem Sie Tag-Ausschlussfilter entweder im Textfeld zum Ausschluss von veränderbaren Tags konfigurieren, wenn Veränderbar ausgewählt ist, oder im Textfeld Unveränderlicher Tag-Ausschluss, wenn Unveränderlich ausgewählt ist.
- Verschlüsselung — Die `AES256` Standardverschlüsselung wird verwendet.
- Repository-Berechtigungen — Ausgelassen, es wird keine Repository-Berechtigungsrichtlinie angewendet.
- Lifecycle-Richtlinie — Ausgelassen, es wird keine Lebenszyklus-Richtlinie angewendet.
- Ressourcen-Tags — Ausgelassen, es werden keine Ressourcen-Tags angewendet.

- Wenn Sie die Unveränderlichkeit von Image-Tags für Repositorys aktivieren, die eine Pull-Through-Cache-Regel verwenden, wird Amazon ECR daran gehindert, Images zu aktualisieren, die dasselbe Tag verwenden.
- Wenn ein Bild zum ersten Mal mithilfe der Pull-Through-Cache-Regel abgerufen wird, ist möglicherweise eine Route zum Internet erforderlich. Unter bestimmten Umständen ist eine Route zum Internet erforderlich. Es empfiehlt sich daher, eine Route einzurichten, um Ausfälle zu vermeiden. Wenn Sie also Amazon ECR so konfiguriert haben, dass ein VPC-Endpunkt eine Schnittstelle verwendet, AWS PrivateLink müssen Sie sicherstellen, dass der erste Pull eine Route zum Internet hat. Eine Möglichkeit, dies zu tun, besteht darin, in derselben VPC ein öffentliches Subnetz mit einem Internet-Gateway zu erstellen und dann den gesamten ausgehenden Datenverkehr von ihrem privaten Subnetz zum öffentlichen Subnetz ins Internet weiterzuleiten. Nachfolgende Image-Pulls unter Verwendung der Pull-Through-Cache-Regel erfordern dies nicht. Weitere Informationen finden Sie unter Beispiel für Routing-Optionen im Benutzerhandbuch von Amazon Virtual Private Cloud.

IAM-Berechtigungen sind erforderlich, um eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung zu synchronisieren

Zusätzlich zu den Amazon-ECR-API-Berechtigungen, die zur Authentifizierung bei einer privaten Registrierung und zum Übertragen und Abrufen von Images erforderlich sind, sind die folgenden zusätzlichen Berechtigungen erforderlich, um Pull-Through-Cache-Regeln effektiv zu verwenden.

- `ecr:CreatePullThroughCacheRule` – Gewährt die Berechtigung zum Erstellen einer neuen Pull-Through-Cache-Regel. Diese Berechtigung muss über eine identitätsbasierte IAM-Richtlinie erteilt werden.
- `ecr:BatchImportUpstreamImage` – Erteilt die Berechtigung, das externe Image abzurufen und in Ihre private Registrierung zu importieren. Diese Berechtigung kann mithilfe der Richtlinie für private Registrierungsberechtigungen, einer identitätsbasierten IAM-Richtlinie oder mithilfe der ressourcenbasierten Repository-Berechtigungsrichtlinie erteilt werden. Weitere Informationen zur Verwendung von Repository-Berechtigungen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.
- `ecr:CreateRepository` – Gewährt die Berechtigung zum Erstellen eines Repositorys in einer privaten Registrierung. Diese Berechtigung ist erforderlich, wenn das Repository, das die zwischengespeicherten Images speichert, noch nicht existiert. Diese Berechtigung kann entweder durch eine identitätsbasierte IAM-Richtlinie oder die Richtlinie für private Registrierungsberechtigungen erteilt werden.

Verwenden von Registrierungsberechtigungen

Private Registrierungsberechtigungen von Amazon ECRs können verwendet werden, um die Berechtigungen einzelner IAM-Entitäten zur Verwendung von Pull-Through-Cache zu nutzen. Wenn eine IAM-Entität mehr Berechtigungen hat, die durch eine IAM-Richtlinie gewährt werden, als die Registrierungsberechtigungsrichtlinie gewährt, hat die IAM-Richtlinie Vorrang. Wenn dem Benutzer beispielsweise `ecr:*`-Berechtigungen gewährt wurden, sind auf Registrierungsebene keine zusätzlichen Berechtigungen erforderlich.

So erstellen Sie eine Richtlinie für private Registrierungsberechtigungen (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.

2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre private Registrierungsberechtigungserklärung konfigurieren möchten.
3. Wählen Sie im Navigationsbereich Private Registrierung, Registrierungsberechtigungen aus.
4. Wählen Sie auf der Seite Registrierungsberechtigungen die Option Anweisung generieren aus.
5. Gehen Sie für jede Richtlinienanweisung für Pull-Through-Cache-Berechtigungen, die Sie erstellen möchten, wie folgt vor.

   a. Wählen Sie für Richtlinientyp, Pull-Through-Cache-Richtlinie aus.
   b. Für Anweisungs-ID, geben Sie einen Namen für die Richtlinie zur Pull-Through-Cache-Anweisung an.
   c. Geben Sie für IAM entities (IAM-Entitäten) die Benutzer, Gruppen oder Rollen an, die in die Richtlinie aufgenommen werden sollen.
   d. Für Repository-Namespace, wählen Sie die Pull-Through-Cache-Regel aus, mit der Sie die Richtlinie verknüpfen möchten.
   e. Für Repository-Namen, geben Sie den Repository-Basisnamen an, für den die Regel angewendet werden soll. Wenn Sie beispielsweise das Amazon-Linux-Repository auf Amazon ECR Public angeben möchten, lautet der Repository-Name `amazonlinux`.

So erstellen Sie eine Richtlinie für private Registrierungsberechtigungen (AWS CLI)

Verwenden Sie den folgenden AWS CLI Befehl, um die privaten Registrierungsberechtigungen mithilfe von anzugeben. AWS CLI

1. Erstellen Sie eine lokale Datei mit dem Namen `ptc-registry-policy.json` mit dem Inhalt der Registrierungsrichtlinie. Das folgende Beispiel gewährt dem `ecr-pull-through-cache-user` die Berechtigung ein Repository zu erstellen und ein Image aus Amazon ECR Public abzurufen. Dabei handelt es sich um die Upstream-Quelle, die der zuvor erstellten Pull-Through-Cache-Regel zugeordnet ist.

   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**
>
> Die `ecr-CreateRepository`-Berechtigung ist nur erforderlich, wenn das Repository, das die zwischengespeicherten Bilder speichert, noch nicht existiert. Zum Beispiel, wenn die Repository-Erstellungsaktion und die Image-Pull-Aktionen von separaten IAM-Prinzipalen wie einem Administrator und einem Entwickler ausgeführt werden.

2. Verwenden Sie den put-registry-policyBefehl, um die Registrierungsrichtlinie festzulegen.

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

Nächste Schritte

Sobald Sie bereit sind, mit der Verwendung von Pull-Through-Cache-Regeln zu beginnen, folgen die nächsten Schritte.

- Erstellen Sie eine Pull-Through-Cache-Regel. Weitere Informationen finden Sie unter Eine Pull-Through-Cache-Regel in Amazon ECR erstellen.
- Erstellen Sie eine Repository-Erstellungsvorlage. Mit einer Repository-Erstellungsvorlage können Sie die Einstellungen für neue Repositorys festlegen, die von Amazon ECR in Ihrem Namen während einer Pull-Through-Cache-Aktion erstellt werden. Weitere Informationen finden Sie unter

Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.

Berechtigungen für kontenübergreifendes ECR auf ECR einrichten PTC

Die Pull-Through-Cache-Funktion von Amazon ECR zu Amazon ECR (ECR to ECR) ermöglicht die automatische Synchronisation von Bildern zwischen Regionen, AWS Konten oder beidem. Mit ECR to ECR PTC können Sie Bilder in Ihre primäre Amazon ECR-Registrierung übertragen und eine Pull-Through-Cache-Regel konfigurieren, um Bilder in nachgelagerten Amazon ECR-Registern zwischenzuspeichern.

IAM-Richtlinien sind für den kontenübergreifenden ECR-zu-ECR-Pull-Through-Cache erforderlich

Um Bilder zwischen Amazon ECR-Registern über verschiedene AWS Konten hinweg zwischenzuspeichern, erstellen Sie eine IAM-Rolle im Downstream-Konto und konfigurieren Sie die Richtlinien in diesem Abschnitt so, dass sie die folgenden Berechtigungen bereitstellen:

- Amazon ECR benötigt Berechtigungen, um in Ihrem Namen Bilder aus der vorgelagerten Amazon ECR-Registrierung abzurufen. Sie können diese Berechtigungen gewähren, indem Sie eine IAM-Rolle erstellen und diese dann in Ihrer Pull-Through-Cache-Regel angeben.
- Der Eigentümer der Upstream-Registrierung muss dem Eigentümer der Cache-Registrierung auch die erforderlichen Berechtigungen zum Abrufen der Bilder in die Ressourcenrichtlinien gewähren.

Richtlinien

- Erstellen einer IAM-Rolle zur Definition der Pull Through-Cache-Berechtigungen
- Erstellen einer Vertrauensrichtlinie für die IAM-Rolle
- Erstellen einer Ressourcenrichtlinie in der vorgelagerten Amazon ECR-Registrierung

Erstellen einer IAM-Rolle zur Definition der Pull Through-Cache-Berechtigungen

Das folgende Beispiel zeigt eine Berechtigungsrichtlinie, die einer IAM-Rolle die Berechtigung erteilt, in Ihrem Namen Bilder aus der vorgelagerten Amazon ECR-Registrierung abzurufen. Wenn Amazon ECR die Rolle übernimmt, erhält es die in dieser Richtlinie angegebenen Berechtigungen.

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": "*"
        }
    ]
}
```

Erstellen einer Vertrauensrichtlinie für die IAM-Rolle

Das folgende Beispiel zeigt eine Vertrauensrichtlinie, die den Amazon ECR Pull-Through-Cache als den AWS Service Principal identifiziert, der die Rolle übernehmen kann.

JSON

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

```
        }
    ]
}
```

Erstellen einer Ressourcenrichtlinie in der vorgelagerten Amazon ECR-Registrierung

Der Besitzer der Amazon ECR-Upstream-Registrierung muss außerdem eine Registrierungsrichtlinie oder eine Repository-Richtlinie hinzufügen, um dem Eigentümer der Downstream-Registrierung die erforderlichen Berechtigungen zur Durchführung der folgenden Aktionen zu gewähren.

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

Eine Pull-Through-Cache-Regel in Amazon ECR erstellen

Für jede Upstream-Registry, die Bilder enthält, die Sie in Ihrer privaten Amazon ECR-Registrierung zwischenspeichern möchten, müssen Sie eine Pull-Through-Cache-Regel erstellen.

Für Upstream-Registrierungen, die eine Authentifizierung mit Geheimnissen erfordern, müssen Sie die Anmeldeinformationen in einem Secrets Manager Manager-Geheimnis speichern. Sie können ein vorhandenes Geheimnis verwenden oder ein neues Geheimnis erstellen. Sie können das Secrets Manager Manager-Geheimnis entweder in der Amazon ECR-Konsole oder in der Secrets Manager Manager-Konsole erstellen. Informationen zum Erstellen eines Secrets Manager Manager-Geheimnisses mit der Secrets Manager Manager-Konsole statt mit der Amazon ECR-Konsole finden Sie unterSpeichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim.

Voraussetzungen

- Stellen Sie sicher, dass Sie über die richtigen IAM-Berechtigungen zum Erstellen von Pull-Through-Cache-Regeln verfügen. Weitere Informationen finden Sie unter IAM-Berechtigungen sind erforderlich, um eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung zu synchronisieren.
- Für Upstream-Registrierungen, die eine Authentifizierung mit Geheimnissen erfordern: Wenn Sie ein vorhandenes Geheimnis verwenden möchten, stellen Sie sicher, dass das Secrets Manager Manager-Geheimnis die folgenden Anforderungen erfüllt:
  - Der Name des Geheimnisses beginnt mit`ecr-pullthroughcache/`. Zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse mit dem `ecr-pullthroughcache/` Präfix an.
  - Das Konto und die Region, in denen sich der geheime Schlüssel befindet, müssen mit dem Konto und der Region übereinstimmen, in denen sich die Pull-Through-Cache-Regel befindet.

So erstellen Sie eine Pull-Through-Cache-Regel (AWS-Managementkonsole)

Die folgenden Schritte zeigen, wie Sie mit der Amazon-ECR-Konsole eine Pull-Through-Cache-Regel und ein Secrets-Manager-Secret erstellen. Informationen zum Erstellen eines Secrets mit der Secrets Manager Manager-Konsole finden Sie unterSpeichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim.

Für Amazon ECR Public, Kubernetes Container Registry oder Quay

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.
4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.
5. Wählen Sie auf der Seite Schritt 1: Geben Sie eine Quelle für Registry entweder Amazon ECR Public, Kubernetes oder Quay aus der Liste der Upstream-Registrierungen aus und klicken Sie dann auf Weiter.
6. Geben Sie auf der Seite Schritt 2: Geben Sie ein Ziel an für das Repository-Präfix von Amazon ECR das Präfix für den Repository-Namespace an, das beim Zwischenspeichern von Images

verwendet werden soll, und wählen Sie dann Weiter aus. Standardmäßig wird ein Namespace ausgefüllt, aber auch ein benutzerdefinierter Namespace kann angegeben werden.

7. Überprüfen Sie auf der Seite Schritt 3: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.
8. Wiederholen Sie den vorangehenden Schritt für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

Für Docker Hub

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.
4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.
5. Wählen Sie auf der Seite Schritt 1: Geben Sie eine Quelle an für Registrierung die Option Docker Hub und dann Weiter aus.
6. Auf der Seite Schritt 2: Authentifizierung konfigurieren müssen Sie für Upstream-Anmeldeinformationen Ihre Authentifizierungsdaten für Docker Hub in einem AWS Secrets Manager -Secret speichern. Sie können ein vorhandenes Secret angeben oder die Amazon-ECR-Konsole verwenden, um ein neues Secret zu erstellen.

   a. Um ein vorhandenes Geheimnis zu verwenden, wählen Sie Vorhandenes AWS Geheimnis verwenden. Wählen Sie unter Geheimer Name in der Auswahlliste Ihr vorhandenes Secret aus und wählen Sie dann Weiter aus.

   > **Note**
   >
   > Zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden. Das Secret muss sich außerdem in demselben Konto und derselben Region befinden, in der die Pull-Through-Cache-Regel erstellt wurde.

   b. Um ein neues Secret zu erstellen, wählen Sie Ein AWS -Secret erstellen aus, gehen Sie wie folgt vor und klicken Sie dann auf Weiter.

i. Geben Sie unter Geheimer Name einen aussagekräftigen Namen für das Secret an. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten.

ii. Geben Sie für Docker Hub-E-Mail Ihre Docker Hub-E-Mail an.

iii. Geben Sie für das Docker-Hub-Zugriffstoken Ihr Docker-Hub-Zugriffstoken an. Weitere Informationen zum Erstellen eines Docker-Hub-Zugriffstokens finden Sie unter Zugriffstoken erstellen und verwalten in der Docker-Dokumentation.

7. Geben Sie auf der Seite Schritt 3: Geben Sie ein Ziel an für das Repository-Präfix von Amazon ECR den Repository-Namespace an, der beim Zwischenspeichern von Images verwendet werden soll, und wählen Sie dann Weiter aus.

   Standardmäßig wird ein Namespace ausgefüllt, aber auch ein benutzerdefinierter Namespace kann angegeben werden.

8. Überprüfen Sie auf der Seite Schritt 4: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.

9. Wiederholen Sie den vorangehenden Schritt für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

Für Container Registry GitHub

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.

2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.

3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.

4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.

5. Wählen Sie auf der Seite Schritt 1: Quelle angeben für Registry GitHub Container Registry, Next aus.

6. Auf der Seite „Schritt 2: Authentifizierung konfigurieren“ müssen Sie für Upstream-Anmeldeinformationen Ihre Authentifizierungsdaten für GitHub Container Registry AWS Secrets Manager geheim speichern. Sie können ein vorhandenes Secret angeben oder die Amazon-ECR-Konsole verwenden, um ein neues Secret zu erstellen.

   a. Um ein vorhandenes Geheimnis zu verwenden, wählen Sie Vorhandenes AWS Geheimnis verwenden aus. Wählen Sie unter Geheimer Name in der Auswahlliste Ihr vorhandenes Secret aus und wählen Sie dann Weiter aus.

> **Note**
>
> Zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden. Das Secret muss sich außerdem in demselben Konto und derselben Region befinden, in der die Pull-Through-Cache-Regel erstellt wurde.

b. Um ein neues Secret zu erstellen, wählen Sie Ein AWS -Secret erstellen aus, gehen Sie wie folgt vor und klicken Sie dann auf Weiter.

   i. Geben Sie unter Geheimer Name einen aussagekräftigen Namen für das Secret an. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten.

   ii. Geben Sie als GitHub Container Registry-Benutzername Ihren GitHub Container Registry-Benutzernamen an.

   iii. Geben Sie für das Zugriffstoken für die GitHub Container Registry Ihr Zugriffstoken für die GitHub Container Registry an. Weitere Informationen zum Erstellen eines GitHub Zugriffstokens finden Sie in der GitHub Dokumentation unter Verwaltung Ihrer persönlichen Zugriffstoken.

7. Geben Sie auf der Seite Schritt 3: Geben Sie ein Ziel an für das Repository-Präfix von Amazon ECR den Repository-Namespace an, der beim Zwischenspeichern von Images verwendet werden soll, und wählen Sie dann Weiter aus.

   Standardmäßig wird ein Namespace ausgefüllt, aber auch ein benutzerdefinierter Namespace kann angegeben werden.

8. Überprüfen Sie auf der Seite Schritt 4: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.

9. Wiederholen Sie den vorangehenden Schritt für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

Für Microsoft Azure Container Registry

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.

2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.

3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.

4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.
5. Führen Sie auf der Seite Schritt 1: Geben Sie eine Quelle an die folgenden Schritte aus.

   a. Wählen Sie für Registry Microsoft Azure Container Registry

   b. Geben Sie unter Quellregistrierungs-URL den Namen Ihrer Microsoft Azure Container Registry an und wählen Sie dann Weiter aus.

   > **Important**
   >
   > Sie müssen nur das Präfix angeben, da das Suffix `.azurecr.io` in Ihrem Namen ausgefüllt wird.

6. Auf der Seite Schritt 2: Authentifizierung konfigurieren müssen Sie für Upstream-Anmeldeinformationen Ihre Authentifizierungsdaten für die Microsoft Azure Container Registry in einem AWS Secrets Manager -Secret speichern. Sie können ein vorhandenes Secret angeben oder die Amazon-ECR-Konsole verwenden, um ein neues Secret zu erstellen.

   a. Um ein vorhandenes Geheimnis zu verwenden, wählen Sie Vorhandenes AWS Geheimnis verwenden. Wählen Sie unter Geheimer Name in der Auswahlliste Ihr vorhandenes Secret aus und wählen Sie dann Weiter aus.

   > **Note**
   >
   > Zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden. Das Secret muss sich außerdem in demselben Konto und derselben Region befinden, in der die Pull-Through-Cache-Regel erstellt wurde.

   b. Um ein neues Secret zu erstellen, wählen Sie Ein AWS -Secret erstellen aus, gehen Sie wie folgt vor und klicken Sie dann auf Weiter.

      i. Geben Sie unter Geheimer Name einen aussagekräftigen Namen für das Secret an. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten.

      ii. Geben Sie für den Microsoft-Azure-Container-Registry-Benutzernamen Ihren Benutzernamen für die Microsoft Azure Container Registry an.

      iii. Geben Sie für den Microsoft-Azure-Container-Registry-Zugriffstoken Ihren Zugriffstoken für die Microsoft Azure Container Registry an. Weitere Informationen zum Erstellen

eines Zugriffstokens für die Microsoft Azure Container Registry finden unter Token erstellen – Portal in der Microsoft Azure-Dokumentation.

7. Geben Sie auf der Seite Schritt 3: Geben Sie ein Ziel an für das Repository-Präfix von Amazon ECR den Repository-Namespace an, der beim Zwischenspeichern von Images verwendet werden soll, und wählen Sie dann Weiter aus.

   Standardmäßig wird ein Namespace ausgefüllt, aber auch ein benutzerdefinierter Namespace kann angegeben werden.

8. Überprüfen Sie auf der Seite Schritt 4: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.

9. Wiederholen Sie den vorangehenden Schritt für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

Für GitLab Container Registry

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.
4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.
5. Wählen Sie auf der Seite Schritt 1: Quelle angeben für Registry GitLab Container Registry, Next aus.
6. Auf der Seite Schritt 2: Authentifizierung konfigurieren müssen Sie für Upstream-Anmeldeinformationen Ihre Authentifizierungsdaten für GitLab Container Registry AWS Secrets Manager geheim speichern. Sie können ein vorhandenes Secret angeben oder die Amazon-ECR-Konsole verwenden, um ein neues Secret zu erstellen.

   a. Um ein vorhandenes Geheimnis zu verwenden, wählen Sie Vorhandenes AWS Geheimnis verwenden aus. Wählen Sie unter Geheimer Name in der Auswahlliste Ihr vorhandenes Secret aus und wählen Sie dann Weiter aus. Weitere Informationen zum Erstellen eines Secrets-Manager-Secrets mit der Secrets-Manager-Konsole finden Sie unter Speichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim.

> **Note**
>
> Zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden. Das Secret muss sich außerdem in demselben Konto und derselben Region befinden, in der die Pull-Through-Cache-Regel erstellt wurde.

b. Um ein neues Secret zu erstellen, wählen Sie Ein AWS -Secret erstellen aus, gehen Sie wie folgt vor und klicken Sie dann auf Weiter.

   i. Geben Sie unter Geheimer Name einen aussagekräftigen Namen für das Secret an. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten.

   ii. Geben Sie als GitLab Container Registry-Benutzername Ihren GitLab Container Registry-Benutzernamen an.

   iii. Geben Sie für das Zugriffstoken für die GitLab Container Registry Ihr Zugriffstoken für die GitLab Container Registry an. Weitere Informationen zur Erstellung eines Zugriffstokens für die GitLab Container Registry finden Sie in der GitLab Dokumentation unter Persönliche Zugriffstoken, Gruppenzugriffstoken oder Projektzugriffstoken.

7. Geben Sie auf der Seite Schritt 3: Geben Sie ein Ziel an für das Repository-Präfix von Amazon ECR den Repository-Namespace an, der beim Zwischenspeichern von Images verwendet werden soll, und wählen Sie dann Weiter aus.

   Standardmäßig wird ein Namespace ausgefüllt, aber auch ein benutzerdefinierter Namespace kann angegeben werden.

8. Überprüfen Sie auf der Seite Schritt 4: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.

9. Wiederholen Sie den vorangehenden Schritt für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

Für Amazon ECR private Registrierung in Ihrem Konto AWS

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.

4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.
5. Wählen Sie auf der Seite Schritt 1: Upstream angeben für Registrierung Amazon ECR Private und Dieses Konto aus. Wählen Sie unter Region die Region für die vorgelagerte Amazon ECR-Registrierung aus, und klicken Sie dann auf Weiter.
6. Wählen Sie auf der Seite Schritt 2: Namespaces angeben für Cache-Namespace aus, ob Sie Pull-Through-Cache-Repositorys mit einem bestimmten Präfix oder ohne Präfix erstellen möchten. Wenn Sie Ein bestimmtes Präfix auswählen, müssen Sie einen Präfixnamen angeben, der als Teil des Namespaces für das Zwischenspeichern von Bildern aus der Upstream-Registrierung verwendet werden soll.
7. Wählen Sie für den Upstream-Namespace aus, ob ein bestimmter Präfix abgerufen werden soll, der in der Upstream-Registrierung vorhanden ist. Wenn Sie kein Präfix auswählen, können Sie Daten aus einem beliebigen Repository in der Upstream-Registrierung abrufen. Geben Sie das Upstream-Repository-Präfix an, wenn Sie dazu aufgefordert werden, und wählen Sie dann Weiter.

   > **Note**
   >
   > Weitere Informationen zum Anpassen von Cache- und Upstream-Namespaces finden Sie unter. Anpassen der Repository-Präfixe für den ECR- und ECR-Pull-Through-Cache
8. Überprüfen Sie auf der Seite Schritt 3: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.
9. Wiederholen Sie diese Schritte für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

Für Amazon ECR private Registrierung von einem anderen Konto AWS

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Sie Ihre privaten Registrierungseinstellungen konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.
4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die Option Regel hinzufügen aus.
5. Wählen Sie auf der Seite Schritt 1: Upstream angeben für Registry die Option Amazon ECR Private and Cross account aus. Wählen Sie unter Region die Region für die vorgelagerte

Amazon ECR-Registrierung aus. Geben Sie unter AWS Konto die Konto-ID für die vorgelagerte Amazon ECR-Registrierung an, und wählen Sie dann Weiter.

6. Wählen Sie auf der Seite Schritt 2: Berechtigungen angeben für die IAM-Rolle eine Rolle aus, die für den kontoübergreifenden Pull-Through-Cache-Zugriff verwendet werden soll, und wählen Sie dann Create aus.

   > **Note**
   >
   > Stellen Sie sicher, dass Sie die IAM-Rolle auswählen, die die in erstellten Berechtigungen verwendet. IAM-Richtlinien sind für den kontenübergreifenden ECR-zu-ECR-Pull-Through-Cache erforderlich

7. Wählen Sie auf der Seite Schritt 3: Namespaces angeben für Cache-Namespace aus, ob Sie Pull-Through-Cache-Repositorys mit einem bestimmten Präfix oder ohne Präfix erstellen möchten. Wenn Sie Ein bestimmtes Präfix auswählen, müssen Sie einen Präfixnamen angeben, der als Teil des Namespaces für das Zwischenspeichern von Bildern aus der Upstream-Registrierung verwendet werden soll.
8. Wählen Sie für den Upstream-Namespace aus, ob ein bestimmter Präfix abgerufen werden soll, der in der Upstream-Registrierung vorhanden ist. Wenn Sie kein Präfix auswählen, können Sie Daten aus einem beliebigen Repository in der Upstream-Registrierung abrufen. Geben Sie das Upstream-Repository-Präfix an, wenn Sie dazu aufgefordert werden, und wählen Sie dann Weiter.

   > **Note**
   >
   > Weitere Informationen zum Anpassen von Cache- und Upstream-Namespaces finden Sie unter. Anpassen der Repository-Präfixe für den ECR- und ECR-Pull-Through-Cache

9. Überprüfen Sie auf der Seite Schritt 4: Überprüfen und erstellen die Konfiguration der Pull-Through-Cache-Regel und wählen Sie dann Erstellen aus.
10. Wiederholen Sie diese Schritte für jeden Pull-Through-Cache, den Sie erstellen möchten. Die Pull-Through-Cache-Regeln werden für jede Region separat erstellt.

So erstellen Sie eine Pull-Through-Cache-Regel (AWS CLI)

Verwenden Sie den AWS CLI Befehl create-pull-through-cache-rule, um eine Pull-Through-Cache-Regel für eine private Amazon ECR-Registrierung zu erstellen. Für Upstream-Registrierungen, die

eine Authentifizierung mit Geheimnissen erfordern, müssen Sie die Anmeldeinformationen in einem Secrets Manager Manager-Geheimnis speichern. Informationen zum Erstellen eines Secrets mit der Secrets Manager Manager-Konsole finden Sie unterSpeichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim.

Die folgenden Beispiele werden für jede unterstützte Upstream-Registrierung bereitgestellt.

Für Amazon ECR Public

Im folgenden Beispiel wird eine Pull-Through-Cache-Regel für die öffentliche Registrierung von Amazon ECR erstellt. Es gibt ein Repository-Präfix von `ecr-public`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `ecr-public/`*`upstream-repository-name`* hat.

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

Für Kubernetes Container Registry

Im folgenden Beispiel wird eine Pull-Through-Cache-Regel für das öffentliche Kubernetes-Registry erstellt. Es gibt ein Repository-Präfix von `kubernetes`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `kubernetes/`*`upstream-repository-name`* hat.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix kubernetes \
     --upstream-registry-url registry.k8s.io \
     --region us-east-2
```

Für Quay

Im folgenden Beispiel wird eine Pull-Through-Cache-Regel für die öffentliche Registrierung von Quay erstellt. Es gibt ein Repository-Präfix von `quay`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `quay/`*`upstream-repository-name`* hat.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix quay \
```

```
    --upstream-registry-url quay.io \
    --region us-east-2
```

Für Docker Hub

Im folgenden Beispiel wird eine Pull-Through-Cache-Regel für die Docker-Hub-Registrierung von Quay erstellt. Es gibt ein Repository-Präfix von `docker-hub`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `docker-hub/`*`upstream-repository-name`* hat. Sie müssen den vollständigen Amazon-Ressourcennamen (ARN) des Secrets mit Ihren Anmeldeinformationen für Docker Hub angeben.

```
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
```

Für Container Registry GitHub

Im folgenden Beispiel wird eine Pull-Through-Cacheregel für die GitHub Container Registry erstellt. Es gibt ein Repository-Präfix von `github`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `github/`*`upstream-repository-name`* hat. Sie müssen den vollständigen Amazon-Ressourcennamen (ARN) des Geheimnisses angeben, das Ihre Anmeldeinformationen für die GitHub Container Registry enthält.

```
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
```

Für Microsoft Azure Container Registry

Im folgenden Beispiel wird eine Pull-Through-Cacheregel für die Microsoft Azure Container Registry erstellt. Es gibt ein Repository-Präfix von `azure`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `azure/`*`upstream-repository-name`* hat. Sie müssen den vollständigen Amazon-Ressourcennamen (ARN) des Secrets mit Ihren Anmeldeinformationen für die Microsoft Azure Container Registry angeben.

```
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
```

Für GitLab Container Registry

Im folgenden Beispiel wird eine Pull-Through-Cacheregel für die GitLab Container Registry erstellt. Es gibt ein Repository-Präfix von `gitlab`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `gitlab/`*`upstream-repository-name` hat. Sie müssen den vollständigen Amazon-Ressourcennamen (ARN) des Geheimnisses angeben, das Ihre Anmeldeinformationen für die GitLab Container Registry enthält.

```
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
```

Für Amazon ECR private Registrierung in Ihrem Konto AWS

Im folgenden Beispiel wird eine Pull-Through-Cache-Regel für die private Amazon ECR-Registrierung für Cross-Region innerhalb desselben AWS Kontos erstellt. Es gibt ein Repository-Präfix von `ecr`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `ecr/`*`upstream-repository-name` hat.

```
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
```

Für Amazon ECR private Registrierung von einem anderen Konto AWS

Im folgenden Beispiel wird eine Pull-Through-Cache-Regel für die private Amazon ECR-Registrierung für Cross-Region innerhalb desselben AWS Kontos erstellt. Es gibt ein Repository-Präfix von `ecr`, was dazu führt, dass jedes Repository, das mit der Pull-Through-Cache-Regel erstellt wurde, das Benennungsschema von `ecr/`*`upstream-repository-name` hat. Sie müssen den vollständigen

Amazon-Ressourcennamen (ARN) der IAM-Rolle mit den in Eine Pull-Through-Cache-Regel in Amazon ECR erstellen erstellten Berechtigungen angeben.

```
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
```

Nächste Schritte

Nachdem Sie Ihre Pull-Through-Cache-Regeln erstellt haben, folgen die nächsten Schritte:

- Erstellen Sie eine Repository-Erstellungsvorlage. Mit einer Repository-Erstellungsvorlage können Sie die Einstellungen für neue Repositorys festlegen, die von Amazon ECR in Ihrem Namen während einer Pull-Through-Cache-Aktion erstellt werden. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.
- Validieren Sie Ihre Pull-Through-Cache-Regeln. Bei der Validierung einer Pull-Through-Cache-Regel stellt Amazon ECR eine Netzwerkverbindung mit der Upstream-Registrierung her und überprüft, ob es auf das Secrets-Manager-Secret zugreifen kann, das die Anmeldeinformationen für die Upstream-Registrierung enthält, und ob die Authentifizierung erfolgreich war. Weitere Informationen finden Sie unter Überprüfung der Pull-Through-Cache-Regeln in Amazon ECR.
- Verwenden Sie zunächst Ihre Pull-Through-Cache-Regeln. Weitere Informationen finden Sie unter Ein Bild mit einer Pull-Through-Cache-Regel in Amazon ECR abrufen.

Überprüfung der Pull-Through-Cache-Regeln in Amazon ECR

Nachdem Sie eine Pull-Through-Cache-Regel erstellt haben, können Sie für Upstream-Registrierungen, die eine Authentifizierung erfordern, überprüfen, ob die Regel ordnungsgemäß funktioniert. Bei der Validierung einer Pull-Through-Cache-Regel stellt Amazon ECR eine Netzwerkverbindung mit der Upstream-Registrierung her, überprüft, ob es auf das Secrets Manager-Geheimnis zugreifen kann, das die Anmeldeinformationen für die Upstream-Registrierung enthält, und überprüft, ob die Authentifizierung erfolgreich war.

Bevor Sie mit der Arbeit mit Ihren Pull-Through-Cache-Regeln beginnen, stellen Sie sicher, dass Sie über die richtigen IAM-Berechtigungen verfügen. Weitere Informationen finden Sie unter IAM-

Berechtigungen sind erforderlich, um eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung zu synchronisieren.

So validieren Sie eine Pull-Through-Cache-Regel (AWS-Managementkonsole)

Die folgenden Schritte zeigen, wie Sie eine Pull-Through-Cache-Regel mit der Amazon-ECR-Konsole validieren.

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie auf der Navigationsleiste die Region aus, die die zu validierende Pull-Through-Cache-Regel enthält.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung, Pull-Through-Cache.
4. Wählen Sie auf der Seite Pull-Through-Cache-Konfiguration die zu validierende Pull-Through-Cache-Regel aus. Wählen Sie dann im Auswahlmenü Aktionen die Option Details anzeigen aus.
5. Wählen Sie auf der Detailseite für die Pull-Through-Cache-Regel das Auswahlmenü Aktionen aus und wählen Sie Authentifizierung überprüfen aus. Amazon ECR zeigt ein Banner mit dem Ergebnis an.
6. Wiederholen Sie diese Schritte für jede Pull-Through-Cache-Regel, die Sie validieren möchten.

So validieren Sie eine Pull-Through-Cache-Regel (AWS CLI)

Der AWS CLI Befehl validate-pull-through-cache-rule wird verwendet, um eine Pull-Through-Cache-Regel für eine private Amazon ECR-Registrierung zu validieren. Im folgenden Beispiel wird das Namespace-Präfix `ecr-public` verwendet. Ersetzen Sie diesen Wert durch den Präfixwert für die zu validierende Pull-Through-Cache-Regel.

```
aws ecr validate-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --region us-east-2
```

In der Antwort gibt der Parameter `isValid` an, ob die Validierung erfolgreich war oder nicht. Falls der Wert `true` ist, konnte Amazon ECR die Upstream-Registrierung erreichen und die Authentifizierung war erfolgreich. Falls der Wert `false` ist, gab es ein Problem und die Validierung schlug fehl. Der Parameter `failure` gibt die Ursache an.

Ein Bild mit einer Pull-Through-Cache-Regel in Amazon ECR abrufen

Die folgenden Beispiele zeigen die Befehlssyntax, die verwendet werden muss, wenn ein Image mithilfe einer Pull-Through-Cache-Regel abgerufen wird. Wenn Sie einen Fehler beim Abrufen eines Upstream-Images mit einer Pull-Through-Cache-Regel erhalten, lesen Sie Behebung von Problemen mit dem Pull-Through-Cache in Amazon ECR für die häufigsten Fehler und wie diese behoben werden können.

Bevor Sie mit der Arbeit mit Ihren Pull-Through-Cache-Regeln beginnen, stellen Sie sicher, dass Sie über die richtigen IAM-Berechtigungen verfügen. Weitere Informationen finden Sie unter IAM-Berechtigungen sind erforderlich, um eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung zu synchronisieren.

> **Note**
>
> Die folgenden Beispiele verwenden die standardmäßigen Amazon ECR-Repository-Namespace-Werte, die AWS-Managementkonsole von verwendet werden. Stellen Sie sicher, dass Sie den von Ihnen konfigurierten Amazon-ECR-Repository-URI verwenden.

Für Amazon ECR Public

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/image_name:tag
```

Kubernetes Container Registry

```
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

Für offizielle Docker-Hub-Images:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

> **Note**
>
> Für offizielle Docker-Hub-Images muss das Präfix `/library` enthalten sein. Für alle anderen Docker-Hub-Repositorys sollten Sie das Präfix `/library` weglassen.

Für alle anderen Docker-Hub-Images:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

GitHub Container-Registrierung

```
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 Container-Registrierung

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

Speichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim

Wenn Sie eine Pull-Through-Cache-Regel für ein Upstream-Repository erstellen, das eine Authentifizierung erfordert, müssen Sie die Anmeldeinformationen in einem Secrets-Manager-Secret

speichern. Für die Verwendung eines Secrets-Manager-Secrets können Kosten anfallen. Weitere Informationen finden Sie unter AWS Secrets Manager Preise.

Die folgenden Verfahren führen Sie Schritt für Schritt durch die Erstellung eines Secrets-Manager-Secrets für jedes unterstützte Upstream-Repository. Sie können optional den Workflow zum Erstellen einer Pull-Through-Cache-Regel in der Amazon-ECR-Konsole verwenden, um das Secret zu erstellen, anstatt das Secret mit der Secrets-Manager-Konsole zu erstellen. Weitere Informationen finden Sie unter Eine Pull-Through-Cache-Regel in Amazon ECR erstellen.

Docker Hub

So erstellen Sie ein Secrets-Manager-Secret für Ihre Anmeldeinformationen für Docker Hub (AWS-Managementkonsole)

1. Öffnen Sie die Secrets Manager Manager-Konsole unter https://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Secret-Typ auswählen die folgenden Schritte aus.
   a. Als Secret-Typ wählen Sie Anderer Secret-Typ aus.
   b. Erstellen Sie in Schlüssel/Wert-Paaren zwei Zeilen für Ihre Anmeldeinformationen für Docker Hub. Sie können bis zu 65536 Bytes im Secret speichern.
      i. Geben Sie für das erste key/value Paar `username` als Schlüssel und Ihren Docker Hub-Benutzernamen als Wert an.
      ii. Geben Sie für das zweite key/value Paar `accessToken` als Schlüssel und Ihr Docker Hub-Zugriffstoken als Wert an. Weitere Informationen zum Erstellen eines Docker-Hub-Zugriffstokens finden Sie unter Zugriffstoken erstellen und verwalten in der Docker-Dokumentation.
   c. Behalten Sie für den Verschlüsselungsschlüssel den AWS KMS key -Standardwert aws/secretsmanager bei und wählen Sie dann Weiter. Für die Verwendung dieses Schlüssels fallen keine Kosten an. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Verschlüsselung und Entschlüsselung von Secrets im Secrets Manager.

> **Important**
>
> Sie müssen den Standard-Verschlüsselungsschlüssel `aws/secretsmanager` verwenden, um Ihr Secret zu verschlüsseln. Amazon ECR unterstützt dafür nicht die Verwendung eines kundenseitig verwalteten Schlüssels (CMK).

4. Führen Sie auf der Seite Secret konfigurieren die folgenden Schritte aus.

   a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten und mit dem Präfix `ecr-pullthroughcache/` versehen sein.

   > **Important**
   >
   > Der Amazon ECR zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden.

   b. (Optional) Im Abschnitt Tags können Sie Tags zu Ihrem Secret hinzufügen. Informationen zu Tagging-Strategien finden Sie unter Tagging von Secrets-Manager-Secrets im Benutzerhandbuch von AWS Secrets Manager . Speichern Sie keine sensiblen Daten in Tags, da sie nicht verschlüsselt sind.

   c. (Optional) Um eine Ressourcenrichtlinie zu Ihrem Secret hinzuzufügen, wählen Sie unter Resource permissions (Ressourcenberechtigungen) die Option Edit permissions (Berechtigungen bearbeiten) aus. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Anhängen einer Berechtigungsrichtlinie an ein Secrets-Manager-Secret.

   d. (Optional) Um Ihr Geheimnis auf ein anderes zu replizieren, wählen Sie unter Secret replizieren die Option Secret AWS-Region replizieren aus. Sie können Ihr Secret jetzt replizieren oder zurückkommen und es später replizieren. Weitere Informationen finden Sie unter Ein Secret an anderen Regionen replizieren im Benutzerhandbuch von AWS Secrets Manager .

   e. Wählen Sie Weiter aus.

5. (Optional) Auf der Seite Rotation konfigurieren können Sie die automatische Rotation aktivieren. Sie können die Rotation auch vorerst ausschalten und später einschalten. Weitere

Informationen finden Sie unter Secrets-Manager-Secrets rotieren im Benutzerhandbuch von AWS Secrets Manager . Wählen Sie Weiter aus.

6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

   Secrets Manager kehrt zur Liste der Secrets zurück. Wenn Ihr Secret nicht angezeigt wird, wählen Sie den Aktualisieren-Button aus.

GitHub Container Registry

Um ein Secrets Manager Manager-Geheimnis für Ihre GitHub Container Registry-Anmeldeinformationen zu erstellen (AWS-Managementkonsole)

1. Öffnen Sie die Secrets Manager Manager-Konsole unter https://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Secret-Typ auswählen die folgenden Schritte aus.

   a. Als Secret-Typ wählen Sie Anderer Secret-Typ aus.

   b. Erstellen Sie in Schlüssel/Wert-Paaren zwei Zeilen für Ihre GitHub Anmeldeinformationen. Sie können bis zu 65536 Bytes im Secret speichern.

      i. Geben Sie für das erste key/value Paar `username` als Schlüssel und Ihren GitHub Benutzernamen als Wert an.

      ii. Geben Sie für das zweite key/value Paar `accessToken` als Schlüssel und Ihr GitHub Zugriffstoken als Wert an. Weitere Informationen zum Erstellen eines GitHub Zugriffstokens finden Sie in der GitHub Dokumentation unter Verwaltung Ihrer persönlichen Zugriffstoken.

   c. Behalten Sie für den Verschlüsselungsschlüssel den AWS KMS key -Standardwert aws/secretsmanager bei und wählen Sie dann Weiter. Für die Verwendung dieses Schlüssels fallen keine Kosten an. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Verschlüsselung und Entschlüsselung von Secrets im Secrets Manager.

> **Important**
>
> Sie müssen den Standard-Verschlüsselungsschlüssel `aws/secretsmanager` verwenden, um Ihr Secret zu verschlüsseln. Amazon ECR unterstützt dafür nicht die Verwendung eines kundenseitig verwalteten Schlüssels (CMK).

4. Führen Sie auf der Seite Configure secret (Secret konfigurieren) die folgenden Schritte aus:

   a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten und mit dem Präfix `ecr-pullthroughcache/` versehen sein.

   > **Important**
   >
   > Der Amazon ECR zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden.

   b. (Optional) Im Abschnitt Tags können Sie Tags zu Ihrem Secret hinzufügen. Informationen zu Tagging-Strategien finden Sie unter Tagging von Secrets-Manager-Secrets im Benutzerhandbuch von AWS Secrets Manager . Speichern Sie keine sensiblen Daten in Tags, da sie nicht verschlüsselt sind.

   c. (Optional) Um eine Ressourcenrichtlinie zu Ihrem Secret hinzuzufügen, wählen Sie unter Resource permissions (Ressourcenberechtigungen) die Option Edit permissions (Berechtigungen bearbeiten) aus. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Anhängen einer Berechtigungsrichtlinie an ein Secrets-Manager-Secret.

   d. (Optional) Um Ihr Geheimnis auf ein anderes zu replizieren, wählen Sie unter Secret replizieren die Option Secret AWS-Region replizieren aus. Sie können Ihr Secret jetzt replizieren oder zurückkommen und es später replizieren. Weitere Informationen finden Sie unter Ein Secret an anderen Regionen replizieren im Benutzerhandbuch von AWS Secrets Manager .

   e. Wählen Sie Weiter aus.

5. (Optional) Auf der Seite Rotation konfigurieren können Sie die automatische Rotation aktivieren. Sie können die Rotation auch vorerst ausschalten und später einschalten. Weitere

Informationen finden Sie unter Secrets-Manager-Secrets rotieren im Benutzerhandbuch von AWS Secrets Manager . Wählen Sie Weiter aus.

6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

   Secrets Manager kehrt zur Liste der Secrets zurück. Wenn Ihr Secret nicht angezeigt wird, wählen Sie den Aktualisieren-Button aus.

Microsoft Azure Container Registry

So erstellen Sie ein Secrets-Manager-Secret für Ihre Anmeldeinformationen für die Microsoft Azure Container Registry (AWS-Managementkonsole)

1. Öffnen Sie die Secrets Manager Manager-Konsole unter https://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Secret-Typ auswählen die folgenden Schritte aus.

   a. Als Secret-Typ wählen Sie Anderer Secret-Typ aus.

   b. Erstellen Sie in Schlüssel/Wert-Paaren zwei Zeilen für Ihre Anmeldeinformationen für Microsoft Azure. Sie können bis zu 65536 Bytes im Secret speichern.

      i. Geben Sie für das erste key/value Paar `username` als Schlüssel und Ihren Microsoft Azure Container Registry-Benutzernamen als Wert an.

      ii. Geben Sie für das zweite key/value Paar `accessToken` als Schlüssel und Ihr Microsoft Azure Container Registry-Zugriffstoken als Wert an. Weitere Informationen zum Erstellen eines Zugriffstokens für Microsoft Azure finden unter Token erstellen – Portal in der Microsoft Azure-Dokumentation.

   c. Behalten Sie für den Verschlüsselungsschlüssel den AWS KMS key -Standardwert aws/secretsmanager bei und wählen Sie dann Weiter. Für die Verwendung dieses Schlüssels fallen keine Kosten an. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Verschlüsselung und Entschlüsselung von Secrets im Secrets Manager.

> **Important**
>
> Sie müssen den Standard-Verschlüsselungsschlüssel `aws/secretsmanager` verwenden, um Ihr Secret zu verschlüsseln. Amazon ECR unterstützt dafür nicht die Verwendung eines kundenseitig verwalteten Schlüssels (CMK).

4. Führen Sie auf der Seite Configure secret (Secret konfigurieren) die folgenden Schritte aus:

   a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten und mit dem Präfix `ecr-pullthroughcache/` versehen sein.

   > **Important**
   >
   > Der Amazon ECR zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden.

   b. (Optional) Im Abschnitt Tags können Sie Tags zu Ihrem Secret hinzufügen. Informationen zu Tagging-Strategien finden Sie unter Tagging von Secrets-Manager-Secrets im Benutzerhandbuch von AWS Secrets Manager . Speichern Sie keine sensiblen Daten in Tags, da sie nicht verschlüsselt sind.

   c. (Optional) Um eine Ressourcenrichtlinie zu Ihrem Secret hinzuzufügen, wählen Sie unter Resource permissions (Ressourcenberechtigungen) die Option Edit permissions (Berechtigungen bearbeiten) aus. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Anhängen einer Berechtigungsrichtlinie an ein Secrets-Manager-Secret.

   d. (Optional) Um Ihr Geheimnis auf ein anderes zu replizieren, wählen Sie unter Secret replizieren die Option Secret AWS-Region replizieren aus. Sie können Ihr Secret jetzt replizieren oder zurückkommen und es später replizieren. Weitere Informationen finden Sie unter Ein Secret an anderen Regionen replizieren im Benutzerhandbuch von AWS Secrets Manager .

   e. Wählen Sie Weiter aus.

5. (Optional) Auf der Seite Rotation konfigurieren können Sie die automatische Rotation aktivieren. Sie können die Rotation auch vorerst ausschalten und später einschalten. Weitere

Informationen finden Sie unter Secrets-Manager-Secrets rotieren im Benutzerhandbuch von AWS Secrets Manager . Wählen Sie Weiter aus.

6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

   Secrets Manager kehrt zur Liste der Secrets zurück. Wenn Ihr Secret nicht angezeigt wird, wählen Sie den Aktualisieren-Button aus.

GitLab Container Registry

Um ein Secrets Manager Manager-Geheimnis für Ihre GitLab Container Registry-Anmeldeinformationen zu erstellen (AWS-Managementkonsole)

1. Öffnen Sie die Secrets Manager Manager-Konsole unter https://console.aws.amazon.com/secretsmanager/.
2. Wählen Sie Store a new secret (Ein neues Secret speichern).
3. Führen Sie auf der Seite Secret-Typ auswählen die folgenden Schritte aus.

   a. Als Secret-Typ wählen Sie Anderer Secret-Typ aus.

   b. Erstellen Sie in Schlüssel/Wert-Paaren zwei Zeilen für Ihre GitLab Anmeldeinformationen. Sie können bis zu 65536 Bytes im Secret speichern.

      i. Geben Sie für das erste key/value Paar `username` als Schlüssel und Ihren GitLab Container Registry-Benutzernamen als Wert an.

      ii. Geben Sie für das zweite key/value Paar `accessToken` als Schlüssel und Ihr GitLab Container Registry-Zugriffstoken als Wert an. Weitere Informationen zur Erstellung eines Zugriffstokens für die GitLab Container Registry finden Sie in der GitLab Dokumentation unter Persönliche Zugriffstoken, Gruppenzugriffstoken oder Projektzugriffstoken.

   c. Behalten Sie für den Verschlüsselungsschlüssel den AWS KMS key -Standardwert aws/secretsmanager bei und wählen Sie dann Weiter. Für die Verwendung dieses Schlüssels fallen keine Kosten an. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Verschlüsselung und Entschlüsselung von Secrets im Secrets Manager.

> **Important**
>
> Sie müssen den Standard-Verschlüsselungsschlüssel `aws/secretsmanager` verwenden, um Ihr Secret zu verschlüsseln. Amazon ECR unterstützt dafür nicht die Verwendung eines kundenseitig verwalteten Schlüssels (CMK).

4. Führen Sie auf der Seite Configure secret (Secret konfigurieren) die folgenden Schritte aus:

   a. Geben Sie einen beschreibenden Secret-Namen und eine Beschreibung ein. Secret-Namen müssen 1–512 Unicode-Zeichen enthalten und mit dem Präfix `ecr-pullthroughcache/` versehen sein.

   > **Important**
   >
   > Der Amazon ECR zeigt AWS-Managementkonsole nur Secrets Manager Manager-Geheimnisse an, deren Namen das `ecr-pullthroughcache/` Präfix verwenden.

   b. (Optional) Im Abschnitt Tags können Sie Tags zu Ihrem Secret hinzufügen. Informationen zu Tagging-Strategien finden Sie unter Tagging von Secrets-Manager-Secrets im Benutzerhandbuch von AWS Secrets Manager . Speichern Sie keine sensiblen Daten in Tags, da sie nicht verschlüsselt sind.

   c. (Optional) Um eine Ressourcenrichtlinie zu Ihrem Secret hinzuzufügen, wählen Sie unter Resource permissions (Ressourcenberechtigungen) die Option Edit permissions (Berechtigungen bearbeiten) aus. Weitere Informationen finden Sie im Benutzerhandbuch von AWS Secrets Manager unter Anhängen einer Berechtigungsrichtlinie an ein Secrets-Manager-Secret.

   d. (Optional) Um Ihr Geheimnis auf ein anderes zu replizieren, wählen Sie unter Secret replizieren die Option Secret AWS-Region replizieren aus. Sie können Ihr Secret jetzt replizieren oder zurückkommen und es später replizieren. Weitere Informationen finden Sie unter Ein Secret an anderen Regionen replizieren im Benutzerhandbuch von AWS Secrets Manager .

   e. Wählen Sie Weiter aus.

5. (Optional) Auf der Seite Rotation konfigurieren können Sie die automatische Rotation aktivieren. Sie können die Rotation auch vorerst ausschalten und später einschalten. Weitere

Informationen finden Sie unter Secrets-Manager-Secrets rotieren im Benutzerhandbuch von AWS Secrets Manager . Wählen Sie Weiter aus.

6. Prüfen Sie auf der Seite Review (Prüfen) die Secret-Details und wählen Sie Store (Speichern).

   Secrets Manager kehrt zur Liste der Secrets zurück. Wenn Ihr Secret nicht angezeigt wird, wählen Sie den Aktualisieren-Button aus.

Anpassen der Repository-Präfixe für den ECR- und ECR-Pull-Through-Cache

Pull-Through-Cache-Regeln unterstützen sowohl das ECR-Repository-Präfix als auch das Upstream-Repository-Präfix. Das ECR-Repository-Präfix ist das Repository-Namespace-Präfix in der Amazon ECR-Cache-Registry, das der Regel zugeordnet ist. Alle Repositorys, die dieses Präfix verwenden, werden zu Pull Through-Cache-fähigen Repositorys für die in der Regel definierte Upstream-Registry. Beispielsweise `prod` gilt das Präfix von für alle Repositorys, die mit beginnen. `prod/` Um eine Vorlage auf alle Repositorys in Ihrer Registrierung anzuwenden, denen keine Pull-Through-Cache-Regel zugeordnet ist, verwenden Sie `ROOT` als Präfix.

> **Important**
>
> Es wird immer ein `/` am Ende des Präfixes angenommen. Wenn Sie `ecr-public` als Präfix angeben, behandelt Amazon ECR dies als `ecr-public/`.

Das Upstream-Repository-Präfix entspricht dem Namen des Upstream-Repositorys. Standardmäßig ist es auf gesetzt`ROOT`, was den Abgleich mit jedem Upstream-Repository ermöglicht. Sie können das Upstream-Repository-Präfix nur festlegen, wenn das Amazon ECR-Repository-Präfix keinen `ROOT` Wert hat.

Die folgende Tabelle zeigt die Zuordnung zwischen Cache-Repository-Namen und Upstream-Repository-Namen auf der Grundlage ihrer Präfixkonfigurationen in den Pull-Through-Cache-Regeln.

Cache-Namespace	Upstream-Namespace	Beziehung zuordnen (Cache-Repository → Upstream-Repository)
ecr-public	ROOT (Standard)	`ecr-public/my-app/image1` → `my-app/image1` `ecr-public/my-app/image2` → `my-app/image2`
WURZEL	WURZEL	`my-app/image1` → `my-app/image1`
Team-A	Team-a	`team-a/myapp/image1` → `team-a/myapp/image1`
meine App	Upstream-App	`my-app/image1` → `upstream-app/image1`

Behebung von Problemen mit dem Pull-Through-Cache in Amazon ECR

Beim Abrufen eines Upstream-Image mit einer Pull-Through-Cache-Regel sind die folgenden Fehler die häufigsten Fehler, die Sie möglicherweise erhalten könnten.

Das Repository ist nicht vorhanden

Ein Fehler, der darauf hinweist, dass das Repository nicht existiert, wird meistens entweder dadurch verursacht, dass das Repository nicht in Ihrer privaten Amazon ECR-Registrierung vorhanden ist, oder dadurch, dass dem IAM-Prinzipal, der das Upstream-Image abruft, keine `ecr:CreateRepository`-Berechtigung erteilt wird. Um diesen Fehler zu beheben, sollten Sie überprüfen, ob der Repository-URI in Ihrem Pull-Befehl korrekt ist, dem IAM-Prinzipal, der das Upstream-Image abruft, die erforderlichen IAM-Berechtigungen erteilt werden, oder dass das Repository, an das das Upstream-Image verschoben werden soll, in Ihrer privaten Amazon ECR-Registrierung erstellt wird, bevor Sie das Upstream-Image abrufen. Weitere Informationen zu den

erforderlichen IAM-Berechtigungen finden Sie unter IAM-Berechtigungen sind erforderlich, um eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung zu synchronisieren.

Es folgt ein Beispiel dieses Fehlers.

```
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'
```

Das angeforderte Bild wurde nicht gefunden

Ein Fehler, der angibt, dass das Image nicht gefunden werden kann, wird am häufigsten dadurch verursacht, dass das Image nicht in der Upstream-Registrierung vorhanden ist, oder dadurch, dass dem IAM-Prinzipal, der das Upstream-Image abruft, keine `ecr:BatchImportUpstreamImage`-Berechtigung erteilt wird, während jedoch das Repository bereits in Ihrer privaten Amazon ECR-Registrierung erstellt wird. Um diesen Fehler zu beheben, sollten Sie überprüfen, ob der Name des Upstream-Image und des Image-Tags korrekt sind und dass der IAM-Prinzipal, der das Upstream-Image abruft, die erforderlichen IAM-Berechtigungen erteilt werden. Weitere Informationen zu den erforderlichen IAM-Berechtigungen finden Sie unter IAM-Berechtigungen sind erforderlich, um eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung zu synchronisieren.

Es folgt ein Beispiel dieses Fehlers.

```
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 Verboten beim Abrufen aus einem Docker Hub-Repository

Wenn Sie aus einem Docker-Hub-Repository abrufen, das als offizielles Docker-Image gekennzeichnet ist, müssen Sie die `/library/` in der von Ihnen verwendeten URI angeben. Beispiel, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/docker-hub/`**`library/`***`image_name:tag`*. Wenn Sie die Option `/library/` für offizielle Docker Hub-Images weglassen, wird ein `403 Forbidden` Fehler zurückgegeben, wenn Sie versuchen, das Image mithilfe einer Pull-Through-Cache-Regel abzurufen. Weitere Informationen finden Sie unter Ein Bild mit einer Pull-Through-Cache-Regel in Amazon ECR abrufen.

Es folgt ein Beispiel dieses Fehlers.

```
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
```

Replikation privater Images in Amazon ECR

Sie können Ihre private Amazon-ECR-Registrierung so konfigurieren, dass sie die Replikation Ihrer Repositorys unterstützt. Amazon ECR unterstützt sowohl die regionen- als auch die kontoübergreifende Replikation. Damit eine kontoübergreifende Replikation stattfinden kann, muss das Zielkonto eine Richtlinie für Registrierungsberechtigungen konfigurieren, um die Replikation aus dem Quell-Registry zu ermöglichen. Weitere Informationen finden Sie unter Private Registrierungsberechtigungen in Amazon ECR.

Themen

Anforderungen an die Richtlinien für die kontoübergreifende Replikation

Damit die kontenübergreifende ECR-Replikation ordnungsgemäß funktioniert, müssen Sie wissen, für welches Konto welche Richtlinien konfiguriert werden müssen. In diesem Abschnitt werden die Richtlinienanforderungen für Quell- und Zielkonten erläutert.

Überblick über die Richtlinienkonfiguration

Für die kontenübergreifende ECR-Replikation ist eine Richtlinienkonfiguration nur für das Zielkonto erforderlich. Für das Quellkonto sind keine speziellen Repository- oder Registrierungsrichtlinien erforderlich.

- Quellkonto: Konfigurieren Sie die Replikationsregeln in den Registrierungseinstellungen. Für Quell-Repositorys sind keine zusätzlichen Richtlinien erforderlich.
- Zielkonto: Konfigurieren Sie eine Richtlinie für Registrierungsberechtigungen, damit das Quellkonto Bilder replizieren kann.

Anforderungen an die Zielregistrierungsrichtlinie

Das Zielkonto muss eine Registrierungsberechtigungsrichtlinie konfigurieren, die dem Quellkonto die Erlaubnis erteilt, die folgenden Aktionen auszuführen:

- `ecr:ReplicateImage`— Ermöglicht dem Quellkonto, Bilder in die Zielregistrierung zu replizieren
- `ecr:CreateRepository`- Ermöglicht ECR, automatisch Repositorys in der Zielregistrierung zu erstellen, falls sie noch nicht existieren

> **Important**
>
> Wenn Sie die `ecr:CreateRepository` Erlaubnis nicht erteilen, müssen Sie manuell Repositorys mit denselben Namen im Zielkonto erstellen, bevor die Replikation erfolgreich sein kann.

Beispiel für eine Zielregistrierungsrichtlinie:

JSON

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

Anforderungen an das Quellkonto

Das Quellkonto muss nur:

- Konfigurieren Sie die Replikationsregeln in den Registrierungseinstellungen, um das Zielkonto und die Regionen anzugeben
- Stellen Sie sicher, dass der IAM-Prinzipal, der die Replikation konfiguriert, über die erforderlichen ECR-Berechtigungen verfügt

Für Quell-Repositorys sind keine zusätzlichen Richtlinien erforderlich. Die Quell-Repositorys benötigen keine Repository-Richtlinien, die Replikationsberechtigungen gewähren.

Häufige Missverständnisse

Im Folgenden finden Sie häufige Missverständnisse zu kontenübergreifenden ECR-Replikationsrichtlinien:

- Irrtum: Das Quell-Repository benötigt eine Richtlinie, die es dem Zielkonto ermöglicht, Bilder zu replizieren.

  Realität: Quell-Repositorys benötigen keine speziellen Richtlinien für die Replikation.
- Irrtum: Sowohl Quell- als auch Zielkonten benötigen Registrierungsrichtlinien.

  Realität: Nur das Zielkonto benötigt eine Registrierungsberechtigungsrichtlinie.
- Irrtum: Repository-Richtlinien und Registrierungsrichtlinien sind dasselbe.

  Realität: Repository-Richtlinien kontrollieren den Zugriff auf einzelne Repositorys, während Registry-Richtlinien Operationen wie Replikation auf Registrierungsebene steuern.

Behebung von Replikationsfehlern

Wenn die kontoübergreifende Replikation fehlschlägt, überprüfen Sie Folgendes:

- Stellen Sie sicher, dass für das Zielkonto eine Registrierungsberechtigungsrichtlinie konfiguriert ist
- Stellen Sie sicher, dass die Registrierungsrichtlinie `ecr:ReplicateImage` sowohl `ecr:CreateRepository` Aktionen als auch umfasst
- Vergewissern Sie sich, dass die Quellkonto-ID in der Zielregistrierungsrichtlinie korrekt angegeben ist

- Prüfen Sie, ob die Ziel-Repositorys existieren (falls `ecr:CreateRepository` nicht gewährt)
- Überprüfen Sie die CloudTrail Protokolle auf fehlgeschlagene `CreateRepository` Aufrufe oder `ReplicateImage` API-Aufrufe

Überlegungen zur privaten Image-Replikation

Bei der Verwendung der privaten Image-Replikation sollte Folgendes beachtet werden.

- Nur Repository-Inhalte, die nach der Konfiguration der Replikation in ein Repository übertragen oder wiederhergestellt wurden, werden repliziert. Bereits vorhandener Inhalt in einem Repository wird nicht repliziert. Wenn ein Image wiederhergestellt wird, nachdem die Replikation aktiviert wurde, wird es repliziert. Wenn es wiederhergestellt wird, bevor die Replikation aktiviert wurde, wird es nicht repliziert.
- Der Repository-Name bleibt in allen Regionen und Konten gleich, wenn die Replikation stattgefunden hat. Amazon ECR unterstützt das Ändern des Repository-Namens während der Replikation nicht.
- Wenn Sie Ihre private Registrierung zum ersten Mal für die Replikation konfigurieren, erstellt Amazon ECR in Ihrem Namen eine serviceverknüpfte IAM-Rolle. Die serviceverknüpfte IAM-Rolle gewährt dem Amazon-ECR-Replikationsservice die Berechtigung, Repositorys zu erstellen und Images in Ihrer Registrierung zu replizieren. Weitere Informationen finden Sie unter Verwendung von dienstgebundenen Rollen für Amazon ECR.
- Damit eine kontoübergreifende Replikation stattfinden kann, muss das Ziel der privaten Registrierung die Erlaubnis erteilen, dass die Quellregistrierung ihre Images replizieren kann. Dies geschieht durch die Festlegung einer Richtlinie für private Registrierungsberechtigungen. Weitere Informationen finden Sie unter Private Registrierungsberechtigungen in Amazon ECR.
- Wenn die Berechtigungsrichtlinie für eine private Registrierung geändert wird, um eine Berechtigung zu entfernen, können alle laufenden Replikationen, die zuvor gewährt wurden, abgeschlossen werden.
- Damit eine regionsübergreifende Replikation stattfinden kann, müssen sowohl das Quell- als auch das Zielkonto für die Region aktiviert sein, bevor Replikationsaktionen innerhalb oder zu dieser Region durchgeführt werden. Weitere Informationen finden Sie unter Verwalten von AWS - Regionen im Allgemeine Amazon Web Services-Referenz.
- Die regionsübergreifende Replikation zwischen AWS Partitionen wird nicht unterstützt. Zum Beispiel ein Repository in `us-west-2` kann nicht in `cn-north-1` repliziert werden. Weitere Informationen zu AWS Partitionen finden Sie unter ARN-Format in der AWS Allgemeinen Referenz.

- Die Replikationskonfiguration für eine private Registrierung kann bis zu 25 eindeutige Ziele für alle Regeln enthalten, wobei die Gesamtzahl der Regeln 10 nicht überschreiten darf. Jede Regel kann bis zu 100 Filter enthalten. Auf diese Weise können separate Regeln für Repositories festgelegt werden, die beispielsweise Images für die Produktion und für Tests enthalten.
- Die Replikationskonfiguration unterstützt die Filterung, welche Repositorys in einer privaten Registrierung repliziert werden, indem ein Repository-Präfix angegeben wird. Ein Beispiel finden Sie unter Beispiel: Konfigurieren der regionsübergreifenden Replikation mithilfe eines Repository-Filters.
- Eine Replikationsaktion erfolgt nur einmal pro Image-Push oder Imagewiederherstellung. Wenn Sie z. B. die regionsübergreifende Replikation von `us-west-2` nach `us-east-1` und von `us-east-1` nach `us-east-2` konfiguriert haben, wird ein Image, das in die Region `us-west-2` verschoben wird, nur in die Region `us-east-1` repliziert, nicht aber in die Region `us-east-2`. Dieses Verhalten gilt sowohl für die regionen- als auch für die kontoübergreifende Replikation.
- Die Mehrheit der Bilder repliziert sich in weniger als 30 Minuten, aber in seltenen Fällen kann die Replikation länger dauern.
- Bei der Registrierungsreplikation werden keine Lösch- oder Archivierungsaktionen ausgeführt. Replizierte Images und Repositorys können gelöscht oder archiviert werden, wenn sie nicht mehr verwendet werden.
- Wenn das zu replizierende Bild im Ziel archiviert ist, wird es im Ziel wiederhergestellt.
- Wenn ein Bild in einer Quellregion archiviert wird, wird es nicht in einer in der Replikationskonfiguration angegebenen Zielregion archiviert.
- Repository-Richtlinien, einschließlich IAM-Richtlinien, und Lebenszyklus-Richtlinien werden nicht repliziert und haben nur Auswirkungen auf das Repository, für das sie definiert sind.
- Repository-Einstellungen werden standardmäßig nicht repliziert. Sie können die Repository-Einstellungen mithilfe von Vorlagen zur Repository-Erstellung replizieren. Zu diesen Einstellungen gehören die Veränderbarkeit von Tags, Verschlüsselung, Repository-Berechtigungen und Lebenszyklusrichtlinien. Weitere Informationen zu Vorlagen für die Erstellung von Repositorys finden Sie unterVorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.
- Wenn die Tag-Unveränderlichkeit in einem Repository aktiviert ist und ein Image repliziert wird, das denselben Tag wie ein bestehendes Image verwendet, wird das Image repliziert, enthält aber nicht den duplizierten Tag. Dies kann dazu führen, dass das Image nicht gekennzeichnet wird.

Beispiele für die Replikation privater Images für Amazon ECR

Die folgenden Beispiele zeigen häufige Anwendungsfälle für die Replikation privater Images. Wenn Sie die Replikation mithilfe von konfigurieren AWS CLI, können Sie die JSON-Beispiele als Ausgangspunkt verwenden, wenn Sie Ihre JSON-Datei erstellen. Wenn Sie die Replikation mithilfe von konfigurieren AWS-Managementkonsole, wird Ihnen bei der Überprüfung Ihrer Replikationsregel auf der Seite Überprüfen und Absenden ein ähnliches JSON angezeigt.

Beispiel: Konfigurieren der regionenübergreifenden Replikation in eine einzige Zielregion

Im Folgenden wird ein Beispiel für die Konfiguration der regionenübergreifenden Replikation innerhalb einer einzelnen Registrierung gezeigt. In diesem Beispiel wird davon ausgegangen, dass Ihre Konto-ID `111122223333` ist und dass Sie diese Replikationskonfiguration in einer anderen Region als `us-west-2` angeben.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Beispiel: Konfigurieren der regionsübergreifenden Replikation mithilfe eines Repository-Filters

Im Folgenden finden Sie ein Beispiel für die Konfiguration der regionsübergreifenden Replikation für Repositorys, die einem Präfixnamenwert entsprechen. In diesem Beispiel wird davon ausgegangen, dass Ihre Konto-ID `111122223333` ist und dass Sie diese Replikationskonfiguration in einer anderen Region als `us-west-1` angeben und über Repositories mit dem Präfix `prod` verfügen.

```
{
 "rules": [{
```

```
  "destinations": [{
   "region": "us-west-1",
   "registryId": "111122223333"
  }],
  "repositoryFilters": [{
   "filter": "prod",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

Beispiel: Konfigurieren der regionenübergreifenden Replikation an mehrere Zielregionen

Im Folgenden wird ein Beispiel für die Konfiguration der regionenübergreifenden Replikation innerhalb einer einzelnen Registrierung gezeigt. In diesem Beispiel wird davon ausgegangen, dass Ihre Konto-ID lautet `111122223333` und dass Sie diese Replikationskonfiguration in einer anderen Region als `us-west-1` oder angeben`us-west-2`.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-1",
                    "registryId": "111122223333"
                },
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

Beispiel: Konfigurieren der kontoübergreifenden Replikation

Im Folgenden finden Sie ein Beispiel für die Konfiguration der kontoübergreifenden Replikation für Ihre Registrierung. In diesem Beispiel wird die Replikation auf das Konto `444455556666` und auf die Region `us-west-2` konfiguriert.

> **Important**
>
> Damit eine kontoübergreifende Replikation stattfinden kann, muss das Zielkonto eine Richtlinie für Registrierungsberechtigungen konfigurieren, um die Replikation zu ermöglichen. Weitere Informationen finden Sie unter Private Registrierungsberechtigungen in Amazon ECR.

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "444455556666"
                }
            ]
        }
    ]
}
```

Beispiel: Festlegen mehrerer Regeln in einer Konfiguration

Im Folgenden finden Sie ein Beispiel für die Konfiguration mehrerer Replikationsregeln für Ihre Registrierung. In diesem Beispiel wird die Replikation für das *111122223333*-Konto mit einer Regel konfiguriert, die Repositorys mit einem Präfix von `prod` in die Region `us-west-2` und Repositorys mit einem Präfix von `test` in die Region `us-east-2` repliziert. Eine Replikationskonfiguration kann bis zu 10 Regeln enthalten, wobei jede Regel bis zu 25 Ziele angeben kann.

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

Beispiel: Alle Replikationseinstellungen werden entfernt

Im Folgenden finden Sie ein Beispiel für das Entfernen aller Replikationseinstellungen aus Ihrer Registrierung. Um Replikationseinstellungen zu entfernen, müssen Sie ein leeres Regelarray konfigurieren.

```
{
    "rules": []
}
```

> **Important**
>
> Durch das Entfernen von Replizierungseinstellungen werden keine zuvor replizierten Repositorys oder Images gelöscht. Sie müssen replizierte Inhalte manuell löschen, wenn sie nicht mehr benötigt werden.

Konfiguration der privaten Image-Replikation in Amazon ECR

Konfigurieren Sie die Replikation pro Region für Ihre private Registrierung. Sie können die regionsübergreifende Replikation oder die kontoübergreifende Replikation konfigurieren.

Beispiele dafür, wie Replikation häufig verwendet wird, finden Sie unter Beispiele für die Replikation privater Images für Amazon ECR.

So konfigurieren Sie die Einstellungen für die Registrierungsreplikation (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositorys.
2. Wählen Sie in der Navigationsleiste die Region aus, für die Sie die Einstellungen für die Registrierungsreplikation konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung.
4. Wählen Sie auf der Seite Private Registry die Option Einstellungen und dann unter Replikationskonfiguration die Option Bearbeiten aus.
5. Wählen Sie auf der Seite Replikation die Option Replikationsregel hinzufügen.
6. Wählen Sie auf der Seite Zieltypen, ob Sie die regionenübergreifende Replikation, die kontoübergreifende Replikation oder beides aktivieren möchten, und wählen Sie dann Weiter.
7. Wenn die regionenübergreifende Replikation aktiviert ist, wählen Sie unter Zielregionen konfigurieren eine oder mehrere Zielregionen aus und wählen dann Weiter.
8. Wenn die kontoübergreifende Replikation aktiviert ist, wählen Sie für die kontoübergreifende Replikation die Einstellung für die kontoübergreifende Replikation für die Registrierung. Geben Sie unter Zielkonto die Konto-ID für das Zielkonto und eine oder mehrere Zielregionen ein, in die repliziert werden soll. Wählen Sie Zielkonto +, um weitere Konten als Replikationsziele zu konfigurieren.

   > **Important**
   >
   > Damit eine kontoübergreifende Replikation stattfinden kann, muss das Zielkonto eine Richtlinie für Registrierungsberechtigungen konfigurieren, um die Replikation zu ermöglichen. Weitere Informationen finden Sie unter Private Registrierungsberechtigungen in Amazon ECR.

9. (Optional) Geben Sie auf der Seite Filter hinzufügen einen oder mehrere Filter für die Replikationsregel an und wählen Sie dann Hinzufügen. Wiederholen Sie diesen Schritt für jeden Filter, den Sie mit der Replikationsaktion verknüpfen möchten. Ein Filter muss als Präfix für den Repository-Namen angegeben werden. Wenn keine Filter hinzugefügt werden, wird der Inhalt aller Repositorys repliziert. Wählen Sie Weiter, wenn Sie alle Filter hinzugefügt haben.
10. Überprüfen Sie auf der Seite Überprüfen und Übermitteln die Konfiguration der Replikationsregel und wählen Sie anschließend Regel übermitteln.

So konfigurieren Sie die Einstellungen für die Registrierungsreplikation (AWS CLI)

1. Erstellen Sie eine JSON-Datei mit den Replikationsregeln, die Sie für Ihre Registrierung definieren müssen. Eine Replikationskonfiguration kann bis zu 10 Regeln enthalten, wobei jede Regel bis zu 25 einzigartige Ziele und 100 Filter angeben kann. Um die regionenübergreifende Replikation innerhalb Ihres eigenen Kontos zu konfigurieren, geben Sie Ihre eigene Konto-ID an. Weitere Beispiele finden Sie unter Beispiele für die Replikation privater Images für Amazon ECR.

   ```
   {
    "rules": [{
     "destinations": [{
      "region": "destination_region",
      "registryId": "destination_accountId"
     }],
     "repositoryFilters": [{
      "filter": "repository_prefix_name",
      "filterType": "PREFIX_MATCH"
     }]
    }]
   }
   ```

2. Erstellen Sie eine Replikationskonfiguration für Ihre Registrierung.

   ```
   aws ecr put-replication-configuration \
        --replication-configuration file://replication-settings.json \
        --region us-west-2
   ```

3. Bestätigen Sie Ihre Registrierungseinstellungen.

   ```
   aws ecr describe-registry \
        --region us-west-2
   ```

Replikationseinstellungen für private Bilder in Amazon ECR entfernen

Um die Replikationseinstellungen für Ihre private Registrierung zu entfernen oder zu deaktivieren, müssen Sie eine leere Replikationskonfiguration konfigurieren. In der gibt es keinen speziellen Befehl zum Entfernen AWS CLI.

Um die Einstellungen für die Replikation der Registrierung zu entfernen (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositorys.
2. Wählen Sie in der Navigationsleiste die Region aus, aus der Sie Ihre Registrierungsreplikationseinstellungen entfernen möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung.
4. Wählen Sie auf der Seite Private Registrierung die Option Einstellungen und dann unter Replikationskonfiguration die Option Bearbeiten aus.
5. Entfernen Sie alle vorhandenen Replikationsregeln, indem Sie für jede Regel die Option Löschen auswählen.
6. Wählen Sie Speichern, um die leere Replikationskonfiguration anzuwenden.

Um die Einstellungen für die Replikation der Registrierung zu entfernen (AWS CLI)

1. Erstellen Sie eine JSON-Datei mit einem leeren Regelarray, um alle Replikationseinstellungen zu entfernen.

```
{
    "rules": []
}
```

2. Wenden Sie die leere Replikationskonfiguration auf Ihre Registrierung an.

```
aws ecr put-replication-configuration \
     --replication-configuration file://empty-replication-settings.json \
     --region us-west-2
```

3. Vergewissern Sie sich, dass die Replikationseinstellungen entfernt wurden.

```
aws ecr describe-registry \
     --region us-west-2
```

   In der Ausgabe sollte ein leeres Feld `replicationConfiguration` ohne Regeln angezeigt werden.

Important

Durch das Entfernen von Replizierungseinstellungen werden keine zuvor replizierten Repositorys oder Images gelöscht. Sie müssen replizierte Inhalte manuell löschen, wenn sie nicht mehr benötigt werden.

Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden

Verwenden Sie Vorlagen zur Erstellung von Amazon ECR-Repositorys, um die Einstellungen für Repositorys zu definieren, die von Amazon ECR in Ihrem Namen erstellt wurden. Die Einstellungen in einer Repository-Erstellungsvorlage werden nur bei der Repository-Erstellung angewendet und haben keine Auswirkung auf bestehende Repositorys oder Repositorys, die mit einer anderen Methode erstellt wurden. Derzeit können Vorlagen für die Erstellung von Repositorys verwendet werden, um während der Repository-Erstellung Einstellungen für die folgenden Funktionen anzuwenden:

- Den Cache durchsuchen
- Beim Push erstellen
- Replikation

So funktionieren Repository-Erstellungsvorlagen

Manchmal muss Amazon ECR in Ihrem Namen ein neues privates Repository erstellen. Beispiel:

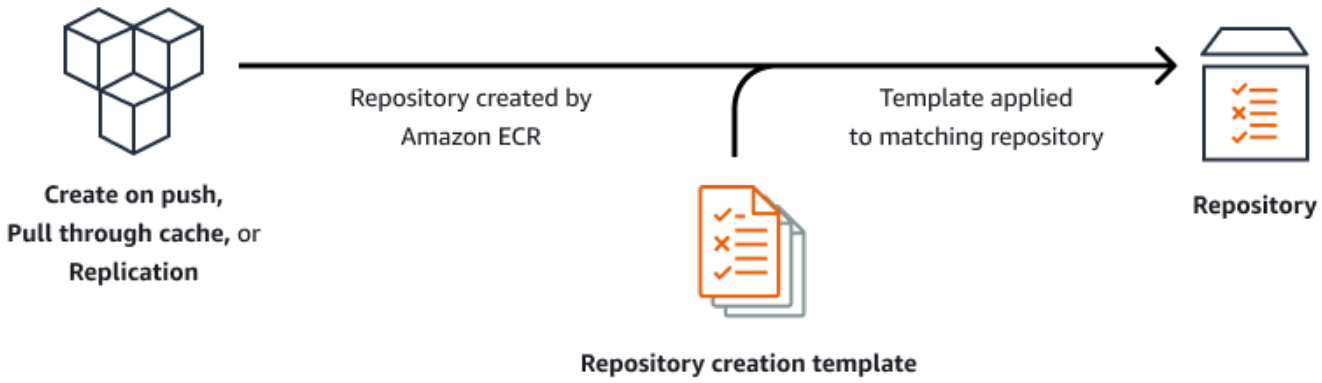
- Wenn Sie zum ersten Mal eine Pull-Through-Cache-Regel verwenden, um den Inhalt eines Upstream-Repositorys abzurufen und in Ihrer privaten Amazon ECR-Registrierung zu speichern.
- Wenn Sie ein Bild in ein Repository übertragen, das noch nicht existiert.
- Wenn Sie möchten, dass Amazon ECR ein Repository in eine andere Region oder ein anderes Konto repliziert.

Wenn es keine Vorlage für die Repository-Erstellung gibt, die Ihrer Pull-Through-Cache-Regel oder Ihrem replizierten Repository entspricht, verwendet Amazon ECR die Standardeinstellungen für das neue Repository. Zu diesen Standardeinstellungen gehören die Deaktivierung der Unveränderlichkeit von Tags, die Verwendung der `AES-256`-Verschlüsselung und die Nichtanwendung von Repository- oder Lebenszyklusrichtlinien.

Wenn es keine Vorlage für die Repository-Erstellung gibt, die mit dem Ziel-Repository für einen Image-Push übereinstimmt, erstellt Amazon ECR kein Repository mit Standardeinstellungen.

Mithilfe einer Vorlage zur Repository-Erstellung können Sie die Einstellungen definieren, die Amazon ECR auf neue Repositorys anwendet, die mit den Aktionen Pull-Through-Cache, Create on Push und Replikation erstellt wurden. Sie können die Unveränderlichkeit von Tags, die Verschlüsselungskonfiguration, die Repository-Berechtigungen, die Lebenszyklusrichtlinie und die Ressourcen-Tags für die neuen Repositorys definieren.

Das folgende Diagramm zeigt den Arbeitsablauf, den Amazon ECR verwendet, wenn eine Repository-Erstellungsvorlage verwendet wird.



Im Folgenden werden die einzelnen Parameter in einer Repository-Erstellungsvorlage detailliert beschrieben.

Präfix

Das Präfix ist das Namespace-Präfix für das Repository, das der Vorlage zugeordnet werden soll. Auf alle Repositorys, die mit diesem Präfix erstellt wurden, werden die in dieser Vorlage definierten Einstellungen angewendet. Das Präfix `prod` würde beispielsweise für alle Repositorys gelten, die mit `prod/` beginnen. Ähnlich würde das Präfix `prod/team` für alle Repositorys gelten, die mit `prod/team/` beginnen. In einer Registrierung, die zwei Vorlagen enthält, wenn eine Vorlage das Präfix „prod“ und die andere das Präfix "prod/team", the template with the prefix "prod/team" will be applied to all repositories whose names start with "prod/team/“ hat.

Um eine Vorlage auf alle Repositorys in Ihrer Registrierung anzuwenden, denen keine Erstellungsvorlage zugeordnet ist, können Sie sie `ROOT` als Präfix verwenden.

> **Important**
>
> Es wird immer ein `/` am Ende des Präfixes angenommen. Wenn Sie `ecr-public` als Präfix angeben, behandelt Amazon ECR dies als `ecr-public/`. Wenn Sie eine Pull-Through-Cache-Regel verwenden, sollten Sie das Repository-Präfix, das Sie bei der

Erstellung der Regel angeben, auch als Präfix für Ihre Repository-Erstellungsvorlage verwenden.

Description

Diese Vorlagenbeschreibung ist optional und wird verwendet, um den Zweck der Vorlage für die Repository-Erstellung zu beschreiben.

Beantragt

Die Einstellung „Angewendet für“ bestimmt, welche von Amazon ECR erstellten Repositorys mit dieser Vorlage erstellt werden. Die gültigen Werte sind `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH` und `REPLICATION`. Zum Beispiel das erste Mal, wenn Sie eine Pull-Through-Cache-Regel verwenden, um den Inhalt eines Upstream-Repositorys abzurufen und in Ihrer privaten Registrierung von Amazon ECR zu speichern. Wenn es keine Repository-Erstellungsvorlage gibt, die Ihrer Pull-Through-Cache-Regel entspricht, verwendet Amazon ECR die Standardeinstellungen für das neue Repository.

Rolle beim Erstellen eines Repositor

Die Rolle zur Repository-Erstellung ist eine IAM-Rolle (ARN), die von Amazon ECR bei der Erstellung und Konfiguration von Repositorys mithilfe von Vorlagen zur Repository-Erstellung übernommen wird. Diese Rolle muss angegeben werden, wenn Repository-Tags and/or KMS in der Vorlage verwendet werden, andernfalls schlägt die Repository-Erstellung fehl.

Veränderlichkeit von Image-Tags

Die Einstellung für die Veränderlichkeit von Tags, die für mit der Vorlage erstellte Repositorys verwendet werden soll. Wenn dieser Parameter ausgelassen wird, wird die Standardeinstellung VERÄNDERLICH verwendet, mit der Image-Tags überschrieben werden können. Dies ist die empfohlene Einstellung für Vorlagen, die für Repositorys verwendet werden, die durch Pull-Through-Cache-Aktionen erstellt wurden. Dadurch wird sichergestellt, dass Amazon ECR die zwischengespeicherten Images aktualisieren kann, wenn die Tags identisch sind.

Wenn UNVERÄNDERLICH angegeben ist, können keine Image-Tags innerhalb des Repositorys verändert werden. Dies verhindert ihre Überschreibung.

Verschlüsselungskonfiguration

> **Important**
>
> Die zweischichtige serverseitige Verschlüsselung mit AWS KMS (DSSE-KMS) ist nur in den AWS GovCloud (US) Regionen verfügbar.

Die Verschlüsselungskonfiguration, die für Repositorys verwendet werden soll, die mit der Vorlage erstellt wurden.

Wenn Sie den Verschlüsselungstyp KMS verwenden, wird der Inhalt des Repository über die serverseitige Verschlüsselung mit dem AWS Key Management Service -Schlüssel in AWS KMS gespeichert. Wenn Sie Ihre Daten verschlüsseln, können Sie entweder den AWS verwalteten AWS KMS Standardschlüssel für Amazon ECR verwenden oder Ihren eigenen AWS KMS Schlüssel angeben, den Sie bereits erstellt haben. AWS KMS Sie können außerdem wählen, ob Sie die einschichtige oder die zweischichtige Verschlüsselung mit verwenden möchten. AWS KMS Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand. Wenn Sie den KMS-Verschlüsselungstyp und ihn für die regionsübergreifende Replikation verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Weitere Informationen finden Sie unter Erstellen einer KMS-Schlüsselrichtlinie für die Replikation.

Wenn Sie den AES256-Verschlüsselungs-Typ verwenden, verwendet Amazon ECR eine serverseitige Verschlüsselung mit von Amazon S3 verwalteten Verschlüsselungsschlüsseln, die die Images im Repository mit einem AES-256-Verschlüsselungs-Algorithmus verschlüsseln. Weitere Informationen finden Sie unter Schützen von Daten mit serverseitiger Verschlüsselung mit Amazon-S3-verwalteten Verschlüsselungsschlüsseln (SSE-S3) im Benutzerhandbuch von Amazon Simple Storage Service.

Repository-Berechtigungen

Die Repository-Richtlinie, die auf Repositorys angewendet werden soll, die mit der Vorlage erstellt wurden. Eine Repository-Richtlinie verwendet ressourcenbasierte Berechtigungen, um den Zugriff auf ein Repository zu kontrollieren. Mit ressourcenbasierten Berechtigungen können Sie festlegen, welche IAM-Benutzer oder -Rollen Zugriff auf ein Repository haben und welche Aktionen sie damit durchführen können. Standardmäßig hat nur das AWS Konto, mit dem das Repository erstellt wurde, Zugriff auf ein Repository. Sie können ein Richtliniendokument anwenden, um zusätzliche Berechtigungen für Ihr Repository zu gewähren oder zu verweigern. Weitere Informationen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.

Lebenszyklusrichtlinie für Repositorys

Die Lebenszyklusrichtlinie, die für Repositorys verwendet werden soll, die mit der Vorlage erstellt wurden. Eine Lebenszyklusrichtlinie bietet mehr Kontrolle über die Lebenszyklusverwaltung von Images in einem privaten Repository. Eine Lebenszyklusrichtlinie enthält eine oder mehrere Regeln, wobei jede Regel eine Aktion für Amazon ECR definiert. Auf diese Weise können Sie die Bereinigung Ihrer Container-Images automatisieren, indem Sie die Images aufgrund ihres Alters oder ihrer Anzahl ablaufen lassen.. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

Ressourcen-Tags

Die Ressourcen-Tags sind Metadaten, die auf das Repository angewendet werden können, um die Kategorisierung und Organisation zu erleichtern. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen. Diese Berechtigung muss auf die Zielregistrierungsrichtlinie angewendet werden, wenn Sie Vorlagen zur Repository-Erstellung mit regionsübergreifender Replikation verwenden.

Vorlage für die Erstellung eines Repositorys in Amazon ECR erstellen

Sie können eine Repository-Erstellungsvorlage erstellen, um die Einstellungen zu definieren, die für Repositorys verwendet werden sollen, die von Amazon ECR in Ihrem Namen während der Aktionen Pull-Through-Cache, Create on Push oder Replikation erstellt wurden. Sobald die Repository-Erstellungsvorlage erstellt wurde, werden die Einstellungen auf alle neu erstellten Repositorys angewendet. Dies hat keine Auswirkungen auf zuvor erstellte Repositorys.

Wenn Sie ein Repository mit Vorlagen einrichten, haben Sie die Möglichkeit, KMS-Schlüssel und Ressourcen-Tags anzugeben. Wenn Sie KMS-Schlüssel, Ressourcen-Tags oder eine Kombination aus beidem in einer oder mehreren Vorlagen verwenden möchten, müssen Sie:

- Erstellen Sie eine benutzerdefinierte Richtlinie für Vorlagen zur Erstellung von Repositorys.
- Erstellen Sie eine IAM-Rolle für Vorlagen zur Repository-Erstellung.

Nach der Konfiguration können Sie die benutzerdefinierte Rolle bestimmten Vorlagen in Ihrer Registrierung zuordnen.

IAM-Berechtigungen zum Erstellen von Vorlagen für die Erstellung von Repositorys

Die folgenden Berechtigungen sind erforderlich, damit ein IAM-Prinzipal Repository-Erstellungsvorlagen verwalten kann. Diese Berechtigungen müssen über eine identitätsbasierte IAM-Richtlinie gewährt werden.

- `ecr:CreateRepositoryCreationTemplate` – Erteilt die Berechtigung zum Erstellen einer Repository-Erstellungsvorlage.
- `ecr:UpdateRepositoryCreationTemplate`— Erteilt die Erlaubnis, eine Vorlage für die Repository-Erstellung zu aktualisieren.
- `ecr:DescribeRepositoryCreationTemplates`— Erteilt die Erlaubnis, Vorlagen für die Erstellung eines Repositorys in einer Registrierung aufzulisten.
- `ecr:DeleteRepositoryCreationTemplate` – Erteilt die Berechtigung zum Löschen einer Repository-Erstellungsvorlage.
- `ecr:CreateRepository`— Erteilt die Erlaubnis, ein Amazon ECR-Repository zu erstellen.
- `ecr:PutLifecyclePolicy` – Erteilt die Berechtigung zum Erstellen einer Lebenszyklusrichtlinie und deren Anwendung auf ein Repository. Diese Berechtigung ist nur erforderlich, wenn die Repository-Erstellungsvorlage eine Lebenszyklusrichtlinie enthält.
- `ecr:SetRepositoryPolicy` – Erteilt die Berechtigung zum Erstellen einer Berechtigungsrichtlinie für ein Repository. Diese Berechtigung ist nur erforderlich, wenn die Repository-Erstellungsvorlage eine Repository-Richtlinie enthält.
- `iam:PassRole`— Erteilt die Erlaubnis, einer Entität zu gestatten, eine Rolle an einen Service oder eine Anwendung zu übergeben. Diese Berechtigung ist für Dienste und Anwendungen erforderlich, die eine Rolle übernehmen müssen, um Aktionen in Ihrem Namen auszuführen.

Erstellen Sie eine benutzerdefinierte Richtlinie für Vorlagen zur Erstellung von Repositorys

Sie können die verwenden AWS-Managementkonsole , um eine Richtlinie zu definieren, die anschließend einer IAM-Rolle zugeordnet wird. Diese IAM-Rolle kann dann bei der Konfiguration einer Vorlage für die Repository-Erstellung als Rolle für die Repository-Erstellung verwendet werden.

AWS-Managementkonsole

Um den JSON-Policy-Editor zu verwenden, um eine benutzerdefinierte Richtlinie für Vorlagen zur Repository-Erstellung zu erstellen.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).
3. Wählen Sie Richtlinie erstellen aus.
4. Wählen Sie im Bereich Policy editor (Richtlinien-Editor) die Option JSON aus.
5. Geben Sie die folgende Richtlinie in das JSON-Feld ein.

   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. Beheben Sie alle Sicherheitswarnungen, Fehler oder allgemeinen Warnungen, die während der Richtlinienvalidierung generiert wurden, und wählen Sie dann Weiter.

7. Wenn Sie mit dem Hinzufügen von Berechtigungen zur Richtlinie fertig sind, wählen Sie Next (Weiter) aus.
8. Geben Sie auf der Seite Review and create (Überprüfen und erstellen) unter Name einen Namen und unter Description (Beschreibung) (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie Permissions defined in this policy (In dieser Richtlinie definierte Berechtigungen), um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden.
9. Wählen Sie Create policy (Richtlinie erstellen) aus, um Ihre neue Richtlinie zu speichern.
10. Erstellen Sie eine Rolle, um diese Richtlinie für die Erstellungsvorlage zuzuweisen, sieheErstellen Sie eine IAM-Rolle für Vorlagen zur Repository-Erstellung.

Erstellen Sie eine IAM-Rolle für Vorlagen zur Repository-Erstellung

Sie können die verwenden, AWS-Managementkonsole um eine Rolle zu erstellen, die von Amazon ECR verwendet werden kann, wenn Sie die Rolle zur Repository-Erstellung in einer Repository-Erstellungsvorlage angeben, die Repository-Tags oder KMS in einer Vorlage verwendet.

AWS-Managementkonsole

Um eine Rolle zu erstellen.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Klicken Sie im Navigationsbereich der Konsole auf Roles (Rollen) und wählen Sie dann Create role (Rolle erstellen).
3. Wählen Sie den Rollentyp Benutzerdefinierte Vertrauensrichtlinie.
4. Fügen Sie im Abschnitt Benutzerdefinierte Vertrauensrichtlinie die unten aufgeführte benutzerdefinierte Vertrauensrichtlinie ein:

JSON

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

```
            "Service": "ecr.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
    ]
}
```

5. Wählen Sie Weiter aus.
6. Aktivieren Sie auf der Seite „Berechtigungen hinzufügen“ in der Liste der Berechtigungsrichtlinien das Kontrollkästchen neben der benutzerdefinierten Richtlinie, die Sie zuvor erstellt haben, und wählen Sie Weiter aus.
7. Geben Sie unter Role name (Rollenname) einen Namen für Ihre Rolle ein. Rollennamen müssen innerhalb Ihres Unternehmens eindeutig sein AWS-Konto. Wenn ein Rollenname in einer Richtlinie oder als Teil eines ARN verwendet wird, muss die Groß-/Kleinschreibung des Rollennamens beachtet werden. Wenn Kunden in der Konsole ein Rollenname angezeigt wird, beispielsweise während des Anmeldevorgangs, wird die Groß-/Kleinschreibung des Rollennamens nicht beachtet. Da verschiedene Entitäten möglicherweise auf die Rolle verweisen, können Sie den Namen der Rolle nach der Erstellung nicht mehr bearbeiten.
8. (Optional) Geben Sie unter Role description (Rollenbeschreibung) eine Beschreibung für die neue Rolle ein.
9. Prüfen Sie die Rolle und klicken Sie dann auf Rolle erstellen.

Erstellen Sie eine Vorlage zur Erstellung eines Repositorys

Sobald Sie die erforderlichen Voraussetzungen für Ihre Vorlagen erfüllt haben, können Sie mit der Erstellung der Vorlagen für die Repository-Erstellung fortfahren.

AWS-Managementkonsole

So erstellen Sie eine Repository-Erstellungsvorlage (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region, in der Sie die Repository-Erstellungsvorlage erstellen möchten.
3. Wählen Sie im Navigationsbereich Private Registrierung und Repository-Erstellungsvorlagen aus.
4. Wählen Sie auf der Seite Repository-Erstellungsvorlagen die Option Vorlage erstellen aus.

5. Wählen Sie auf der Seite Schritt 1: Vorlage definieren unter Vorlagendetails die Option Ein bestimmtes Präfix aus, um die Vorlage auf ein bestimmtes Repository-Namespace-Präfix anzuwenden, oder wählen Sie Beliebiges Präfix in Ihrer ECR-Registrierung, um die Vorlage auf alle Repositorys anzuwenden, die keiner anderen Vorlage in der Region entsprechen.

   a. Wenn Sie Ein bestimmtes Präfix wählen, geben Sie unter Präfix das Repository-Namespace-Präfix an, auf das die Vorlage angewendet werden soll. Es wird immer ein `/` am Ende des Präfixes angenommen. Das Präfix `prod` würde beispielsweise für alle Repositorys gelten, die mit `prod/` beginnen. Ähnlich würde das Präfix `prod/team` für alle Repositorys gelten, die mit `prod/team/` beginnen.

   b. Wenn Sie Beliebiges Präfix in Ihrer ECR-Registrierung wählen, wird das Präfix auf `ROOT` gesetzt.

6. Geben Sie unter Beantragt für an, für welche Amazon ECR-Workflows diese Vorlage gelten soll. Die Optionen lauten `PULL_THROUGH_CACHE`, `CREATE_ON_PUSH` und `REPLICATION`.

7. Geben Sie unter Vorlagenbeschreibung eine optionale Beschreibung für die Vorlage ein und wählen Sie dann Weiter.

8. Geben Sie auf der Seite Schritt 2: Konfiguration für Repository-Erstellung hinzufügen die Konfiguration der Repository-Einstellungen an, die auf Repositorys angewendet werden soll, die mit der Vorlage erstellt wurden.

   a. Wählen Sie für Veränderlichkeit von Image-Tags die zu verwendende Einstellung für die Veränderlichkeit von Tags. Weitere Informationen finden Sie unter Verhindern, dass Bild-Tags in Amazon ECR überschrieben werden.

      - Veränderbar — Wählen Sie diese Option, wenn Sie möchten, dass Bild-Tags überschrieben werden. Empfohlen für Repositorys, die Pull-Through-Cache-Aktionen verwenden, um sicherzustellen, dass Amazon ECR zwischengespeicherte Bilder aktualisieren kann. Um Tag-Updates für einige veränderbare Tags zu deaktivieren, geben Sie außerdem Tag-Namen ein oder verwenden Sie Platzhalter (*), um mehrere ähnliche Tags im Textfeld zum Ausschluss veränderbarer Tags abzugleichen.
      - Unveränderlich — Wählen Sie diese Option, wenn Sie verhindern möchten, dass Bild-Tags überschrieben werden. Sie gilt für alle Tags und Ausschlüsse im Repository, wenn Sie ein Bild mit vorhandenem Tag pushen. Amazon ECR gibt eine zurück, `ImageTagAlreadyExistsException` wenn Sie versuchen, ein Bild mit einem vorhandenen Tag zu pushen. Um Tag-Updates für einige unveränderliche Tags zu aktivieren, geben Sie außerdem Tagnamen ein oder verwenden Sie Platzhalter

(*), um mehrere ähnliche Tags im Textfeld zum Ausschluss unveränderlicher Tags abzugleichen.

b. Wählen Sie für die Verschlüsselungskonfiguration die zu verwendende Verschlüsselungseinstellung aus. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

Wenn AES-256 ausgewählt ist, verwendet Amazon ECR eine serverseitige Verschlüsselung mit von Amazon Simple Storage Service verwalteten Verschlüsselungsschlüsseln, die Ihre Daten im Ruhezustand mit dem branchenüblichen AES-256-Verschlüsselungsalgorithmus verschlüsseln. Dies wird ohne zusätzliche Kosten angeboten.

Wenn AWS -KMS ausgewählt ist, verwendet Amazon ECR serverseitige Verschlüsselung mit Schlüsseln, die in AWS Key Management Service (AWS KMS) gespeichert sind. Wenn Sie Ihre Daten verschlüsseln, können Sie entweder den standardmäßigen AWS verwalteten Schlüssel verwenden, der von Amazon ECR verwaltet wird, oder Ihren eigenen AWS KMS Schlüssel angeben, der als vom Kunden verwalteter Schlüssel bezeichnet wird. AWS KMS

> **Note**
>
> Die Verschlüsselungseinstellungen für ein Repository können nicht geändert werden, sobald das Repository erstellt wurde.

c. Geben Sie für Repository-Berechtigungen die Richtlinie für Repository-Berechtigungen an, die auf Repositorys angewendet werden soll, die mit dieser Vorlage erstellt wurden. Sie können optional das Auswahlmenü verwenden, um eines der JSON-Beispiele für die häufigsten Anwendungsfälle auszuwählen. Weitere Informationen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.

d. Geben Sie unter Repository-Lebenszyklusrichtlinie die Repository-Lebenszyklusrichtlinie an, die auf Repositorys angewendet werden soll, die mit dieser Vorlage erstellt wurden. Sie können optional das Auswahlmenü verwenden, um eines der JSON-Beispiele für die häufigsten Anwendungsfälle auszuwählen. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.

e. Geben Sie für AWS Repository-Tags die Metadaten in Form von Schlüssel-Wert-Paaren an, die den mit dieser Vorlage erstellten Repositorys zugeordnet werden sollen, und

wählen Sie dann Weiter. Weitere Informationen finden Sie unter Kennzeichnen eines privaten Repositorys in Amazon ECR.

f. Wählen Sie für die Rolle zum Erstellen eines Repositorys eine benutzerdefinierte IAM-Rolle aus dem Drop-down-Menü aus, die für Vorlagen zur Repository-Erstellung verwendet werden soll, wenn Repository-Tags oder KMS in der Vorlage verwendet werden (Erstellen Sie eine IAM-Rolle für Vorlagen zur Repository-ErstellungEinzelheiten finden Sie unter). Wählen Sie dann Weiter.

9. Überprüfen Sie auf der Seite Schritt 3: Überprüfen und erstellen die Einstellungen, die Sie für die Repository-Erstellungsvorlage angegeben haben. Sie können die Option Bearbeiten auswählen, um Änderungen vorzunehmen. Wählen Sie anschließend Erstellen.

AWS CLI

Der create-repository-creation-template AWS CLI Befehl wird verwendet, um eine Vorlage zur Erstellung eines Repositorys für Ihre private Registrierung zu erstellen.

So erstellen Sie eine Repository-Erstellungsvorlage (AWS CLI)

1. Verwenden Sie den AWS CLI , um ein Skelett für den create-repository-creation-templateBefehl zu generieren.

```
aws ecr create-repository-creation-template \
    --generate-cli-skeleton
```

In der Ausgabe des Befehls wird die vollständige Syntax der Vorlage für die Repository-Erstellung angezeigt.

```
{
"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. Erstellen Sie eine Datei `repository-creation-template.json` mit dem Namen der Ausgabe des vorherigen Schritts. Diese Vorlage legt einen KMS-Verschlüsselungsschlüssel für jedes Repository fest, das unter `prod/*` einer Repository-Richtlinie erstellt wurde, die das Pushen und Abrufen von Bildern in future Repositorys ermöglicht, legt eine Lebenszyklusrichtlinie fest, nach der Bilder, die älter als zwei Wochen sind, ablaufen, und legt eine benutzerdefinierte Rolle fest, mit der ECR auf den KMS-Schlüssel zugreifen und das Ressourcen-Tag future Repositorys zuweisen `examplekey` kann.

```
{
"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. Verwenden Sie den folgenden Befehl, um eine Vorlage für die Erstellung eines Repositorys zu erstellen. Stellen Sie sicher, dass Sie den Namen der im vorherigen Schritt erstellten Konfigurationsdatei anstelle des Namens `repository-creation-template.json` im folgenden Beispiel angeben.

```
aws ecr create-repository-creation-template \
    --cli-input-json file://repository-creation-template.json
```

Eine Vorlage zur Erstellung eines Repositorys aktualisieren

Sie können eine Vorlage für die Erstellung eines Repositorys bearbeiten, wenn Sie deren Konfigurationen ändern müssen. Sobald die Vorlage für die Repository-Erstellung bearbeitet wurde, gelten die neuen Konfigurationen für die bestehende Vorlage.

> **Important**
>
> Dies hat keine Auswirkungen auf zuvor erstellte Repositorys.

AWS-Managementkonsole

Um eine Vorlage für die Erstellung eines Repositorys zu bearbeiten (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der sich die zu bearbeitende Repository-Erstellungsvorlage befindet.
3. Wählen Sie im Navigationsbereich Private Registrierung und dann Einstellungen aus.
4. Wählen Sie in der Navigationsleiste die Vorlagen für die Repository-Erstellung aus.
5. Wählen Sie auf der Seite Vorlagen für die Repository-Erstellung die Vorlage für die Repository-Erstellung aus, die Sie bearbeiten möchten.
6. Wählen Sie im Dropdownmenü Aktionen die Option Bearbeiten aus.
7. Überprüfen und aktualisieren Sie die Konfigurationseinstellungen.
8. Wählen Sie „Aktualisieren“, um die Konfigurationen der neuen Erstellungsvorlage zu übernehmen.

AWS CLI

Um eine Vorlage für die Erstellung eines Repositorys zu bearbeiten (AWS CLI)

- Verwenden Sie den update-repository-creation-templateBefehl, um eine vorhandene Vorlage für die Erstellung eines Repositorys zu aktualisieren. Sie müssen den `prefix` Wert der Vorlage angeben. Im folgenden Beispiel wird eine Vorlage zur Erstellung eines Repositorys mit dem `prod` Präfix aktualisiert.

  ```
  aws ecr update-repository-creation-template \
       --prefix prod \
       --image-tag-mutability="IMMUTABLE_WITH_EXCLUSION" \
       --image-tag-mutability-exclusion-filters filterType=WILDCARD, filter=latest
  ```

  In der Ausgabe des Befehls werden die Details der aktualisierten Vorlage für die Repository-Erstellung angezeigt.

Löschen einer Repository-Erstellungsvorlage in Amazon ECR

Sie können eine Repository-Erstellungsvorlage löschen, wenn Sie sie nicht mehr verwenden. Sobald eine Vorlage für die Repository-Erstellung gelöscht wurde, erben alle neu erstellten Repositorys, die während eines Pull-Through-Cache- oder Replikationsvorgangs unter dem zugehörigen Präfix erstellt wurden, die Standardeinstellungen, sofern keine andere passende Vorlage gefunden wird, siehe. So funktionieren Repository-Erstellungsvorlagen

> **Important**
>
> Dies hat keine Auswirkungen auf zuvor erstellte Repositorys.

AWS-Managementkonsole

So löschen Sie eine Repository-Erstellungsvorlage (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region, in der Sie die Repository-Erstellungsvorlage löschen möchten.
3. Wählen Sie im Navigationsbereich Private Registrierung und Repository-Erstellungsvorlagen aus.
4. Wählen Sie auf der Seite Repository-Erstellungsvorlagen die Repository-Erstellungsvorlage aus, die Sie löschen möchten.
5. Wählen Sie im Auswahlmenü Aktionen Löschen aus.

AWS CLI

So löschen Sie eine Repository-Erstellungsvorlage (AWS CLI)

- Verwenden Sie den Befehl delete-repository-creation-template.html, um eine vorhandene Vorlage zur Erstellung eines Repositorys zu löschen. Sie müssen den `prefix` Wert der Vorlage angeben. Im folgenden Beispiel wird eine Vorlage zur Erstellung eines Repositorys mit dem `prod` Präfix gelöscht.

```
aws ecr delete-repository-creation-template \
     --prefix prod
```

In der Ausgabe des Befehls werden die Details der gelöschten Repository-Erstellungsvorlage angezeigt.

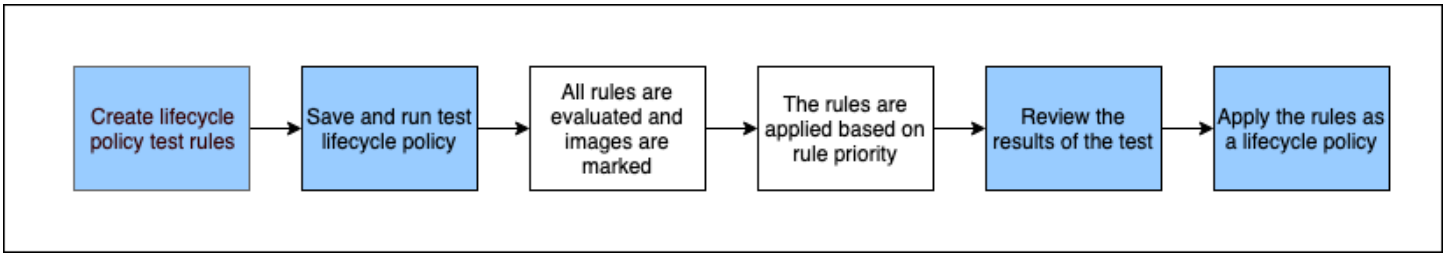
Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR

Amazon ECR-Lebenszyklusrichtlinien bieten mehr Kontrolle über das Lebenszyklusmanagement von Images in einem privaten Repository. Eine Lebenszyklusrichtlinie enthält eine oder mehrere Regeln, und jede Regel definiert eine Aktion für Amazon ECR. Auf der Grundlage der Ablaufkriterien in der Lebenszyklusrichtlinie können Bilder innerhalb von 24 Stunden archiviert werden oder sie können entsprechend den in der Lebenszyklusrichtlinie angegebenen Kriterien archiviert werden. Wenn Amazon ECR eine Aktion auf der Grundlage einer Lebenszyklusrichtlinie ausführt, wird diese Aktion als Ereignis in AWS CloudTrail erfasst. Weitere Informationen finden Sie unter Protokollierung von Amazon ECR-Aktionen mit AWS CloudTrail.

Wie Lebenszyklusrichtlinien funktionieren

Eine Lebenszyklusrichtlinie besteht aus einer oder mehreren Regeln, die festlegen, welche Images in einem Repository ablaufen sollen. Wenn Sie den Einsatz von Lebenszyklusrichtlinien in Erwägung ziehen, ist es wichtig, die Vorschau der Lebenszyklusrichtlinie zu verwenden, um zu bestätigen, welche Images die Lebenszyklusrichtlinie ablaufen lässt, bevor sie auf ein Repository angewendet wird. Sobald eine Lebenszyklusrichtlinie auf ein Repository angewendet wird, sollten Sie davon ausgehen, dass Images innerhalb von 24 Stunden, nachdem sie die Ablaufkriterien erfüllt haben, ablaufen. Wenn Amazon ECR eine Aktion basierend auf einer Lebenszyklusrichtlinie durchführt, wird dies als Ereignis in AWS CloudTrail angegeben. Weitere Informationen finden Sie unter Protokollierung von Amazon ECR-Aktionen mit AWS CloudTrail.

Das folgende Diagramm zeigt den Workflow der Lebenszyklusrichtlinie.



1. Erstellen Sie eine oder mehrere Testregeln.
2. Speichern Sie die Testregeln und führen Sie die Vorschau aus.
3. Der Lifecycle Policy Evaluator geht alle Regeln durch und markiert die Images, auf die sich jede Regel auswirkt.

4. Der Lifecycle-Policy-Evaluator wendet dann die Regeln auf der Grundlage der Regelpriorität an und zeigt an, welche Bilder im Repository als abgelaufen oder archiviert gelten. Eine niedrigere Regelprioritätsnummer bedeutet eine höhere Priorität. Beispielsweise hat eine Regel mit Priorität 1 Vorrang vor einer Regel mit Priorität 2.
5. Überprüfen Sie die Testergebnisse und stellen Sie sicher, dass die Bilder, die als abgelaufen oder archiviert markiert sind, Ihren Vorstellungen entsprechen.
6. Wenden Sie die Testregeln als Lebenszyklusrichtlinie für das Repository an.
7. Sobald die Lebenszyklusrichtlinie erstellt wurde, sollten Sie damit rechnen, dass Bilder innerhalb von 24 Stunden, nachdem sie die Ablaufkriterien erfüllen, abgelaufen sind oder archiviert werden.

Regeln für die Bewertung der Lebenszyklusrichtlinie

Der Lifecycle-Policy-Evaluator ist für das Parsen des Klartext-JSON der Lifecycle-Policy, die Bewertung aller Regeln und die anschließende Anwendung dieser Regeln basierend auf der Regelpriorität auf die Images im Repository zuständig. Im Folgenden wird die Logik des Lifecycle-Policy-Evaluators ausführlicher erläutert. Beispiele finden Sie unter Beispiele für Lebenszyklusrichtlinien in Amazon ECR.

- Wenn Referenzartefakte in einem Repository vorhanden sind, laufen die Amazon ECR-Lebenszyklusrichtlinien automatisch ab oder archivieren diese Artefakte innerhalb von 24 Stunden nach dem Löschen oder Archivieren des betreffenden Bilds.
- Alle Regeln werden gleichzeitig ausgewertet, unabhängig von der Priorität der Regeln. Nachdem alle Regeln ausgewertet wurden, werden sie entsprechend der Regelpriorität angewendet.
- Ein Bild ist abgelaufen oder wird nach genau einer oder gar keiner Regel archiviert.
- Ein Bild, das den Tagging-Anforderungen einer Regel entspricht, kann nicht aufgrund einer Regel mit niedrigerer Priorität abgelaufen oder archiviert werden.
- Regeln können Bilder, die durch Regeln mit höherer Priorität gekennzeichnet sind, niemals markieren, sie können sie aber dennoch so identifizieren, als ob sie nicht abgelaufen oder archiviert wären.
- Der Satz aller Regeln, die eine bestimmte Speicherklasse auswählen, muss einen eindeutigen Satz von Präfixen enthalten.
- Nur eine Regel, die eine bestimmte Speicherklasse auswählt, darf Bilder ohne Tags auswählen.
- Wenn in einer Manifestliste auf ein Bild verwiesen wird, kann es nicht abgelaufen oder archiviert werden, ohne dass die Manifestliste zuerst gelöscht oder archiviert wurde.

- Das Ablaufdatum wird immer nach `pushed_at_time` oder geordnet `transitioned_at_time` und ältere Images laufen immer vor neueren ab. Wenn ein Bild archiviert und dann zu einem beliebigen Zeitpunkt in der Vergangenheit wiederhergestellt wurde, `last_activated_at` wird das Bild anstelle von verwendet`pushed_at_time`.
- Eine Lebenszyklusrichtlinienregel kann entweder `tagPatternList` oder`tagPrefixList` angeben, aber nicht beide. Eine Lebenszyklusrichtlinie kann jedoch mehrere Regeln enthalten, wobei unterschiedliche Regeln sowohl Muster- als auch Präfixlisten verwenden können. Ein Bild ist erfolgreich zugeordnet, wenn alle Tags im `tagPrefixList` Wert `tagPatternList` oder mit einem der Bild-Tags abgeglichen wurden.
- Die Parameter `tagPrefixList` oder `tagPatternList` dürfen nur verwendet werden, wenn der `tagStatus` auf `tagged` lautet.
- Bei Verwendung von `tagPatternList` stimmt ein Image erfolgreich überein, wenn es dem Platzhalterfilter entspricht. Wenn beispielsweise ein Filter von angewendet `prod*` wird, würde er Bild-Tags entsprechen, deren Name `prod` beispielsweise mit `prodprod1`, oder beginnt. `production-team1` Wenn ein Filter von angewendet `*prod*` wird, würde er auch Bild-Tags entsprechen, deren Name beispielsweise oder enthält`prod`. `repo-production` `prod-team`

  > **Important**
  >
  > Es gibt eine Obergrenze von vier Platzhaltern (`*`) pro Zeichenfolge. Zum Beispiel ist `["*test*1*2*3", "test*1*2*3*"]` gültig, `["test*1*2*3*4*5*6"]` aber ungültig.

- Bei Verwendung dieser Option wird ein Bild erfolgreich zugeordnet`tagPrefixList`, wenn alle Platzhalterfilter im `tagPrefixList` Wert mit einem der Tags des Bilds abgeglichen wurden.
- Der `countUnit` Parameter wird nur verwendet, wenn er `sinceImagePushed`sinceImagePulled, oder `countType` `sinceImageTransitioned` ist.
- Mit `countType = imageCountMoreThan` werden Bilder auf der Grundlage von den jüngsten zu den ältesten sortiert, `pushed_at_time` und dann sind alle Bilder, die die angegebene Anzahl überschreiten, abgelaufen oder archiviert.
- Mit sind alle Bilder`countType = sinceImagePushed`, `pushed_at_time` die älter als die angegebene Anzahl von Tagen `countNumber` sind, abgelaufen oder archiviert.
- Mit werden alle Bilder `last_recorded_pulltime` archiviert`countType = sinceImagePulled`, die älter als die angegebene Anzahl von Tagen `countNumber` sind. Wenn ein Bild nie abgerufen wurde, `pushed_at_time` wird das Bild anstelle von verwendet`last_recorded_pulltime`. Wenn ein Bild zu einem beliebigen Zeitpunkt in der

Vergangenheit archiviert und dann wiederhergestellt wurde, aber seit der Wiederherstellung des Images nie abgerufen wurde, `last_activated_at` wird das Bild anstelle von verwendet`last_recorded_pulltime`.

- Mit sind alle archivierten Bilder `countType = sinceImageTransitionedlast_archived_at`, die älter als die angegebene Anzahl von Tagen `countNumber` sind, abgelaufen.
- Das Ablaufdatum wird immer nach älteren Bildern geordnet `pushed_at_time` und läuft immer vor neueren ab.

Erstellen einer Lifecycle-Policy-Vorschau in Amazon ECR

Sie können eine Lifecycle-Policy-Vorschau verwenden, um die Auswirkungen einer Lebenszyklus-Richtlinie auf ein Image-Repository zu sehen, bevor Sie sie anwenden. Es gilt als Best Practice, eine Vorschau zu erstellen, bevor eine Lebenszyklusrichtlinie auf ein Repository angewendet wird.

> **Note**
>
> Wenn Sie die Amazon ECR-Replikation verwenden, um Kopien eines Repositorys in verschiedenen Regionen oder Konten zu erstellen, beachten Sie, dass eine Lebenszyklusrichtlinie nur eine Aktion für Repositorys in der Region ausführen kann, in der sie erstellt wurde. Wenn Sie die Replikation aktiviert haben, sollten Sie daher erwägen, eine Lebenszyklusrichtlinie für jede Region und jedes Konto zu erstellen, in das Sie Ihre Repositorys replizieren.

So erstellen Sie eine Lebenszyklus-Richtlinienvorschau (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositorys.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der das Repository enthalten ist, für das Lebenszyklus-Richtlinienvorschau ausgeführt werden soll.
3. Wählen Sie im Navigationsbereich unter Private Registrierung die Option Repositorys aus.
4. Wählen Sie auf der Seite Private Repositorys ein Repository aus und verwenden Sie dann das Drop-down-Menü Aktionen, um die Option Lebenszyklusrichtlinien auszuwählen.
5. Wählen Sie auf der Seite mit den Regeln für die Lebenszyklusrichtlinien die Optionen Testregeln bearbeiten, Regel erstellen aus.
6. Geben Sie die folgenden Details für jede Lebenszyklusrichtlinienregel an.

a. Geben Sie für Regelpriorität eine Nummer für die Regelpriorität ein. Die Regelpriorität bestimmt, in welcher Reihenfolge die Lebenszyklusrichtlinienregeln angewendet werden. Eine niedrigere Zahl bedeutet eine höhere Priorität. Beispielsweise hat eine Regel mit Priorität 1 Vorrang vor einer Regel mit Priorität 2.

b. Geben Sie für Regelbeschreibung eine Beschreibung für die Lebenszyklusrichtlinienregel ein.

c. Wählen Sie für Image-Status die Optionen Markiert (Platzhalterabgleich), Markiert (Präfixabgleich), Nicht markiert oder Beliebig aus.

> **Important**
>
> Wenn Sie mehrere Tags angeben, werden nur die Images mit allen angegebenen Tags ausgewählt.

d. Wenn Sie Markiert (Platzhalterabgleich) für Image-Status ausgewählt haben, können Sie für Tags für Platzhalterabgleich angeben eine Liste von Image-Tags mit einem Platzhalter (*) angeben, für die Sie gemäß Ihrer Lebenszyklusrichtlinie Aktionen durchführen möchten. Wenn Ihre Images beispielsweise als `prod`, `prod1`, `prod2` usw. markiert sind, würden Sie `prod*` angeben, um für alle Aktionen durchzuführen. Wenn Sie mehrere Tags angeben, werden nur die Images mit allen angegebenen Tags ausgewählt.

> **Important**
>
> Es gibt eine Obergrenze von vier Platzhaltern (*) pro Zeichenfolge. Zum Beispiel ist `["*test*1*2*3", "test*1*2*3*"]` gültig, `["test*1*2*3*4*5*6"]` aber ungültig.

e. Wenn Sie Markiert (Präfixabgleich) für Image-Status ausgewählt haben, können Sie für Tags für Präfixabgleich angeben eine Liste von Image-Tags angeben, für die Sie gemäß Ihrer Lebenszyklusrichtlinie Aktionen durchführen möchten.

f. Wählen Sie für Abgleichskriterien die Option Tage seit der Erstellung des Images, Tage seit der letzten aufgezeichneten Abrufzeit, Tage seit der Archivierung des Bilds oder Anzahl der Bilder aus und geben Sie dann einen Wert an.

g. Wählen Sie für Regelaktion entweder Ablaufen oder Archivieren aus.

h. Wählen Sie Speichern.

7. Erstellen Sie weitere Test-Lebenszyklusrichtlinienregeln, indem Sie die Schritte 5 bis 7 wiederholen.
8. Um die Lebenszyklus-Richtlinienvorschau auszuführen, wählen Sie Save and run test (Speichern und Test ausführen).
9. Überprüfen Sie unter Imageübereinstimmungen für Testlebenszyklusregeln (Image-Übereinstimmungen für Lebenszyklus-Testregeln) die Wirkung Ihrer Lebenszyklus-Richtlinienvorschau.
10. Wenn Sie mit den Vorschauergebnisse zufrieden sind, wählen Sie Anwendung als Lebenszyklusrichtlinie, um eine Lebenszyklusrichtlinie mit den angegebenen Regeln zu erstellen. Sie sollten damit rechnen, dass die betroffenen Bilder nach der Anwendung einer Lebenszyklusrichtlinie innerhalb von 24 Stunden abgelaufen sind oder archiviert werden.
11. Wenn Sie mit den Vorschauergebnissen nicht zufrieden sind, können Sie eine oder mehrere Testlebenszyklusregeln löschen und eine oder mehrere Regeln erstellen, um sie zu ersetzen und dann den Test zu wiederholen.

Erstellen einer Lebenszyklusrichtlinie für ein Repository in Amazon ECR

Verwenden Sie eine Lebenszyklus-Richtlinie, um eine Reihe von Regeln zu erstellen, die ablaufen, oder archivieren Sie ungenutzte Repository-Images. Nach der Erstellung einer Lebenszyklus-Richtlinie sind die betroffenen Images innerhalb von 24 Stunden abgelaufen oder archiviert.

> **Note**
>
> Wenn Sie die Amazon ECR-Replikation verwenden, um Kopien eines Repositorys in verschiedenen Regionen oder Konten zu erstellen, beachten Sie, dass eine Lebenszyklusrichtlinie nur eine Aktion für Repositorys in der Region ausführen kann, in der sie erstellt wurde. Wenn Sie die Replikation aktiviert haben, sollten Sie daher erwägen, eine Lebenszyklusrichtlinie für jede Region und jedes Konto zu erstellen, in das Sie Ihre Repositorys replizieren.

Voraussetzung

Bewährtes Verfahren: Erstellen Sie eine Vorschau der Lifecycle-Richtlinien, um zu überprüfen, ob die Bilder abgelaufen sind oder ob die nach Ihren Lebenszyklus-Richtlinien archivierten Bilder Ihren

Vorstellungen entsprechen. Detaillierte Anweisungen finden Sie unter Erstellen einer Lifecycle-Policy-Vorschau in Amazon ECR.

So erstellen Sie eine Lebenszyklusrichtlinie (AWS-Managementkonsole)

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/Repositorys.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der das Repository enthalten ist, für das eine Lebenszyklusrichtlinien erstellt werden soll.
3. Wählen Sie im Navigationsbereich unter Private Registrierung die Option Repositorys aus.
4. Wählen Sie auf der Seite Private Repositorys ein Repository aus und verwenden Sie dann das Drop-down-Menü Aktionen, um die Option Lebenszyklusrichtlinien auszuwählen.
5. Wählen Sie auf der Seite mit den Lebenszyklusrichtlinienregeln die Option Regel erstellen aus.
6. Geben Sie die folgenden Details für Ihre Lebenszyklusrichtlinienregel ein.

    a. Geben Sie für Regelpriorität eine Nummer für die Regelpriorität ein. Die Regelpriorität bestimmt, in welcher Reihenfolge die Lebenszyklusrichtlinienregeln angewendet werden. Eine niedrigere Regelprioritätsnummer bedeutet eine höhere Priorität. Beispielsweise hat eine Regel mit Priorität 1 Vorrang vor einer Regel mit Priorität 2.

    b. Geben Sie für Regelbeschreibung eine Beschreibung für die Lebenszyklusrichtlinienregel ein.

    c. Wählen Sie für Image-Status die Optionen Markiert (Platzhalterabgleich), Markiert (Präfixabgleich), Nicht markiert oder Beliebig aus.

    > **Important**
    >
    > Wenn Sie mehrere Tags angeben, werden nur die Images mit allen angegebenen Tags ausgewählt.

    d. Wenn Sie Markiert (Platzhalterabgleich) für Image-Status ausgewählt haben, können Sie für Tags für Platzhalterabgleich angeben eine Liste von Image-Tags mit einem Platzhalter (*) angeben, für die Sie gemäß Ihrer Lebenszyklusrichtlinie Aktionen durchführen möchten. Wenn Ihre Images beispielsweise als `prod`, `prod1`, `prod2` usw. markiert sind, würden Sie `prod*` angeben, um für alle Aktionen durchzuführen. Wenn Sie mehrere Tags angeben, werden nur die Images mit allen angegebenen Tags ausgewählt.

> **Important**
>
> Es gibt eine Obergrenze von vier Platzhaltern (*) pro Zeichenfolge. Zum Beispiel ist `["*test*1*2*3", "test*1*2*3*"]` gültig, `["test*1*2*3*4*5*6"]` aber ungültig.

e. Wenn Sie Markiert (Präfixabgleich) für Image-Status ausgewählt haben, können Sie für Tags für Präfixabgleich angeben eine Liste von Image-Tags angeben, für die Sie gemäß Ihrer Lebenszyklusrichtlinie Aktionen durchführen möchten.

f. Wählen Sie für Abgleichskriterien die Option Tage seit der Erstellung des Images, Tage seit der letzten aufgezeichneten Abrufzeit, Tage seit der Archivierung des Bilds oder Anzahl der Bilder aus und geben Sie dann einen Wert an.

g. Wählen Sie für Regelaktion entweder Ablaufen oder Archivieren aus.

h. Wählen Sie Speichern.

7. Erstellen Sie weitere Lebenszyklus-Richtlinienregeln, indem Sie die Schritte 5 bis 7 wiederholen.

So erstellen Sie eine Lebenszyklusrichtlinie (AWS CLI)

1. Ermitteln Sie den Namen des Repositorys, für das die Lebenszyklusrichtlinie erstellt werden soll.

```
aws ecr describe-repositories
```

2. Erstellen Sie eine lokale Datei mit dem Namen `policy.json` mit dem Inhalt der Lebenszyklusrichtlinie. Beispiele für Lebenszyklus-Richtlinien finden Sie unter Beispiele für Lebenszyklusrichtlinien in Amazon ECR.

3. Erstellen Sie eine Lebenszyklusrichtlinie, indem Sie den Namen des Repositorys angeben und auf die von Ihnen erstellte JSON-Datei der Lebenszyklusrichtlinie verweisen.

```
aws ecr put-lifecycle-policy \
      --repository-name repository-name \
      --lifecycle-policy-text file://policy.json
```

Beispiele für Lebenszyklusrichtlinien in Amazon ECR

Im Folgenden finden Sie Beispiele für Lebenszyklusrichtlinien, die die Syntax zeigen.

Weitere Informationen zu Richtlinieneigenschaften finden Sie unterEigenschaften der Lebenszyklusrichtlinie in Amazon ECR. Anweisungen zum Erstellen einer Lebenszyklusrichtlinie mithilfe von finden Sie unterSo erstellen Sie eine Lebenszyklusrichtlinie (AWS CLI). AWS CLI

Vorlage für Lebenszykluspolitik

Der Inhalt Ihrer Lebenszyklus-Richtlinie wird bewertet, bevor sie einem Repository zugeordnet wird. Nachfolgend sehen Sie die JSON-Syntaxvorlage für die Lebenszyklus-Richtlinie.

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

Filterung nach dem Alter der Images

Das folgende Beispiel zeigt die Lebenszyklusrichtliniensyntax für eine Richtlinie, die Images mit einem Tag ablaufen lässt, das mit `prod` beginnt. Dazu wird eine `tagPatternList` für `prod*` und die Sucheinschränkung „älter als 14 Tage“ verwendet.

```
{
    "rules": [
        {
            "rulePriority": 1,
```

```
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtern nach der Anzahl an Images

Das folgende Beispiel zeigt die Lebenszyklusrichtliniensyntax für eine Richtlinie, die nur ein Image ohne Tags beibehält und alle anderen ablaufen lässt:

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtern nach mehreren Regeln

Die folgenden Beispiele verwenden mehrere Regeln in einer Lebenszyklus-Richtlinie. Dafür werden ein Beispiel-Repository und eine Beispiel-Lebenszyklusrichtlinie gezeigt, ebenso wie eine Erklärung des Ergebnisses.

Beispiel A

Repository-Inhalt:

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: vor 10 Tagen
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: vor 9 Tagen
- Image C, Taglist: ["beta-3"], Pushed: vor 8 Tagen

Text der Lebenszyklus-Richtlinie:

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

Die Logik dieser Lebenszyklus-Richtlinie wäre:

- Regel 1 identifiziert Images, die mit dem Präfix `prod` markiert sind. Sie sollte die Images beginnend mit dem ältesten markieren, bis es ein oder weniger verbleibende Images gibt, für die eine Übereinstimmung vorliegt. Sie markiert Image A für den Ablauf.
- Regel 2 identifiziert Images, die mit dem Präfix `beta` markiert sind. Sie sollte die Images beginnend mit dem ältesten markieren, bis es ein oder weniger verbleibende Images gibt, für die eine Übereinstimmung vorliegt. Sie markiert Image A und Image B für den Ablauf. Image A wurde jedoch bereits von Regel 1 verarbeitet, und wenn Image B abgelaufen wäre, würde dies Regel 1 verletzen, deshalb wird es übersprungen.
- Ergebnis: Image A ist abgelaufen.

Beispiel B

Dies ist dasselbe Repository wie im vorigen Beispiel, aber die Prioritätsreihenfolge der Regel wird geändert, um das Ergebnis zu verdeutlichen.

Repository-Inhalt:

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: vor 10 Tagen
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: vor 9 Tagen
- Image C, Taglist: ["beta-3"], Pushed: vor 8 Tagen

Text der Lebenszyklus-Richtlinie:

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

Die Logik dieser Lebenszyklus-Richtlinie wäre:

- Regel 1 identifiziert Images, die mit dem Präfix `beta` markiert sind. Sie sollte die Images beginnend mit dem ältesten markieren, bis es ein oder weniger verbleibende Images gibt, für die eine Übereinstimmung vorliegt. Sie verarbeitet alle drei Images und würde Image A und Image B für den Ablauf markieren.
- Regel 2 identifiziert Images, die mit dem Präfix `prod` markiert sind. Sie sollte die Images beginnend mit dem ältesten markieren, bis es ein oder weniger verbleibende Images gibt, für die eine Übereinstimmung vorliegt. Sie würde keine Images verarbeiten, weil alle verfügbaren Images bereits von Regel 1 verarbeitet wurden, es würden also keine weiteren Images markiert.
- Ergebnis: Image A und B laufen ab.

Filtern nach mehreren Tags in einer einzigen Regel

Die folgenden Beispiele zeigen die Lebenszyklusrichtliniensyntax für mehrere Tag-Muster innerhalb einer einzigen Regel. Dafür werden ein Beispiel-Repository und eine Beispiel-Lebenszyklusrichtlinie gezeigt, ebenso wie eine Erklärung des Ergebnisses.

Beispiel A

Wenn mehrere Tag-Muster innerhalb einer einzigen Regel angegeben sind, müssen die Images mit allen aufgelisteten Tag-Mustern übereinstimmen.

Repository-Inhalt:

- Image A, Taglist: ["alpha-1"], Pushed: vor 12 Tagen
- Image B, Taglist: ["beta-1"], Pushed: vor 11 Tagen
- Image C, Taglist: ["alpha-2", "beta-2"], Pushed: vor 10 Tagen
- Image D, Taglist: ["alpha-3"], Pushed: vor 4 Tagen
- Image E, Taglist: ["beta-3"], Pushed: vor 3 Tagen
- Image F, Taglist: ["alpha-4", "beta-4"], Pushed: vor 2 Tagen

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Die Logik dieser Lebenszyklus-Richtlinie wäre:

- Regel 1 identifiziert Images, die mit dem Präfix `alpha` und `beta` markiert sind. Sie verarbeitet die Images C und F. Sie sollte Images markieren, die älter als fünf Tage sind, das wäre Image C.
- Ergebnis: Image C läuft ab.

Beispiel B

Das folgende Beispiel veranschaulicht, dass Tags nicht exklusiv sind.

Repository-Inhalt:

- Image A, Taglist: ["alpha-1", "beta-1", "gamma-1"], Pushed: vor 10 Tagen
- Image B, Taglist: ["alpha-2", "beta-2"], Pushed: vor 9 Tagen
- Image C, Taglist: ["alpha-3", "beta-3", "gamma-2"], Pushed: vor 8 Tagen

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Die Logik dieser Lebenszyklus-Richtlinie wäre:

- Regel 1 identifiziert Images, die mit dem Präfix `alpha` und `beta` markiert sind. Sie verarbeitet alle Images. Sie sollte die Images beginnend mit dem ältesten markieren, bis es ein oder weniger verbleibende Images gibt, für die eine Übereinstimmung vorliegt. Sie markiert Image A und B für den Ablauf.
- Ergebnis: Image A und B laufen ab.

Filterung auf alle Images

Die folgenden Beispiele für Lebenszyklusrichtlinien geben alle Images mit unterschiedlichen Filtern an. Dafür werden ein Beispiel-Repository und eine Beispiel-Lebenszyklusrichtlinie gezeigt, ebenso wie eine Erklärung des Ergebnisses.

Beispiel A

Nachfolgend sehen Sie die Syntax der Lebenszyklus-Richtlinie für eine Richtlinie, die für alle Regeln gilt, aber nur ein Image beibehält und alle anderen ablaufen lässt.

Repository-Inhalt:

- Image A, Taglist: ["alpha-1"], vor 4 Tagen
- Image B, Taglist: ["beta-1"], vor 3 Tagen
- Image C, Taglist: [], Pushed: vor 2 Tagen
- Image D, Taglist: ["alpha-2"], Pushed: vor 1 Tag

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Die Logik dieser Lebenszyklus-Richtlinie wäre:

- Regel 1 identifiziert alle Images. Sie sieht die Images A, B, C und D. Sie soll alle Images außer dem neuesten auslaufen lassen. Sie kennzeichnet die Images A, B und C für den Ablauf.
- Ergebnis: Image A, B und C laufen ab.

Beispiel B

Das folgende Beispiel zeigt eine Lebenszyklus-Richtlinie, die alle Regeltypen in einer einzigen Regel kombiniert.

Repository-Inhalt:

- Image A, Taglist: ["alpha-", "beta-1", "-1"], Pushed: vor 4 Tagen
- Image B, Taglist: [], Pushed: vor 3 Tagen
- Image C, Taglist: ["alpha-2"], Pushed: vor 2 Tagen
- Image D, Taglist: ["git hash"], Pushed: vor 1 Tag
- Image E, Taglist: [], Pushed: vor 1 Tag

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

Die Logik dieser Lebenszyklus-Richtlinie wäre:

- Regel 1 identifiziert Images, die mit dem Präfix `alpha` markiert sind. Sie identifiziert die Images A und C. Sie sollte das neueste Image beibehalten und die restlichen für den Ablauf markieren. Sie markiert Image A für den Ablauf.
- Regel 2 identifiziert Images ohne Tags. Sie identifiziert die Images B und E. Sie sollte alle Images, die älter als einen Tag sind, für den Ablauf markieren. Sie markiert Image B für den Ablauf.
- Regel 3 identifiziert alle Images. Sie identifiziert die Images A, B, C, D und E. Sie sollte das neueste Image beibehalten und die restlichen für den Ablauf markieren. Sie kann jedoch die Images A, B, C oder E nicht markieren, weil sie von Regeln mit höherer Priorität identifiziert wurden. Sie markiert Image D für den Ablauf.
- Ergebnis: Image A, B und D laufen ab.

Archivbeispiele

Die folgenden Beispiele zeigen Lebenszyklusrichtlinien, mit denen Bilder archiviert werden, anstatt sie zu löschen.

Archivierung von Bildern, die älter als eine bestimmte Anzahl von Tagen sind

Das folgende Beispiel zeigt eine Lebenszyklusrichtlinie, mit der Bilder archiviert werden, deren Tags mit beginnen`prod`, die älter als 30 Tage sind:

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

Archivierung von Bildern, die nicht innerhalb einer bestimmten Anzahl von Tagen abgerufen wurden

Das folgende Beispiel zeigt eine Lebenszyklusrichtlinie, mit der Bilder archiviert werden, die seit 90 Tagen nicht abgerufen wurden:

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

Kombination von Archivierungs- und Löschregeln

Das folgende Beispiel zeigt eine Lebenszyklusrichtlinie, die Bilder archiviert, die älter als 30 Tage sind, und anschließend Bilder, die länger als 365 Tage archiviert wurden, dauerhaft löscht:

> **Note**
>
> Archivierte Bilder haben eine Mindestspeicherdauer von 90 Tagen. Sie können keine Lebenszyklusrichtlinien konfigurieren, mit denen Bilder gelöscht werden, die sich seit weniger als 90 Tagen im Archiv befinden. Wenn Sie Bilder löschen müssen, die seit weniger als 90 Tagen archiviert wurden, müssen Sie die batch-delete-image API verwenden. Die Mindestspeicherdauer von 90 Tagen wird Ihnen jedoch in Rechnung gestellt.

```
{
    "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": "Delete images archived for more than 365 days",
            "selection": {
                "tagStatus": "any",
                "storageClass": "archive",
                "countType": "sinceImageTransitioned",
                "countUnit": "days",
                "countNumber": 365
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Eigenschaften der Lebenszyklusrichtlinie in Amazon ECR

Lebenszyklusrichtlinien haben die folgenden Eigenschaften.

Beispiele für Lebenszyklusrichtlinien finden Sie unterBeispiele für Lebenszyklusrichtlinien in Amazon ECR. Anweisungen zum Erstellen einer Lebenszyklusrichtlinie mithilfe von finden Sie unterSo erstellen Sie eine Lebenszyklusrichtlinie (AWS CLI). AWS CLI

Priorität der Regel

`rulePriority`

Typ: Ganzzahl

Erforderlich: Ja

Legt die Reihenfolge fest, in der die Regeln angewendet werden, von unten nach oben. Eine Lebenszyklus-Richtlinienregel mit der Priorität von 1 wird zuerst angewendet, eine Regel mit der Priorität von 2 folgt und so weiter. Wenn Sie einer Lebenszyklusrichtlinie Regeln hinzufügen, müssen Sie ihr einen eindeutigen Wert für `rulePriority` zuweisen. Werte müssen für alle Regeln in einer Richtlinie nicht sequentiell sein. Eine Regel mit dem `tagStatus`-Wert `any` muss den höchsten Wert für `rulePriority` haben und als letzte ausgewertet werden.

Description

`description`

Typ: Zeichenfolge

Erforderlich: nein

(Optional) Beschreibt den Zweck einer Regel innerhalb einer Lebenszyklus-Richtlinie.

Tag-Status

`tagStatus`

Typ: Zeichenkette

Erforderlich: Ja

Legt fest, ob die von Ihnen hinzugefügte Lebenszyklusrichtlinienregel ein Tag für ein Image angibt. Zulässige Optionen sind `tagged`, `untagged` oder `any`. Wenn Sie `any` angeben, wird die Regel auf alle Images angewandt. Wenn Sie angeben`tagged`, müssen Sie auch einen `tagPrefixList` Wert oder einen `tagPatternList` Wert angeben. Wenn Sie angeben`untagged`, müssen Sie sowohl als `tagPrefixList` auch weglassen.
`tagPatternList`

Tag-Muster-Liste

`tagPatternList`

Typ: list[string]

Erforderlich: ja, wenn `tagStatus` auf „tagged“ (markiert) gesetzt und `tagPrefixList` nicht angegeben ist

Bei der Erstellung einer Lebenszyklusrichtlinie für Images mit Tags empfiehlt es sich, eine `tagPatternList` zu verwenden, um anzugeben, welche Tags ablaufen sollen. Sie geben eine Liste mit durch Kommas voneinander getrennten Image-Tag-Mustern an, die Platzhalter (*) enthalten können, die Sie in Ihren Lebenszyklusrichtlinien-Aktionen ausführen wollen. Wenn Ihre Images beispielsweise als `prod`, `prod1`, `prod2` usw. markiert sind, würden Sie die Tag-Musterliste `prod*` verwenden, um sie alle anzugeben. Wenn Sie mehrere Tags angeben, werden nur die Images mit allen angegebenen Tags ausgewählt.

> **Important**
>
> Es gibt eine Obergrenze von vier Platzhaltern (*) pro Zeichenfolge. Zum Beispiel ist `["*test*1*2*3", "test*1*2*3*"]` gültig, `["test*1*2*3*4*5*6"]` aber ungültig.

Tag-Präfix-Liste

`tagPrefixList`

Typ: list[string]

Erforderlich: ja, wenn `tagStatus` auf „tagged“ (markiert) gesetzt und `tagPatternList` nicht angegeben ist

Wird nur verwendet, wenn Sie `"tagStatus": "tagged"` angegeben haben, aber keine `tagPatternList`. Sie müssen eine Liste mit durch Kommas voneinander getrennten Image-Tag-Präfixen angeben, die Sie in Ihrer Lebenszyklusrichtlinienaktionen ausführen wollen. Wenn Ihre Images beispielsweise als `prod`, `prod1`, `prod2` usw. markiert sind, würden Sie das Tag-Präfix `prod` verwenden, um sie alle anzugeben. Wenn Sie mehrere Tags angeben, werden nur die Images mit allen angegebenen Tags ausgewählt.

Speicherklasse

`storageClass`

Type: Zeichenkette

Erforderlich: ja, wenn `countType` `sinceImageTransitioned`

Die Regel wählt nur Bilder dieser Speicherklasse aus. Wenn Sie einen Wert `countType` von `imageCountMoreThan`sinceImagePushed, oder verwendensinceImagePulled, ist der einzige unterstützte Wertstandard. Bei Verwendung des Zähltyps von `sinceImageTransitioned` ist dies erforderlich, und der einzige unterstützte Wert istarchive. Wenn Sie diesen Wert weglassen, `standard` wird der Wert von verwendet.

Art der Zählung

`countType`

Typ: Zeichenkette

Erforderlich: Ja

Geben Sie einen Zählertyp an, der auf die Images angewendet wird.

Wenn `countType` auf `imageCountMoreThan` gesetzt ist, geben Sie auch `countNumber` an, um eine Regel zu erstellen, die eine Obergrenze für die Anzahl der Images festlegt, die in Ihrem Repository vorhanden sein dürfen. Wenn aufsinceImagePushed, `sinceImagePulled` oder gesetzt `countType` istsinceImageTransitioned, geben `countUnit` Sie auch ein Zeitlimit für die Bilder an, die in Ihrem Repository existieren. `countNumber`

Zähleinheit

`countUnit`

Type: Zeichenkette

Erforderlich: ja, nur wenn auf `sinceImagePushedsinceImagePulled`, oder gesetzt `countType` ist `sinceImageTransitioned`

Geben Sie eine Zähleinheit von `days` an, um diese als Zeiteinheit festzulegen, zusätzlich zu `countNumber`, der Anzahl der Tage.

Dies sollte nur angegeben werden, wenn `sinceImagePushedsinceImagePulled`, oder `countType` ist`sinceImageTransitioned`; ein Fehler tritt auf, wenn Sie eine Zähleinheit angeben, obwohl es sich um einen anderen Wert `countType` handelt.

Anzahl

`countNumber`

Typ: Ganzzahl

Erforderlich: Ja

Geben Sie eine Anzahl an. Akzeptable Werte sind positive Ganzzahlen (`0` ist kein akzeptierter Wert).

Wenn der verwendete `countType` `imageCountMoreThan` ist, ist der Wert die maximale Anzahl der Images, die Sie in Ihrem Repository beibehalten wollen. Wenn der verwendete `countType` `sinceImagePushed` ist, ist der Wert die maximale Altersgrenze für Ihre Images. Wenn der Wert `countType` verwendet wird`sinceImagePulled`, entspricht der Wert der maximalen Anzahl von Tagen, seit das Bild zuletzt abgerufen wurde. Wenn der Wert `countType` verwendet wird`sinceImageTransitioned`, entspricht der Wert der maximalen Anzahl von Tagen seit der Archivierung des Images.

Action

`type`

Typ: Zeichenfolge

Erforderlich: Ja

Geben Sie einen Aktionstyp an. Die unterstützten Werte sind `expire` (zum Löschen von Bildern) und `transition` (zum Verschieben von Bildern in den Archivspeicher).

`targetStorageClass`

Type: Zeichenkette

Erforderlich: ja, `type` wenn `transition`

Die Speicherklasse, auf die die Lebenszyklusrichtlinie das Image umstellen soll. `archive`ist der einzige unterstützte Wert.

Ausschlüsse für Pull-Time-Updates

Amazon ECR aktualisiert den `LastRecordedPullTime` Zeitstempel bei jedem Abruf, mit Ausnahme von Abrufen durch Inspector. AWS Mit Ausschlüssen für Pull-Time-Updates können Sie die IAM-Rolle angeben ARNs , die die Abrufzeiten von Bildern nicht aktualisieren soll, wenn Bilder abgerufen werden, z. B. beim Abrufen von Bildern durch Scanner von Drittanbietern (wie Crowdstrike, Snyk und Trivy). Dies ist nützlich für Images, die zu CI/CD Testzwecken oder für Zwecke verwendet werden, bei denen Sie nicht möchten, dass die Pull-Time die Entscheidungen über die Lebenszyklus-Richtlinien beeinflusst.

Wenn eine Rolle in der Ausschlussliste ein Bild abruft, bleibt die Abrufzeit unverändert. Bei jeder anderen Rolle wird die Pullzeit weiterhin aktualisiert (aktuelles Verhalten). Sie können bis zu 100 Ausschlüsse pro Konto konfigurieren.

Ausschlüsse für Pull-Time-Updates verwalten

Um Ausschlüsse für Pull-Time-Updates zu verwalten, benötigen Sie die folgenden IAM-Berechtigungen:

- `ecr:CreatePullTimeUpdateExclusion`— Erteilt die Erlaubnis, einen Rollen-ARN zur Ausschlussliste hinzuzufügen.
- `ecr:DeletePullTimeUpdateExclusion`— Erteilt die Erlaubnis, einen Rollen-ARN aus der Ausschlussliste zu entfernen.
- `ecr:ListPullTimeUpdateExclusions`— Erteilt die Erlaubnis, alle Rollen ARNs in der Ausschlussliste aufzulisten.

> **Note**
>
> Sie benötigen keine `iam:PassRole` Genehmigung. Amazon ECR übernimmt nicht die Rolle, eine Aktion auszuführen. Es verwendet lediglich die Ausschlusskonfiguration, ARNs um zu bestimmen, ob die Abrufzeit des Images aktualisiert werden soll.

Sie können Ausschlüsse für Pull-Time-Updates mit der Amazon ECR-Konsole oder der CLI verwalten. AWS

AWS-Managementkonsole

Um Ausnahmen für Pull-Time-Updates zu verwalten ()AWS-Managementkonsole

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/ private-registry/repositories
2. Wählen Sie in der Navigationsleiste die Region aus.
3. Wählen Sie im Navigationsbereich Private Registrierung, Funktionen und Einstellungen und anschließend Ausnahmen für Pull-Time-Updates aus.
4. Um einen Ausschluss hinzuzufügen, wählen Sie Ausschluss hinzufügen, geben Sie den Rollen-ARN ein und wählen Sie dann Hinzufügen.
5. Um einen Ausschluss zu entfernen, wählen Sie den Rollen-ARN aus der Liste aus und klicken Sie auf Löschen.
6. Um alle Ausnahmen anzuzeigen, werden in der Liste alle konfigurierten Rollen ARNs angezeigt.

AWS CLI

Um einen Ausschluss für Pull-Time-Updates zu erstellen

- Verwenden Sie den create-pull-time-update-exclusion Befehl, um der Ausschlussliste einen Rollen-ARN hinzuzufügen:

```
aws ecr create-pull-time-update-exclusion \
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

Der Befehl gibt den Rollen-ARN und den Erstellungszeitstempel zurück:

```
{
   "roleArn": "arn:aws:iam::123456789012:role/scanner-role",
   "createdAt": 1745531331.0
}
```

Um einen Ausschluss für Pull-Time-Updates zu löschen

- Verwenden Sie den delete-pull-time-update-exclusion Befehl, um einen Rollen-ARN aus der Ausschlussliste zu entfernen:

```
aws ecr delete-pull-time-update-exclusion \
    --role-arn arn:aws:iam::123456789012:role/scanner-role
```

Der Befehl gibt den Rollen-ARN zurück, der gelöscht wurde:

```
{
  "roleArn": "arn:aws:iam::123456789012:role/scanner-role"
}
```

Um Ausschlüsse für Pull-Time-Updates aufzulisten

1. Verwenden Sie den list-pull-time-update-exclusions Befehl, um alle Rollen ARNs in der Ausschlussliste aufzulisten:

   ```
   aws ecr list-pull-time-update-exclusions
   ```

   Wenn keine Ausnahmen konfiguriert sind, gibt der Befehl eine leere Liste zurück:

   ```
   {
      "pullTimeUpdateExclusions": []
   }
   ```

   Wenn Ausschlüsse konfiguriert sind, gibt der Befehl die Rollenliste zurück: ARNs

   ```
   {
      "pullTimeUpdateExclusions": [
           "arn:aws:iam::123456789012:role/security-role"
       ]
   }
   ```

2. Verwenden Sie die Parameter `--max-results` und`--next-token`, um die Ergebnisse zu paginieren:

   ```
   aws ecr list-pull-time-update-exclusions \
       --max-results 4
   ```

Der Befehl gibt bis zur angegebenen Anzahl von Ergebnissen zurück und gibt an, `nextToken` ob mehr Ergebnisse verfügbar sind:

```
{
   "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..."
}
```

Um die nächste Ergebnisseite abzurufen, verwenden Sie die `nextToken` aus der vorherigen Antwort:

```
aws ecr list-pull-time-update-exclusions \
    --max-results 4 \
    --next-token ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79...
```

Überlegungen zu Ausnahmen von Pull-Time-Updates

Beachten Sie Folgendes, wenn Sie Ausnahmen für Pull-Time-Updates verwenden:

- Die Standardseitengröße für Ausschlüsse von Auflistungen ist 100. Sie können die Seitennummerierung mit `maxResults` und `nextToken` -Parametern verwenden.
- Es werden nur gültige IAM-Rollen ARNs im richtigen ARN-Format akzeptiert.
- Wenn Sie versuchen, eine Ausnahme zu erstellen, die bereits existiert, erhalten Sie eine `ExclusionAlreadyExistsException` Fehlermeldung. Wenn Sie versuchen, eine Ausnahme zu löschen, die nicht existiert, erhalten Sie eine `ExclusionNotFoundException` Fehlermeldung.

Sicherheit in Amazon Elastic Container Registry

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS -Compliance-Programme regelmäßig. Informationen zu den Compliance-Programmen, die für Amazon ECR gelten, finden Sie unter AWS Services im Geltungsbereich nach Compliance-Programm.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von Amazon ECR anwenden können. Die folgenden Themen zeigen Ihnen, wie Sie Amazon ECR konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Ihnen helfen, Ihre Amazon ECR-Ressourcen zu überwachen und zu sichern.

Topics

Identity and Access Management für Amazon Elastic Container Registry

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert (mit Berechtigungen ausgestattet) werden kann, um Amazon ECR-Ressourcen zu nutzen. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung bei Amazon Elastic Container Registry - Identität und Zugriff).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie Amazon Elastic Container Registry mit IAM funktioniert).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Verwenden Sie möglichst temporäre Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer für den Zugriff AWS mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer-

zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für Verbundbenutzerzugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, dienstübergreifenden Zugriff und Anwendungen, die auf Amazon ausgeführt werden. EC2 Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an Identitäten oder Ressourcen anhängen. AWS Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie Amazon Elastic Container Registry mit IAM funktioniert

Bevor Sie IAM verwenden, um den Zugriff auf Amazon ECR zu verwalten, sollten Sie verstehen, welche IAM-Features für die Verwendung mit Amazon ECR verfügbar sind. Einen allgemeinen Überblick darüber, wie Amazon ECR und andere AWS Services mit IAM zusammenarbeiten, finden Sie unter AWS Services That Work with IAM im IAM-Benutzerhandbuch.

Themen

Identitätsbasierte Amazon-ECR-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Amazon ECR unterstützt bestimmte Aktionen, Ressourcen und Zustandsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Richtlinienaktionen in Amazon ECR verwenden das folgende Präfix vor der Aktion: `ecr:`. Um beispielsweise jemandem die Erlaubnis zu erteilen, ein Amazon ECR-Repository mit der Amazon ECR-`CreateRepository`API-Operation zu erstellen, nehmen Sie die `ecr:CreateRepository`-Aktion in seine Richtlinie auf. Richtlinienanweisungen müssen entweder ein – `Action`oder ein `NotAction`-Element enthalten. Amazon ECR definiert einen eigenen Satz von Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service durchführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "ecr:Describe*"
```

Eine Liste der Amazon-ECR-Aktionen finden Sie unter Aktionen, Ressourcen und Zustandsschlüssel für Amazon Elastic Container Registry im IAM User Guide.

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Amazon ECR-Repository-Ressource hat den folgenden ARN:

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon Resource Names (ARNs) und AWS Service Namespaces.

Um beispielsweise das `my-repo`-Repository in der `us-east-1`- Region in Ihrer Anweisung anzugeben, verwenden Sie den folgenden ARN:

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

Um alle Repositorys anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, trennen Sie sie durch Kommas. ARNs

```
"Resource": [
      "resource1",
      "resource2"
```

Eine Liste der Amazon ECR-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von Amazon Elastic Container Registry definierte Ressourcen im IAM-Benutzerhandbuch. Um zu erfahren, mit welchen Aktionen Sie den ARN einer jeden Ressource angeben können, siehe Von Amazon Elastic Container Registry definierte Aktionen.

Bedingungsschlüssel

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Amazon ECR definiert seinen eigenen Satz von Konditionsschlüsseln und unterstützt auch die Verwendung einiger globaler Konditionsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Die meisten Amazon ECR-Aktionen unterstützen die`aws:ResourceTag`und `ecr:ResourceTag`Bedingungsschlüssel. Weitere Informationen finden Sie unter Verwenden Tag-basierter Zugriffskontrolle.

Eine Liste der Amazon-ECR-Bedingungsschlüssel finden Sie unter Condition Keys Defined by Amazon Elastic Container Registry im IAM-Benutzerhandbuch. Um zu erfahren, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, siehe Actions Defined by Amazon Elastic Container Registry.

Beispiele

Beispiele für identitätsbasierte Amazon ECR-Richtlinien finden Sie unterBeispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien .

Ressourcenbasierte Amazon-ECR-Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die angeben, welche Aktionen ein bestimmter Auftraggeber auf einer Amazon ECR-Ressource durchführen kann und unter welchen Bedingungen. Amazon ECR unterstützt ressourcenbasierte Berechtigungsrichtlinien für Amazon ECR-Repositories. Ressourcenbasierte Richtlinien ermöglichen die Erteilung von Nutzungsberechtigungen für andere -Konten pro Ressource. Sie können auch eine ressourcenbasierte Richtlinie verwenden, AWS um einem Dienst den Zugriff auf Ihre Amazon ECR-Repositories zu ermöglichen.

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Principal und die Ressource in unterschiedlichen AWS Konten befinden, müssen Sie der Prinzipalentität auch die Erlaubnis erteilen, auf die Ressource zuzugreifen. Sie erteilen Berechtigungen, indem Sie der Entität eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, benötigen Sie in der identitätsbasierten Richtlinie keine zusätzlichen Amazon ECR-Repository-Berechtigungen. Weitere Informationen finden Sie unter Wie sich IAM-Rollen von ressourcenbasierten Richtlinien unterscheiden im IAM-Benutzerhandbuch.

Der Amazon ECR-Service unterstützt nur einen Typ von ressourcenbasierten Richtlinien, die sogenannte Repository Policy, die an ein Repository angehängt wird. Diese Richtlinie definiert, welche Prinzipal-Entitäten (Konten, Benutzer, Rollen und verbundene Benutzer) Aktionen auf dem Container durchführen können. Weitere Informationen zum Anfügen einer ressourcenbasierten Richtlinie an ein Repository finden Sie unter Richtlinien für private Repositorys in Amazon ECR.

> **Note**
>
> In einer Amazon ECR-Repository-Richtlinie unterstützt das Richtlinienelement `Sid` zusätzliche Zeichen und Entfernungen, die in IAM-Richtlinien nicht unterstützt werden.

Beispiele

Beispiele für ressourcenbasierte Amazon ECR-Richtlinien finden Sie unter Beispiele für Richtlinien für private Repositorien in Amazon ECR,

Autorisierung basierend auf Amazon ECR-Tags

Sie können Tags an Amazon ECR-Ressourcen anhängen oder Tags in einer Anfrage an Amazon ECR übergeben. Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `ecr:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder `aws:TagKeys`Bedingung verwenden. Weitere Informationen zum Taggen von Amazon ECR-Ressourcen finden Sie unter Kennzeichnen eines privaten Repositorys in Amazon ECR.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Verwenden Tag-basierter Zugriffskontrolle.

Amazon ECR IAM-Rollen

Eine IAM-Rolle ist eine Entität innerhalb Ihres AWS Kontos, die über bestimmte Berechtigungen verfügt.

Verwendung temporärer Anmeldeinformationen mit Amazon ECR

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Sie erhalten temporäre Sicherheitsanmeldedaten, indem Sie AWS STS API-Operationen wie AssumeRoleoder GetFederationTokenaufrufen.

Amazon ECR unterstützt die Verwendung temporärer Anmeldeinformationen.

Service-verknüpfte Rollen

Mit dienstbezogenen Rollen können AWS Dienste auf Ressourcen in anderen Diensten zugreifen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem IAM-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.

Amazon ECR unterstützt Service-verknüpfte Rollen. Weitere Informationen finden Sie unter Verwendung von dienstgebundenen Rollen für Amazon ECR.

Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien

Standardmäßig verfügen Benutzer und Rollen nicht über die Berechtigung, Amazon-ECR-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Amazon ECR definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Elastic Container Registry in der Service Authorization Reference.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Themen

- Bewährte Methoden für Richtlinien
- Verwendung der Amazon ECR-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Zugriff auf ein Amazon ECR-Repository

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand Amazon-ECR-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr

verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren

Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwendung der Amazon ECR-Konsole

Um auf die Konsole von Amazon Elastic Container Registry zugreifen zu können, müssen Sie über eine Mindestanzahl von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon ECR-Ressourcen in Ihrem AWS Konto aufzulisten und einzusehen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Um sicherzustellen, dass diese Entitäten weiterhin die Amazon ECR-Konsole verwenden können, fügen Sie die `AmazonEC2ContainerRegistryReadOnly` AWS verwaltete Richtlinie zu den Entitäten hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch:

Informationen zu den Berechtigungen für diese Richtlinie finden Sie unter AmazonElasticContainerRegistryPublicReadOnly in der Referenz zu von AWS verwalteten Richtlinien.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die den API-Operation entsprechen, die Sie ausführen möchten.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder 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": "*"
        }
    ]
}
```

Zugriff auf ein Amazon ECR-Repository

In diesem Beispiel möchten Sie einem Benutzer in Ihrem AWS Konto Zugriff auf eines Ihrer Amazon ECR-Repositorys gewähren. `my-repo` Sie möchten dem Benutzer auch erlauben, Images zu übertragen, abzurufen und aufzulisten.

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"
      }
   ]
}
```

Verwenden Tag-basierter Zugriffskontrolle

Mit der Amazon ECR `CreateRepository` API-Aktion können Sie Tags angeben, wenn Sie das Repository erstellen. Weitere Informationen finden Sie unter Kennzeichnen eines privaten Repositorys in Amazon ECR.

Damit Benutzer Repositorys bei der Erstellung markieren können, müssen sie über die Berechtigung zur Verwendung der Aktion verfügen, mit der die Ressource erstellt wird (z. B. `ecr:CreateRepository`). Wenn Tags in der Aktion angegeben werden, mit der die Ressource erstellt wird, führt Amazon eine zusätzliche Autorisierung für die `ecr:CreateRepository`-Aktion aus, um die Berechtigungen der Benutzer zum Erstellen von Tags zu überprüfen.

Sie können die Tag-basierte Zugriffskontrolle über IAM-Richtlinien verwenden. Im Folgenden sind einige Beispiele aufgeführt.

Die folgende Richtlinie würde einem Benutzer nur erlauben, ein Repository als `key=environment,value=dev` zu erstellen oder zu kennzeichnen.

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"
                }
            }
        }
    ]
}
```

Die folgende Richtlinie würde es einem Benutzer ermöglichen, Bilder aus allen Repositorys abzurufen, sofern sie nicht als gekennzeichnet sind. `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 verwaltete Richtlinien für Amazon Elastic Container Registry

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet AWS wird. AWS Verwaltete Richtlinien dienen dazu, Berechtigungen für viele gängige Anwendungsfälle bereitzustellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

Amazon ECR bietet mehrere verwaltete Richtlinien, die Sie an IAM-Identitäten oder Amazon-Instances anhängen können. EC2 Diese verwalteten Richtlinien ermöglichen unterschiedliche Kontrollstufen für den Zugriff auf Amazon ECR-Ressourcen und API-Operationen. Weitere Informationen zu jedem in diesen Richtlinien erwähnten API-Vorgang finden Sie unter Aktionen in der Amazon Elastic Container Registry API-Referenz.

Themen

- Amazon EC2 ContainerRegistryFullAccess
- Amazon EC2 ContainerRegistryPowerUser
- Amazon EC2 ContainerRegistryPullOnly
- Amazon EC2 ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- ECRTemplateServiceRolePolicy
- Amazon ECR-Updates für AWS verwaltete Richtlinien

Amazon EC2 ContainerRegistryFullAccess

Sie können die `AmazonEC2ContainerRegistryFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen. Diese Richtlinie gewährt Administratorzugriff auf Amazon ECR-Ressourcen und gewährt einer IAM-Identität (z. B. einem Benutzer, einer Gruppe oder Rolle) Zugriff auf die AWS Services, in die Amazon ECR integriert ist, um alle Amazon ECR-Funktionen nutzen zu können. Die Verwendung dieser Richtlinie ermöglicht den Zugriff auf alle Amazon ECR-Funktionen, die in der AWS-Managementkonsole verfügbar sind.

Die Berechtigungen für diese Richtlinie finden Sie unter Amazon EC2 ContainerRegistryFullAccess in der Referenz zu AWS verwalteten Richtlinien.

Amazon EC2 ContainerRegistryPowerUser

Sie können die `AmazonEC2ContainerRegistryPowerUser`-Richtlinie an Ihre IAM-Identitäten anfügen. Diese Richtlinie gewährt administrative Berechtigungen, die es IAM-Benutzern erlauben, Repositorys zu lesen und zu schreiben, aber sie erlaubt ihnen nicht, Repositorys zu löschen oder die auf sie angewandten Richtliniendokumente zu ändern.

Die Berechtigungen für diese Richtlinie finden Sie unter Amazon EC2 ContainerRegistryPowerUser in der Referenz zu AWS verwalteten Richtlinien.

Amazon EC2 ContainerRegistryPullOnly

Sie können die `AmazonEC2ContainerRegistryPullOnly`-Richtlinie an Ihre IAM-Identitäten anfügen. Diese Richtlinie erteilt die Erlaubnis, Container-Images aus Amazon ECR abzurufen. Wenn die Registrierung für den Pull-Through-Cache aktiviert ist, ermöglicht sie auch Pulls, um ein Bild aus einer Upstream-Registrierung zu importieren.

Die Berechtigungen für diese Richtlinie finden Sie unter Amazon EC2 ContainerRegistryPullOnly in der Referenz zu AWS verwalteten Richtlinien.

Amazon EC2 ContainerRegistryReadOnly

Sie können die `AmazonEC2ContainerRegistryReadOnly`-Richtlinie an Ihre IAM-Identitäten anfügen. Diese Richtlinie gewährt Amazon ECR nur Leseberechtigungen. Dazu gehört auch die Möglichkeit, Repositorys und Images innerhalb der Repositories aufzulisten. Es beinhaltet auch die Möglichkeit, Images von Amazon ECR mit der Docker CLI zu ziehen.

Die Berechtigungen für diese Richtlinie finden Sie unter Amazon EC2 ContainerRegistryReadOnly in der Referenz zu AWS verwalteten Richtlinien.

`AWSECRPullThroughCache_ServiceRolePolicy`

Sie können die verwaltete `AWSECRPullThroughCache_ServiceRolePolicy`-IAM-Richtlinie nicht mit Ihren IAM-Entitäten verknüpfen. Diese Richtlinie ist an eine dienstbezogene Rolle angehängt, die es Amazon ECR ermöglicht, Images durch den Pull-Through-Cache-Workflow an Ihre Repositorys zu übertragen. Weitere Informationen finden Sie unter Serviceverknüpfte Amazon-ECR-Rolle für Pull-Through-Cache.

`ECRReplicationServiceRolePolicy`

Sie können die verwaltete `ECRReplicationServiceRolePolicy`-IAM-Richtlinie nicht mit Ihren IAM-Entitäten verknüpfen. Diese Richtlinie ist mit einer servicegebundenen Rolle verknüpft, die es Amazon ECR ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter Verwendung von dienstgebundenen Rollen für Amazon ECR.

`ECRTemplateServiceRolePolicy`

Sie können die verwaltete `ECRTemplateServiceRolePolicy`-IAM-Richtlinie nicht mit Ihren IAM-Entitäten verknüpfen. Diese Richtlinie ist mit einer servicegebundenen Rolle verknüpft, die es Amazon ECR ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter Verwendung von dienstgebundenen Rollen für Amazon ECR.

Amazon ECR-Updates für AWS verwaltete Richtlinien

Hier finden Sie Informationen zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon ECR seit Beginn der Nachverfolgung dieser Änderungen durch diesen Service. Um automatisch über Änderungen auf dieser Seite informiert zu werden, abonnieren Sie den RSS-Feed auf der Seite Amazon ECR Document history.

Änderungen	Beschreibung	Datum
Serviceverknüpfte Amazon-ECR-Rolle für Pull-Through-Cache – Aktualisierung auf eine bestehende Richtlinie	Amazon ECR hat der `AWSECRPullThroughCache_ServiceRolePolicy` -Richtlinie neue Berechtigungen hinzugefügt. Diese Berechtigungen ermöglichen es Amazon ECR, Bilder aus der privaten ECR-Registrierung abzurufen. Dies ist erforderlich, wenn Sie eine Pull-Through-Cache -Regel verwenden, um Bilder aus einer anderen privaten Amazon ECR-Registrierung zwischenzuspeichern.	12. März 2025

Änderungen	Beschreibung	Datum
Amazon EC2 Container RegistryPullOnly — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt, die Amazon ECR nur Pull-Berechtigungen gewährt.	10. Oktober 2024
ECRTemplateServiceRolePolicy – Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt. Diese Richtlinie ist mit der `ECRTemplateServiceRolePolicy` serviceverknüpften Rolle für die Funktion „Vorlage zur Erstellung von Repositorys“ verknüpft.	20. Juni 2024
AWSECRPullThroughCache_ServiceRolePolicy – Aktualisierung auf eine bestehende Richtlinie	Amazon ECR hat der `AWSECRPullThroughCache_ServiceRolePolicy` -Richtlinie neue Berechtigungen hinzugefügt. Diese Berechtigungen ermöglichen Amazon ECR, den verschlüsselten Inhalt eines Secrets-Manager-Secrets abzurufen. Dies ist erforderlich, wenn eine Pull-Through-Cache-Regel verwendet wird, um Images aus einer Upstream-Registrierung zwischenzuspeichern, für das eine Authentifizierung erforderlich ist.	15. November 2023

Änderungen	Beschreibung	Datum
AWSECRPullThroughCache_ServiceRolePolicy – Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt. Diese Richtlinie wird mit der `AWSServiceRoleForECRPullThroughCache` -Serviceverknüpfte Rolle für das Pull-Through-Cache-Feature zugeordnet.	29. November 2021
ECRReplicationServiceRolePolicy – Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt. Diese Richtlinie wird mit der `AWSServiceRoleForECRReplication` - Serviceverknüpfte Rolle für das Replikations-Feature zugeordnet.	4. Dezember 2020
Amazon EC2 Container RegistryFullAccess — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryFullAccess` -Richtlinie neue Berechtigungen hinzugefügt. Diese Berechtigungen erlauben es Prinzipalen, die serviceverknüpfte Amazon ECR-Rolle zu erstellen.	4. Dezember 2020

Änderungen	Beschreibung	Datum
Amazon EC2 Container RegistryReadOnly — Aktualisierung einer bestehenden Richtlinie	Amazon ECR fügte der Richtlinie neue Berechtigungen hinzu, `AmazonEC2ContainerRegistryReadOnly` die es den Auftraggebern ermöglichen, Lebenszyklusrichtlinien zu lesen, Tags aufzulisten und die Scanergebnisse für Images zu beschreiben.	10. Dezember 2019
Amazon EC2 Container RegistryPowerUser — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryPowerUser` -Richtlinie neue Berechtigungen hinzugefügt. Sie ermöglichen es den Auftraggebern, Lebenszyklusrichtlinien zu lesen, Tags aufzulisten und die Scanergebnisse für Images zu beschreiben.	10. Dezember 2019
Amazon EC2 Container RegistryFullAccess — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryFullAccess` -Richtlinie neue Berechtigungen hinzugefügt. Sie ermöglichen es den Schulleitern, nach Verwaltungsereignissen oder AWS CloudTrail Insights-Ereignissen zu suchen, die von CloudTrail erfasst wurden.	10. November 2017

Änderungen	Beschreibung	Datum
Amazon EC2 Container RegistryReadOnly — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryReadOnly` - Richtlinie neue Berechtigungen hinzugefügt. Sie ermöglichen es den Auftraggebern, Amazon-ECR-Images zu beschreiben.	11. Oktober 2016
Amazon EC2 Container RegistryPowerUser — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryPowerUser` - Richtlinie neue Berechtigungen hinzugefügt. Sie ermöglichen es den Auftraggebern, Amazon-ECR-Images zu beschreiben.	11. Oktober 2016
Amazon EC2 Container RegistryReadOnly — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt, die Amazon ECR nur Leseberechtigungen gewährt. Diese Berechtigungen umfassen die Möglichkeit, Repositories und Images innerhalb der Repositories aufzulisten. Sie umfassen auch die Möglichkeit, mit der Docker-CLI Images aus Amazon ECR zu ziehen.	21. Dezember 2015

Änderungen	Beschreibung	Datum
Amazon EC2 Container RegistryPowerUser — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt, die Benutzern Administratorrechte gewährt, die es Benutzern ermöglichen, in Repositorys zu lesen und in sie zu schreiben, es ihnen jedoch nicht erlaubt, Repositorys zu löschen oder die für sie geltenden Richtlinien zu ändern.	21. Dezember 2015
Amazon EC2 Container RegistryFullAccess — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt. Diese Richtlinie gewährt vollen Zugang zu Amazon ECR.	21. Dezember 2015
Amazon ECR begann mit der Verfolgung von Änderungen	Amazon ECR hat damit begonnen, Änderungen für AWS verwaltete Richtlinien nachzuverfolgen.	24. Juni 2021

Verwendung von dienstgebundenen Rollen für Amazon ECR

Amazon Elastic Container Registry (Amazon ECR) verwendet AWS Identity and Access Management (IAM) service-verknüpfte Rollen, um die für die Nutzung der Replikations- und Pull-Through-Cache-Funktionen erforderlichen Berechtigungen bereitzustellen. Eine servicegebundene Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit Amazon ECR verknüpft ist. Die serviceverknüpfte Rolle ist von Amazon ECR vordefiniert. Es enthält alle Berechtigungen, die der Service zur Unterstützung der Replikation und Pull-Through-Cache-Features für Ihre private Registrierung benötigt. Nachdem Sie die Replikation oder den Pull-Through-Cache für Ihre Registrierung konfiguriert haben, wird automatisch eine serviceverknüpfte Rolle in Ihrem Namen erstellt. Weitere Informationen finden Sie unter Private Registrierungseinstellungen in Amazon ECR.

Eine servicegebundene Rolle erleichtert das Einrichten der Replikation und des Pull-Through-Cache mit Amazon ECR. Das liegt daran, dass Sie bei Verwendung dieser Rolle nicht alle erforderlichen

Berechtigungen manuell hinzufügen müssen. Amazon ECR definiert die Berechtigungen seiner mit dem Service verbundenen Rollen, und sofern nicht anders definiert, kann nur Amazon ECR seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Die Berechtigungsrichtlinie kann mit keiner anderen IAM-Entität verknüpft werden.

Sie können die entsprechende serviceverknüpfte Rolle erst löschen, nachdem Sie entweder die Replikation oder den Pull-Through-Cache in Ihrer Registrierung deaktiviert haben.aben. Dies stellt sicher, dass Sie die Berechtigungen, die Amazon ECR für diese Features benötigt, nicht versehentlich entfernen.

Informationen über andere Dienste, die dienstgebundene Rollen unterstützen, finden Sie unter AWS Dienste, die mit IAM arbeiten. Suchen Sie auf dieser verknüpften Seite nach den Diensten, die in der Spalte Dienstverknüpfte Rolle den Wert Ja haben. Wählen Sie ein Ja mit einem Link, um die entsprechende dienstbezogene Rollendokumentation für diesen Dienst anzuzeigen.

Themen

Unterstützte Regionen für Amazon ECR-Service-verknüpfte Rollen

Amazon ECR unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Amazon-ECR-Service verfügbar ist. Weitere Informationen zur Verfügbarkeit der Amazon-ECR-Region finden Sie unter AWS -Regionen und -Endpunkte.

Serviceverknüpfte Amazon-ECR-Rolle für die Replikation

Amazon ECR verwendet eine serviceverknüpfte Rolle mit dem Namen AWSServiceRoleForECRReplication, die es Amazon ECR ermöglicht, Bilder über mehrere Konten hinweg zu replizieren.

Service-gebundene Rollenberechtigungen für Amazon ECR

Die AWSService RoleFor ECRReplication serviceverknüpfte Rolle vertraut darauf, dass die folgenden Dienste die Rolle übernehmen:

- `replication.ecr.amazonaws.com`

Die folgende `ECRReplicationServiceRolePolicy` Richtlinie für Rollenberechtigungen erlaubt Amazon ECR, die folgenden Aktionen auf Ressourcen anzuwenden:

JSON

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

> **Note**
>
> Es `ReplicateImage` handelt sich um eine interne API, die Amazon ECR für die Replikation verwendet und die nicht direkt aufgerufen werden kann.

Sie müssen die Berechtigungen so konfigurieren, dass eine IAM-Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Service-Linked Role Permissions im IAM-Benutzerhandbuch.

Erstellen einer servicegebundenen Rolle für Amazon ECR

Sie müssen die serviceverknüpfte Amazon ECR-Rolle nicht manuell erstellen. Wenn Sie die Replikationseinstellungen für Ihre Registrierung in der AWS-Managementkonsole, der oder der AWS CLI AWS API konfigurieren, erstellt Amazon ECR die serviceverknüpfte Rolle für Sie.

Wenn Sie diese dienstverknüpfte Rolle löschen und erneut erstellen müssen, können Sie die Rolle in Ihrem Konto auf dieselbe Weise neu erstellen. Wenn Sie die Replikationseinstellungen für Ihre Registrierung konfigurieren, erstellt Amazon ECR die serviceverknüpfte Rolle erneut für Sie.

Bearbeiten einer serviceverknüpften Rolle für Amazon ECR

Amazon ECR erlaubt es nicht, die AWSService RoleFor ECRReplication serviceverknüpfte Rolle manuell zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach der Erstellung einer serviceverknüpften Rolle nicht bearbeitet werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften Rolle für Amazon ECR

Wenn Sie ein Feature oder einen Dienst, für den eine dienstgebundene Rolle erforderlich ist, nicht mehr benötigen, empfehlen wir Ihnen, diese Rolle zu löschen. Auf diese Weise haben Sie keine ungenutzte Einheit, die nicht aktiv überwacht oder gepflegt wird. Sie müssen jedoch die Replikationskonfiguration für Ihre Registrierung in jeder Region entfernen, bevor Sie die dienstverknüpfte Rolle manuell löschen können.

> **Note**
>
> Wenn Sie versuchen, Ressourcen zu löschen, während der Amazon ECR-Service die Rollen noch verwendet, kann Ihre Löschaktion fehlschlagen. Wenn dies der Fall ist, warten Sie einige Minuten und versuchen Sie es erneut.

Um Amazon ECR-Ressourcen zu löschen, die verwendet werden von AWSService RoleFor ECRReplication

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, für die Ihre Replikationskonfiguration festgelegt ist.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung.
4. Wählen Sie auf der Seite Private Registrierung im Abschnitt Replikationskonfiguration die Option Bearbeiten.
5. Um alle Ihre Replikationsregeln zu löschen, wählen Sie Alle löschen. Dieser Schritt erfordert eine Bestätigung.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die AWSServiceRoleForECRReplicationserviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Leitfaden.

Serviceverknüpfte Amazon-ECR-Rolle für Pull-Through-Cache

Amazon ECR verwendet eine serviceverknüpfte Rolle mit AWSServiceRoleForECRPullThroughCachedem Namen, die Amazon ECR die Erlaubnis erteilt, in Ihrem Namen Aktionen durchzuführen, um Cache-Aktionen abzuschließen. Weitere Informationen zum Pull-Through-Cache finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.

Service-gebundene Rollenberechtigungen für Amazon ECR

Die AWSServiceRoleForECRPullThroughCacheserviceverknüpfte Rolle vertraut darauf, dass der folgende Service die Rolle übernimmt.

- `pullthroughcache.ecr.amazonaws.com`

Details zu Berechtigungen

Die Berechtigungsrichtlinie `AWSECRPullThroughCache_ServiceRolePolicy` ist mit der dienstverknüpften Rolle verbunden. Diese verwaltete Richtlinie gewährt Amazon ECR die Erlaubnis, die folgenden Aktionen durchzuführen. Weitere Informationen finden Sie unter `AWSECRPullThroughCache_ServiceRolePolicy`.

- `ecr`— Ermöglicht dem Amazon ECR-Service, Bilder abzurufen und in ein privates Repository zu übertragen.
- `secretsmanager:GetSecretValue`— Ermöglicht dem Amazon ECR-Service, den verschlüsselten Inhalt eines AWS Secrets Manager Geheimnisses abzurufen. Dies ist erforderlich, wenn eine Pull-Through-Cache-Regel verwendet wird, um Images aus einer Upstream-Registrierung zwischenzuspeichern, für das eine Authentifizierung in Ihrer privaten Registry erforderlich ist. Diese Berechtigung gilt nur für Secrets mit dem Namenspräfix `ecr-pullthroughcache/`.

Die `AWSECRPullThroughCache_ServiceRolePolicy`-Richtlinie enthält das folgende JSON.

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}"
                }
            }
        }
    ]
}
```

Sie müssen die Berechtigungen so konfigurieren, dass eine IAM-Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigung im IAM-Benutzerhandbuch.

Erstellen einer servicegebundenen Rolle für Amazon ECR

Sie müssen die serviceverknüpfte Amazon-ECR-Rolle für Pull-Through-Cache nicht manuell erstellen. Wenn Sie eine Pull-Through-Cache-Regel für Ihre private Registrierung in der AWS-Managementkonsole AWS CLI, der oder der AWS API erstellen, erstellt Amazon ECR die serviceverknüpfte Rolle für Sie.

Wenn Sie diese dienstverknüpfte Rolle löschen und erneut erstellen müssen, können Sie die Rolle in Ihrem Konto auf dieselbe Weise neu erstellen. Wenn Sie eine Pull-Through-Cache-Regel für Ihre private Registrierung erstellen, erstellt Amazon ECR die serviceverknüpfte Rolle erneut für Sie, falls sie noch nicht vorhanden ist.

Bearbeiten einer serviceverknüpften Rolle für Amazon ECR

Amazon ECR erlaubt es nicht, die AWSServiceRoleForECRPullThroughCacheserviceverknüpfte Rolle manuell zu bearbeiten. Nachdem Sie eine serviceverknüpfte Rolle erstellt haben, können Sie den Namen der Rolle nicht mehr ändern, da verschiedene Entitäten auf die Rolle verweisen könnten. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften Rolle für Amazon ECR

Wenn Sie ein Feature oder einen Dienst, für den eine dienstgebundene Rolle erforderlich ist, nicht mehr benötigen, empfehlen wir Ihnen, diese Rolle zu löschen. Auf diese Weise haben Sie keine ungenutzte Einheit, die nicht aktiv überwacht oder gepflegt wird. Sie müssen jedoch die Pull-Through-Cache-Regeln für Ihre Registrierung in jeder Region löschen, bevor Sie die serviceverknüpfte Rolle manuell löschen können.

> **Note**
>
> Wenn Sie versuchen, Ressourcen zu löschen, während der Amazon-ECR-Service die Rolle noch verwendet, kann Ihre Löschaktion fehlschlagen. Wenn dies der Fall ist, warten Sie einige Minuten und versuchen Sie es erneut.

So löschen Sie die durch die serviceverknüpfte Rolle AWSServiceRoleForECRPullThroughCache verwendeten Amazon-ECR-Ressourcen

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der Ihre Pull-Through-Cache-Regeln erstellt werden.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung.
4. Wählen Sie auf der Seite Private Registry im Abschnitt Pull-Through-Cache-Konfiguration die Option Bearbeiten aus.
5. Wählen Sie für jede von Ihnen erstellte Pull-Through-Cache-Regel die Regel aus und wählen Sie dann Regel löschen.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die AWSServiceRoleForECRPullThroughCacheserviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Leitfaden.

Mit dem Amazon ECR Service verknüpfte Rolle für Vorlagen zur Repository-Erstellung

Amazon ECR verwendet eine servicebezogene Rolle mit dem Namen AWSServiceRoleForECRTemplate, die Amazon ECR die Erlaubnis erteilt, in Ihrem Namen Aktionen durchzuführen, um Aktionen zur Erstellung von Repository-Vorlagen abzuschließen.

Service-gebundene Rollenberechtigungen für Amazon ECR

Die AWSServiceRoleForECRTemplateserviceverknüpfte Rolle vertraut darauf, dass der folgende Service die Rolle übernimmt.

- `ecr.amazonaws.com`

Details zu Berechtigungen

Die Berechtigungsrichtlinie `ECRTemplateServiceRolePolicy` ist mit der dienstverknüpften Rolle verbunden. Diese verwaltete Richtlinie erteilt Amazon ECR die Erlaubnis, Repository-Erstellungsaktionen in Ihrem Namen durchzuführen.

Die `ECRTemplateServiceRolePolicy`-Richtlinie enthält das folgende JSON.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
     "Sid": "CreateRepositoryWithTemplate",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Sie müssen die Berechtigungen so konfigurieren, dass eine IAM-Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigung im IAM-Benutzerhandbuch.

Erstellen einer servicegebundenen Rolle für Amazon ECR

Sie müssen die mit dem Amazon ECR Service verknüpfte Rolle für die Repository-Erstellungsvorlage nicht manuell erstellen. Wenn Sie eine Vorlagenregel für die Repository-Erstellung für Ihre private Registrierung in der AWS-Managementkonsole AWS CLI, der oder der AWS API erstellen, erstellt Amazon ECR die serviceverknüpfte Rolle für Sie.

Wenn Sie diese dienstverknüpfte Rolle löschen und erneut erstellen müssen, können Sie die Rolle in Ihrem Konto auf dieselbe Weise neu erstellen. Wenn Sie eine Regel zur Erstellung eines Repositorys für Ihre private Registrierung erstellen, erstellt Amazon ECR die serviceverknüpfte Rolle erneut für Sie, sofern sie noch nicht vorhanden ist.

Bearbeiten einer serviceverknüpften Rolle für Amazon ECR

Amazon ECR erlaubt es nicht, die AWSServiceRoleForECRTemplateserviceverknüpfte Rolle manuell zu bearbeiten. Nachdem Sie eine serviceverknüpfte Rolle erstellt haben, können Sie den Namen der Rolle nicht mehr ändern, da verschiedene Entitäten auf die Rolle verweisen könnten. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften Rolle für Amazon ECR

Wenn Sie ein Feature oder einen Dienst, für den eine dienstgebundene Rolle erforderlich ist, nicht mehr benötigen, empfehlen wir Ihnen, diese Rolle zu löschen. Auf diese Weise haben Sie keine ungenutzte Einheit, die nicht aktiv überwacht oder gepflegt wird. Sie müssen jedoch die Regeln für die Erstellung von Repositorys für Ihre Registrierung in jeder Region löschen, bevor Sie die serviceverknüpfte Rolle manuell löschen können.

> **Note**
>
> Wenn Sie versuchen, Ressourcen zu löschen, während der Amazon-ECR-Service die Rolle noch verwendet, kann Ihre Löschaktion fehlschlagen. Wenn dies der Fall ist, warten Sie einige Minuten und versuchen Sie es erneut.

So löschen Sie die durch die serviceverknüpfte Rolle AWSServiceRoleForECRTemplate verwendeten Amazon-ECR-Ressourcen

1. Öffnen Sie die Amazon ECR-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Ihre Repository-Erstellungsregeln erstellt werden.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung.
4. Wählen Sie auf der Seite Private Registry im Abschnitt Vorlagen für die Repository-Erstellung die Option Bearbeiten aus.
5. Wählen Sie für jede Regel zur Erstellung eines Repositorys, die Sie erstellt haben, die Regel aus und klicken Sie dann auf Regel löschen.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die AWSServiceRoleForECRTemplateserviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Leitfaden.

Fehlerbehebung bei Amazon Elastic Container Registry - Identität und Zugriff

Die folgenden Informationen helfen Ihnen bei der Diagnose und Behebung häufiger Probleme, die bei der Arbeit mit Amazon ECR und IAM auftreten können.

Themen

Ich bin nicht befugt, eine Aktion in Amazon ECR durchzuführen

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `ecr:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `ecr:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam durchzuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der Aktion „`iam:PassRole`“ autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon ECR übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon ECR durchzuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Amazon ECR-Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Um zu erfahren, ob Amazon E CR diese Features unterstützt, siehe Wie Amazon Elastic Container Registry mit IAM funktioniert.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen in AWS-Konten Ihrem Besitz gewähren können, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , dem Sie gehören.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Datenschutz bei Amazon ECR

Das AWS Modell der gilt für den Datenschutz in Amazon Elastic Container Service. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon ECS oder anderen AWS-Services über die Konsole AWS CLI, API oder arbeiten AWS SDKs. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen

externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Themen

Verschlüsselung im Ruhezustand

> **Important**
>
> Die zweischichtige serverseitige Verschlüsselung mit AWS KMS (DSSE-KMS) ist nur in den AWS GovCloud (US) Regionen verfügbar.

Amazon ECR speichert Images in Amazon-S3-Buckets, die Amazon ECR verwaltet. Standardmäßig verwendet Amazon ECR eine serverseitige Verschlüsselung mit von Amazon S3 verwalteten Verschlüsselungsschlüsseln, die Ihre Daten im Ruhezustand mit einem AES-256-Verschlüsselungsalgorithmus verschlüsseln. Dies erfordert kein Handeln Ihrerseits und wird ohne zusätzliche Kosten angeboten. Weitere Informationen finden Sie unter Schützen von Daten mit serverseitiger Verschlüsselung mit Amazon S3-verwalteten Verschlüsselungsschlüsseln (SSE-S3) im Benutzerhandbuch von Amazon Simple Storage Service.

Für mehr Kontrolle über die Verschlüsselung Ihrer Amazon ECR-Repositorys können Sie die serverseitige Verschlüsselung mit KMS-Schlüsseln verwenden, die in AWS Key Management Service () gespeichert sind.AWS KMS Wenn Sie Ihre Daten verschlüsseln, können Sie entweder den Standard verwenden Von AWS verwalteter Schlüssel, der von Amazon ECR verwaltet wird, oder Ihren eigenen KMS-Schlüssel (als vom Kunden verwalteter Schlüssel bezeichnet) angeben. AWS KMS Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung mit in gespeicherten KMS-Schlüsseln AWS KMS (SSE-KMS) im Amazon Simple Storage Service-Benutzerhandbuch.

Sie können wählen, ob Sie zwei Verschlüsselungsebenen auf Ihre Amazon ECR-Images anwenden möchten, indem Sie die zweischichtige serverseitige Verschlüsselung mit AWS KMS () verwenden. DSSE-KMS DSSE-KMSDie Option ist ähnlich wieSSE-KMS, wendet jedoch zwei einzelne Verschlüsselungsebenen anstelle einer Ebene an. Weitere Informationen finden Sie unter Verwenden der serverseitigen Dual-Layer-Verschlüsselung mit AWS KMS Schlüsseln (DSSE-KMS).

Jedes Amazon ECR-Repository hat eine Verschlüsselungskonfiguration, die bei der Erstellung des Repositorys festgelegt wird. Sie können für jedes Repository unterschiedliche Verschlüsselungskonfigurationen verwenden. Weitere Informationen finden Sie unter Erstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern.

Wenn ein Repository mit aktivierter AWS KMS Verschlüsselung erstellt wird, wird ein KMS-Schlüssel verwendet, um den Inhalt des Repositorys zu verschlüsseln. Darüber hinaus fügt Amazon ECR dem KMS-Schlüssel einen AWS KMS Zuschuss hinzu, wobei das Amazon ECR-Repository als Principal des Zuschussempfängers fungiert.

Im Folgenden erfahren Sie, wie Amazon ECR mit AWS KMS integriert ist, um Ihre Repositories zu verschlüsseln und zu entschlüsseln:

1. Beim Erstellen eines Repositorys sendet Amazon ECR einen DescribeKeyAufruf an, AWS KMS um den Amazon-Ressourcennamen (ARN) des in der Verschlüsselungskonfiguration angegebenen KMS-Schlüssels zu überprüfen und abzurufen.
2. Amazon ECR sendet zwei CreateGrantAnfragen AWS KMS zur Erstellung von Zuschüssen für den KMS-Schlüssel, damit Amazon ECR Daten mithilfe des Datenschlüssels ver- und entschlüsseln kann.
3. Beim Pushen eines Images wird eine GenerateDataKeyAnfrage gestellt, die den KMS-Schlüssel AWS KMS angibt, der für die Verschlüsselung der Bildebene und des Manifests verwendet werden soll.
4. AWS KMS generiert einen neuen Datenschlüssel, verschlüsselt ihn unter dem angegebenen KMS-Schlüssel und sendet den verschlüsselten Datenschlüssel, der zusammen mit den Metadaten der Bildebene und dem Imagemanifest gespeichert wird.
5. Beim Abrufen eines Bilds wird eine Decrypt-Anfrage an gesendet AWS KMS, in der der verschlüsselte Datenschlüssel angegeben wird.
6. AWS KMS entschlüsselt den verschlüsselten Datenschlüssel und sendet den entschlüsselten Datenschlüssel an Amazon S3.
7. Der Datenschlüssel wird zur Entschlüsselung der Image-Ebene verwendet, bevor die Image-Ebene abgerufen wird.
8. Wenn ein Repository gelöscht wird, sendet Amazon ECR zwei RetireGrantAnfragen an, AWS KMS um die für das Repository erstellten Zuschüsse zurückzuziehen.

Überlegungen

Die folgenden Punkte sollten bei der Verwendung von AWS KMS basierter Verschlüsselung (SSE-KMSoderDSSE-KMS) mit Amazon ECR berücksichtigt werden.

- Wenn Sie Ihr Amazon ECR-Repository mit KMS-Verschlüsselung erstellen und keinen KMS-Schlüssel angeben, verwendet Amazon ECR standardmäßig einen Von AWS verwalteter Schlüssel mit dem Alias`aws/ecr`. Dieser KMS-Schlüssel wird in Ihrem Konto erstellt, wenn Sie zum ersten Mal ein Repository mit aktivierter KMS-Verschlüsselung erstellen.
- Die Konfiguration der Repository-Verschlüsselung kann nach der Erstellung eines Repositorys nicht geändert werden.
- Wenn Sie die KMS-Verschlüsselung mit Ihrem eigenen KMS-Schlüssel verwenden, muss sich dieser Schlüssel in derselben Region wie Ihr Repository befinden.
- Die Bewilligungen, die Amazon ECR in Ihrem Namen erstellt, sollten nicht widerrufen werden. Wenn Sie die Genehmigung widerrufen, die Amazon ECR die Erlaubnis erteilt, die AWS KMS Schlüssel in Ihrem Konto zu verwenden, kann Amazon ECR nicht auf diese Daten zugreifen, keine neuen Bilder verschlüsseln, die in das Repository übertragen werden, oder sie entschlüsseln, wenn sie abgerufen werden. Wenn Sie eine Förderung für Amazon ECR widerrufen, tritt die Änderung sofort in Kraft. Um Zugriffsrechte zu widerrufen, sollten Sie das Repository löschen, anstatt die Gewährung zu widerrufen. Wenn ein Repository gelöscht wird, hebt Amazon ECR die Förderungen in Ihrem Namen auf.
- Die Verwendung von Schlüsseln ist mit Kosten verbunden. AWS KMS Weitere Informationen finden Sie unter AWS Key Management Service Preise.
- Die Verwendung der serverseitigen Dual-Layer-Verschlüsselung ist mit Kosten verbunden. Weitere Informationen finden Sie unter Amazon ECR — Preise

Erforderliche IAM-Berechtigungen

Wenn Sie ein Amazon ECR-Repository mit serverseitiger Verschlüsselung unter Verwendung von AWS KMS erstellen oder löschen, hängen die erforderlichen Berechtigungen von dem spezifischen KMS-Schlüssel ab, den Sie verwenden.

Erforderliche IAM-Berechtigungen bei Verwendung von Von AWS verwalteter Schlüssel für Amazon ECR

Wenn die AWS KMS Verschlüsselung für ein Amazon ECR-Repository aktiviert ist, aber kein KMS-Schlüssel angegeben ist, wird standardmäßig der Von AWS verwalteter Schlüssel für Amazon ECR

verwendet. Wenn der AWS-managed KMS-Schlüssel für Amazon ECR zum Verschlüsseln eines Repositorys verwendet wird, kann jeder Principal, der über die Berechtigung zum Erstellen eines Repositorys verfügt, auch die AWS KMS Verschlüsselung für das Repository aktivieren. Der IAM-Prinzipal, der das Repository löscht, muss jedoch die `kms:RetireGrant`-Berechtigung haben. Dadurch können die Grants, die dem AWS KMS Schlüssel bei der Erstellung des Repositorys hinzugefügt wurden, zurückgezogen werden.

Die folgende Beispiel-IAM-Richtlinie kann als Inline-Richtlinie zu einem Benutzer hinzugefügt werden, um sicherzustellen, dass er über die Mindestberechtigungen verfügt, die zum Löschen eines Repositorys mit aktivierter Verschlüsselung erforderlich sind. Der KMS-Schlüssel, der zur Verschlüsselung des Repositorys verwendet wird, kann über den Parameter resource angegeben werden.

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"
        }
    ]
}
```

Erforderliche IAM-Berechtigungen bei Verwendung eines vom Kunden verwalteten Schlüssels

Beim Erstellen eines Repositorys mit aktivierter AWS KMS Verschlüsselung mithilfe eines vom Kunden verwalteten Schlüssels sind für den Benutzer oder die Rolle, die das Repository erstellt, Berechtigungen sowohl für die KMS-Schlüsselrichtlinie als auch für die IAM-Richtlinie erforderlich.

Bei der Erstellung Ihres eigenen KMS-Schlüssels können Sie entweder die von AWS KMS erstellte Standard-Schlüsselrichtlinie verwenden oder Ihre eigene angeben. Um sicherzustellen, dass der vom

Kunden verwaltete Schlüssel vom Kontoinhaber verwaltet werden kann, sollte die Schlüsselrichtlinie für den KMS-Schlüssel alle AWS KMS Aktionen für den Root-Benutzer des Kontos zulassen. Die Schlüsselrichtlinie kann um zusätzliche Berechtigungen erweitert werden, doch sollte zumindest der Root-Benutzer die Berechtigung erhalten, den KMS-Schlüssel zu verwalten. Damit der KMS-Schlüssel nur für Anfragen verwendet werden kann, die ihren Ursprung in Amazon ECR haben, können Sie den ViaService Bedingungsschlüssel kms: mit dem `ecr.<region>.amazonaws.com` Wert verwenden.

Die folgende Beispielschlüsselrichtlinie gewährt dem AWS Konto (Root-Benutzer), dem der KMS-Schlüssel gehört, vollen Zugriff auf den KMS-Schlüssel. Weitere Informationen zu dieser Beispielschlüsselrichtlinie finden Sie unter Erlaubt Zugriff auf das AWS Konto und aktiviert IAM-Richtlinien im AWS Key Management Service Entwicklerhandbuch.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

Der IAM-Benutzer, die IAM-Rolle oder das AWS Konto, das Ihre Repositorys erstellt `kms:CreateGrantkms:RetireGrant`, muss zusätzlich zu den erforderlichen Amazon `kms:DescribeKey` ECR-Berechtigungen über die Berechtigungen, und verfügen.

> **Note**
>
> Die `kms:RetireGrant`-Berechtigung muss der IAM-Richtlinie des Benutzers oder der Rolle hinzugefügt werden, der/die das Repository erstellt. Die Berechtigungen `kms:CreateGrant`

und `kms:DescribeKey` können entweder der Schlüsselrichtlinie für den KMS-Schlüssel oder der IAM-Richtlinie des Benutzers oder der Rolle, die das Repository erstellt, hinzugefügt werden. Weitere Informationen zur Funktionsweise von AWS KMS Berechtigungen finden Sie unter AWS KMS API-Berechtigungen: Referenz zu Aktionen und Ressourcen im Entwicklerhandbuch.AWS Key Management Service

Die folgende Beispiel-IAM-Richtlinie kann als Inline-Richtlinie zu einem Benutzer hinzugefügt werden, um sicherzustellen, dass er über die Mindestberechtigungen verfügt, die erforderlich sind, um ein Repository mit aktivierter Verschlüsselung zu erstellen und das Repository zu löschen, wenn er es nicht mehr benötigt. Der zur Verschlüsselung des Repositorys verwendete AWS KMS key -Schlüssel kann mit dem Ressourcen-Parameter angegeben werden.

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"
        }
    ]
}
```

Erlauben Sie einem Benutzer, KMS-Schlüssel in der Konsole aufzulisten, wenn er ein Repository erstellt

Wenn Sie die Amazon ECR-Konsole zum Erstellen eines Repositorys verwenden, können Sie einem Benutzer die Berechtigung erteilen, die vom Kunden verwalteten KMS-Schlüssel in der Region

aufzulisten, wenn Sie die Verschlüsselung für das Repository aktivieren. Das folgende Beispiel für eine IAM-Richtlinie zeigt die Berechtigungen, die für die Auflistung Ihrer KMS-Schlüssel und -Aliase bei Verwendung der Konsole erforderlich sind.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
}
```

Überwachung der Interaktion zwischen Amazon ECR mit AWS KMS

Sie können AWS CloudTrail damit die Anfragen verfolgen, an die Amazon ECR in AWS KMS Ihrem Namen sendet. Die Protokolleinträge im Protokoll enthalten einen Verschlüsselungskontextschlüssel, um sie leichter identifizierbar zu machen. CloudTrail

Amazon ECR-Verschlüsselungskontext

Ein Verschlüsselungskontext ist ein Satz von Schlüssel-Wert-Paaren, der beliebige nicht geheime Daten enthält. Wenn Sie einen Verschlüsselungskontext in eine Anforderung zur Verschlüsselung von Daten einbeziehen, wird der Verschlüsselungskontext AWS KMS kryptografisch an die verschlüsselten Daten gebunden. Zur Entschlüsselung der Daten müssen Sie denselben Verschlüsselungskontext übergeben.

In seinen Anfragen GenerateDataKeyund Decrypt verwendet Amazon ECR einen Verschlüsselungskontext mit zwei Name-Wert-Paaren, die das verwendete Repository und den verwendeten Amazon S3 S3-Bucket identifizieren. AWS KMS Dies wird im folgenden Beispiel veranschaulicht. Die Namen variieren nicht, aber die kombinierten Verschlüsselungskontextwerte sind für jeden Wert unterschiedlich.

```
"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"
}
```

Sie können den Verschlüsselungskontext verwenden, um diese kryptografischen Vorgänge in Prüfaufzeichnungen und Protokollen wie Amazon CloudWatch Logs zu identifizieren AWS CloudTrailund als Voraussetzung für die Autorisierung in Richtlinien und Zuschüssen zu verwenden.

Der Amazon ECR-Verschlüsselungskontext besteht aus zwei Name-Wert-Paaren.

- aws:s3:arn - Das erste Name-Wert-Paar identifiziert den Bucket. Der Schlüssel lautet `aws:s3:arn`. Der Wert ist der Amazon Resource Name (ARN) des Amazon S3-Buckets.

  ```
  "aws:s3:arn": "ARN of an Amazon S3 bucket"
  ```

  Wenn die ARN des Buckets z. B. `arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df` ist, würde der Verschlüsselungskontext das folgende Paar enthalten.

  ```
  "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
  ```

- aws:ecr:arn – Das zweite Name-Wert-Paar identifiziert den Amazon Resource Name (ARN) des Repositorys. Der Schlüssel lautet `aws:ecr:arn`. Der Wert ist der ARN des Repositorys.

  ```
  "aws:ecr:arn": "ARN of an Amazon ECR repository"
  ```

  Wenn der ARN des Repository beispielsweise `arn:aws:ecr:us-west-2:111122223333:repository/repository-name` lautet, würde der Verschlüsselungskontext das folgende Paar enthalten.

  ```
  "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
  ```

Fehlerbehebung

Wenn Sie ein Amazon ECR-Repository mit der Konsole löschen und das Repository erfolgreich gelöscht wurde, Amazon ECR aber nicht in der Lage ist, die zu Ihrem KMS-Schlüssel für Ihr Repository hinzugefügten Grants zurückzuziehen, erhalten Sie die folgende Fehlermeldung.

```
The repository [{repository-name}] has been deleted successfully but the grants created
 by the kmsKey [{kms_key}] failed to be retired
```

In diesem Fall können Sie die AWS KMS Zuschüsse für das Repository selbst zurückziehen.

Um AWS KMS Zuschüsse für ein Repository manuell zurückzuziehen

1. Listet die Grants für den AWS KMS Schlüssel auf, der für das Repository verwendet wird. Der `key-id`-Wert ist in der Fehlermeldung enthalten, die Sie von der Konsole erhalten. Sie können den `list-keys` Befehl auch verwenden, um Von AWS verwaltete Schlüssel sowohl die als auch die vom Kunden verwalteten KMS-Schlüssel in einer bestimmten Region in Ihrem Konto aufzulisten.

   ```
   aws kms list-grants \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
        --region us-west-2
   ```

   Die Ausgabe enthält einen `EncryptionContextSubset` mit dem Amazon Resource Name (ARN) Ihres Repositorys. Auf diese Weise können Sie feststellen, welche der zum Schlüssel hinzugefügten Förderungen diejenige ist, die Sie aufheben möchten. Der `GrantId`-Wert wird verwendet, wenn die Förderung im nächsten Schritt aufgehoben wird.

2. Alle Grants für den AWS KMS Schlüssel, der dem Repository hinzugefügt wurde, zurückziehen. Ersetzen Sie den Wert für *`GrantId`* durch die ID des Grants aus der Ausgabe des vorherigen Schritts.

   ```
   aws kms retire-grant \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
        --grant-id GrantId \
        --region us-west-2
   ```

Compliance-Validierung für Amazon Elastic Container Registry

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Sicherheit der Infrastruktur in Amazon Elastic Container Registry

Als verwalteter Service ist Amazon Elastic Container Registry durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Amazon ECR zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Sie können diese API-Vorgänge von jedem Netzwerkstandort aus aufrufen, aber Amazon ECR unterstützt ressourcenbasierte Zugriffsrichtlinien, die Einschränkungen auf der Grundlage der Quell-IP-Adresse enthalten können. Sie können auch Amazon ECR-Richtlinien verwenden, um den Zugriff von bestimmten Amazon Virtual Private Cloud (Amazon VPC) -Endpunkten oder bestimmten zu kontrollieren. VPCs Dadurch wird der Netzwerkzugriff auf eine bestimmte Amazon ECR-Ressource effektiv nur von der spezifischen VPC innerhalb des Netzwerks isoliert. AWS Weitere Informationen finden Sie unter VPC-Endpunkte mit Amazon ECR-Schnittstelle ()AWS PrivateLink.

VPC-Endpunkte mit Amazon ECR-Schnittstelle ()AWS PrivateLink

Sie können die Sicherheit Ihrer VPC verbessern, indem Sie Amazon ECR so konfigurieren, dass es einen Schnittstellen-VPC-Endpunkt verwendet. VPC-Endgeräte werden von einer Technologie unterstützt AWS PrivateLink, mit der Sie privat APIs über private IP-Adressen (sowohl als auch IPv4) auf Amazon ECR zugreifen können. IPv6 AWS PrivateLink schränkt den gesamten Netzwerkverkehr zwischen Ihrer VPC und Amazon ECR auf das Amazon-Netzwerk ein. Sie benötigen kein Internet-Gateway, kein NAT-Gerät und kein Virtual Private Gateway.

Weitere Informationen zu AWS PrivateLink VPC-Endpunkten finden Sie unter VPC-Endpunkte im Amazon VPC-Benutzerhandbuch.

Überlegungen für Amazon ECR VPC-Endpunkte

Bevor Sie VPC-Endpunkte für Amazon ECR konfigurieren, sollten Sie die folgenden Punkte beachten.

- Damit Ihre auf EC2 Amazon-Instances gehosteten Amazon ECS-Aufgaben private Images von Amazon ECR abrufen können, erstellen Sie die VPC-Endpunkte der Schnittstelle für Amazon ECS. Weitere Informationen finden Sie unter Interface VPC Endpoints (AWS PrivateLink) im Amazon Elastic Container Service Developer Guide.
- Amazon ECS-Aufgaben, die auf Fargate gehostet werden und Container-Images von Amazon ECR beziehen, können den Zugriff auf die spezifische VPC, die ihre Aufgaben verwenden, und auf den VPC-Endpunkt, den der Dienst verwendet, beschränken, indem sie der IAM-Rolle für die Aufgabenausführung Bedingungsschlüssel hinzufügen. Weitere Informationen finden Sie unter Optionale IAM-Berechtigungen für Fargate-Aufgaben, die Amazon ECR-Images über Schnittstellenendpunkte abrufen im Amazon Elastic Container Service Entwicklerleitfaden.
- Die Sicherheitsgruppe für den VPC-Endpunkt müssen eingehende Verbindungen auf Port 443 aus dem privaten Subnetz der VPC zulassen.
- Amazon ECR VPC-Endpunkte unterstützen Dual-Stack- (IPv4 und) Konnektivität. IPv6 Wenn Sie einen Dual-Stack-VPC-Endpunkt erstellen, kann er den Datenverkehr IPv4 sowohl über private IP-Adressen als auch über IPv6 private IP-Adressen verarbeiten.
- VPC-Endpunkte unterstützen öffentliche Amazon ECR Repositorys über den AWS API-SDK-Endpunkt in USA Ost (Nord-Virginia).
- VPC-Endpunkte unterstützen nur AWS bereitgestelltes DNS über Amazon Route 53. Wenn Sie Ihre eigene DNS verwenden möchten, können Sie die bedingte DNS-Weiterleitung nutzen. Weitere Informationen finden Sie unter DHCP Options Sets im Amazon VPC-Benutzerhandbuch.

- Wenn Ihre Container über bestehende Verbindungen zu Amazon S3 verfügen, werden deren Verbindungen möglicherweise kurz unterbrochen, wenn Sie den Amazon S3-Gateway-Endpunkt hinzufügen. Wenn Sie diese Unterbrechung vermeiden möchten, erstellen Sie eine neue VPC, die den Amazon S3-Gateway-Endpunkt verwendet, und migrieren Sie dann Ihren Amazon ECS-Cluster und seine Container in die neue VPC.
- Wenn ein Image zum ersten Mal mit einer Pull-Through-Cache-Regel abgerufen wird und Sie Amazon ECR für die Verwendung eines Schnittstellen-VPC-Endpunkts mit AWS PrivateLink konfiguriert haben, müssen Sie in derselben VPC ein öffentliches Subnetz mit einem NAT-Gateway erstellen und leiten Sie dann den gesamten ausgehenden Datenverkehr von ihrem privaten Subnetz ins Internet zum NAT-Gateway, damit der Abruf funktioniert. Nachfolgende Image-Abrufe erfordern dies nicht. Weitere Informationen finden Sie unter Szenario: Zugriff auf das Internet aus einem privaten Subnetz im Benutzerhandbuch für Amazon Virtual Private Cloud.
- Für Workloads, die FIPS 140-3-validierte kryptografische Module erfordern, unterstützt Amazon ECR FIPS-Endpunkte für VPC-Endpunkte.

Überlegungen für Windows-Images

Images, die auf dem Windows-Betriebssystem basieren, enthalten Artefakte, die aufgrund von Lizenzbeschränkungen nicht weitergegeben werden dürfen. Wenn Sie Windows-Images in ein Amazon ECR-Repository übertragen, werden die Ebenen, die diese Artefakte enthalten, standardmäßig nicht übertragen, da sie als Fremdebenen betrachtet werden. Wenn die Artefakte von Microsoft bereitgestellt werden, werden die fremden Schichten von der Microsoft Azure-Infrastruktur abgerufen. Aus diesem Grund sind neben der Erstellung der VPC-Endpunkte weitere Schritte erforderlich, damit Ihre Container diese fremden Schichten von Azure beziehen können.

Es ist möglich, dieses Verhalten beim Pushen von Windows-Images auf Amazon ECR durch Verwendung des `--allow-nondistributable-artifacts`-Flags im Docker-Daemon außer Kraft zu setzen. Wenn dieses Flag aktiviert ist, werden die lizenzierten Ebenen zu Amazon ECR gepusht, wodurch diese Images von Amazon ECR über den VPC-Endpunkt abgerufen werden können, ohne dass ein zusätzlicher Zugriff auf Azure erforderlich ist.

> **Important**
>
> Die Verwendung des `--allow-nondistributable-artifacts`-Flags entbindet Sie nicht von der Verpflichtung, die Bedingungen der Windows-Container-Basis-Image-Lizenz

einzuhalten; Sie können keine Windows-Inhalte für die öffentliche Weitergabe oder die Weitergabe durch Dritte bereitstellen. Die Verwendung in Ihrer eigenen Umgebung ist erlaubt.

Um die Verwendung dieses Flags für Ihre Docker-Installation zu aktivieren, müssen Sie die Docker-Daemon-Konfigurationsdatei ändern, die je nach Docker-Installation in der Regel in den Einstellungen oder im Menü "Präferenzen" unter dem Abschnitt "Docker-Engine" oder durch direkte Bearbeitung der `C:\ProgramData\docker\config\daemon.json`-Datei konfiguriert werden kann.

Im Folgenden finden Sie ein Beispiel für die erforderliche Konfiguration. Ersetzen Sie den Wert durch den Repository-URI, an den Sie die Images weitergeben möchten.

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
    ]
}
```

Nachdem Sie die Konfigurationsdatei des Docker-Daemon geändert haben, müssen Sie den Docker-Daemon neu starten, bevor Sie versuchen, Ihr Image zu übertragen. Bestätigen Sie, dass der Push funktioniert hat, indem Sie überprüfen, ob die Basisebene in Ihr Repository gepusht wurde.

> **Note**
>
> Die Basisebenen für Windows-Images sind groß. Die Größe der Ebene führt zu einer längeren Push-Zeit und zusätzlichen Speicherkosten in Amazon ECR für diese Images. Aus diesen Gründen empfehlen wir, diese Option nur dann zu nutzen, wenn sie unbedingt erforderlich ist, um die Bauzeit und die laufenden Lagerkosten zu reduzieren. Das `mcr.microsoft.com/windows/servercore`-Image ist beispielsweise etwa 1,7 GiB groß, wenn es in Amazon ECR komprimiert wird.

Erstellen der VPC Endpunkte für Amazon ECR

Um die VPC-Endpunkte für den Amazon ECR-Service zu erstellen, verwenden Sie das Verfahren Erstellung eines Interface-Endpunktes im Amazon VPC Benutzerhandbuch.

Amazon ECR VPC-Endpunkte unterstützen Dual-Stack- (IPv4 und) Konnektivität. IPv6 Wenn Sie einen Dual-Stack-VPC-Endpunkt erstellen, verarbeitet er automatisch den Datenverkehr IPv4

sowohl über private IP-Adressen als auch über IPv6 private IP-Adressen. Der Endpunkt leitet den Datenverkehr mit der entsprechenden IP-Version weiter, die auf der Netzwerkkonfiguration Ihres Clients und den Funktionen des Endpunkts basiert.

Wenn Sie bereits VPC-Endpoints haben und IPv4 zu Dual-Stack-Endpoints migrieren möchten, können Sie Ihre vorhandenen Endpoints so ändern, dass sie Dual-Stack-Konnektivität unterstützen, oder neue Dual-Stack-Endpoints erstellen. Stellen Sie beim Erstellen oder Ändern von Endpoints sicher, dass Ihre VPC und Subnetze die IP-Version unterstützen, die Sie verwenden möchten. Nach der Erstellung von Dual-Stack-Endpunkten unterstützen die Endpunkte sowohl als auch. IPv4 IPv6

Amazon ECS-Aufgaben, die auf EC2 Amazon-Instances gehostet werden, erfordern sowohl Amazon ECR-Endpunkte als auch den Amazon S3-Gateway-Endpunkt.

Für Amazon ECS-Aufgaben, die auf Fargate mit der Plattformversion `1.4.0` oder höher gehostet werden, sind sowohl Amazon ECR VPC-Endpunkte als auch die Amazon S3-Gateway-Endpunkte erforderlich.

Auf Fargate gehostete Amazon ECS-Aufgaben, die die Plattformversion `1.3.0` oder eine frühere Version verwenden, benötigen nur die Datei com.amazonaws. ***region***.ecr.dkr Amazon ECR VPC-Endpunkt und Amazon S3 S3-Gateway-Endpunkte.

> **Note**
>
> Die Reihenfolge, in der die Endpunkte erstellt werden, ist unerheblich.

com.amazonaws. ***region***.ecr.dkr

Dieser Endpunkt wird für die Docker Registry verwendet. APIs Docker-Client-Befehle wie `push` und `pull` verwenden diesen Endpunkt.

Wenn Sie diesen Endpunkt erstellen, müssen Sie einen privaten DNS-Hostnamen aktivieren. Stellen Sie dazu sicher, dass die Option Privaten DNS-Namen aktivieren in der Amazon VPC-Konsole ausgewählt ist, wenn Sie den VPC-Endpunkt erstellen.

Verwenden Sie für FIPS 140-3-konforme Verbindungen den FIPS-Endpunktnamen com.amazonaws. ***region***.ecr-fips.dkr

com.amazonaws. ***region***.ecr.api

> **Note**
>
> Der angegebene *region* Wert steht für die Regionskennung für eine AWS Region, die von Amazon ECR unterstützt wird, z. B. `us-east-2` für die Region USA Ost (Ohio).

Verwenden Sie für FIPS 140-3-konforme Verbindungen die FIPS-Endpunktnamen: com.amazonaws. *region*.ecr-fips.dkr und com.amazonaws. ***region***.ecr-fips.api.

Dieser Endpunkt wird für Aufrufe an die Amazon ECR-API verwendet. API-Aktionen wie `DescribeImages` und `CreateRepository` betreffen diesen Endpunkt.

Wenn dieser Endpunkt erstellt wird, haben Sie die Möglichkeit, einen privaten DNS-Hostnamen zu aktivieren. Aktivieren Sie diese Einstellung, indem Sie Privaten DNS-Namen aktivieren in der VPC-Konsole auswählen, wenn Sie den VPC-Endpunkt erstellen. Wenn Sie einen privaten DNS-Hostnamen für den VPC-Endpunkt aktivieren, aktualisieren Sie Ihr SDK oder AWS CLI auf die neueste Version, sodass die Angabe einer Endpunkt-URL bei der Verwendung des SDK oder AWS CLI nicht erforderlich ist.

Verwenden Sie für FIPS 140-3-konforme Verbindungen den FIPS-Endpunktnamen com.amazonaws. ***region***.ecr-fips.api.

Wenn Sie einen privaten DNS-Hostnamen aktivieren und ein SDK oder eine AWS CLI Version verwenden, die vor dem 24. Januar 2019 veröffentlicht wurde, müssen Sie den Parameter verwenden, um die `--endpoint-url` Schnittstellenendpunkte anzugeben. Das folgende Beispiel zeigt das Format für die Endpunkt-URL.

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

Wenn Sie keinen privaten DNS-Hostnamen für den VPC-Endpunkt aktivieren, müssen Sie den `--endpoint-url`-Parameter verwenden und die VPC-Endpunkt-ID für den Schnittstellenendpunkt angeben. Das folgende Beispiel zeigt das Format für die Endpunkt-URL.

```
aws ecr create-repository --repository-name name --endpoint-url
 https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

Verwenden Sie für FIPS 140-3-konforme Verbindungen die FIPS-Endpunkt-URL:

```
aws ecr create-repository --repository-name name --endpoint-url https://api.ecr-fips.region.amazonaws.com
```

Erstellen Sie den Amazon S3-Gateway-Endpunkt

Damit Ihre Amazon ECS-Aufgaben private Images der von Amazon ECR abrufen können, müssen Sie einen Gateway-Endpunkt für Amazon S3 erstellen. Der Gateway-Endpunkt ist erforderlich, da Amazon ECR Amazon S3 zum Speichern Ihrer Image-Ebenen verwendet. Wenn Ihre Container-Images von Amazon ECR herunterladen, müssen sie auf Amazon ECR zugreifen, um das Image-Manifest zu erhalten, und dann auf Amazon S3, um die eigentlichen Image-Ebenen herunterzuladen. Nachfolgend ist der Amazon Resource Name (ARN) des Amazon S3-Buckets angegeben, der die Ebenen für jedes Docker-Image enthält.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

> **Note**
>
> Wenn Sie einen Dual-Stack-VPC-Endpunkt für Amazon ECR erstellen, müssen Sie auch einen Dual-Stack-Amazon S3-Gateway- oder Interface-Endpunkt erstellen. Einzelheiten finden Sie in der S3-Dokumentation.

Verwenden Sie das Verfahren Erstellen eines Gateway-Endpunkts im Amazon VPC Benutzerhandbuch, um den folgenden Amazon S3-Gateway-Endpunkt für Amazon ECR zu erstellen. Achten Sie bei der Erstellung des Endpunkts darauf, dass Sie die Route-Tabellen für Ihre VPC auswählen.

com.amazonaws. ***region***. 3

Der Amazon S3-Gateway-Endpunkt verwendet ein IAM-Richtliniendokument, um den Zugriff auf den Dienst zu beschränken. Die Vollzugriffs-Richtlinie kann verwendet werden, da alle Beschränkungen, die Sie in Ihren IAM-Aufgabenrollen oder anderen IAM-Benutzerrichtlinien festgelegt haben, weiterhin zusätzlich zu dieser Richtlinie gelten. Wenn Sie den Zugriff auf den Amazon S3-Bucket auf die für die Verwendung von Amazon ECR erforderlichen Mindestberechtigungen beschränken möchten, siehe .Mindestberechtigungen für Amazon S3-Buckets für Amazon ECR

Mindestberechtigungen für Amazon S3-Buckets für Amazon ECR

Der Amazon S3-Gateway-Endpunkt verwendet ein IAM-Richtliniendokument, um den Zugriff auf den Service zu beschränken. Um nur die minimalen Amazon S3-Bucket-Berechtigungen für Amazon ECR zuzulassen, beschränken Sie den Zugriff auf das Amazon S3-Bucket, das Amazon ECR verwendet, wenn Sie das IAM-Richtliniendokument für den Endpunkt erstellen.

In der folgenden Tabelle werden die von Amazon ECR benötigten Amazon S3-Bucket-Richtlinienberechtigungen beschrieben.

Berechtigung	Description
`arn:aws:s3:::prod-` *`region`*`-starport-layer-bucket/*`	Ermöglicht den Zugriff auf den Amazon S3-Bucket, der die Schichten für jedes Docker-Image enthält. Stellt den Regionsbezeichner für eine von Amazon ECR unterstützte AWS Region dar, z. B. `us-east-2` für die Region US East (Ohio).

Beispiel

Das folgende Beispiel veranschaulicht, wie der Zugriff auf die Amazon S3-Buckets, die für Amazon ECR-Vorgänge erforderlich sind, bereitgestellt wird.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

Erstellen Sie den CloudWatch Logs-Endpunkt

Amazon ECS-Aufgaben, die den Starttyp Fargate verwenden und eine VPC ohne Internet-Gateway verwenden, die auch den **`awslogs`** Protokolltreiber verwenden, um CloudWatch Protokollinformationen an Logs zu senden, erfordern, dass Sie die Datei com.amazonaws erstellen. ***`region`***VPC-Endpunkt der Schnittstelle .logs für CloudWatch Logs. Weitere Informationen finden Sie unter Verwenden von CloudWatch Protokollen mit VPC-Endpunkten der Benutzeroberfläche von Amazon Logs im Amazon CloudWatch Logs-Benutzerhandbuch.

Erstellen Sie eine Endpunktrichtlinie für Ihre Amazon ECR VPC-Endpunkte

Eine VPC-Endpunktrichtlinie ist eine IAM-Ressourcenrichtlinie, die Sie einem Endpunkt beim Erstellen oder Ändern des Endpunkts zuordnen. Wenn Sie bei der Erstellung eines Endpunkts keine Richtlinie AWS anhängen, hängt eine Standardrichtlinie für Sie an, die vollen Zugriff auf den Service ermöglicht. -Benutzerrichtlinien oder servicespezifische Richtlinien werden durch Endpunktrichtlinien nicht überschrieben oder ersetzt. Endpunktrichtlinien steuern unabhängig vom Endpunkt den Zugriff auf den angegebenen Service. Endpunktrichtlinien müssen im JSON-Format erstellt werden. Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon VPC User Guide.

Wir empfehlen, eine einzige IAM-Ressourcenrichtlinie zu erstellen und sie an beide Amazon ECR VPC-Endpunkte anzuhängen.

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für Amazon ECR. Diese Richtlinie ermöglicht es einer bestimmten IAM-Rolle, Images von Amazon ECR zu beziehen.

```
{
 "Statement": [{
  "Sid": "AllowPull",
  "Principal": {
   "AWS": "arn:aws:iam::1234567890:role/role_name"
  },
  "Action": [
   "ecr:BatchGetImage",
   "ecr:GetDownloadUrlForLayer",
                   "ecr:GetAuthorizationToken"
  ],
  "Effect": "Allow",
  "Resource": "*"
 }]
```

```
}
```

Das folgende Beispiel für eine Endpunktrichtlinie verhindert, dass ein bestimmtes Repository gelöscht wird.

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

Das folgende Beispiel für eine Endpunktrichtlinie vereint die beiden vorherigen Beispiele in einer einzigen Richtlinie.

```
{
 "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": "*"
   }
  ]
}
```

So ändern Sie die VPC-Endpunktrichtlinie für Amazon ECR

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Wenn Sie die VPC-Endpunkte für Amazon ECR noch nicht erstellt haben, siehe Erstellen der VPC Endpunkte für Amazon ECR.
4. Wählen Sie den Amazon-ECR-VPC-Endpunkt aus, dem Sie eine Richtlinie hinzufügen möchten, und wählen Sie die Registerkarte Richtlinie in der unteren Hälfte des Bildschirms.
5. Wählen Sie Richtlinie bearbeiten und nehmen Sie die Änderungen an der Richtlinie vor.
6. Wählen Sie Speichern, um die Änderung zu speichern.

Gemeinsam genutzte Subnetze

Sie können VPC-Endpunkte in Subnetzen, die mit Ihnen geteilt werden, nicht erstellen, beschreiben, ändern oder löschen. Sie können die VPC-Endpunkte jedoch in Subnetzen verwenden, die mit Ihnen geteilt werden.

Serviceübergreifende Confused-Deputy-Prävention

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine juristische Stelle, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine privilegiertere juristische Stelle zwingen kann, die Aktion auszuführen. Ein AWS dienstübergreifender Identitätswechsel kann zu einem Problem mit dem verwirrten Stellvertreter führen. Ein dienstübergreifender Identitätswechsel

kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der aufrufende Service kann manipuliert werden, um seine Berechtigungen zu verwenden, um Aktionen auf die Ressourcen eines anderen Kunden auszuführen, für die er sonst keine Zugriffsberechtigung haben sollte. Um dies zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Wir empfehlen die Verwendung der globalen Bedingungskontextschlüssel `aws:SourceArn` oder `aws:SourceAccount` in ressourcenbasierten Richtlinien, um die Berechtigungen, die Amazon ECR einem anderen Service erteilt, auf eine bestimmte Ressource zu beschränken. Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel `aws:SourceArn` mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel, `arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*`.

Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie beide globale Bedingungskontextschlüssel verwenden, um Berechtigungen einzuschränken.

Der `aws:SourceArn`-Wert muss ResourceDescription lauten.

Das folgende Beispiel zeigt, wie Sie die Kontextschlüssel `aws:SourceArn` und die `aws:SourceAccount` globalen Bedingungsschlüssel in einer Amazon ECR-Repository-Richtlinie verwenden können, um den AWS CodeBuild Zugriff auf die Amazon ECR-API-Aktionen zu ermöglichen, die für die Integration mit diesem Service erforderlich sind, und gleichzeitig das Problem des verwirrten Stellvertreters zu verhindern.

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"
            }
         }
      }
   ]
}
```

Amazon ECR-Überwachung

Sie können Ihre Amazon ECR-API-Nutzung mit Amazon überwachen CloudWatch, das Rohdaten aus Amazon ECR sammelt und zu lesbaren Metriken nahezu in Echtzeit verarbeitet. Diese Statistiken werden über einen Zeitraum von zwei Wochen aufgezeichnet, sodass Sie auf historische Informationen zugreifen und sich einen Überblick über Ihre API-Nutzung verschaffen können. Amazon ECR-Metrikdaten werden automatisch innerhalb von einer Minute CloudWatch an gesendet. Weitere Informationen zu CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Amazon ECR liefert Metriken, die auf Ihrer API-Nutzung für Autorisierungs-, Image-Push- und Image-Pull-Aktionen basieren.

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon ECR und Ihren AWS Lösungen. Wir empfehlen Ihnen, Überwachungsdaten aus den Ressourcen zu sammeln, aus denen Ihre AWS Lösung besteht, damit Sie einen etwaigen Fehler an mehreren Stellen leichter debuggen können. Bevor Sie mit der Überwachung von Amazon ECR beginnen, sollten Sie jedoch einen Überwachungsplan erstellen, der Antworten auf die folgenden Fragen enthält:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Der nächste Schritt besteht darin, eine Basislinie für die normale Amazon ECR-Leistung in Ihrer Umgebung zu erstellen, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Speichern Sie bei der Überwachung von Amazon ECR historische Überwachungsdaten, damit Sie sie mit neuen Leistungsdaten vergleichen, normale Leistungsmuster und Leistungsanomalien erkennen und Methoden zur Behebung von Problemen entwickeln können.

Themen

- Amazon ECR-Nutzungsberichte
- Amazon-ECR-Repository-Metriken
- Amazon ECR-Ereignisse und EventBridge
- Protokollierung von Amazon ECR-Aktionen mit AWS CloudTrail

Visualisierung Ihrer Service Quotas und Einstellung von Alarmen

Sie können die CloudWatch Konsole verwenden, um Ihre Servicekontingenten zu visualisieren und zu sehen, wie Ihre aktuelle Nutzung im Vergleich zu Servicekontingenten abschneidet. Sie können auch Alarme festlegen, damit Sie benachrichtigt werden, wenn Sie sich einem Kontingent nähern.

So visualisieren Sie ein Service Quotas und legen optional einen Alarm fest

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Auf der Registerkarte Alle Metriken wählen Sie Nutzung und dann Nach AWS Ressourcen.

   Die Liste der Service Quotas-Nutzungsmetriken wird angezeigt.
4. Aktivieren Sie das Kontrollkästchen neben einer der Metriken.

   Das Diagramm zeigt Ihre aktuelle Nutzung dieser AWS Ressource.
5. Gehen Sie wie folgt vor, um Service Quotas in das Diagramm aufzunehmen:

   a. Wählen Sie die Registerkarte Graphed metrics (Grafisch dargestellte Metriken) aus.

   b. Wählen Sie Math expression (Mathematischer Ausdruck), Start with an empty expression (Mit einem leeren Ausdruck beginnen). Geben Sie dann in der neuen Zeile unter Details **`SERVICE_QUOTA(m1)`** ein.

      Dem Diagramm wird eine neue Linie hinzugefügt, die Service Quotas für die in der Metrik dargestellten Ressource anzeigt.
6. Um Ihre aktuelle Nutzung als Prozentsatz des Kontingents anzuzeigen, fügen Sie einen neuen Ausdruck hinzu oder ändern Sie den aktuellen SERVICE_QUOTA-Ausdruck. Verwenden Sie für den neuen Ausdruck **`m1/60/SERVICE_QUOTA(m1)*100`**.
7. (Optional) Gehen Sie wie folgt vor, um einen Alarm festzulegen, der Sie benachrichtigt, wenn Sie sich Service Quotas nähern:

a. Wählen Sie in der **`m1/60/SERVICE_QUOTA(m1)*100`**-Zeile unter Aktionen das Alarmsymbol aus. Es sieht aus wie eine Glocke.

   Die Seite „Alarmerstellung“ wird angezeigt.

b. Vergewissern Sie sich unter Conditions (Bedingungen), dass der Threshhold type (Schwellenwert-Typ) Static (Statisch) ist und Whenever Expression1 ist auf Greater (Größer) festgelegt ist. Unter als geben Sie **`80`** ein. Dadurch wird ein Alarm ausgelöst, der in den Zustand ALARM übergeht, wenn die Nutzung 80 Prozent des Kontingents überschreitet.

c. Wählen Sie Weiter aus.

d. Auf der nächsten Seite können Sie ein Amazon SNS-Thema auswählen oder ein neues erstellen. Dieses Thema wird benachrichtigt, wenn der Alarm in den ALARM-Status wechselt. Wählen Sie anschließend Weiter.

e. Geben Sie auf der nächsten Seite einen Namen und eine Beschreibung für den Alarm ein und wählen Sie dann Next (Weiter).

f. Wählen Sie Alarm erstellen aus.

Amazon ECR-Nutzungsmetriken

Sie können CloudWatch Nutzungsmetriken verwenden, um einen Überblick über die Ressourcennutzung Ihres Kontos zu erhalten. Verwenden Sie diese Kennzahlen, um Ihre aktuelle Servicenutzung in CloudWatch Diagrammen und Dashboards zu visualisieren.

Die Nutzungsmetriken von Amazon ECR entsprechen den AWS Servicekontingenten. Sie können Alarme konfigurieren, mit denen Sie benachrichtigt werden, wenn sich Ihre Nutzung einem Servicekontingent nähert. Weitere Informationen über Amazon ECR Service Quotas finden Sie unter Amazon ECR Service Quotas.

Amazon ECR veröffentlicht die folgenden Metriken im Namespace `AWS/Usage`.

Metrik	Description
`CallCount`	Die Anzahl der API-Aktionsaufrufe von Ihrem Konto. Die Ressourcen werden durch die Dimensionen definiert, die der Metrik zugeordnet sind.

Metrik	Description
	Die nützlichste Statistik für diese Metrik ist `SUM`, mit der die Summe der Werte aller Mitwirkenden während der definierten Periode dargestellt wird.

Die folgenden Dimensionen werden verwendet, um die von Amazon ECR veröffentlichten Nutzungsmetriken zu verfeinern.

Dimension	Description
`Service`	Der Name des AWS Dienstes, der die Ressource enthält. Für Amazon ECR-Nutzungsmetriken lautet der Wert für diese Dimension `ECR`.
`Type`	Der Typ von Entität, die gemeldet wird. Derzeit ist der einzige gültige Wert für Amazon ECR-Nutzungsmetriken `API`.
`Resource`	Der Typ der Ressource, die ausgeführt wird. Derzeit liefert Amazon ECR Informationen über Ihre API-Nutzung für die folgenden API-Aktionen. • `GetAuthorizationToken` • `BatchCheckLayerAvailability` • `InitiateLayerUpload` • `UploadLayerPart` • `CompleteLayerUpload` • `PutImage` • `BatchGetImage` • `GetDownloadUrlForLayer`
`Class`	Die Klasse der nachverfolgten Ressource. Derzeit verwendet Amazon ECR die Klassendimension nicht.

Amazon ECR-Nutzungsberichte

AWS bietet ein kostenloses Berichtstool namens Cost Explorer, mit dem Sie die Kosten und die Nutzung Ihrer Amazon ECR-Ressourcen analysieren können.

Verwenden Sie den Cost Explorer, um Diagramme Ihrer Nutzung und Kosten anzuzeigen. Sie können die Daten der vorherigen 13 Monate anzeigen und prognostizieren, wie viel Sie wahrscheinlich für die nächsten drei Monate ausgeben werden. Sie können den Cost Explorer verwenden, um Muster in Ihren Ausgaben für AWS -Ressourcen im Verlauf der Zeit zu sehen, Bereiche zu identifizieren, die eine genauere Untersuchung erfordern, und Trends auszumachen, die Ihnen helfen, Ihre Kosten zu verstehen. Sie können auch Zeitbereiche für die Daten angeben und die Daten nach Tagen oder Monate anzeigen lassen.

Die Messdaten in Ihren Kosten- und Nutzungsberichten zeigen die Nutzung in allen Ihren Amazon ECR-Repositories. Weitere Informationen finden Sie unter Markieren von Ressourcen für die Fakturierung.

Weitere Informationen zur Erstellung eines AWS Kosten- und Nutzungsberichts finden Sie unter AWS Kosten- und Nutzungsbericht im AWS Billing Benutzerhandbuch.

Amazon-ECR-Repository-Metriken

Amazon ECR sendet Metriken zur Anzahl der Repository-Abrufe an Amazon CloudWatch. Amazon ECR-Metrikdaten werden automatisch CloudWatch in Zeitabständen von 1 Minute an gesendet. Weitere Informationen zu CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Themen

- CloudWatch Metriken aktivieren
- Verfügbare Metriken und Dimensionen
- Amazon ECR-Metriken mit der CloudWatch Konsole anzeigen

CloudWatch Metriken aktivieren

Amazon ECR sendet Repository-Metriken automatisch für alle Repositorys. Sie müssen keine manuellen Schritte unternehmen.

Verfügbare Metriken und Dimensionen

In den folgenden Abschnitten sind die Metriken und Dimensionen aufgeführt, die Amazon ECR an Amazon CloudWatch sendet.

Amazon-ECR-Metriken

Amazon ECR stellt Metriken bereit, mit denen Sie Ihre Repositorys überwachen können. Sie können die Pullcount messen.

Der `AWS/ECR`-Namespace enthält die folgenden Metriken.

`RepositoryPullCount`

Die Gesamtzahl der Pulls für die Images im Repository.

Gültige Dimensionen: `RepositoryName`.

Gültige Statistiken: Durschnitt, Minimum, Maximum, Summe, Datenstichproben. Die nützlichste Statistik ist Sum.

Unit: Integer.

Dimensionen für Amazon-ECR-Metriken

Amazon-ECR-Metriken verwenden den `AWS/ECR`-Namespace und stellen Metriken für folgende Dimensionen bereit.

`RepositoryName`

Diese Dimension filtert die Daten, die Sie für alle Container-Images in einem bestimmten Repository anfordern.

Amazon ECR-Metriken mit der CloudWatch Konsole anzeigen

Sie können die Amazon ECR-Repository-Metriken auf der CloudWatch Konsole anzeigen. Die CloudWatch Konsole bietet eine detaillierte und anpassbare Anzeige Ihrer Ressourcen. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Amazon ECR-Ereignisse und EventBridge

Amazon EventBridge ermöglicht es Ihnen, Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse im AWS Rahmen von Services werden EventBridge nahezu in Echtzeit übermittelt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind, durchzuführende automatisierte Aktionen einschließen, wenn sich für ein Ereignis eine Übereinstimmung mit einer Regel ergibt. Die folgenden Aktionen können beispielsweise automatisch ausgelöst werden:

- Ereignisse zu Protokollgruppen in CloudWatch Logs hinzufügen
- Eine AWS Lambda Funktion aufrufen
- Amazon EC2 Run Command aufrufen
- Weiterleiten des Ereignisses an Amazon Kinesis Data Streams
- Aktivierung einer AWS Step Functions Zustandsmaschine
- Benachrichtigen eines Amazon SNS-Themas oder einer Amazon SQS-Warteschlange

Weitere Informationen finden Sie unter Erste Schritte mit Amazon EventBridge im EventBridge Amazon-Benutzerhandbuch.

Beispielereignisse von Amazon ECR

Nachfolgend finden Sie ein Beispiel für Ereignisse aus Amazon ECR. Ereignisse werden auf bestmögliche Weise ausgegeben.

Ereignis für einen abgeschlossenen Image-Push

Das folgende Ereignis wird gesendet, wenn jeder Image-Push abgeschlossen ist. Weitere Informationen finden Sie unter Ein Docker-Image in ein privates Amazon ECR-Repository übertragen.

```
{
    "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"
    }
}
```

Ereignis für eine Pull-Through-Cache-Aktion

Das folgende Ereignis wird gesendet, wenn versucht wird, eine Pull-Through-Cache-Aktion auszuführen. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung.

```
{
    "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",
    }
}
```

Ereignis für einen abgeschlossenen Image-Scan (grundlegendes Scanning)

Wenn der grundlegende Scan für Ihre Registrierung aktiviert ist, wird das folgende Ereignis gesendet, wenn jeder Image-Scan abgeschlossen ist. Der `finding-severity-counts`-Parameter gibt

nur einen Wert für einen Schweregrad zurück, wenn ein solcher vorhanden ist. Wenn das Image beispielsweise keine Ergebnisse auf `CRITICAL`-Ebene enthält, wird keine kritische Zählung zurückgegeben. Weitere Informationen finden Sie unter Bilder auf Betriebssystemschwachstellen in Amazon ECR scannen.

> **Note**
>
> Weitere Informationen zu Ereignissen, die Amazon Inspector ausgibt, wenn das erweiterte Scannen aktiviert ist, finden Sie unter EventBridge Ereignisse, die zum erweiterten Scannen in Amazon ECR gesendet wurden.

```
{
    "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": []
    }
}
```

Ereignis für eine Änderungsbenachrichtigung für eine Ressource mit aktiviertem verbessertem Scannen (erweitertes Scannen)

Wenn das erweiterte Scannen für Ihre Registrierung aktiviert ist, wird das folgende Ereignis von Amazon ECR gesendet, wenn es eine Änderung an einer Ressource gibt, für die das erweiterte

Scannen aktiviert ist. Dazu gehören neue Repositorys, die Untersuchungshäufigkeit für ein Repository, das geändert wird, oder wenn Images in Repositorys mit aktiviertem erweiterten Scannen erstellt oder gelöscht werden. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.


```
{
 "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"
 }
}
```


Ereignis für eine Image-Löschung

Das folgende Ereignis wird gesendet, wenn ein Image gelöscht wird. Weitere Informationen finden Sie unter Löschen eines Bilds in 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"
    }
}
```


Ereignis für eine Bildarchivierungsaktion

Das folgende Ereignis wird gesendet, wenn ein Bild archiviert wird. Das `target-storage-class` Feld wird auf gesetzt`ARCHIVE`. Das Ereignis umfasst die Medientypen Manifest und Artifact, mit denen der Typ des archivierten Inhalts identifiziert werden kann.


```
{
    "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"
    }
}
```

Ereignis für eine Image-Wiederherstellungsaktion

Das folgende Ereignis wird gesendet, wenn ein archiviertes Image wiederhergestellt wird. Das `target-storage-class` Feld wird auf gesetztSTANDARD. Das Ereignis enthält ein `last-activated-at` Feld, das anzeigt, wann das Bild zuletzt wiederhergestellt wurde.

```
{
    "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"
    }
}
```

Ereignis für eine Aktion zur Wiederherstellung des Referrers

Das folgende Ereignis wird gesendet, wenn ein archivierter Referrer (Referenzartefakt wie eine SBOM, Signatur oder Bescheinigung) wiederhergestellt wird. Beachten Sie, dass das dazu dient, `detail-type` es von normalen `ECR Referrer Action` Bildaktionen zu unterscheiden. Die `artifact-media-type` Felder `manifest-media-type` und identifizieren den spezifischen Typ des Referrers, der wiederhergestellt wird. In diesem Beispiel wird ein SBOM-Artefakt wiederhergestellt.

```
{
    "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"
    }
}
```

Ereignis für eine abgeschlossene Image-Replikation

Das folgende Ereignis wird gesendet, wenn jede Image-Replikation abgeschlossen ist. Weitere Informationen finden Sie unter Replikation privater Images in 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"
  }
}
```

Ereignis für eine fehlgeschlagene Image-Replikation

Das folgende Ereignis wird gesendet, wenn eine Bildreplikation fehlschlägt. Das `result` Feld enthält zusätzliche Fehlerinformationen `FAILED` und kann in den Ereignisdetails enthalten sein.

```
{
  "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"
  }
}
```

Protokollierung von Amazon ECR-Aktionen mit AWS CloudTrail

Amazon ECR ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Amazon ECR ausgeführt wurden. CloudTrail erfasst die folgenden Amazon ECR-Aktionen als Ereignisse:

- Alle API-Aufrufe, einschließlich Aufrufen von der Amazon ECR-Konsole

- Alle Aktionen, die aufgrund der Verschlüsselungseinstellungen in Ihren Repositories durchgeführt werden
- Alle Aktionen, die aufgrund von Lebenszyklus-Richtlinienregeln durchgeführt werden, einschließlich erfolgreicher sowie erfolgloser Aktionen

> **Important**
>
> Aufgrund der Größenbeschränkungen einzelner CloudTrail Ereignisse sendet Amazon ECR bei Lebenszyklus-Richtlinienaktionen, bei denen 10 oder mehr Bilder abgelaufen sind, mehrere Ereignisse an CloudTrail. Darüber hinaus enthält Amazon ECR maximal 100 Tags pro Image.

Wenn ein Trail erstellt wird, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon ECR. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand dieser Informationen können Sie feststellen, welche Anfrage an Amazon ECR gestellt wurde, von welcher IP-Adresse sie ausging, wer die Anfrage gestellt hat, wann sie gestellt wurde und weitere Details.

Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.

Amazon ECR-Informationen in CloudTrail

CloudTrail ist für Ihr AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn eine Aktivität in Amazon ECR auftritt, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Amazon ECR, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole erstellen, können Sie den Trail auf eine einzelne Region oder auf alle Regionen anwenden. Der Trail protokolliert Ereignisse in der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste so konfigurieren, dass sie die in den CloudTrail Protokollen gesammelten Ereignisdaten analysieren und darauf reagieren. Weitere Informationen finden Sie unter:

- Einen Trail für dein AWS Konto erstellen
- AWS Serviceintegrationen mit Protokollen CloudTrail
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle Amazon ECR API-Aktionen werden von der Amazon Elastic Container Registry API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Wenn Sie allgemeine Aufgaben ausführen, werden in den CloudTrail Protokolldateien Abschnitte für jede API-Aktion generiert, die Teil dieser Aufgabe ist. Wenn Sie beispielsweise ein Repository erstellen`GetAuthorizationToken`, `CreateRepository` werden `SetRepositoryPolicy` Abschnitte in den CloudTrail Protokolldateien generiert. Wenn Sie ein Image in ein Repository pushen, werden `InitiateLayerUpload`-, `UploadLayerPart`-, `CompleteLayerUpload`- und `PutImage`-Abschnitte generiert. Bei einem Abrufen des Images werden `GetDownloadUrlForLayer` und `BatchGetImage`-Abschnitte generiert. Wenn Sie ein Bild archivieren oder wiederherstellen, wird ein `UpdateImageStorageClass` Abschnitt generiert. Wenn OCI Clients, die die OCI 1.1 Spezifikation unterstützen, die Liste der Referrer oder Referenzartefakte für ein Bild mithilfe der Referrer-API abrufen, wird ein Ereignis ausgelöst. `ListImageReferrers` CloudTrail Beispiele für diese gängigen Aufgaben finden Sie unter CloudTrail Beispiele für Protokolleinträge.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder -Benutzeranmeldeinformationen ausgeführt wurde.
- Ob die Anfrage mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen föderierten Benutzer ausgeführt wurde
- Ob die Anfrage von einem anderen Dienst gestellt wurde AWS

Weitere Informationen hierzu finden Sie unter dem CloudTrail-Element `userIdentity`.

Verstehen der Amazon ECR-Protokolldateieinträge

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion,

Anforderungsparameter und andere Informationen. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

CloudTrail Beispiele für Protokolleinträge

Im Folgenden finden Sie Beispiele für CloudTrail Protokolleinträge für einige häufig vorkommende Amazon ECR-Aufgaben.

Diese Beispiele wurden für eine bessere Lesbarkeit formatiert. In einer CloudTrail Protokolldatei sind alle Einträge und Ereignisse in einer einzigen Zeile zusammengefasst. Darüber hinaus wurde dieses Beispiel auf einen einzigen Amazon ECR-Eintrag beschränkt. In einer echten CloudTrail Protokolldatei sehen Sie Einträge und Ereignisse von mehreren Diensten. AWS

> **Important**
>
> Die Quelle IPAddress ist die IP-Adresse, von der aus die Anfrage gestellt wurde. Bei Aktionen, die von der Servicekonsole ausgehen, bezieht sich die angegebene Adresse auf Ihre zugrunde liegende Ressource, nicht auf den Konsolen-Webserver. Für Dienste in AWS wird nur der DNS-Name angezeigt. Wir werten die Authentifizierung immer noch anhand der Quell-IP des Clients aus, auch wenn sie so bearbeitet wurde, dass sie dem AWS DNS-Namen dient.

Themen

Beispiel: Repository-Aktion erstellen

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die Aktion demonstriert. `CreateRepository`


```
{
    "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"
}
```

Beispiel: AWS KMS**`CreateGrant`** API-Aktion beim Erstellen eines Amazon ECR-Repositorys

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die AWS KMS `CreateGrant` Aktion beim Erstellen eines Amazon ECR-Repositorys mit aktivierter KMS-Verschlüsselung demonstriert. Für jedes Repository, das mit aktivierter KMS-Verschlüsselung erstellt wurde, sollten Sie zwei `CreateGrant` Protokolleinträge sehen. 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"
}
```

Beispiel: Aktion zum Pushen eines Images

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der einen Image-Push demonstriert, der die `PutImage` Aktion verwendet.

> **Note**
>
> Wenn Sie ein Bild übertragen, werden Sie in den CloudTrail Protokollen auch `CompleteLayerUpload` Verweise auf `InitiateLayerUploadUploadLayerPart`, und sehen.


```
{
    "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"
}
```

Beispiel: Aktion zum Abrufen eines Images

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der einen Image-Pull demonstriert, bei dem die `BatchGetImage` Aktion verwendet wird.

> **Note**
>
> Wenn Sie die Pull-Übertragung eines Image durchführen und das Image nicht lokal vorhanden ist, erscheinen in den CloudTrail -Protokollen zudem `GetDownloadUrlForLayer`-Verweise.

```
{
    "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"
}
```


Beispiel: Image-Lebenszyklus-Richtlinien-Aktion

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der zeigt, wann ein Image aufgrund einer Lebenszyklus-Richtlinienregel abgelaufen ist. Dieser Ereignistyp kann durch Filtern nach `PolicyExecutionEvent` für das Ereignisnamensfeld gefunden werden.

Wenn Sie eine Lifecycle-Richtlinienvorschau testen, generiert Amazon ECR einen CloudTrail Protokolleintrag mit dem Feld für den Ereignisnamen von`DryRunEvent`, mit exakt derselben Struktur wie der`PolicyExecutionEvent`. Indem Sie den Namen des Ereignisses in ändern`DryRunEvent`, können Sie stattdessen nach Probelaufereignissen filtern.

> **Important**
>
> Aufgrund der Größenbeschränkungen einzelner CloudTrail Ereignisse sendet Amazon ECR bei Lebenszyklus-Richtlinienaktionen, bei denen 10 oder mehr Bilder abgelaufen sind, mehrere Ereignisse an CloudTrail. Darüber hinaus enthält Amazon ECR maximal 100 Tags pro Image.


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

Beispiel: Aktion zur Bildarchivierung

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der zeigt, dass ein Bild mithilfe der `UpdateImageStorageClass` Aktion archiviert wird, bei der die `targetStorageClass` Einstellung auf `ARCHIVE` gesetzt ist.

```
{
    "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"
}
```

Beispiel: Aktion zur Wiederherstellung eines Bilds

Die folgenden Beispiele zeigen CloudTrail Protokolleinträge, die zeigen, dass ein Image wiederhergestellt wird. Wenn Sie ein archiviertes Image wiederherstellen, werden zwei Ereignisse generiert:

1. Ein API-Aufrufereignis, wenn die Wiederherstellung initiiert wird
2. Ein Dienstereignis, wenn der asynchrone Wiederherstellungsvorgang abgeschlossen ist

API-Aufrufereignis (Initiierung der Wiederherstellung)

Das folgende Beispiel zeigt den ersten API-Aufruf zur Wiederherstellung eines Images mithilfe der `UpdateImageStorageClass` Aktion mit `targetStorageClass` set to`STANDARD`. Die Antwort zeigt den Image-Status als`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"
}
```

Serviceereignis (Abschluss der Wiederherstellung)

Das folgende Beispiel zeigt das Dienstereignis, das generiert wird, wenn der asynchrone Wiederherstellungsvorgang abgeschlossen ist. Dieser Ereignistyp kann durch Filtern nach `ImageActivationEvent` für das Ereignisnamensfeld gefunden werden. Der `serviceEventDetails` Abschnitt enthält das Wiederherstellungsergebnis und den endgültigen Image-Status.

```
{
    "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"
}
```

Beispiel: Aktion „Bild-Referrer“

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der zeigt, wann ein OCI 1.1 kompatibler Client mithilfe der API eine Liste von Referrern oder Referenzartefakten für ein Bild abruft. `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"
}
```

Amazon ECR mit einem AWS SDK verwenden

AWS Software Development Kits (SDKs) sind für viele beliebte Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK für Go	AWS SDK für Go Codebeispiele
AWS SDK für Java	AWS SDK für Java Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Codebeispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Codebeispiele
AWS SDK für .NET	AWS SDK für .NET Codebeispiele
AWS SDK für PHP	AWS SDK für PHP Codebeispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Codebeispiele
AWS SDK für Ruby	AWS SDK für Ruby Codebeispiele
AWS SDK für Rust	AWS SDK für Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK für Swift	AWS SDK für Swift Codebeispiele

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Provide feedback (Feedback geben) auswählen.

Codebeispiele für Amazon ECR mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon ECR mit einem AWS Software Development Kit (SDK) verwendet wird.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon ECR mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte

Hello Amazon ECR

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon ECR beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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()));
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
```

```
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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)
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS SDK für Python (Boto3).

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon ECR AWS SDKs
  - Hello Amazon ECR
  - Lernen Sie die Grundlagen von Amazon ECR mit einem AWS SDK kennen
  - Aktionen für Amazon ECR mit AWS SDKs
    - Verwendung CreateRepository mit einem AWS SDK oder CLI
    - Verwendung DeleteRepository mit einem AWS SDK oder CLI
    - Verwendung DescribeImages mit einem AWS SDK oder CLI
    - Verwendung DescribeRepositories mit einem AWS SDK oder CLI
    - Verwendung GetAuthorizationToken mit einem AWS SDK oder CLI

- Verwendung GetRepositoryPolicy mit einem AWS SDK oder CLI
- Verwendung ListImages mit einem AWS SDK oder CLI
- PushImageCmdMit einem AWS SDK verwenden
- Verwendung PutLifeCyclePolicy mit einem AWS SDK oder CLI
- Verwendung SetRepositoryPolicy mit einem AWS SDK oder CLI
- Verwendung StartLifecyclePolicyPreview mit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung von Amazon ECR AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Elastic Container Registry mit verwenden können AWS SDKs.

Beispiele

- Hello Amazon ECR
- Lernen Sie die Grundlagen von Amazon ECR mit einem AWS SDK kennen
- Aktionen für Amazon ECR mit AWS SDKs
  - Verwendung CreateRepository mit einem AWS SDK oder CLI
  - Verwendung DeleteRepository mit einem AWS SDK oder CLI
  - Verwendung DescribeImages mit einem AWS SDK oder CLI
  - Verwendung DescribeRepositories mit einem AWS SDK oder CLI
  - Verwendung GetAuthorizationToken mit einem AWS SDK oder CLI
  - Verwendung GetRepositoryPolicy mit einem AWS SDK oder CLI
  - Verwendung ListImages mit einem AWS SDK oder CLI
  - PushImageCmdMit einem AWS SDK verwenden
  - Verwendung PutLifeCyclePolicy mit einem AWS SDK oder CLI
  - Verwendung SetRepositoryPolicy mit einem AWS SDK oder CLI
  - Verwendung StartLifecyclePolicyPreview mit einem AWS SDK oder CLI

Hello Amazon ECR

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon ECR beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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()));
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}")
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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)
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS -SDK für Python (Boto3).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon ECR mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen von Amazon ECR mit einem AWS SDK kennen

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Amazon-ECR-Repository.
- Legen Sie Repository-Richtlinien fest.
- Repository abrufen URIs.
- Rufen Sie Amazon-ECR-Autorisierungs-Token ab.
- Legen Sie Richtlinien für den Lebenszyklus von Amazon-ECR-Repositorys fest.
- Pushen Sie ein Docker-Image in ein Amazon-ECR-Repository.
- Verifizieren Sie das Vorhandensein eines Images in einem Amazon-ECR-Repository.
- Listen Sie die Amazon-ECR-Repositorys für Ihr Konto auf und erhalten Sie detaillierte Informationen dazu.
- Löschen Sie Amazon-ECR-Repositorys.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
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.");
           }
       }
   }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
```

```
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
```

```
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
           Throwable cause = e.getCause();
           if (cause instanceof EcrException ex) {
               if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                   System.out.println("The Amazon ECR repository already exists,
moving on...");
                   DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                       .repositoryNames(repoName)
                       .build();
                   DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                   return
describeResponse.repositories().get(0).repositoryArn();
               } else {
                   throw new RuntimeException(ex);
               }
           } else {
               throw new RuntimeException(e);
           }
       }
   }
```

```
   /**
    * Deletes an ECR (Elastic Container Registry) repository.
    *
    * @param repoName the name of the repository to delete.
    * @throws IllegalArgumentException if the repository name is null or empty.
    * @throws EcrException if there is an error deleting the repository.
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " +
repoName + " repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
               }
           }
       });

       // Wait for the CompletableFuture to complete
       response.join();
   }
```

```
   private static DockerClient getDockerClient() {
       String osName = System.getProperty("os.name");
       if (osName.startsWith("Windows")) {
           // Make sure Docker Desktop is running.
           String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
           DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
           dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
       } else {
           dockerClient = DockerClientBuilder.getInstance().build();
       }
       return dockerClient;
   }

   /**
    * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
    *
    * @return the configured ECR asynchronous client.
    */
   private static EcrAsyncClient getAsyncClient() {

       /*
        The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
        and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
        It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
        provide a non-blocking, event-driven approach to HTTP requests and
responses.
        */
       SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
           .maxConcurrency(50)  // Adjust as needed.
           .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
           .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
           .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
           .build();

       ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
```

```
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
```

```
                    }
                }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
```

```
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null &&
!describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
```

```
        DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
```

```
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
           }
       });
       response.join();
   }

   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
```

```
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
       });

       GetRepositoryPolicyResponse result = response.join();
       return result != null ? result.policyText() : null;
   }

   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
```

```
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
            {
             "Version":"2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
                   throw (RepositoryPolicyNotFoundException) cause;
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
```

```
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
```

```
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
```

```
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
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.")
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
```

```
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
```

```
                    "action": {
                        "type": "expire"
                    }
                }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
```

```
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
```

```
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws RepositoryAlreadyExistsException if the repository exists.
    * @throws EcrException         if an error occurs while creating the
repository.
    */
   suspend fun createECRRepository(repoName: String?): String? {
       val request =
           CreateRepositoryRequest {
               repositoryName = repoName
           }

       return try {
           EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
               val response = ecrClient.createRepository(request)
               response.repository?.repositoryArn
           }
       } catch (e: RepositoryAlreadyExistsException) {
           println("Repository already exists: $repoName")
           repoName?.let { getRepoARN(it) }
       } catch (e: EcrException) {
           println("An error occurred: ${e.message}")
           null
       }
   }

   suspend fun getRepoARN(repoName: String): String? {
       // Fetch the existing repository's ARN.
       val describeRequest =
           DescribeRepositoriesRequest {
               repositoryNames = listOf(repoName)
           }
       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val describeResponse =
ecrClient.describeRepositories(describeRequest)
           return describeResponse.repositories?.get(0)?.repositoryArn
       }
   }
```

```
    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
```

```
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)
```

```
            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String =
 repoData?.repositoryUri?.split("/")?.get(0) ?: ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
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)
```

ECRWrapper Klasse, die Amazon ECR-Aktionen umschließt.

```
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
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Amazon ECR mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne Amazon ECR-Aktionen mit AWS SDKs ausführen. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Elastic Container Registry.

Beispiele

- Verwendung CreateRepository mit einem AWS SDK oder CLI
- Verwendung DeleteRepository mit einem AWS SDK oder CLI
- Verwendung DescribeImages mit einem AWS SDK oder CLI
- Verwendung DescribeRepositories mit einem AWS SDK oder CLI
- Verwendung GetAuthorizationToken mit einem AWS SDK oder CLI
- Verwendung GetRepositoryPolicy mit einem AWS SDK oder CLI
- Verwendung ListImages mit einem AWS SDK oder CLI
- PushImageCmdMit einem AWS SDK verwenden

- Verwendung PutLifeCyclePolicy mit einem AWS SDK oder CLI
- Verwendung SetRepositoryPolicy mit einem AWS SDK oder CLI
- Verwendung StartLifecyclePolicyPreview mit einem AWS SDK oder CLI

Verwendung **`CreateRepository`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRepository` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: So erstellen Sie ein Repository

Im folgenden Beispiel für `create-repository` wird ein Repository innerhalb des angegebenen Namespace im Standard-Registry für ein Konto erstellt.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Repositorys im Amazon-ECR-Benutzerhandbuch.

Beispiel 2: So erstellen Sie ein Repository, das mit der Unveränderlichkeit von Image-Tags konfiguriert ist

Im folgenden Beispiel für `create-repository` wird im Standard-Registry für ein Konto ein Repository erstellt, das für die Unveränderlichkeit von Tags konfiguriert ist.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Weitere Informationen finden Sie unter Veränderlichkeit von Image-Tags im Amazon-ECR-Benutzerhandbuch.

Beispiel 3: So erstellen Sie ein Repository, das mit einer Scan-Konfiguration konfiguriert ist

Im folgenden Beispiel für `create-repository` wird im Standard-Registry für ein Konto ein Repository erstellt, das für die Durchführung eines Scans auf Schwachstellen beim Pushen von Images konfiguriert ist.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
```

```
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Weitere Informationen finden Sie unter Scannen von Images im Amazon-ECR-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateRepositoryin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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);
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateRepositoryin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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
        }
    }
```

- Einzelheiten zur API finden Sie CreateRepositoryin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie CreateRepositoryin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteRepository`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRepository` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So löschen Sie ein Repository

Der folgende `delete-repository`-Beispielbefehl erzwingt das Löschen des angegebenen Repositorys im Standard-Registry für ein Konto. Das `--force`-Flag ist erforderlich, wenn das Repository Images enthält.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
    }
}
```

Weitere Informationen finden Sie unter Löschen eines Repositorys im Amazon-ECR-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteRepositoryin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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();
    }
```

- Einzelheiten zur API finden Sie DeleteRepositoryin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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")
        }
    }
```

- Einzelheiten zur API finden Sie DeleteRepositoryin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DeleteRepositoryin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeImages`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeImages` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie ein Image in einem Repository

Im folgenden Beispiel für `describe-images` werden Details zu einem Image im `cluster-autoscaler`-Repository mit dem Tag `v1.13.6` angezeigt.

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Ausgabe:

```
{
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
```

```
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeImagesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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();
    }
```

- Einzelheiten zur API finden Sie DescribeImagesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
    * 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.")
           }
       }
   }
```

- Einzelheiten zur API finden Sie DescribeImagesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DescribeImagesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeRepositories`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeRepositories` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie die Repositorys in einem Registry

In diesem Beispiel werden die Repositorys im Standard-Registry für ein Konto beschrieben.

Befehl:

```
aws ecr describe-repositories
```

Ausgabe:

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

- Einzelheiten zur API finden Sie DescribeRepositoriesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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();
    }
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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 ""
            }
        }
```

```
    }
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon ECR mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetAuthorizationToken`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAuthorizationToken` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie ein Autorisierungstoken für Ihr Standard-Registry ab

Mit dem folgenden `get-authorization-token`-Beispielbefehl wird ein Autorisierungstoken für Ihr Standard-Registry abgerufen.

```
aws ecr get-authorization-token
```

Ausgabe:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * 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();
    }
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetRepositoryPolicy`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetRepositoryPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie die Repository-Richtlinie für ein Repository ab

Im folgenden Beispiel für `get-repository-policy` werden Details zur Repository-Richtlinie für das `cluster-autoscaler`-Repository angezeigt.

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Ausgabe:

```
{
    "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}"
}
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * 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;
    }
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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
        }
    }
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListImages`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListImages` verwendet wird.

CLI

AWS CLI

So listen Sie die Images in einem Repository auf

Im folgenden Beispiel für `list-images` wird eine Liste der Images im `cluster-autoscaler`-Repository angezeigt.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Ausgabe:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListImagesin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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(())
}
```

- Einzelheiten zur API finden Sie ListImagesin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon ECR mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

PushImageCmdMit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie `PushImageCmd` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * 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();
    }
```

- Einzelheiten zur API finden Sie PushImageCmdin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * 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)
            }
        }
    }
```

- Einzelheiten zur API finden Sie PushImageCmdin der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon ECR mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`PutLifeCyclePolicy`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutLifeCyclePolicy` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Lebenszyklusrichtlinie

Im folgenden Beispiel für `put-lifecycle-policy` wird eine Lebenszyklusrichtlinie für das angegebene Repository im Standard-Registry für ein Konto erstellt.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Inhalt von `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
```

```
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Ausgabe:

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

Weitere Informationen finden Sie unter Lebenszyklusrichtlinien im Amazon-ECR-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutLifeCyclePolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

Beispiel, das eine Richtlinie für ein Ablaufdatum festlegt.

```
    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)
        )
```

- Einzelheiten zur API finden Sie PutLifeCyclePolicyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SetRepositoryPolicy`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetRepositoryPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So legen Sie die Repository-Richtlinie für ein Repository fest

Im folgenden Beispiel für `set-repository-policy` wird eine in einer Datei enthaltene Repository-Richtlinie an das `cluster-autoscaler`-Repository angehängt.

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Inhalt von `my-policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Ausgabe:

```
{
    "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}"
}
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * 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();
    }
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

Beispiel, das einer IAM-Rolle Download-Zugriff gewährt.

```
    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)
        )
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon ECR mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`StartLifecyclePolicyPreview`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartLifecyclePolicyPreview` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So erstellen Sie eine Lebenszyklus-Richtlinienvorschau

Im folgenden Beispiel für `start-lifecycle-policy-preview` wird eine Lebenszyklus-Richtlinienvorschau, die durch eine JSON-Datei definiert ist, für das angegebene Repository erstellt.

```
aws ecr start-lifecycle-policy-preview \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Inhalt von `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
```

```
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Ausgabe:

```
{
   "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"
}
```

- Einzelheiten zur API finden Sie StartLifecyclePolicyPreviewin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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();
    }
```

- Einzelheiten zur API finden Sie StartLifecyclePolicyPreviewin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * 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.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie StartLifecyclePolicyPreviewin der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon ECR mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Amazon ECR Service Quotas

Die folgende Tabelle enthält die Standard Service Quotas für Amazon Elastic Container Registry (Amazon ECR).

Name	Standard	Anpas	Description
Einfache Bildscans pro 24 Stunden	Jede unterstützte Region: 100 000	Nein	Die maximale Anzahl von Bildern, die innerhalb eines Zeitraums von 24 Stunden im aktuellen Konto und in der Region mithilfe des Standards cans gescannt werden können. Dieses Limit umfasst sowohl das Scannen im Push-Modu s als auch das manuelle Scannen.
Filter pro Regel in einer Replikati onskonfiguration	Jede unterstützte Region: 100	Nein	Die maximale Anzahl von Filtern pro Regel in einer Replikationskonfiguration.
Image pro Repository	Jede unterstützte Region: 100 000	Ja	Die maximale Anzahl von Images pro Repository.
Layer parts (Layer-Teile)	Jede unterstützte Region: 4.200	Nein	Die maximale Anzahl von Ebenenteilen. Dies gilt nur, wenn Sie Amazon ECR API-Aktionen direkt verwenden, um mehrteili ge Uploads für Image-Push-Vorgänge zu initiieren.

Name	Standard	Anpas	Description
Lifecycle policy length (Lebenszy klusrichtlinienlänge)	Jede unterstützte Region: 30.720	Nein	Die maximale Anzahl von Zeichen in einer Lebenszyklusrichtlinie.
Max. Layer-Segmentgröße	Jede unterstützte Region: 10	Nein	Die maximale Größe (MiB) eines Layer-Teils. Dies gilt nur, wenn Sie Amazon ECR API-Aktio nen direkt verwenden, um mehrteilige Uploads für Image-Push-Vorgänge zu initiieren.
Maximum layer size (Maximale Layer-Größe)	Jede unterstützte Region: 52 000	Nein	Die maximale Größe (MiB) einer Ebene.
Min. Layer-Segmentgröße	Jede unterstützte Region: 5	Nein	Die Mindestgröße (MiB) eines Layer-Teils. Dies gilt nur, wenn Sie Amazon ECR API-Aktio nen direkt verwenden, um mehrteilige Uploads für Image-Push-Vorgänge zu initiieren.
Pull-Through-Cache-Regeln pro Registry	Jede unterstützte Region: 50	Nein	Die maximale Anzahl von Pull-Through-Cache-Regeln.

Name	Standard	Anpas	Description
Rate der BatchCheckLayerAvailability Anfragen	Jede unterstützte Region: 1 000 pro Sekunde	Ja	Die maximale Anzahl von BatchCheckLayerAvailability Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Bei einem Image-Push in ein Repository wird bei jeder Image-Ebene überprüft, ob es zuvor hochgeladen wurde. Wenn es hochgeladen wurde, wird die Image-Ebene übersprungen.
Rate der BatchGetImage Anfragen	Jede unterstützte Region: 2 000 pro Sekunde	Ja	Die maximale Anzahl von BatchGetImage Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Wenn ein Bild abgerufen wird, wird die BatchGetImage API einmal aufgerufen, um das Image-Manifest abzurufen. Wenn Sie eine Erhöhung des Kontingents für diese API beantragen, überprüfen Sie auch Ihre GetDownloadUrlForLayer Nutzung.

Name	Standard	Anpas	Description
Rate der CompleteLayerUpload Anfragen	Jede unterstützte Region: 100 pro Sekunde	Ja	Die maximale Anzahl von CompleteLayerUploa d Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Wenn ein Bild übertragen wird, wird die CompleteL ayerUpload API für jede neue Bildebene einmal aufgerufen, um zu überprüfen, ob der Upload abgeschlossen ist.
Rate der GetAuthorizationToken Anfragen	Jede unterstützte Region: 500 pro Sekunde	Ja	Die maximale Anzahl von GetAuthorizationTo ken Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können.

Name	Standard	Anpas	Description
Rate der GetDownloadUrlForLayer Anfragen	Jede unterstützte Region: 3.000 pro Sekunde	Ja	Die maximale Anzahl von GetDownloadUrlForLayer Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Wenn ein Bild abgerufen wird, wird die GetDownloadUrlForLayer API einmal pro Bildebene aufgerufen, die noch nicht zwischengespeichert ist. Wenn Sie eine Erhöhung des Kontingents für diese API beantragen, überprüfen Sie auch Ihre BatchGetImage Nutzung.
Rate der InitiateLayerUpload Anfragen	Jede unterstützte Region: 100 pro Sekunde	Ja	Die maximale Anzahl von InitiateLayerUpload Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Wenn ein Bild übertragen wird, wird die InitiateLayerUpload API einmal pro Bildebene aufgerufen, die noch nicht hochgeladen wurde. Ob eine Bildebene hochgeladen wurde oder nicht, hängt von der BatchCheckLayerAvailability API-Aktion ab.

Name	Standard	Anpas	Description
Rate der PutImage Anfragen	Jede unterstützte Region: 10 pro Sekunde	Ja	Die maximale Anzahl von PutImage Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Wenn ein Bild übertragen wird und alle neuen Bildebenen hochgeladen wurden, wird die PutImage API einmal aufgerufen, um das Image-Manifest und die mit dem Bild verknüpften Tags zu erstellen oder zu aktualisieren.
Rate der UploadLayerPart Anfragen	Jede unterstützte Region: 500 pro Sekunde	Ja	Die maximale Anzahl von UploadLayerPart Anfragen, die Sie pro Sekunde in der aktuellen Region stellen können. Wenn ein Bild übertragen wird, wird jede neue Bildebene in Teilen hochgeladen und die UploadLayerPart API wird für jeden neuen Teil der Bildebene einmal aufgerufen.
Rate der Image-Scans	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von Image-Scans pro Image und pro 24 Stunden.

Name	Standard	Anpas	Description
Registered repositories (Registrierte Repositorys)	Jede unterstützte Region: 100 000	Ja	Die maximale Anzahl von Repositorys, die Sie in diesem Konto in der aktuellen Region erstellen können.
Rules per lifecycle policy (Regeln pro Lebenszyklusrichtlinie)	Jede unterstützte Region: 50	Nein	Die maximale Anzahl von Regeln in einer Lebenszyklusrichtlinie
Regeln pro Replikationskonfiguration	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von Regeln in einer Replikationskonfiguration.
Tags pro Image	Jede unterstützte Region: 1.000	Nein	Die maximale Anzahl von Tags pro Image
Eindeutige Ziele für alle Regeln in einer Replikationskonfiguration	Jede unterstützte Region: 25	Nein	Die maximale Anzahl von eindeutigen Zielen für alle Regeln in einer Replikationskonfiguration.

Verwalten Ihrer Amazon ECR Service Quotas in der AWS-Managementkonsole

Amazon ECR ist in Service Quotas integriert, einen AWS Service, mit dem Sie Ihre Kontingente von einem zentralen Ort aus einsehen und verwalten können. Weitere Informationen finden Sie unter Was ist Service Quotas? im Service Quotas-Benutzerhandbuch.

Service Quotas macht es einfach, den Wert aller Amazon ECR Service Quotas nachzuschlagen.

So zeigen Sie Amazon ECR Service Quotas an (AWS-Managementkonsole)

1. Öffnen Sie die Service Quotas-Konsole unter https://console.aws.amazon.com/servicequotas/.
2. Wählen Sie im Navigationsbereich AWS -Services.

3. Suchen Sie in der Liste der AWS -Services nach Amazon Elastic Container Registry (Amazon ECR) und wählen Sie diese aus.

   In der Liste der Service-Kontingente können Sie den Namen des Servicekontingents, den angewendeten Wert (falls verfügbar), das AWS Standardkontingent und die Frage, ob der Kontingentwert anpassbar ist, sehen.
4. Wählen Sie den Kontingentnamen, um zusätzliche Informationen zu einem Service Quota anzuzeigen, z. B. seine Beschreibung.

Informationen zur Beantragung einer Erhöhung der Quota finden Sie unter Beantragung einer Erhöhung der Quota im Service Quotas-Benutzerhandbuch.

Einen CloudWatch Alarm zur Überwachung der API-Nutzungsmetriken erstellen

Amazon ECR stellt CloudWatch Nutzungsmetriken bereit, die den AWS Servicekontingenten für alle APIs an der Registrierung beteiligten Personen entsprechen, die an der Authentifizierung, Image-Push und Image-Pull beteiligt sind. In der Service Quotas-Konsole können Sie Ihre Nutzung in einem Diagramm visualisieren und Alarme konfigurieren, die Sie warnen, wenn Ihre Nutzung eine Service Quota erreicht. Weitere Informationen finden Sie unter Amazon ECR-Nutzungsmetriken.

Gehen Sie wie folgt vor, um einen CloudWatch Alarm zu erstellen, der auf einer der Amazon ECR-API-Nutzungsmetriken basiert.

So erstellen Sie einen Alarm basierend auf Ihren Amazon ECR-Nutzungsquoten (AWS-Managementkonsole)

1. Öffnen Sie die Service Quotas-Konsole unter https://console.aws.amazon.com/servicequotas/.
2. Wählen Sie im Navigationsbereich AWS -Services.
3. Suchen Sie in der Liste der AWS Serives nach Amazon Elastic Container Registry (Amazon ECR) und wählen Sie diese aus.
4. Wählen Sie in der Liste Service Quotas die Amazon ECR-Nutzungsquota aus, für das Sie einen Alarm erstellen möchten.
5. Wählen Sie im Bereich Amazon CloudWatch Events-Alarme die Option Erstellen aus.
6. Wählen Sie bei Alarmschwellenwert den Prozentsatz des angewendeten Kontingentwerts aus, den Sie als Alarmwert festlegen möchten.

7. Geben Sie bei Alarmname einen Namen für den Alarm ein und wählen Sie dann Erstellen aus.

Amazon ECR-Fehlerbehebung

Dieses Kapitel hilft Ihnen bei der Suche nach Diagnoseinformationen für Amazon ECR und enthält Schritte zur Fehlerbehebung für häufig auftretende Probleme und Fehlermeldungen.

Themen

Behebung von Docker-Befehlen und Problemen bei der Verwendung von Amazon ECR

In einigen Fällen kann die Ausführung eines Docker-Befehls für Amazon ECR zu einer Fehlermeldung führen. Nachstehend finden Sie einige häufige Fehlermeldungen und mögliche Lösungen.

Themen

Docker-Protokolle enthalten keine erwarteten Fehlermeldungen

Um mit dem Debuggen von Docker-Problemen zu beginnen, aktivieren Sie zunächst die Docker-Debugging-Ausgabe auf dem Docker-Daemon, der auf Ihren Host-Instances ausgeführt wird. Wenn Sie Bilder verwenden, die aus Amazon ECR auf Amazon ECS-Container-Instances abgerufen wurden, finden Sie weitere Informationen unter Konfiguration der ausführlichen Ausgabe aus dem Docker-Daemon im Amazon Elastic Container Service Developer Guide.

Fehler: "Überprüfung des Dateisystems fehlgeschlagen" oder "404: Image nicht gefunden" beim Abrufen eines Images aus einem Amazon-ECR-Repository

Sie erhalten möglicherweise den Fehler `Filesystem verification failed`, wenn Sie den Befehl docker pull verwenden, um ein Image aus einem Amazon ECR-Repository mit Docker 1.9 oder höher zu pullen. Sie können den Fehler `404: Image not found` erhalten, wenn Sie Docker-Versionen vor 1.9 verwenden.

Nachfolgend finden Sie einige mögliche Ursachen und deren Erläuterungen.

Der lokale Datenträger ist voll.

- Wenn die lokale Festplatte, auf der Sie den Befehl docker pull ausführen, voll ist, kann sich der SHA-1-Hash, der für die lokale Datei berechnet wurde, von dem unterscheiden, der von Amazon ECR berechnet wurde. Vergewissern Sie sich, dass auf Ihrer lokalen Festplatte noch genügend freier Speicherplatz vorhanden ist, um das Docker-Image, das Sie pullen, zu speichern. Sie können alte Images auch löschen, um mehr Speicherplatz für neue freizusetzen. Mit dem Befehl docker images können Sie eine Liste aller lokal heruntergeladenen Docker-Images und deren Größe aufrufen.

Der Client kann aufgrund eines Netzwerkfehlers keine Verbindung zum Remote-Repository herstellen.

- Aufrufe eines Amazon ECR-Repositoriums erfordern eine funktionierende Verbindung zum Internet. Überprüfen Sie die Netzwerkeinstellungen und stellen Sie sicher, dass andere Tools und Anwendungen auf Ressourcen im Internet zugreifen können. Wenn Sie docker pull auf einer EC2 Amazon-Instance in einem privaten Subnetz arbeiten, stellen Sie sicher, dass das Subnetz über eine Route zum Internet verfügt. Verwenden Sie einen NAT-Server (Network Address Translation) oder ein verwaltetes NAT-Gateway.

- Derzeit erfordern Aufrufe an ein Amazon ECR-Repository auch einen Netzwerkzugang durch Ihre Unternehmensfirewall zu Amazon Simple Storage Service (Amazon S3). Wenn Ihre Organisation Firewall-Software oder ein NAT-Gerät verwendet, das Service-Endpunkte zulässt, stellen Sie sicher, dass die Amazon S3-Service-Endpunkte für Ihre aktuelle Region zugelassen sind.

- Wenn Sie Docker hinter einem HTTP-Proxy nutzen, können Sie die entsprechenden Proxy-Einstellungen für Docker konfigurieren. Weitere Informationen finden Sie unter HTTP proxy in der Docker-Dokumentation.

Fehler: "Filesystem Layer Verification Failed" beim Abrufen von Images aus Amazon ECR

Sie können die Fehlermeldung `image image-name not found` erhalten, wenn Sie Images mit dem Befehl docker pull pullen. Bei der Überprüfung der Docker-Protokolle wird vielleicht folgender Fehler angezeigt:

```
filesystem layer verification failed for digest sha256:2b96f...
```

Dieser Fehler zeigt an, dass eine oder mehrere der Ebenen für Ihr Image nicht heruntergeladet werden konnten. Nachfolgend finden Sie einige mögliche Ursachen und deren Erläuterungen.

Sie verwenden eine ältere Docker-Version.

> Dieser Fehler tritt in einem sehr kleinen Prozentsatz der Fälle auf, wenn eine ältere Version als Docker 1.10 verwendet wird. Führen Sie ein Upgrade des Docker-Clients auf Version 1.10 oder neuer aus.

Für den Client ist ein Netzwerk- oder Datenträgerfehler aufgetreten.

> Aufgrund eines vollen Datenträgers oder eines Netzwerkproblems konnten nicht alle Layer heruntergeladen werden; dies wurde bereits zuvor für die Meldung `Filesystem verification failed` dargelegt. Befolgen Sie die obigen Empfehlungen, um sicherzustellen, dass Ihr Dateisystem nicht voll ist und dass Sie den Zugriff auf Amazon S3 von Ihrem Netzwerk aus aktiviert haben.

HTTP 403-Fehler oder "keine grundlegenden Berechtigungsnachweise"-Fehler beim Pushen zum Repository

Gelegentlich kann ein `HTTP 403 (Forbidden)`-Fehler oder die Fehlermeldung `no basic auth credentials` vom Befehl docker push oder docker pull zurückgegeben werden, auch wenn Sie Docker erfolgreich für den Befehl aws ecr get-login-password authentifiziert haben. Für dieses Problem sind folgende Ursachen bekannt:

Sie haben die Authentifizierung für eine andere Region ausgeführt.

> Authentifizierungsanforderungen sind an bestimmte Regionen geknüpft und nicht regionenübergreifend verwendbar. Wenn Sie beispielsweise ein Autorisierungs-Token aus US

West (Oregon) erhalten, können Sie es nicht zur Authentifizierung gegenüber Ihren Repositories in US East (N. Virginia) verwenden. Stellen Sie zum Beheben des Problems sicher, dass Sie ein Authentifizierungs-Token aus derselben Region abgerufen haben, in der Ihr Repository vorhanden ist. Weitere Informationen finden Sie unter the section called “Registrierungsauthentifizierung”.

Sie haben sich authentifiziert, um in ein Repository zu pushen, für das Sie keine Berechtigung haben

Sie verfügen nicht über die erforderlichen Berechtigungen, um einen Push in das Repository durchzuführen. Weitere Informationen finden Sie unter Richtlinien für private Repositorys in Amazon ECR.

Ihr Token ist abgelaufen.

Standardmäßig laufen Autorisierungs-Token, die mit der Operation `GetAuthorizationToken` abgerufen wurden, nach 12 Stunden ab.

Im Programm zur Verwaltung von Anmeldeinformationen `wincred` liegt ein Fehler vor.

Einige Versionen von Docker für Windows verwenden einen Anmeldeinformationsmanager namens`wincred`, der den von generierten Docker-Anmeldebefehl nicht richtig verarbeitet aws ecr get-login-password (weitere Informationen finden Sie unter `CredsStoreFails` with private Repositories). Sie können den ausgegebenen Docker-Anmeldebefehl ausführen, aber wenn Sie versuchen, Images zu pushen oder zu pullen, schlagen diese Befehle fehl. Dieser Fehler lässt sich umgehen, indem Sie das `https://`-Schema aus dem Registrierungsargument des Docker-Anmeldebefehls entfernen, der eine Ausgabe des Befehls aws ecr get-login-password ist. Nachstehend finden Sie ein Beispiel für den Docker-Anmeldebefehl ohne HTTPS-Schema.

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Fehlersuche bei Amazon ECR-Fehlermeldungen

In einigen Fällen wird ein API-Aufruf, den Sie über die Amazon ECR-Konsole oder dann initiiert haben, mit einer Fehlermeldung AWS CLI beendet. Nachstehend finden Sie einige häufige Fehlermeldungen und mögliche Lösungen.

HTTP 429: Zu viele Anfragen oder ThrottleException

Möglicherweise erhalten Sie bei einer `429: Too Many Requests` oder mehreren Amazon ECR-Aktionen oder API-Aufrufen einen `ThrottleException` Fehler oder einen Fehler. Dies

deutet darauf hin, dass Sie einen einzelnen Endpunkt in Amazon ECR wiederholt über ein kurzes Intervall aufrufen und dass Ihre Anforderungen gedrosselt werden. Eine solche Drosselung wird vorgenommen, wenn die Aufrufe eines einzelnen Endpunkts durch denselben Benutzer einen festgelegten Grenzwert für einen bestimmten Zeitraum überschreiten.

Jedem API-Vorgang in Amazon ECR sind Ratendrosselungen zugeordnet. Beispielsweise liegt die Drosselung für die Aktion `GetAuthorizationToken` bei 20 Transaktionen pro Sekunde (TPS), mit einer maximal zulässigen Steigerung auf 200 TPS. In jeder Region erhält jedes Konto einen Bucket, in dem ein Guthaben von bis zu 200 `GetAuthorizationToken`-API-Transaktionen gespeichert werden kann. Dieses Guthaben wird mit einer Rate von 20 pro Sekunde aufgefüllt. Bei einem Bucket-Guthaben von 200 können Sie 200 `GetAuthorizationToken`-API-Transaktionen pro Sekunde in einer Sekunde ausführen und anschließend 20 Transaktionen pro Sekunde (unbegrenzt). Weitere Informationen zu den Ratenlimits für Amazon ECR finden Sie APIs unterAmazon ECR Service Quotas.

Zur Behebung von Drosselungsfehlern implementieren Sie eine Wiederholungsfunktion mit inkrementellem Backoff in den Code. Weitere Informationen finden Sie unter Verhalten bei Wiederholungsversuchen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. Eine weitere Option besteht darin, eine Erhöhung des Ratenlimits zu beantragen, was Sie über die Service Quotas Quota-Konsole tun können. Weitere Informationen finden Sie unterVerwalten Ihrer Amazon ECR Service Quotas in der AWS-Managementkonsole. .

HTTP 403: "User [arn] is not authorized to perform [operation]"

Möglicherweise erhalten Sie den folgenden Fehler, wenn Sie versuchen, eine Aktion mit Amazon ECR durchzuführen:

```
$ 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: *
```

Dies deutet darauf hin, dass Ihr Benutzer keine Berechtigungen für die Verwendung von Amazon ECR hat oder dass diese Berechtigungen nicht korrekt eingerichtet sind. Insbesondere, wenn Sie Aktionen gegen ein Amazon ECR-Repository durchführen, überprüfen Sie, ob dem Benutzer die Berechtigungen für den Zugriff auf dieses Repository gewährt wurden. Weitere Informationen zum Erstellen und Überprüfen von Berechtigungen für Amazon ECR finden Sie unter Identity and Access Management für Amazon Elastic Container Registry.

HTTP 404-Fehler: "Das Repository existiert nicht"

Wenn Sie ein noch nicht vorhandenes Docker Hub-Repository angeben, wird dieses von Docker Hub automatisch angelegt. Bei Amazon ECR müssen neue Repositories explizit erstellt werden, bevor sie verwendet werden können. So wird verhindert, dass aus Versehen neue Repositorys erstellt werden (z. B. aufgrund eines Tippfehlers). Außerdem wird auf diese Weise sichergestellt, dass den neuen Repositorys eine geeignete Sicherheitszugriffsrichtlinie zugewiesen wird. Weitere Informationen zum Erstellen von Repositorys finden Sie unter Private Repositories von Amazon ECR.

Fehler: Interaktive Anmeldung von einem Nicht-TTY-Gerät aus nicht möglich

Wenn Sie den Fehler `Cannot perform an interactive login from a non TTY device` erhalten, sollten die folgenden Schritte zur Fehlerbehebung hilfreich sein.

- Stellen Sie sicher, dass Sie AWS CLI Version 2 verwenden und dass auf Ihrem System keine widersprüchliche Version von AWS CLI Version 1 installiert ist. Weitere Informationen finden Sie unter Installieren oder Aktualisierung auf die neueste Version von AWS CLI.
- Stellen Sie sicher, dass Sie Ihre AWS CLI mit gültigen Anmeldeinformationen konfiguriert haben. Weitere Informationen finden Sie unter Installieren oder Aktualisierung auf die neueste Version von AWS CLI.
- Stellen Sie sicher, dass die Syntax Ihres AWS CLI Befehls korrekt ist.

Podman mit Amazon ECR verwenden

Durch die Verwendung Podman mit Amazon ECR können Unternehmen die Sicherheit und Einfachheit von nutzen und Podman gleichzeitig von der Skalierbarkeit und Zuverlässigkeit von Amazon ECR für die Container-Imageverwaltung profitieren. Indem sie die beschriebenen Schritte und Befehle befolgen, können Entwickler und Administratoren ihre Container-Workflows rationalisieren, die Sicherheit erhöhen und die Ressourcennutzung optimieren. Da die Containerisierung weiter an Dynamik gewinnt, bietet Amazon ECR eine robuste Podman und flexible Lösung für die Verwaltung und Bereitstellung containerisierter Anwendungen.

Verwenden von Podman zur Authentifizierung bei Amazon ECR

Vor der Interaktion mit Amazon ECR ist eine Authentifizierung erforderlich. Podman Dies kann erreicht werden, indem Sie den `aws ecr get-login-password` Befehl zum Abrufen eines Authentifizierungstokens ausführen und dann dieses Token zusammen mit dem `podman login` Befehl zur Authentifizierung bei Amazon ECR verwenden.

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

Verwenden des Amazon ECR Credential Helper mit Podman

Amazon ECR bietet einen Docker-Credential-Helper, der mit Podman funktioniert. Der Credential Helper erleichtert das Speichern und Verwenden von Docker-Anmeldeinformationen beim Push und Abrufen von Bildern an Amazon ECR. Informationen zu Installations- und Konfigurationsschritten finden Sie unter Amazon ECR Docker Credential Helper.

Important

Podman unterstützt die Spezifikation nur teilweise. docker-creds-helper Podman unterstützt das `credHelpers` Schlüsselwort in der Docker-Konfiguration, unterstützt das Schlüsselwort jedoch nicht. `credsStore`

Um den Amazon ECR Credential Helper mit Podman zu verwenden, konfigurieren Sie Ihre Docker-Konfigurationsdatei mit dem folgenden Format: `credHelpers`

```
{
    "credHelpers": {
```

```
        "public.ecr.aws": "ecr-login",
        "<aws_account_id>.dkr.ecr.<region>.amazonaws.com": "ecr-login"
    }
}
```

Die folgende `credsStore` Konfiguration wird von Podman nicht unterstützt:

```
{
    "credsStore": "ecr-login"
}
```

> **Note**
>
> Der Amazon ECR Docker Credential Helper unterstützt derzeit keine Multi-Faktor-Authentifizierung (MFA).

Mit Podman Bilder aus Amazon ECR abrufen

Nach erfolgreicher Authentifizierung können Container-Images mithilfe des `podman pull` Befehls mit der vollständigen Amazon ECR-Repository-URI aus Amazon ECR abgerufen werden.

```
podman pull aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Container für Amazon ECR ausführen mit Podman

Sobald das gewünschte Image abgerufen wurde, kann ein Container mit dem Befehl instanziiert werden. `podman run`

```
podman run -d aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

Bilder auf Amazon ECR übertragen mit Podman

Um ein lokales Bild an Amazon ECR zu übertragen, muss das Bild zunächst mit der Amazon ECR-Repository-URI gekennzeichnet werden. Anschließend kann der `podman push` Befehl verwendet werden`podman tag`, um das Bild in Amazon ECR hochzuladen.

```
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
```

Dokumentverlauf

Die folgende Tabelle beschreibt die wichtigsten Änderungen in der Dokumentation seit der letzten Version von Amazon ECR. Wir aktualisieren die Dokumentation regelmäßig, um das Feedback, das Sie uns senden, einzuarbeiten.

Änderungen	Beschreibung	Date
Von ECR verwaltetes Signieren	Amazon ECR unterstützt jetzt das Signieren von verwalteten Container-Images, um Ihre Sicherhei tslage zu verbessern und den betrieblichen Aufwand für die Einrichtung der Signatur zu vermeiden. Mit der Signierung von Container-Images können Sie überprüfen, ob Bilder aus vertrauenswürdigen Quellen stammen. Weitere Informationen finden Sie unter Verwaltetes Signieren.	21. November 2025
IPv6 Unterstützung für AWS PrivateLink (VPC-Endpunkte)	Unterstützung für Dual-Stack IPv4 - (und IPv6) Konnektivität für Amazon ECR VPC-Endpunkte hinzugefügt, die von. AWS PrivateLink Sie können jetzt Dual-Stack-VPC-Endpoints erstellen, die den Datenverkehr sowohl über private IP-Adressen als auch über IPv6 private IPv4 IP-Adressen verarbeiten. Weitere Informationen finden Sie unter VPC-Endpu nkte mit Amazon ECR-Schnittstelle ()AWS PrivateLink.	21. November 2025
Funktion „ECR-Archiv“	Amazon ECR hat Unterstützung für die Archivier ung von Bildern für die langfristige Aufbewahrung hinzugefügt. Weitere Informationen finden Sie unter Archivieren eines Bilds in Amazon ECR.	19. November 2025
Es wurde aktualisiert und bietet nun Unterstützung für Ausschlussmuster zur Unveränderlichkeit von Image-Tags	Amazon ECR hat die aktualisierten Funktionen zum Taggen von Bildern aktualisiert, sodass beim Erstellen und Aktualisieren von Repositorys Ausschlussmuster für die Unveränderlichkeit von Image-Tags berücksic htigt werden. Sie können jetzt Tags angeben, die auch dann aktualisiert werden können, wenn die	23. Juli 2025

Änderungen	Beschreibung	Date
	Unveränderlichkeit von Tags in einem Repository aktiviert ist, indem Sie Platzhaltermuster (wie,,, `dev-*`) definieren, um bestimmte Tags von den `latest` Unveränderlichkeitsregeln `v*.beta` auszuschließen und gleichzeitig die Unveränderlichkeit für alle anderen Tags beizubehalten. Weitere Informationen finden Sie unter Erstellen eines privaten Amazon ECR-Repositorys zum Speichern von Bildern.	
Verbessertes Bildscannen wurde aktualisiert, um Einblicke in die Bildnutzung zu erhalten	Amazon ECR hat die aktualisierten Funktionen für erweitertes Scannen von Bildern aktualisiert, sodass Sie jetzt besser nachvollziehen können, wie Bilder auf Amazon EKS und Amazon ECS verwendet werden. Weitere Informationen finden Sie unter Scannen von Images auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR.	16. Juni 2025
IPv6 unterstützen	Es wurde Unterstützung für Anfragen an Amazon ECR-Registries hinzugefügt, die sowohl IPv4 -only- als auch Dual-Stack-Endpunkte (IPv4 und) verwenden. IPv6 Weitere Informationen finden Sie unter Anfragen an Amazon ECR-Registries stellen.	30. April 2025
Unterstützung für private Amazon ECR-Registry hinzugefügt, um den Cache zu durchsuchen	Amazon ECR hat Unterstützung für die Erstellung von Pull-Through-Cache-Regeln für die private Amazon ECR-Registrierung hinzugefügt. Weitere Informationen erhalten Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung und Serviceverknüpfte Amazon-ECR-Rolle für Pull-Through-Cache.	12. März 2025

Änderungen	Beschreibung	Date
Unterstützung für die Festlegung des Geltungsbereichs von Registrierungsrichtlinien wurde hinzugefügt	Amazon ECR hat Unterstützung für die Konfiguration des Geltungsbereichs der Registrierungsrichtlinien für Ihre private Registrierung hinzugefügt. Weitere Informationen finden Sie unter Private Registry-Berechtigungen in Amazon ECR und Amazon ECR Private Registry.	23. Dezember 2024
Amazon EC2 Container RegistryPullOnly — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt, die Amazon ECR nur Pull-Berechtigungen gewährt.	10. Oktober 2024
Über den Docker/OCI-Client weitergeleitete Operationen in Ereignissen verweisen jetzt auf CloudTrail `ecr.amazonaws.com`	Der Wert `ecr.amazonaws.com` ersetzt `AWS Internal` die Felder Benutzeragent (`userAgent`) und Quell-IP-Adresse (`sourceIPAddress`) für Ereignisse, die mit Client-Endpunkten verknüpft sind. CloudTrail Docker/OCI Beispiele finden Sie unter Beispiel: Aktion zum Abrufen eines Images und Beispiel: Aktion zum Pushen eines Images.	1. Juli 2024
Beschreibung der neuen Amazon ECR-Servicerolle für Vorlagen zur Erstellung von Repositorys hinzugefügt.	Amazon ECR verwendet eine servicebezogene Rolle mit dem Namen AWSServiceRoleForECRTemplate, die Amazon ECR die Erlaubnis erteilt, in Ihrem Namen Aktionen durchzuführen, um Aktionen zur Erstellung von Repository-Vorlagen abzuschließen. Weitere Informationen finden Sie unter Mit dem Amazon ECR Service verknüpfte Rolle für Vorlagen zur Repository-Erstellung.	20. Juni 2024
Die `ECRTemplateServiceRolePolicy` serviceverknüpfte Rolle wurde hinzugefügt.	Die `ECRTemplateServiceRolePolicy` serviceverknüpfte Rolle wurde hinzugefügt. Weitere Informationen finden Sie unter `ECRTemplateServiceRolePolicy`.	20. Juni 2024

Änderungen	Beschreibung	Date
Regions- und kontenübergreifende Replikation wurde den China Regionen hinzugefügt.	Amazon ECR hat der Region China Unterstützung hinzugefügt, um zu filtern, welche Repositorys repliziert werden. Weitere Informationen finden Sie unter Replikation privater Images in Amazon ECR.	15. Mai 2024
GitLab Container-Registry zum Durchrufen von Cache-Regeln hinzugefügt	Amazon ECR hat Unterstützung für die Erstellung von Pull-Through-Cache-Regeln für die GitLab Container-Registry hinzugefügt. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung.	8. Mai 2024
Lebenszyklusrichtlinien in Amazon ECR unterstützen nach einem Update die Verwendung von Platzhaltern	Amazon ECR unterstützt jetzt die Verwendung von Platzhaltern in einer Lebenszyklusrichtlinie. Dazu wird der Parameter `tagPatternList` in einer Lebenszyklusrichtlinienregel verwendet. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.	18. Dezember 2023
Repository-Erstellungsvorlagen in Amazon ECR	Amazon ECR unterstützt jetzt Repository-Erstellungsvorlagen. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache-, Create-On-Push- oder Replikationsaktion erstellt wurden.	15. November 2023
Pull-Through-Cache von Amazon ECR hinzugefügt, unterstützt für authentifizierte Upstream-Registrierungen	Amazon ECR unterstützt jetzt die Verwendung von Upstream-Registrierungen, die eine Authentifizierung für Ihre Pull-Through-Cache-Regeln erfordern. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung.	15. November 2023

Änderungen	Beschreibung	Date
AWSECRPullThroughCache_ServiceRolePolicy— Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AWSECRPullThroughCache_ServiceRolePolicy` -Richtlinie neue Berechtigungen hinzugefügt. Diese Berechtigungen ermöglichen Amazon ECR, den verschlüsselten Inhalt eines Secrets-Manager-Secrets abzurufen. Dies ist erforderlich, wenn eine Pull-Through-Cache-Regel verwendet wird, um Images aus einer Upstream-Registrierung zwischenzuspeichern, für das eine Authentifizierung erforderlich ist.	15. November 2023
Amazon-ECR-Image-Signierung	Amazon ECR und AWS Signer zusätzliche Unterstützung für die Erstellung und Übertragung von Container-Image-Signaturen mithilfe des Notary-Clients. Weitere Informationen finden Sie unter Bilder in Amazon ECR signieren.	6. Juni 2023
Kubernetes-Container-Registry wurde hinzugefügt, um Cache-Regeln abzurufen	Amazon ECR hat Unterstützung für das Erstellen von Pull-Through-Cache-Regeln für das Kubernetes-Container-Registry hinzugefügt. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer privaten Amazon ECR-Registrierung.	1. Juni 2023
Unterstützung für erweiterte Scandauer von Amazon ECR	Amazon Inspector hat Unterstützung für die Einstellung der Dauer hinzugefügt, für die Ihre Repositorys überwacht werden, wenn erweitertes Scannen aktiviert ist. Weitere Informationen finden Sie unter Änderung der erweiterten Scandauer für Bilder in Amazon Inspector.	28. Juni 2022
Amazon ECR sendet Metriken zur Pull-Anzahl von Repositorys an Amazon CloudWatch	Amazon ECR sendet Metriken zur Anzahl der Repository-Abrufe an Amazon CloudWatch. Weitere Informationen finden Sie unter Amazon-ECR-Repository-Metriken.	6. Januar 2022

Änderungen	Beschreibung	Date
Erweiterte Replikationsunterstützung	Amazon ECR hat die Unterstützung für die Filterung der Repositories, die repliziert werden, erweitert. Weitere Informationen finden Sie unter Replikation privater Images in Amazon ECR.	21 September 2021
AWS verwaltete Richtlinien für Amazon ECR	Amazon ECR hat eine Dokumentation der AWS verwalteten Richtlinien hinzugefügt. Weitere Informationen finden Sie unter AWS verwaltete Richtlinien für Amazon Elastic Container Registry.	24 Juni 2021
Regions- und kontoüber greifende Replikation	Amazon ECR hat Unterstützung für die Konfiguration von Replikationseinstellungen für Ihre private Registrierung hinzugefügt. Weitere Informationen finden Sie unter Private Registrierungseinstellungen in Amazon ECR.	8 Dezember 2020
Unterstützung von OCI-Artefakten	Amazon ECR hat Unterstützung für das Pushen und Pullen von Open Container Initiative (OCI)-Artefakten hinzugefügt. Ein neuer Parameter `artifactMediaType` wurde der `DescribeImages` -API-Antwort hinzugefügt, um die Art des Artefakts anzugeben. Weitere Informationen finden Sie unter Übertragung eines Helm-Diagramms in ein privates Amazon ECR-Repository.	24. August 2020
Verschlüsselung im Ruhezustand	Amazon ECR hat Unterstützung für die Konfiguration der Verschlüsselung für Ihre Repositories mit serverseitiger Verschlüsselung mit vom Kunden verwalteten Schlüsseln, die in AWS Key Management Service (AWS KMS) gespeichert sind, hinzugefügt. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.	29. Juli 2020

Änderungen	Beschreibung	Date
Images mit mehreren Architekturen	Amazon ECR hat Unterstützung für die Erstellung und Übertragung von Docker-Manifestlisten hinzugefügt, die für Multi-Architektur-Images verwendet werden. Weitere Informationen finden Sie unter Übertragung eines Images mit mehreren Architekturen in ein privates Amazon ECR-Repository.	28. April 2020
Amazon ECR-Nutzungsmetriken	Amazon ECR hat CloudWatch Nutzungsmetriken hinzugefügt, die einen Überblick über die Ressourcennutzung Ihres Kontos bieten. Sie haben auch die Möglichkeit, CloudWatch Alarme sowohl in der Konsole als auch in der CloudWatch Service Quotas Quota-Konsole zu erstellen, um Benachrichtigungen zu erhalten, wenn sich Ihre Nutzung Ihrem zugewiesenen Servicekontingent nähert. Weitere Informationen finden Sie unter Amazon ECR-Nutzungsmetriken.	28. Februar 2020
Aktualisierte Amazon ECR Service Quotas	Die Amazon ECR Service Quotas wurden aktualisiert, um Kontingente pro API einzubeziehen. Weitere Informationen finden Sie unter Amazon ECR Service Quotas.	19. Februar 2020
`get-login-password` -Befehl hinzugefügt	Unterstützung für get-login-password wurde hinzugefügt. Dadurch wird eine einfache und sichere Methode zum Abrufen eines Autorisierungs-Tokens bereitgestellt. Weitere Informationen finden Sie unter Verwendung eines Autorisierungs-Tokens.	4. Feb 2020

Änderungen	Beschreibung	Date
Scannen von Images	Hinzufügung von Unterstützung für das Scannen von Images, das beim Identifizieren von Softwares chwachstellen in Ihren Container-Images hilft. Amazon ECR verwendet die Common Vulnerabilities and Exposures (CVEs) -Datenbank aus dem Open-Sour ce-Projekt CoreOS Clair und stellt Ihnen eine Liste der Scanergebnisse zur Verfügung. Weitere Informationen finden Sie unter Bilder auf Softwareschwachstellen in Amazon ECR scannen.	24. Okt. 2019
VPC-Endpunktrichtlinie	Unterstützung für die Einstellung einer IAM-Richtlinie auf den Amazon ECR-Schnittstelle VPC-Endpunkten wurde hinzugefügt. Weitere Informationen finden Sie unter Erstellen Sie eine Endpunktrichtlinie für Ihre Amazon ECR VPC-Endpunkte.	26. September 2019
Veränderlichkeit von Image-Tags	Zusätzliche Unterstützung für die Konfiguration eines Repository als unveränderlich, um zu verhindern, dass Image Tags überschrieben werden. Weitere Informationen finden Sie unter Verhinder n, dass Bild-Tags in Amazon ECR überschrieben werden.	25. Juli 2019
Schnittstellen-VPC -Endpunkte (AWS PrivateLink)	Unterstützung für die Konfiguration von VPC-Endpu nkten mit Schnittstelle hinzugefügt. AWS PrivateLink So können Sie eine private Verbindung zwischen Ihrer VPC und Amazon ECR herstellen, ohne dass ein Zugang über das Internet, eine NAT-Instance, eine VPN-Verbindung oder Direct Connect. Weitere Informationen finden Sie unter VPC-Endpu nkte mit Amazon ECR-Schnittstelle ()AWS PrivateLink.	25. Januar 2019

Änderungen	Beschreibung	Date
Ressourcen-Markierung	Amazon ECR unterstützt nun das Hinzufügen von Metadaten-Tags zu Ihren Repositorys. Weitere Informationen finden Sie unter Kennzeichnen eines privaten Repositorys in Amazon ECR.	18. Dez. 2018
Amazon ECR-Namen sänderung	Amazon Elastic Container Registry wird umbenannt (zuvor Amazon EC2 Container Registry).	21. Nov. 2017
Lebenszyklus-Richtlinien	Mit Amazon ECR-Lebenszyklusrichtlinien können Sie das Lebenszyklusmanagement von Images in einem Repository festlegen. Weitere Informationen finden Sie unter Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR.	11. Okt. 2017
Amazon ECR-Unterstützung für Docker-Image-Manifest 2, Schema 2	Amazon ECR unterstützt jetzt Docker Image Manifest V2 Schema 2 (verwendet mit Docker Version 1.10 und neuer). Weitere Informationen finden Sie unter Unterstützung des Container-Image-Manifestformats in Amazon ECR.	27. Jan. 2017
Amazon ECR Allgemeine Verfügbarkeit	Amazon Elastic Container Registry (Amazon ECR) ist ein verwalteter AWS Docker-Registrierungsservice, der sicher, skalierbar und zuverlässig ist.	21. Dez. 2015

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.