aws

Benutzerhandbuch

Amazon ECR

API-Version 2015-09-21

Amazon ECR: Benutzerhandbuch

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 So können bestimmte Benutzer oder Amazon EC2-Instanzen auf Ihre Container-Repositories 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.

Amazon ECR-Komponenten

Amazon ECR enthält die folgenden Komponenten:

Registrierung

Für jedes AWS Konto wird eine private Amazon ECR-Registrierung bereitgestellt. Sie können ein oder mehrere Repositorys in Ihrer Registrierung erstellen und Docker-Images, Open Container Initiative (OCI) -Images und OCI-kompatible Artefakte darin speichern. Weitere Informationen finden Sie unter Private Amazon-ECR-Registrierung.

Autorisierungs-Token

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 Amazon ECR-Image-Repository enthält Ihre Docker-Images, Open Container Initiative (OCI)-Images und OCI-kompatible Artefakte. 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.

Image

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.

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 Software-Sicherheitslücken in Amazon scannen ECR.
- 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 ECR privaten Amazon-Registrierung.

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 nur für die Datenmenge, die Sie in Ihren Repositories speichern, und für den Datentransfer aus Ihren Image-Pushes und Pulls. 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 Management Console 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 bereits Amazon EC2 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 Amazon-EC2-Instance mit einem Amazon-Linux-2023-AMI

1. Starten Sie eine Instance mit dem neuesten Amazon-Linux-2023-AMI. Weitere Informationen finden Sie unter Launching an Instance im Amazon EC2 EC2-Benutzerhandbuch.
2. Verbinden Sie sich mit der Instance. Weitere Informationen finden Sie unter Connect to Your Linux Instance im Amazon EC2 EC2-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 werden Sie ein Docker-Image einer einfachen Webanwendung erstellen und es auf Ihrem lokalen System oder einer Amazon-EC2-Instance testen.

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 Wert für Öffentliche DNS für den Server. Dabei handelt es sich um dieselbe Adresse, mit der Sie eine Verbindung mit der Instance per SSH herstellen. 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 Sie *machine-name 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: Authentifizieren der 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-ECR () LoginCommandAWS Tools for Windows PowerShell

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

Schritt 3: Erstellen eines Repositorys

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 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
   ```

   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 Amazon EC2-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.

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.us-west-2.amazonaws.com`.
- Standardmäßig hat Ihr Konto Lese- und Schreibzugriff auf die Repositories in Ihrer privaten Registrierung. Benutzer benötigen jedoch Berechtigungen, um die Amazon ECR-APIs aufzurufen und Bilder in 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 über Regionen in Ihrer eigenen privaten Registrierung und über separate Konten hinweg repliziert werden, indem Sie die Replikation für Ihre private Registrierung konfigurieren. Weitere Informationen finden Sie unter Replikation privater Bilder in Amazon ECR.

Authentifizierung bei privaten Registern in Amazon ECR

Sie können die SDKs AWS Management Console, das oder die AWS SDKs verwenden AWS CLI, um private Repositorys 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-ECR () LoginCommandAWS 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 Befehl bereitgestellte Autorisierungstoken übergeben. get-authorization-token AWS CLI

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 ermöglicht die Kontrolle über die Replikation und die Berechtigungen für den Pull-Through-Cache. Weitere Informationen finden Sie unter Private Registrierungsberechtigungen in Amazon ECR.
- Pull-Through-Cache-Regeln – Mit einer Pull-Through-Cache-Regel können Sie Images aus einer Upstream-Registrierung in Ihrer privaten Registrierung von Amazon ECR zwischenspeichern. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer ECR privaten Amazon-Registrierung.

- Replikationskonfiguration – Die Replikationskonfiguration wird verwendet, um zu steuern, ob Ihre Repositorys zwischen AWS -Regionen oder -Konten kopiert werden. Weitere Informationen finden Sie unter Replikation privater Bilder in Amazon ECR.
- Repository-Erstellungsvorlagen – Eine Repository-Erstellungsvorlage 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 erstellt wurden. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache- oder Replikationsaktion erstellt wurden.
- Konfiguration Scanning – Standardmäßig ist für Ihre Registrierung die grundlegende Überprüfung 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 Software-Sicherheitslücken in Amazon scannen ECR.

Private Registrierungsberechtigungen in Amazon ECR

Amazon ECR verwendet eine Registrierungsrichtlinie, um einem AWS -Prinzipal auf privater Registry-Ebene Berechtigungen zu erteilen. Diese Berechtigungen werden verwendet, um den Zugriff auf die Replikation und Pull-Through-Cache-Features zu erweitern.

Amazon ECR erzwingt die folgenden Berechtigungen nur auf privater Registrierungsebene. Wenn der Registrierungsrichtlinie zusätzliche Aktionen hinzugefügt werden, tritt ein Fehler auf.

- `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.

> **Note**
>
> Obwohl es möglich ist, einer Richtlinie für private Registrierungsberechtigungen die `ecr:*`-Aktion zuzufügen, wird es als bewährte Methode angesehen, nur die erforderlichen Aktionen basierend auf dem Feature hinzuzufügen, die Sie verwenden, 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 immer noch stattfinden, wenn die Aktion `ecr:CreateRepository` aus der Registrierungsberechtigungsanweisung entfernt wird. Für eine erfolgreiche Replikation müssen Sie jedoch Repositories mit demselben Namen innerhalb Ihres Kontos erstellen.

Beispiel: Erlauben Sie dem Root-Benutzer eines Quellkontos, alle Repositorys zu replizieren

Die folgende Richtlinie für Registrierungsberechtigungen ermöglicht es dem Root-Benutzer eines Quellkontos, alle Repositorys zu replizieren.

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

```
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_id:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/*"
            ]
        }
    ]
}
```

Beispiel: Erlaube Root-Benutzern von mehreren Konten

Die folgende Richtlinie für Registrierungsberechtigungen besteht aus zwei Aussagen. Jede Anweisung ermöglicht es dem Root-Benutzer eines Quellkontos, alle Repositorys zu replizieren.

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

```
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id:repository/*"
            ]
        }
    ]
}
```

Beispiel: Erlauben Sie dem Root-Benutzer eines Quellkontos, alle Repositories mit dem Präfix **`prod-`** zu replizieren.

Die folgende Richtlinie für Registrierungsberechtigungen ermöglicht es dem Root-Benutzer eines Quellkontos, alle Repositorys zu replizieren, die mit beginnen. `prod-`

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

Erteilen von Registrierungsberechtigungen für die kontoübergreifende Replikation in Amazon ECR

Der kontoübergreifende Richtlinientyp wird verwendet, um einem AWS -Prinzipal Berechtigungen zu erteilen und die Replikation der Repositorys von einer Quellregistrierung in Ihre Registrierung zu ermöglichen. 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 angezeigt CloudTrail.

So konfigurieren Sie eine Berechtigungsrichtlinie für die Replikation (AWS Management Console)

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, Registrierungsberechtigungen 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 Kontoübergreifende Richtlinie.
    b. Geben Sie für Auszugs-ID eine eindeutige Auszugs-ID ein. Dieses Feld wird als `Sid` für die Registrierungsrichtlinie verwendet.
    c. Geben Sie unter Konten die Konto-IDs für jedes Konto ein, dem Sie Berechtigungen erteilen möchten. Wenn Sie mehrere Konto-IDs angeben, trennen Sie diese durch ein Komma.
6. Erweitern Sie den Abschnitt Richtlinienvorschau, um die Richtlinie für Registrierungsberechtigungen zu überprüfen.

7. Nachdem Sie die Richtlinie bestätigt haben, wählen Sie Zu Richtlinie hinzufügen, um die Richtlinie in Ihrer Registrierung zu 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.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"ReplicationAccessCrossAccount",
            "Effect":"Allow",
            "Principal":{
                "AWS":"arn:aws:iam::source_account_id:root"
            },
            "Action":[
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-west-2:your_account_id: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 Management Console)

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`.

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 Aufrufe an die Amazon-ECR-APIs zu tätigen und Images zu und von 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.

Ein privates Amazon ECR-Repository zum Speichern von Bildern erstellen

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 Management Console zu erstellen. Anweisungen zum Erstellen eines Repositorys mithilfe von finden Sie AWS CLI unterSchritt 3: Erstellen eines Repositorys.

So erstellen Sie ein Repository (AWS Management Console)

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 auf der Seite Repositorys die Option Private Repositorys und dann Repository erstellen aus.
4. Stellen Sie bei den Sichtbarkeitseinstellungen sicher, dass Privat ausgewählt ist.
5. 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.

6. Wählen Sie für die Unveränderlichkeit von Tags die Einstellung für die Veränderlichkeit von Tags für das Repository. Repositories, die mit unveränderlichen Tags konfiguriert sind, verhindern,

dass Image-Tags überschrieben werden. Weitere Informationen finden Sie unter Verhindern, dass Bild-Tags in Amazon überschrieben werden ECR.

7. Obwohl Sie bei Scannen Sie bei Push 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 Software-Sicherheitslücken in Amazon scannen ECR.
8. Wählen Sie für die KMS-Verschlüsselung aus, ob die Verschlüsselung der Bilder im Repository mit aktiviert werden soll. AWS Key Management Service Wenn die KMS-Verschlüsselung aktiviert ist, verwendet Amazon ECR standardmäßig einen Von AWS verwalteter Schlüssel (KMS-Schlüssel) mit dem Alias`aws/ecr`. Dieser Schlüssel wird in Ihrem Konto erstellt, wenn Sie zum ersten Mal ein Repository mit aktivierter KMS-Verschlüsselung erstellen. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.
9. Wenn die KMS-Verschlüsselung aktiviert ist, wählen Sie Kundenverschlüsselungseinstellungen (erweitert), um Ihren eigenen KMS-Schlüssel auszuwählen. Der KMS-Schlüssel muss sich in der gleichen Region wie der Cluster befinden. Wählen Sie Create an AWS KMS Key, um zur AWS KMS Konsole zu navigieren und Ihren eigenen Schlüssel zu erstellen.
10. Wählen Sie Repository erstellen aus.

Nächste Schritte

Um die Schritte zum Pushen 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 ECR privates Amazon-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 Management Console 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 Management Console angezeigte Image zurückgibt.

So zeigen Sie Repository-Informationen (AWS Management Console) 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 anzeigen ECR.

     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 Ein Bild in Amazon löschen 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 Management Console, 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 Management Console)

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. Überprüfen Sie im Fenster Delete ***Repository-Name***, ob die ausgewählten Repositorys wirklich gelöscht werden sollen, und wählen Sie dann 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 Kontrolle des

Benutzerzugriffs auf verschiedenen 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.

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

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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeRepoImage",
            "Effect": "Allow",
            "Action": [
                "ecr:DescribeImages",
                "ecr:DescribeRepositories"
            ],
            "Resource": ["arn:aws:ecr:region:account-id: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 Kontrolle des Benutzerzugriffs auf verschiedenen 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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPushPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:user/push-pull-user-1",
                    "arn:aws:iam::account-id:user/push-pull-user-2"
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:GetDownloadUrlForLayer",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ]
        }
    ]
}
```

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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPush",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:root"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:CompleteLayerUpload",
                "ecr:InitiateLayerUpload",
                "ecr:PutImage",
                "ecr:UploadLayerPart"
            ]
        }
    ]
}
```

Die folgende Repository-Richtlinie ermöglicht es einigen Benutzern, Images abzurufen (*pull-user-1* und *pull-user-2*), während diese einem anderen Benutzer (*admin-user*) vollen Zugriff gewährt.

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPull",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id:user/pull-user-1",
                    "arn:aws:iam::account-id:user/pull-user-2"
```

```
                ]
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        },
        {
            "Sid": "AllowAll",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:user/admin-user"
            },
            "Action": [
                "ecr:*"
            ]
        }
    ]
}
```

Beispiel: Allen verweigern

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

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

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 gibt den `54.240.143.*`-Bereich der zulässigen IP-Adressen des Internetprotokoll-4-Adressen (IPv4) an.

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

```
{
    "Version": "2012-10-17",
    "Id": "ECRPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "ecr:*",
            "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 im Amazon ECR-Beispiel für CodeBuild im AWS CodeBuild Benutzerhandbuch.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:region: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 Management Console 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 Kontrolle des Benutzerzugriffs auf verschiedenen 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 Anweisung 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 Konto-IDs eine AWS Kontonummer an (z. B.111122223333), um die Abrechnung auf alle Benutzer eines bestimmten AWS Kontos anzuwenden. 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 unter Ihrem AWS Konto aus, auf die die Abrechnung angewendet werden soll.

  > **Note**
  >
  > Bei komplizierteren Repository-Richtlinien, die derzeit nicht in der unterstützt werden AWS Management Console, 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 ECR Amazon-Repositorys zu helfen, können Sie neuen oder bestehenden ECR Amazon-Repositorys mithilfe von AWS Ressourcen-Tags Ihre eigenen Metadaten zuweisen. Sie könnten beispielsweise eine Reihe von Tags für die ECR Amazon-Repositorys Ihres Kontos definieren, mit deren Hilfe Sie den Besitzer jedes Repositorys nachverfolgen können.

Grundlagen zu Tags (Markierungen)

Tags haben für Amazon keine semantische Bedeutung ECR und werden ausschließlich als Zeichenfolge 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 mit Tags über die ECR Amazon-Konsole AWS CLI, The und Amazon arbeiten ECRAPI.

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 Stichwörtern in IAM Richtlinien finden Sie unterthe section called “Verwenden Tag-basierter Zugriffskontrolle”.

Markieren von Ressourcen für die Fakturierung

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

1. Öffnen Sie die ECR Amazon-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 oderAPI)

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

- AWS CLI - Tag-Ressource
- APIAktion - 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 Management Console)

1. Öffnen Sie die ECR Amazon-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 löschen, indem Sie das AWS CLI oder ein verwendenAPI.

- AWS CLI - Untag-Ressource
- APIAktion - UntagResource

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

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

Private Bilder bei Amazon ECR

Amazon ECR speichert Docker-Images, Open Container Initiative (OCI) -Images und OCI kompatible Artefakte in privaten Repositorys. Sie können den Docker oder Ihren bevorzugten Client verwendenCLI, um Bilder in Ihre Repositorys zu übertragen und von dort abzurufen.

Mit ECR Amazon-Unterstützung für OCI v1.1 können Sie Referenzartefakte speichern und verwalten, die von den OCI APIReferrern definiert werden. Zu den Artefakten gehören Signaturen, Softwarelisten (SBoMs), Helm-Diagramme, 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 ECR privaten Amazon-Repository Seiten Signieren eines in einem ECR privaten Amazon-Repository gespeicherten Bildes 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 ECR privates Amazon-Repository übertragen

Sie können Ihre Docker-Images, Manifestlisten und Open Container Initiative (OCI) -Images und kompatiblen Artefakte per Push in Ihre privaten Repositorys übertragen.

Amazon bietet ECR 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.

Themen

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

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

Ein Benutzer muss sich bei jeder ECR Amazon-Registrierung, in die er Bilder übertragen möchte, authentifizieren, indem er ein Autorisierungstoken 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, um ein Bild in ein bestimmtes Repository zu übertragen. Das Repository muss als vollständiger Amazon-Ressourcenname (ARN) angegeben werden.

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

```
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage"
            ],
            "Resource": "arn:aws:ecr:region: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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CompleteLayerUpload",
                "ecr:GetAuthorizationToken",
                "ecr:UploadLayerPart",
                "ecr:InitiateLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

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

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

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

Um ein Docker-Image in ein ECR Amazon-Repository zu übertragen

Das ECR Amazon-Repository muss vorhanden sein, bevor Sie das Image übertragen können. Weitere Informationen finden Sie unter the section called “Erstellen eines Repositorys zum Speichern von Bildern”.

1. Authentifizieren Sie Ihren Docker-Client bei der ECR Amazon-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.

   Führen Sie den Befehl aus, um Docker bei einer ECR Amazon-Registrierung zu authentifizieren. aws ecr get-login-password Wenn Sie das Authentifizierungstoken an den docker login Befehl übergeben, verwenden Sie den Wert `AWS` für den Benutzernamen und geben Sie die ECR Amazon-Registrierung an, bei der URI 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 Registrierung existiert, in die Sie den Push durchführen wollen, erstellen Sie es. Weitere Informationen finden Sie unter Ein privates Amazon ECR-Repository zum Speichern von Bildern erstellen.
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 mit dem identifizieren *repository:tag* Wert oder die Bild-ID in der resultierenden Befehlsausgabe.

4. Kennzeichnen Sie Ihr Bild mit der zu verwendenden Kombination aus ECR Amazon-Registrierung, Repository und optionaler Image-Tag-Name. Die Registrierung hat das Format *aws_account_id*.dkr.ecr.*us-west-2*.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.

   Das folgende Beispiel kennzeichnet ein lokales Bild mit der ID *e9ae3c220b23* als*aws_account_id*.dkr.ecr.*us-west-2*.amazonaws.com/my-repository:tag.

   ```
   docker tag e9ae3c220b23 aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag
   ```

5. Pushen Sie das Image mit dem Befehl docker push:

   ```
   docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag
   ```

6. (Optional) Fügen Sie Ihrem Bild weitere Tags hinzu und übertragen Sie diese Tags an Amazon, ECR indem Sie Step 4 und wiederholenStep 5.

Ein Image mit mehreren Architekturen in ein ECR privates Amazon-Repository übertragen

Sie können Images mit mehreren Architekturen in ein ECR Amazon-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 wie andere ECR Amazon-Images abgerufen oder in einer ECS Amazon-Aufgabendefinition oder EKS Amazon-Pod-Spezifikation referenziert werden.

Voraussetzungen

- Aktivieren Sie in Ihrem Docker CLI experimentelle Funktionen. Informationen zu experimentellen Funktionen finden Sie unter Experimentelle Funktionen in der Docker-Dokumentation.
- Das ECR Amazon-Repository muss vorhanden sein, bevor Sie das Image übertragen können. 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 ECR privates Amazon-Repository verschieben.

Um ein Docker-Image mit mehreren Architekturen in ein Amazon-Repository zu übertragen ECR

1. Authentifizieren Sie Ihren Docker-Client bei der ECR Amazon-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.

   Führen Sie den Befehl aus, um Docker bei einer ECR Amazon-Registrierung zu authentifizieren. aws ecr get-login-password Wenn Sie das Authentifizierungstoken an den docker login Befehl übergeben, verwenden Sie den Wert `AWS` für den Benutzernamen und geben Sie die ECR Amazon-Registrierung an, bei der URI 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.us-west-2.amazonaws.com/my-repository aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:image_one_tag aws_account_id.dkr.ecr.us-west-2.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.us-west-2.amazonaws.com/my-repository
```

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

```
docker manifest push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository
```

Ein Helm-Diagramm in ein ECR privates Amazon-Repository verschieben

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

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

Um ein Helm-Diagramm in ein ECR Amazon-Repository zu übertragen

1. Installieren Sie die neueste Version des Helm-Clients. Diese Schritte wurden mit Helm Version `3.8.2` geschrieben. 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 ein ConfigMap im `templates` Ordner.

```
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
```

Output

```
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 Ein privates Amazon ECR-Repository zum Speichern von Bildern erstellen.

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

5. Authentifizieren Sie Ihren Helm-Client bei der ECR Amazon-Registrierung, in die Sie Ihr Helm-Diagramm 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.

```
aws ecr get-login-password \
     --region us-west-2 | helm registry login \
```

```
    --username AWS \
    --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
```

6. Drücken Sie die Steuerkarte mit dem Befehl helm push. Die Ausgabe sollte das ECR Amazon-Repository URI und den SHA Digest enthalten.

```
helm push helm-test-chart-0.1.0.tgz oci://aws_account_id.dkr.ecr.us-
west-2.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 die Helm-Configmap und beginnen Sie mit AmazonEKS. Weitere Informationen finden Sie unter Installation eines Helm-Diagramms auf einem Amazon EKS-Cluster.

Signieren eines in einem ECR privaten Amazon-Repository gespeicherten Bildes

Amazon ECR lässt sich integrieren AWS Signer , um Ihnen die Möglichkeit zu bieten, Ihre Container-Images zu signieren. Sie können sowohl Ihre Container-Images als auch die Signaturen in Ihren privaten Repositorys speichern.

Überlegungen

Folgendes sollte bei der Verwendung von Amazon ECR Image Signing beachtet werden.

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

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.
- Lassen Sie ein Container-Image zum Signieren in einem ECR privaten Amazon-Repository speichern. Weitere Informationen finden Sie unter Ein Bild in ein ECR privates Amazon-Repository übertragen.

Konfiguration der Authentifizierung für Notarkunden

Bevor Sie mithilfe der Notation eine Signatur erstellen könnenCLI, müssen Sie den 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, den Amazon ECR Docker Credential Helper als Ihren Anmeldeinformationsspeicher zu verwenden. Weitere Informationen zur Installation und Konfiguration von Amazon ECR Credential Helper finden Sie unter Amazon ECR Docker Credential Helper.

Signieren eines Images

Die folgenden Schritte können verwendet werden, um die Ressourcen zu erstellen, die zum Signieren eines Container-Images und zum Speichern der Signatur in einem ECR privaten Amazon-Repository erforderlich sind. Die Notation signiert Images mithilfe des Digest.

So signieren Sie ein Image

1. Erstellen Sie mithilfe der AWS Signer Signaturplattform ein `Notation-OCI-SHA384-ECDSA` Signaturprofil. Sie können optional eine Gültigkeitsdauer der Signatur mithilfe des Parameters `--signature-validity-period` angeben. Dieser Wert kann mit `DAYS`, `MONTHS` oder `YEARS` angegeben werden. Wenn kein Wert für den Gültigkeitszeitraum angegeben wird, wird der Standardwert 135 Monate verwendet.

   ```
   aws signer put-signing-profile --profile-name ecr_signing_profile --platform-id
    Notation-OCI-SHA384-ECDSA
   ```

   > **Note**
   >
   > Der Name des Signaturprofils unterstützt nur alphanumerische Zeichen und den Unterstrich (_).

2. Authentifizieren Sie den Notation-Client bei Ihrem Standard-Registry. Das folgende Beispiel verwendet die AWS CLI , um die Notation bei einer ECR privaten Amazon-Registrierung CLI zu authentifizieren.

```
aws ecr get-login-password --region region | notation login --username AWS --
password-stdin 111122223333.dkr.ecr.region.amazonaws.com
```

3. Verwenden Sie die Notation, CLI um das Bild zu signieren, und geben Sie das Bild anhand des Repository-Namens und des SHA Digest an. Dadurch wird die Signatur erstellt und in dasselbe ECR private Amazon-Repository übertragen, in dem sich das signierte Bild befindet.

   Im folgenden Beispiel signieren wir ein Bild im `curl` Repository mit SHA `sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE` Digest.

```
notation
 sign 111122223333.dkr.ecr.region.amazonaws.com/
curl@sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE --plugin
 "com.amazonaws.signer.notation.plugin" --id "arn:aws:signer:region:111122223333:/
signing-profiles/ecrSigningProfileName"
```

Nächste Schritte

Nachdem Sie Ihr Container-Image signiert haben, können Sie die Signatur lokal überprüfen. Anweisungen zur Überprüfung eines Images finden Sie unter Lokales Überprüfen eines Images nach dem Signieren im AWS Signer Entwicklerhandbuch.

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

Sie können den ORAS Client verwenden, um Signaturen und andere Artefakte vom Referenztyp aus einem ECR privaten Amazon-Repository aufzulisten und zu löschen. Das Löschen von Signaturen und anderen Referenzartefakten ähnelt dem Löschen eines Bilds (siheEin Bild in Amazon löschen ECR). So listen Sie Artefakte auf und löschen Signaturen:

Um Bildartefakte mit dem zu verwalten 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 ECR Amazon-Bild aufzulisten, verwenden Sie`oras discover`, gefolgt von einem Bildnamen:

```
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. Führen Sie den folgenden Befehl aus ORASCLI, um eine Signatur mithilfe des oben angegebenen Beispiels zu löschen:

```
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 anzeigen ECR

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

- Bild 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 Management Console)

1. Öffnen Sie die ECR Amazon-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. In den Repositorien: ***repository_name***Wählen Sie auf der Seite das Bild aus, dessen Details Sie sich ansehen möchten.

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

Wenn Sie ein Docker-Image ausführen möchten, das in Amazon verfügbar istECR, können Sie es mit dem docker pull Befehl 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.

Informationen zur Verwendung eines ECR Amazon-Images in einer ECS Amazon-Aufgabendefinition finden Sie unterVerwendung von Amazon ECR-Bildern mit Amazon ECS.

> **Important**
>
> Amazon ECR verlangt, dass Benutzer über die Erlaubnis verfügen, `ecr:GetAuthorizationToken` API über eine IAM Richtlinie Aufrufe zu tätigen, bevor sie sich bei einer Registrierung authentifizieren und Bilder aus einem beliebigen ECR Amazon-Repository per Push oder Pull abrufen 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.

Um ein Docker-Image aus einem ECR Amazon-Repository abzurufen

1. Authentifizieren Sie Ihren Docker-Client bei der ECR Amazon-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 Beispiel-Registrierung enthält das Repository `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 *repository-url* `not found: does not exist or no pull access` Fehlermeldung erhalten, müssen Sie Ihren Docker-Client möglicherweise bei Amazon authentifizieren. ECR Weitere Informationen finden Sie unter Authentifizierung bei privaten Registern in Amazon ECR.

Das Amazon Linux-Container-Image abrufen

Das Amazon Linux-Container-Image besteht aus denselben Softwarekomponenten, die in Amazon Linux enthalten sindAMI. Das Amazon Linux-Container-Image kann in jeder Umgebung als Basis-Image für Docker-Workloads verwendet werden. Wenn Sie Amazon Linux AMI für Anwendungen in Amazon verwendenEC2, können Sie Ihre Anwendungen mit dem Amazon Linux-Container-Image containerisieren.

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

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

Um das Amazon Linux-Container-Image von Amazon ECR Public abzurufen

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. Das Amazon Linux-Container-Image in der Amazon ECR Public Gallery finden Sie unter 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
```

Ein Bild in Amazon löschen 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 OCI Version 1.1 ECR unterstützte, weisen bei einigen Signaturen Bildindizes oder Manifestlisten auf sie hin. Wenn Sie ein Image aus einer Version vor OCI Version 1.1 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 Management Console)

1. Öffnen Sie die ECR Amazon-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. In den Repositorien: ***repository_name***Wählen Sie auf der Seite das Feld links neben dem Bild aus, das Sie löschen möchten, und wählen Sie Löschen.
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
```

Ein Bild in Amazon neu taggen ECR

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 das Manifest für ein Bild mit dem Tag *latest*, im Repository, *amazonlinux*, wird in eine Umgebungsvariable namens geschrieben *MANIFEST*.

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

2. Verwenden Sie die `--image-tag` Option des put-image Befehls, um das Image-Manifest ECR mit einem neuen Tag an Amazon zu senden. In diesem Beispiel ist das Bild gekennzeichnet als *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 das Cmdlet Get-ECRImageBatch, um die Beschreibung des neu zu kennzeichnenden Images abzurufen und sie in eine Umgebungsvariable zu schreiben. In diesem Beispiel ein Bild mit dem Tag *latest*, im Repository, *amazonlinux*, wird in die Umgebungsvariable geschrieben, *$Image*.

   > **Note**
   >
   > Wenn das Cmdlet Get-ECRImageBatch auf Ihrem System nicht verfügbar ist, lesen Sie bitte den Abschnitt Einrichten des AWS Tools for Windows PowerShell im AWS Tools for Windows PowerShell Benutzerhandbuch.

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

2. Schreiben Sie das Manifest des Bildes in den *$Manifest* Umgebungsvariable.

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

3. Verwenden Sie die `-ImageTag` Option des Write-ECRImage Cmdlets, um das Image-Manifest ECR mit einem neuen Tag an Amazon zu senden. 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 überschrieben werden ECR

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 Management Console 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 unterEin privates Amazon ECR-Repository zum Speichern von Bildern erstellen.

Einstellung der Veränderbarkeit von Bild-Tags ()AWS Management Console

Um die Veränderbarkeit von Bild-Tags festzulegen

1. Öffnen Sie die ECR Amazon-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 linken Navigationsbereich Repositorys aus.
4. Wählen Sie auf der Seite Repositorys die Registerkarte Privatund wählen anschließend das zu bearbeitende Repository aus und klicken Sie auf Bearbeiten.
5. Wählen Sie für die Unveränderlichkeit von Tags die Einstellung für die Veränderlichkeit von Tags für das Repository. Repositories, die mit unveränderlichen Tags konfiguriert sind, verhindern, dass Image-Tags überschrieben werden. Weitere Informationen finden Sie unter Verhindern, dass Bild-Tags in Amazon überschrieben werden ECR.
6. 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 Software-Sicherheitslücken in Amazon scannen ECR.
7. Für Verschlüsselungseinstellungen ist dies ein Nur-Ansichtsfeld, da die Verschlüsselungseinstellungen für ein Repository nicht geändert werden können, sobald das Repository erstellt wurde.
8. 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)

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

- Neu- ECRRepository ()AWS Tools for Windows PowerShell

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

Um die Mutabilitätseinstellungen für Image-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)

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

- Schreiben Sie- ECRImageTagMutability ()AWS Tools for Windows PowerShell

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

Unterstützung für das Container-Image-Manifestformat 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.

Konvertierung des ECR Amazon-Image-Manifests

Wenn Sie Images zu und von Amazon übertragen und abrufenECR, kommuniziert Ihr Container-Engine-Client (z. B. Docker) mit der Registry, um sich auf ein Manifestformat zu einigen, das vom Client und der Registry verstanden wird, das für das Image verwendet werden soll.

Wenn Sie ein Image ECR mit Docker Version 1.9 oder früher an Amazon übertragen, wird das Image-Manifestformat als Docker Image Manifest V2 Schema 1 gespeichert. Wenn Sie ein Image

ECR mit Docker Version 1.10 oder höher an Amazon übertragen, wird das Image-Manifestformat als Docker Image Manifest V2 Schema 2 gespeichert.

Wenn Sie ein Bild anhand eines Tags ECR von Amazon abrufen, ECR gibt Amazon das Image-Manifestformat zurück, das im Repository gespeichert ist. Das Format wird nur zurückgegeben, wenn es vom Client verarbeitet werden kann. Wenn das gespeicherte Image-Manifestformat vom Client nicht verstanden wird, ECR konvertiert Amazon das Image-Manifest in ein Format, das verstanden wird. Wenn beispielsweise ein Docker 1.9-Client ein Image-Manifest anfordert, das als Docker Image Manifest V2 Schema 2 gespeichert ist, ECR gibt Amazon das Manifest im Format Docker Image Manifest V2 Schema 1 zurück. In der folgenden Tabelle werden die verfügbaren Konvertierungen beschrieben, die von Amazon unterstützt werdenECR, wenn ein Bild nach Tag abgerufen wird:

Vom Client angefordertes Schema	Auf V2 übertragenECR, Schema 1	Auf V2 verschoben, Schema 2 ECR	Auf ECR als verschoben OCI
V2, Schema 1	Keine Konvertierung erforderlich	Konvertiert in V2, Schema 1	Konvertiert in V2, Schema 1
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	Übersetzt nach OCI	Keine Konvertierung erforderlich

Important

Wenn Sie ein Image per Digest abrufen, ist keine Konvertierung verfügbar. Ihr Kunde muss das Image-Manifestformat verstehen, das in Amazon gespeichert istECR. 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 dasselbe Bild nach Tag anfordern, ECR übersetzt Amazon das Image-Manifest in ein Format, das der Kunde verstehen kann. 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 Berechtigungen zum Erstellen der `ecr:BatchGetImage`-, `ecr:GetDownloadUrlForLayer`-, und `ecr:GetAuthorizationToken`-APIs verfügen.

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-Rolle	AWS verwaltete IAM-Richtlinie
Amazon ECS auf Amazon-EC2-Instances	Verwenden Sie die IAM-Rolle der Container-Instance, die der Amazon-EC2-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	`AmazonECSTaskExecutionRolePolicy` Weitere Informationen finden Sie unter AmazonECSTaskExecutionRolePolicy im Entwicklerhandbuch für Amazon Elastic Container Service.

Starttyp	IAM-Rolle	AWS verwaltete IAM-Richtlinie
	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 Amazon_ECS-Cluster registriert ist. Weitere Informationen finden Sie unter Amazon ECS-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.

Important

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

```
{
    "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.

```
{
    "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 AWS 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 Ein Helm-Diagramm in ein ECR privates Amazon-Repository verschieben.
- 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 auf Software-Sicherheitslücken in Amazon scannen ECR

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 Scanversion zurückkehren, sind die etablierten Scans verfügbar.

- Verbessertes Scannen — Amazon ist ECR in Amazon Inspector integriert, um automatisiertes, kontinuierliches Scannen Ihrer Repositorys zu ermöglichen. Ihre Container-Images werden sowohl auf Betriebssysteme als auch auf Schwachstellen im Programmiersprachenpaket 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.
- Einfaches Scannen — Amazon ECR bietet zwei Versionen des Standardscans an, die die Common Vulnerabilities and Exposures (CVEs) -Datenbank verwenden: die aktuelle GA-Version, die das Open-Source-Projekt Clair verwendet, und eine verbesserte Version des Basisscans, die unsere native Technologie verwendet. AWS Beim einfachen Scannen konfigurieren Sie Ihre Repositorys so, dass sie bei Push scannen, oder Sie können manuelle Scans durchführen. Amazon ECR stellt dann eine Liste der Scanergebnisse bereit. Das grundlegende Scannen bietet Folgendes:
  - Betriebssystem-Scans.
  - Zwei Scanfrequenzen: Manuell und Scannen auf Tastendruck.

> **Important**
>
> Die neue Version von Basic Scanning unterstützt `imageScanFindingsSummary` und `imageScanStatus` in der nicht `DescribeImages`API. Um diese anzuzeigen, verwenden Sie die `DescribeImageScanFindings`API.

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

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. Alle Repositorys, die nicht mit einem Scan-bei-Push-Filter für einfaches Scannen übereinstimmen, werden auf die manuelle Scan-Frequenz umgestellt, was bedeutet, dass Sie den Scan manuell auslösen müssen.

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 erweitertes Scannen verwenden und separate Filter für das Scannen bei Push und das kontinuierliche Scannen angeben, bei denen mehrere Filter mit demselben Repository übereinstimmen, ECR setzt Amazon den Filter für kontinuierliches Scannen über den Scan-on-Push-Filter für dieses Repository durch.

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

Scannen Sie Bilder auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR

Amazon ECR Enhanced Scanning ist eine Integration mit Amazon Inspector, die Schwachstellenscans 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 direkt bei Amazon Inspector 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 dieser Funktion fallen keine zusätzlichen Kosten von Amazon ECR an. Amazon Inspector erhebt jedoch Kosten für das Scannen Ihrer Bilder. Weitere Informationen erhalten Sie unter Amazon Inspector: Preise.
- Erweitertes Scannen wird in den folgenden Regionen nicht unterstützt:
  - Naher Osten (UAE) (`me-central-1`)
  - Asien-Pazifik (Hyderabad) (`ap-south-2`)

  - Israel (Tel Aviv) (`il-central-1`)
  - Asien-Pazifik (Melbourne) (`ap-southeast-4`)
  - Europa (Spanien) (`eu-south-2`)
- Amazon Inspector unterstützt das Scannen nach bestimmten Betriebssystemen. Eine vollständige Liste finden Sie unter Unterstützte Betriebssysteme — ECR Amazon-Scanning im Amazon Inspector-Benutzerhandbuch.
- Amazon Inspector verwendet eine servicebezogene IAM Rolle, die die Berechtigungen bereitstellt, die für erweiterte Scans Ihrer Repositorys erforderlich sind. Die serviceverknüpfte IAM Rolle wird automatisch von Amazon Inspector erstellt, wenn das erweiterte Scannen für Ihre private Registrierung aktiviert wird. 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 ECR in den letzten 30 Tagen, basierend auf dem Image-Push-Zeitstempel, an Amazon gesendet oder in den letzten 90 Tagen abgerufen 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.
- Alle Bilder, die ECR nach der Aktivierung des erweiterten Scannens an Amazon gesendet werden, werden kontinuierlich für die konfigurierte Dauer gescannt. Die Standarddauer ist Lifetime. Diese Einstellung kann über die Amazon Inspector-Konsole festgelegt werden. Weitere Informationen finden Sie unter Änderung der erweiterten Scandauer für Bilder in Amazon Inspector.
- Wenn das erweiterte Scannen für Ihre ECR private Amazon-Registrierung aktiviert ist, werden Repositorys, die den Scanfiltern entsprechen, 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 gescannt werden ECR.
- Wenn Sie separate Filter für Scan on Push und kontinuierliches Scannen angeben, bei denen mehrere Filter demselben Repository entsprechen, setzt Amazon den Filter ECR für kontinuierliches Scannen über den Filter für Scan bei Push-Zugriff für dieses Repository durch.
- Wenn erweitertes Scannen aktiviert ist, ECR sendet Amazon ein Ereignis an, EventBridge wenn die Scan-Frequenz für ein Repository geändert wird. Amazon Inspector gibt Ereignisse aus, EventBridge wenn ein erster Scan abgeschlossen ist und wenn ein Bildscan-Ergebnis erstellt, aktualisiert oder geschlossen wird.

IAMFür erweitertes Scannen in Amazon sind Berechtigungen erforderlich ECR

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

Die folgende IAM Richtlinie gewährt die erforderlichen Berechtigungen für die Aktivierung und Nutzung des erweiterten Scannens. Dazu gehören die Berechtigungen, die Amazon Inspector benötigt, um die serviceverknüpfte IAM Rolle zu erstellen, sowie die Amazon Inspector API Inspector-Berechtigungen, die erforderlich sind, um das erweiterte Scannen ein- und auszuschalten und die Scanergebnisse abzurufen.

```
{
    "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 IAM erforderlichen Berechtigungen verfügen, um erweitertes Scannen zu konfigurieren. Weitere Informationen finden Sie unter IAMFür erweitertes Scannen in Amazon sind Berechtigungen erforderlich ECR.

AWS Management Console

Um das erweiterte Scannen für Ihre private Registrierung zu aktivieren

1. Öffnen Sie die ECR Amazon-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, Einstellungen, Scannen 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 Save (Speichern) aus.
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 angegeben `rules` sind, ECR legt Amazon die Scan-Konfiguration standardmäßig auf kontinuierliches Scannen für alle Repositorys fest.

```
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
```

Änderung der erweiterten Scandauer für Bilder in Amazon Inspector

Sie können die Anzahl der Tage ändern, an denen Amazon Inspector die Bilder in Ihren ECR privaten Amazon-Repositorys kontinuierlich scannt. Wenn das erweiterte Scannen für Ihre ECR private Amazon-Registrierung aktiviert ist, überwacht der Amazon Inspector-Service standardmäßig kontinuierlich Ihre Repositorys, bis entweder das Bild gelöscht oder das erweiterte Scannen deaktiviert wird. Die Dauer, in der Amazon Inspector Ihre Images scannt, kann mithilfe der Amazon-Inspector-Einstellungen geändert werden. Die verfügbaren Scandauern sind Lifetime (default) Lebensdauer (Standard), 180 days (180 Tage), und 30 days (30 Tage). Wenn die Scandauer für ein Repository abgelaufen ist, wird der Scanstatus `SCAN_ELIGIBILITY_EXPIRED` angezeigt, wenn Ihre Scan-Schwachstellen aufgelistet werden. Weitere Informationen finden Sie unter Ändern der Dauer des ECR automatischen erneuten Scans durch Amazon im Amazon Inspector-Benutzerhandbuch.

So ändern Sie die Einstellung für die erweiterte Scandauer

1. Öffnen Sie die Amazon Inspector Inspector-Konsole unter https://console.aws.amazon.com/inspector/v2/home.
2. Erweitern Sie im linken Navigationsbereich Settings (Einstellungen) und wählen Sie dann General (Allgemeines).
3. Wählen Sie auf der Seite Einstellungen unter Dauer des ECR erneuten Scans eine Einstellung aus und klicken Sie dann auf Speichern.

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

Wenn erweitertes Scannen aktiviert ist, ECR sendet Amazon ein Ereignis an, EventBridge wenn die Scan-Frequenz für ein Repository geändert wird. 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 gesendet, ECR wenn es eine Änderung mit einer Ressource gibt, für die erweitertes 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 Software-Sicherheitslücken in Amazon scannen ECR.

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

```
    "repository-name": "repository-3",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
 }
}
```

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 Scannen Sie Bilder auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR.

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 Scannen Sie Bilder auf Sicherheitslücken in Betriebssystemen und Programmiersprachenpaketen in Amazon ECR.

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",
                        "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 Softwareschwachstellen 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 Problembehebung beim Scannen von Bildern in Amazon ECR.

AWS Management Console

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

Um die Ergebnisse des Bildscans abzurufen

1. Öffnen Sie die ECR Amazon-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 Images unter der Spalte Schwachstellen die Option Ergebnisse anzeigen für das Image aus, für das die Scanergebnisse abgerufen werden sollen.

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 mit `imageTag` oder `imageDigest` ein Bild angeben. Beide können mit dem Befehl list-images CLI abgerufen werden.

- 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 scannen ECR

> **Note**
>
> Es wird empfohlen, die verbesserte Version des Standardscans zu verwenden.

Amazon ECR bietet zwei Versionen von Standardscans an, die die Common Vulnerabilities and Exposures (CVEs) -Datenbank verwenden:

- Die verbesserte Version des einfachen Scannens, die AWS native Technologie verwendet (AWS_NATIVE).

- Die vorherige Basisversion für das Scannen, die das Open-Source-Projekt Clair verwendet. Weitere Informationen zu Clair finden Sie unter Clair. GitHub

Amazon ECR verwendet den Schweregrad für a CVE aus der vorgelagerten Vertriebsquelle, sofern verfügbar. Andernfalls wird die Bewertung des Common Vulnerability Scoring System (CVSS) verwendet. Die CVSS Bewertung kann verwendet werden, um die Bewertung des Schweregrads der NVD Sicherheitslücke zu erhalten. Weitere Informationen finden Sie unter Bewertungen des Schweregrads der NVD Sicherheitslücke.

Beide Versionen von Amazon ECR Basic Scanning unterstützen Filter, mit denen Sie angeben können, welche Repositorys bei Push gescannt werden sollen. Für alle Repositorien, 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.

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

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 das Scannen nach Sicherheitslücken ECR möglicherweise nicht mehr. Alle Ergebnisse, die Amazon ECR für ein eingestelltes Betriebssystem generiert, sollten nur zu Informationszwecken verwendet werden. Nachfolgend sind die aktuell unterstützten Betriebssysteme und Versionen aufgeführt.

Betriebssystem	Version
Alpine Linux (Alpine)	3.19

Betriebssystem	Version
Alpines Linux (Alpin)	3.18
Alpines Linux (Alpin)	3.17
Alpines Linux (Alpin)	3.16
Alpines Linux (Alpin)	3,20
Amazon Linux (2AL2)	AL2
Amazon Linux 2023 (AL2023)	AL2023
CentOS Linux (CentOS)	7
Debian-Server (Bücherwurm)	12
Debian-Server (Bullseye)	11
Debian-Server (Buster)	10
Oracle Linux (Oracle)	9
Oracle Linux (Oracle)	8
Oracle Linux (Oracle)	7
Ubuntu (Mond)	23,04
Ubuntu (Jammy)	22,04 () LTS
Ubuntu (fokal)	20,04 () LTS
Ubuntu (Bionisch)	18,04 () ESM
Ubuntu (Xenial)	16,04 () ESM
Ubuntu (Vertrauenswürdig)	14.04 () ESM
RedHat Enterprise Linux () RHEL	7

Betriebssystem	Version
RedHat Enterprise Linux (RHEL)	8
RedHat Enterprise Linux (RHEL)	9

Konfiguration des grundlegenden Scannens für Bilder in Amazon ECR

Standardmäßig ECR aktiviert Amazon 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 ECR Amazon-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 Registrierung, Scannen 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 gescannt werden ECR.

Umstellung auf das verbesserte grundlegende Scannen von Bildern in Amazon ECR

> **Important**
>
> Für neue Benutzer werden Ihre Registrierungen bei der Erstellung automatisch so konfiguriert, dass sie die `AWS_NATIVE` Scantechnologie verwenden. Sie müssen nichts

unternehmen. Amazon empfiehlt ECR nicht, zur vorherigen Scantechnologie zurückzukehren. `CLAIR`

AWS Management Console

Um das verbesserte Standardscannen für Ihre private Registrierung zu aktivieren

1. Öffnen Sie die ECR Amazon-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, Settings, Scanning aus.
4. Wählen Sie auf der Konfigurationsseite für den Scantyp die Option Verbessertes Standardscannen 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 gescannt werden ECR.

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 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 arugents 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 `AWS_NATIVE` geändert haben, wird nicht empfohlen, zu `CLAIR` zurückzukehren.

```
aws ecr put-account-setting --name 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 Problembehebung beim Scannen von Bildern in Amazon ECR.

AWS Management Console

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

1. Öffnen Sie die ECR Amazon-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 ECRImageScanFindingAWS 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 Problembehebung beim Scannen von Bildern in Amazon ECR.

AWS Management Console

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

Um die Ergebnisse des Bildscans abzurufen

1. Öffnen Sie die ECR Amazon-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 Images unter der Spalte Schwachstellen die Option Details für das Image aus, für das die Scanergebnisse abgerufen werden sollen.

AWS CLI

Verwenden Sie den folgenden AWS CLI Befehl, um die Ergebnisse des Bildscans mithilfe von abzurufen. AWS CLI Sie können mit `imageTag` oder `imageDigest` ein Bild angeben. Beide können mit dem Befehl list-images CLI abgerufen werden.

- 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- () ECRImageScanFindingAWS 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
```

Problembehebung beim Scannen von Bildern in Amazon ECR

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

UnsupportedImageError

Möglicherweise erhalten Sie eine `UnsupportedImageError` Fehlermeldung, wenn Sie versuchen, einen einfachen Scan für ein Bild durchzuführen, das mit einem Betriebssystem erstellt wurde, für das Amazon ECR das einfache Scannen von Bildern nicht unterstützt. Amazon ECR unterstützt das Scannen von Paketen nach Sicherheitslücken für Hauptversionen der Distributionen Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine und RHEL Linux. Sobald eine Distribution den Support durch ihren Anbieter verliert, unterstützt Amazon das Scannen nach Sicherheitslücken ECR möglicherweise nicht mehr. Amazon ECR unterstützt das Scannen von Bildern, die aus dem Docker-Scratch-Image erstellt wurden, nicht.

> **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 Sicherheitslücke wurde von der CVE Quelle keine Priorität zugewiesen.
- Der Sicherheitslücke wurde eine Priorität zugewiesen, die Amazon nicht ECR erkannte.

Um den Schweregrad und die Beschreibung einer Sicherheitslücke zu ermitteln, können Sie sie CVE direkt von der Quelle aus einsehen.

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 zunächst aktivieren, erkennt Amazon Inspector nur Bilder, die ECR in den letzten 30 Tagen an Amazon gesendet 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 die Dauer des ECR erneuten Scans in der Amazon Inspector Inspector-Konsole geändert wird und diese Zeit verstrichen ist, wird der Scanstatus des Bilds in den Ursachencode geändert`expired`, und alle zugehörigen Ergebnisse für das Bild werden so geplant, dass alle zugehörigen Ergebnisse für das Bild geschlossen werden. `inactive` Dies führt dazu, dass die ECR Amazon-Konsole den Scanstatus als auflistet`SCAN_ELIGIBILITY_EXPIRED`.

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

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

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

- Docker Hub, Microsoft Azure Container Registry, GitHub Container Registry und GitLab Container Registry (Authentifizierung erforderlich)
- Amazon ECR Public, die Kubernetes-Container-Image-Registry und Quay (erfordert keine Authentifizierung)

Für GitLab Container Registry ECR unterstützt Amazon den Pull-Through-Cache nur mit GitLab software-as-a-service Offering, GitLab .com.

Für die Upstream-Registries, für die eine Authentifizierung erforderlich ist, müssen Sie Ihre Anmeldeinformationen AWS Secrets Manager geheim speichern. Die ECR Amazon-Konsole macht es Ihnen leicht, das Secrets Manager Manager-Geheimnis für jede der authentifizierten Upstream-Registries 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.

Nachdem Sie eine Pull-Through-Cache-Regel für die Upstream-Registrierung erstellt haben, ziehen Sie mithilfe Ihrer ECR privaten Amazon-Registrierung einfach ein Bild aus dieser Upstream-Registrierung abURI. Amazon erstellt ECR dann ein Repository und speichert dieses Bild in Ihrer privaten Registrierung. Bei Ihren nachfolgenden Pull-Requests des zwischengespeicherten Images mit einem bestimmten Tag ECR überprüft Amazon in der Upstream-Registry, ob es eine neue Version des Images mit diesem spezifischen Tag gibt, und versucht, das Image in Ihrer privaten Registrierung mindestens einmal alle 24 Stunden zu aktualisieren.

Vorlagen zur Erstellung von Repositorien

Amazon ECR hat Unterstützung für Vorlagen zur Erstellung von Repositorys hinzugefügt, die sich derzeit in der Vorschauversion befinden. Dadurch können Sie mithilfe von Pull-Through-

Cache-Regeln die Anfangskonfigurationen für neue Repositorys festlegen, die von ECR Amazon 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- 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 von Amazon ECR mithilfe einer Pull-Through-Cache-Regel nicht.
- Beim Abrufen von Bildern mithilfe des Pull-Through-Cache werden die ECR FIPS Amazon-Servicendpunkte nicht unterstützt, wenn ein Bild zum ersten Mal abgerufen wird. Die Nutzung der Amazon ECR FIPS Service Endpoints funktioniert jedoch bei nachfolgenden Pulls.
- Wenn ein zwischengespeichertes Bild über die ECR private Registrierung von Amazon abgerufen wirdURI, 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 Bild durch die ECR private Registrierung von Amazon abgerufen wirdURI, ECR überprüft Amazon das Upstream-Repository mindestens einmal alle 24 Stunden, um zu überprüfen, ob es sich bei dem zwischengespeicherten Image um die neueste Version handelt. Wenn sich in der Upstream-Registrierung ein neueres Bild befindet, ECR versucht Amazon, das zwischengespeicherte Bild zu aktualisieren. Dieser Timer basiert auf dem letzten Abruf des zwischengespeicherten Images.

- Wenn Amazon ECR das Image aus irgendeinem Grund nicht aus der Upstream-Registrierung aktualisieren kann und das Bild abgerufen wird, wird das zuletzt zwischengespeicherte Bild trotzdem 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 Multiarchitektur-Image mithilfe einer Pull-Through-Cache-Regel abgerufen wird, werden die Manifestliste und jedes Image, auf das in der Manifestliste verwiesen wird, in das ECR Amazon-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 Amazon die Berechtigungen bereitstellt, die Amazon benötigt, ECR um das Repository zu erstellen, den geheimen Secrets Manager-Wert für die Authentifizierung abzurufen und das zwischengespeicherte Bild in Ihrem Namen zu übertragen. Die serviceverknüpfte IAM Rolle wird automatisch erstellt, wenn eine Pull-Through-Cache-Regel erstellt wird. Weitere Informationen finden Sie unter ECRService-verknüpfte Rolle mit Amazon für den Pull-Through-Cache.
- Standardmäßig verfügt der IAM Principal, der das zwischengespeicherte Bild abruft, über die ihm gemäß seiner IAM Richtlinie gewährten Berechtigungen. Sie können die Amazon ECR Private Registry Permissions Policy verwenden, um die Berechtigungen einer IAM Entität weiter einzuschränken. Weitere Informationen finden Sie unter Verwenden von Registrierungsberechtigungen.
- ECRAmazon-Repositorys, die mit dem Pull-Through-Cache-Workflow erstellt wurden, werden wie jedes andere ECR Amazon-Repository behandelt. Alle Repository-Features wie Replikation und Image-Scan werden unterstützt.
- Wenn Amazon in Ihrem Namen mithilfe einer Pull-Through-Cache-Aktion ein neues Repository ECR erstellt, werden die folgenden Standardeinstellungen auf das Repository angewendet, sofern es keine passende Vorlage für die Repository-Erstellung gibt. Sie können eine Vorlage für die Erstellung eines Repositorys verwenden, um die Einstellungen zu definieren, die auf Repositorys angewendet werden, die Amazon in ECR Ihrem Namen erstellt hat. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache- oder Replikationsaktion erstellt wurden.
  - Unveränderlichkeit von Tags – Diese Option ist deaktiviert, Tags sind veränderlich und können überschrieben werden.

  - Verschlüsselung – Die `AES256`-Standardverschlüsselung wird verwendet.
  - Repository-Berechtigungen – Ausgelassen, es wird keine Repository-Berechtigungsrichtlinie angewendet.
  - Lebenszyklusrichtlinie – Ausgelassen, es wird keine Lebenszyklusrichtlinie angewendet.
  - Ressourcen-Tags – Ausgelassen, es werden keine Ressourcen-Tags angewendet.
- Wenn Sie die Unveränderlichkeit von Image-Tags für Repositorys mithilfe einer Pull-Through-Cache-Regel aktivieren, wird Amazon ECR daran gehindert, Bilder zu aktualisieren, die dasselbe Tag verwenden.
- Wenn ein Bild zum ersten Mal mithilfe der Pull-Through-Cache-Regel abgerufen wird, ist möglicherweise eine Verbindung 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 für die Verwendung eines VPC Schnittstellenendpunkts konfiguriert haben, AWS PrivateLink müssen Sie sicherstellen, dass der erste Pull eine Route zum Internet hat. Eine Möglichkeit, dies zu tun, besteht darinVPC, in demselben ein öffentliches Subnetz mit einem Internet-Gateway zu erstellen und dann den gesamten ausgehenden Datenverkehr von ihrem privaten Subnetz zum öffentlichen Subnetz an das 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.

IAMBerechtigungen, die erforderlich sind, um eine Upstream-Registrierung mit einer ECR privaten Amazon-Registrierung zu synchronisieren

Zusätzlich zu den ECR API Amazon-Berechtigungen, die für die Authentifizierung bei einer privaten Registrierung und für das Push- und Pull von Bildern erforderlich sind, sind die folgenden zusätzlichen Berechtigungen erforderlich, um Pull-Through-Cache-Regeln effektiv nutzen zu können.

- `ecr:CreatePullThroughCacheRule` – Gewährt die Berechtigung zum Erstellen einer neuen Pull-Through-Cache-Regel. Diese Genehmigung muss im Rahmen einer IAM identitätsbasierten Richtlinie erteilt werden.
- `ecr:BatchImportUpstreamImage` – Erteilt die Berechtigung, das externe Image abzurufen und in Ihre private Registrierung zu importieren. Diese Berechtigung kann mithilfe der Berechtigungsrichtlinie für private Registrierungen, einer identitätsbasierten Richtlinie oder mithilfe

der IAM Richtlinie für ressourcenbasierte Repository-Berechtigungen 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 durch die Berechtigungsrichtlinie für private Registrierungen erteilt werden.
- `ecr:TagResource`— Erteilt die Erlaubnis, Metadaten-Tags zu einer ECR Amazon-Ressource hinzuzufügen. Diese Berechtigung ist nur erforderlich, wenn Sie ein Image abrufen, das eine Pull-Through-Cache-Regel verwendet, der eine Repository-Erstellungsvorlage zugeordnet ist, die so konfiguriert ist, dass sie dem Repository Ressourcen-Tags hinzufügt. Diese Genehmigung muss im Rahmen einer identitätsbasierten IAM Richtlinie erteilt werden.

Verwenden von Registrierungsberechtigungen

Die ECR privaten Registrierungsberechtigungen von Amazon können verwendet werden, um die Berechtigungen einzelner IAM Entitäten zur Nutzung des Pull-Through-Cache einzuschränken. Wenn einer IAM Entität durch eine IAM Richtlinie mehr Berechtigungen gewährt werden, als die Richtlinie für Registrierungsberechtigungen 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 Management Console)

1. Öffnen Sie die ECR Amazon-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 IAMEntitäten 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 erteilt die `ecr-pull-through-cache-user` Berechtigung, ein Repository zu erstellen und ein Bild von Amazon ECR Public abzurufen. Amazon Public ist die Upstream-Quelle, die mit der zuvor erstellten Pull-Through-Cache-Regel verknüpft ist.

```
{
  "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. Dies ist

beispielsweise der Fall, wenn die Aktion zum Erstellen des Repositorys und die Aktionen zum Abrufen von Images von unterschiedlichen 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 erstellen ECR.
- Erstellen Sie eine Repository-Erstellungsvorlage. Mit einer Vorlage für die Erstellung eines Repositorys können Sie festlegen, welche Einstellungen für neue Repositorys verwendet werden sollen, die Amazon in ECR Ihrem Namen während einer Pull-Through-Cache-Aktion erstellt hat. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache- oder Replikationsaktion erstellt wurden.

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

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

Für Upstream-Registrierungen, die eine Authentifizierung 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 ECR Amazon-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 ECR Amazon-Konsole finden Sie unterSpeichern Sie Ihre Upstream-Repository-Anmeldeinformationen AWS Secrets Manager geheim.

Voraussetzungen

- Vergewissern Sie sich, dass Sie über die erforderlichen IAM Berechtigungen zum Erstellen von Pull-Through-Cache-Regeln verfügen. Weitere Informationen finden Sie unter IAMBerechtigungen, die erforderlich sind, um eine Upstream-Registrierung mit einer ECR privaten Amazon-Registrierung zu synchronisieren.
- Für Upstream-Registrierungen, die eine Authentifizierung erfordern: Wenn Sie ein vorhandenes Secret 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 Management Console 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 Management Console)

Die folgenden Schritte zeigen, wie Sie mithilfe der ECR Amazon-Konsole eine Pull-Through-Cache-Regel und ein Secrets Manager-Geheimnis 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 ECR Amazon-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 entweder Amazon ECR Public, Kubernetes oder Quay aus der Liste der Upstream-Registries aus und klicken Sie dann auf Weiter.
6. Geben Sie auf Schritt 2: Zielseite angeben für das ECRAmazon-Repository-Präfix das Repository-Namespace-Präfix an, das beim Zwischenspeichern von Bildern verwendet werden

soll, die aus der öffentlichen Quellregistrierung abgerufen wurden, und wählen Sie dann Weiter. 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 ECR Amazon-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 Geheimnis angeben oder die ECR Amazon-Konsole verwenden, um ein neues Geheimnis 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 Management Console 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 Docker-Hub-Benutzername Ihren Docker-Hub-Benutzernamen 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 eine Zielseite für das ECRAmazon-Repository-Präfix den Repository-Namespace an, der beim Zwischenspeichern von Bildern verwendet werden soll, die aus der öffentlichen Quellregistrierung abgerufen wurden, und wählen Sie dann Weiter.

   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 ECR Amazon-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 Geheimnis angeben oder die ECR Amazon-Konsole verwenden, um ein neues Geheimnis 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 Management Console 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 eine Zielseite für das ECRAmazon-Repository-Präfix den Repository-Namespace an, der beim Zwischenspeichern von Bildern verwendet werden soll, die aus der öffentlichen Quellregistrierung abgerufen wurden, und wählen Sie dann Weiter.

   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 ECR Amazon-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 für Quellregistrierung URL den Namen Ihrer Microsoft Azure-Container-Registry an und wählen Sie dann Weiter.

   > **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 Geheimnis angeben oder die ECR Amazon-Konsole verwenden, um ein neues Geheimnis 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 Management Console 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 eine Zielseite für das ECRAmazon-Repository-Präfix den Repository-Namespace an, der beim Zwischenspeichern von Bildern verwendet werden soll, die aus der öffentlichen Quellregistrierung abgerufen wurden, und wählen Sie dann Weiter.

   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 GitLab

1. Öffnen Sie die ECR Amazon-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 Geheimnis angeben oder die ECR Amazon-Konsole verwenden, um ein neues Geheimnis 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. 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 Management Console 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 eine Zielseite für das ECRAmazon-Repository-Präfix den Repository-Namespace an, der beim Zwischenspeichern von Bildern verwendet werden soll, die aus der öffentlichen Quellregistrierung abgerufen wurden, und wählen Sie dann Weiter.

   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.

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 ECR private Amazon-Registry zu erstellen. Für Upstream-Registrierungen, für die eine Authentifizierung erforderlich ist, müssen Sie die Anmeldeinformationen in einem Secrets-Manager-Secret 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 Amazon ECR Public Registry 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 die 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 Geheimnisses angeben, das Ihre Docker Hub-Anmeldeinformationen enthält.

```
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 GitHub Container Registry

Im folgenden Beispiel wird eine Pull-Through-Cacheregel für die GitHub Container Registry 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 `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 Geheimnisses angeben, das Ihre Anmeldeinformationen für die Microsoft Azure Container Registry enthält.

```
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
```

Nächste Schritte

Nachdem Sie Ihre Pull-Through-Cache-Regeln erstellt haben, folgen die nächsten Schritte:

- Erstellen Sie eine Repository-Erstellungsvorlage. Mit einer Vorlage für die Erstellung eines Repositorys können Sie festlegen, welche Einstellungen für neue Repositorys verwendet werden sollen, die Amazon in ECR Ihrem Namen während einer Pull-Through-Cache-Aktion erstellt hat. Weitere Informationen finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache- oder Replikationsaktion erstellt wurden.

- Validieren Sie Ihre Pull-Through-Cache-Regeln. Bei der Validierung einer Pull-Through-Cache-Regel ECR stellt Amazon eine Netzwerkverbindung mit der Upstream-Registrierung her und überprüft, ob es auf das Secrets Manager-Geheimnis 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 abrufen ECR.

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 ECR stellt Amazon 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 entsprechenden Berechtigungen verfügen. IAM Weitere Informationen finden Sie unter IAMBerechtigungen, die erforderlich sind, um eine Upstream-Registrierung mit einer ECR privaten Amazon-Registrierung zu synchronisieren.

So validieren Sie eine Pull-Through-Cache-Regel (AWS Management Console)

Die folgenden Schritte zeigen, wie Sie eine Pull-Through-Cache-Regel mithilfe der ECR Amazon-Konsole validieren.

1. Öffnen Sie die ECR Amazon-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 ECR private Amazon-Registry 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. Wenn `true` Amazon die Upstream-Registrierung erreichen ECR konnte und die Authentifizierung erfolgreich war. 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 abrufen ECR

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 entsprechenden IAM Berechtigungen verfügen. Weitere Informationen finden Sie unter IAMBerechtigungen, die erforderlich sind, um eine Upstream-Registrierung mit einer ECR privaten Amazon-Registrierung zu synchronisieren.

> **Note**
>
> Die folgenden Beispiele verwenden die standardmäßigen ECR Amazon-Repository-Namespace-Werte, die von AWS Management Console verwendet werden. Stellen Sie

sicher, dass Sie das ECR private Amazon-Repository verwendenURI, das Sie konfiguriert haben.

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 ECR Amazon-Konsole verwenden, um das Geheimnis zu erstellen, anstatt das Geheimnis mit der Secrets Manager-Konsole zu erstellen. Weitere Informationen finden Sie unter Eine Pull-Through-Cache-Regel in Amazon erstellen ECR.

Docker Hub

So erstellen Sie ein Secrets-Manager-Secret für Ihre Anmeldeinformationen für Docker Hub (AWS Management Console)

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 Schlüssel/Wert-Paar `username` als Schlüssel und Ihren Docker-Hub-Benutzernamen als Wert an.

   ii. Geben Sie für das zweite Schlüssel/Wert-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 unterstützt die Verwendung eines vom Kunden verwalteten Schlüssels (CMK) dafür ECR nicht.

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**
   >
   > Amazon zeigt ECR AWS Management Console nur Secrets Manager 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) Wählen Sie unter Geheimnis replizieren die Option Geheimnis replizieren aus, um Ihr Geheimnis auf ein anderes AWS-Region zu replizieren. 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.

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.

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 Management Console)

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 Schlüssel/Wert-Paar `username` als Schlüssel und Ihren GitHub Benutzernamen als Wert an.

ii. Geben Sie für das zweite Schlüssel/Wert-Paar `accessToken` als Schlüssel und Ihr GitHub Zugriffstoken als Wert an. Weitere Informationen zum Erstellen eines GitHub Zugriffstokens finden Sie in der Dokumentation unter Verwaltung Ihrer persönlichen Zugriffstoken. GitHub

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 unterstützt die Verwendung eines vom Kunden verwalteten Schlüssels (CMK) dafür ECR nicht.

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**
>
> Amazon zeigt ECR AWS Management Console nur Secrets Manager 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) Wählen Sie unter Geheimnis replizieren die Option Geheimnis replizieren aus, um Ihr Geheimnis auf ein anderes AWS-Region zu replizieren. 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.

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.

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 Management Console)

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 Schlüssel/Wert-Paar `username` als Schlüssel und Ihren Benutzernamen für die Microsoft Azure Container Registry als Wert an.

      ii. Geben Sie für das zweite Schlüssel/Wert-Paar `accessToken` als Schlüssel und Ihr Zugriffstoken für die Microsoft Azure Container Registry 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 unterstützt die Verwendung eines vom Kunden verwalteten Schlüssels (CMK) dafür ECR nicht.

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**
   >
   > Amazon zeigt ECR AWS Management Console nur Secrets Manager 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) Wählen Sie unter Geheimnis replizieren die Option Geheimnis replizieren aus, um Ihr Geheimnis auf ein anderes AWS-Region zu replizieren. 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.

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.
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 Management Console)

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 Schlüssel/Wert-Paar `username` als Schlüssel und Ihren GitLab Container Registry-Benutzernamen als Wert an.

      ii. Geben Sie für das zweite Schlüssel/Wert-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 Dokumentation unter Persönliche Zugriffstoken, Gruppenzugriffstoken oder Projektzugriffstoken. GitLab

   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 unterstützt die Verwendung eines vom Kunden verwalteten Schlüssels (CMK) dafür ECR nicht.

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**
   >
   > Amazon zeigt ECR AWS Management Console nur Secrets Manager 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) Wählen Sie unter Geheimnis replizieren die Option Geheimnis replizieren aus, um Ihr Geheimnis auf ein anderes AWS-Region zu replizieren. 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.

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.
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.

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 angibt, dass das Repository nicht existiert, wird meistens dadurch verursacht, dass entweder das Repository nicht in Ihrer ECR privaten Amazon-Registrierung vorhanden ist oder dass dem IAM Prinzipal, der das Upstream-Image abruft, keine `ecr:CreateRepository` Erlaubnis erteilt wurde. Um diesen Fehler zu beheben, sollten Sie überprüfen, ob das Repository URI in Ihrem Pull-Befehl korrekt ist, dass dem IAM Principal, der das Upstream-Image abruft, die erforderlichen IAM Berechtigungen erteilt wurden oder dass das Repository für das Upstream-Image, in das gepusht werden soll, in Ihrer ECR privaten Amazon-Registrierung erstellt wurde, bevor Sie den Upstream-Image-Pull durchführen. Weitere Informationen zu den erforderlichen IAM Berechtigungen finden Sie unter IAMBerechtigungen, die erforderlich sind, um eine Upstream-Registrierung mit einer ECR privaten Amazon-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 darauf hinweist, dass das Image nicht gefunden werden kann, wird meistens dadurch verursacht, dass entweder das Image nicht in der Upstream-Registrierung vorhanden ist oder dass dem `ecr:BatchImportUpstreamImage` IAM Prinzipal, der das Upstream-Image abruft, nicht erteilt wurde, das Repository jedoch bereits in Ihrer ECR privaten Amazon-Registrierung erstellt wurde. Um diesen Fehler zu beheben, sollten Sie überprüfen, ob das Upstream-Image und der Image-Tag-Name korrekt sind und ob sie vorhanden sind und dass dem IAM Prinzipal, der das Upstream-Image abruft, die erforderlichen IAM Berechtigungen erteilt wurden. Weitere Informationen zu den erforderlichen IAM Berechtigungen finden Sie unterIAMBerechtigungen, die erforderlich sind, um eine Upstream-Registrierung mit einer ECR privaten Amazon-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 das `/library/` in das URI von Ihnen verwendete einbeziehen. Beispiel, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/docker-hub/`**`library/`***`image_name:tag`*. Wenn Sie die `/library/` für offizielle Docker-Hub-Images überspringen, 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 abrufen ECR.

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 Bilder 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

Überlegungen zur privaten Image-Replikation

Bei der Verwendung der privaten Image-Replikation sollte Folgendes beachtet werden.

- Nur Repository-Inhalt, der nach der Konfiguration der Replikation in ein Repository gepusht wurde, wird repliziert. Bereits vorhandener Inhalt in einem Repository wird nicht repliziert. Sobald die Replikation für ein Repository konfiguriert ist, synchronisiert Amazon ECR Ziel und Quelle.
- 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 Verwenden von serviceverknüpften 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 Partitionen wird nicht unterstützt. AWS 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 findet nur einmal pro Image-Push statt. 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.
- Die Registrierungsreplikation führt keine Löschvorgänge durch. Replizierte Images und Repositories können manuell gelöscht werden, wenn sie nicht mehr verwendet werden.
- Repository-Richtlinien, einschließlich IAM-Richtlinien, und Lebenszyklus-Richtlinien werden nicht repliziert und haben nur Auswirkungen auf das Repository, für das sie definiert sind.
- Die Repository-Einstellungen werden nicht repliziert. Die Einstellungen für die Unveränderlichkeit von Tags, das Scannen von Images und die KMS-Verschlüsselung sind bei allen Repositorys, die aufgrund einer Replikationsaktion erstellt wurden, standardmäßig deaktiviert. Die Einstellungen für die Unveränderlichkeit von Tags und das Scannen von Images können nach der Erstellung des Repositorys geändert werden. Die Einstellung gilt jedoch nur für Images, die nach der Änderung der Einstellung verschoben 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 Management Console, wird Ihnen bei der Überprüfung Ihrer Replikationsregel auf der Seite Überprüfen und Absenden ein ähnliches JSON-Format 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 `111122223333` ist und dass Sie diese Replikationskonfiguration in einer anderen Region als `us-west-1` oder `us-west-2` angeben.

```
{
    "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 Konto *111122223333* 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"
   }]
  }
 ]
}
```

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 Management Console)

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 Sie die Einstellungen für die Registrierungsreplikation konfigurieren möchten.
3. Wählen Sie im Navigationsbereich die Option Private Registrierung.
4. Wählen Sie auf der Seite Private Registrierung im Abschnitt Replikation die Option Bearbeiten.
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
```

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

Verwenden Sie Vorlagen zur Erstellung von ECR Amazon-Repositorys, um die Einstellungen für Repositorys zu definieren, die von Amazon in ECR 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 durchforsten
- Replikation

Vorlagen für die Erstellung von Repositorys werden 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`

So funktionieren Repository-Erstellungsvorlagen

Es kann vorkommen, dass Amazon in Ihrem Namen ein neues privates Repository erstellen ECR muss. Beispielsweise:

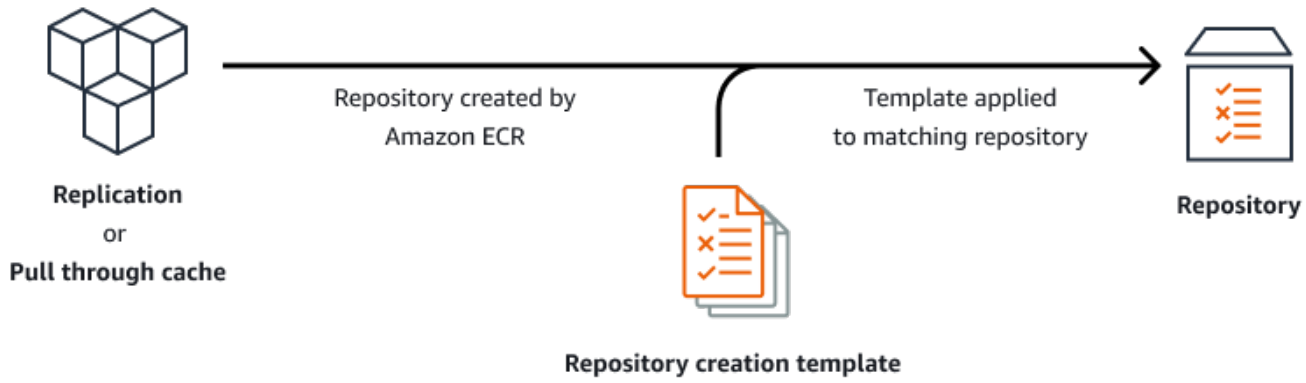
- Wenn Sie zum ersten Mal eine Pull-Through-Cache-Regel verwenden, um den Inhalt eines Upstream-Repositorys abzurufen und in Ihrer ECR privaten Amazon-Registrierung zu speichern.
- Wenn Sie möchtenECR, dass Amazon ein Repository in eine andere Region oder ein anderes Konto repliziert.

Wenn es keine Vorlage für die Repository-Erstellung gibt, die Ihrer Pull-Through-Cache-Regel oder Ihrem replizierten Repository entspricht, ECR verwendet Amazon 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.

Mithilfe einer Vorlage für die Erstellung eines Repositorys können Sie die Einstellungen definieren, die Amazon auf neue Repositorys ECR anwendet, die über den Pull-Through-Cache und die Replikationsaktionen 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 Workflow, den Amazon ECR verwendet, wenn eine Vorlage zur Repository-Erstellung mit einer Pull-Through-Cache-Aktion 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. Wenn in einer Registrierung, die zwei Vorlagen enthält, eine Vorlage das Präfix „prod“ und die andere das Präfix „prod/team“ hat, wird die Vorlage mit dem Präfix „prod/team“ auf alle Repositorys angewendet, deren Namen mit „prod/team/“ beginnen.

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, ECR behandelt Amazon das 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.

Beschreibung

Diese Vorlagenbeschreibung ist optional und wird verwendet, um den Zweck der Vorlage für die Repository-Erstellung zu beschreiben.

Beantragt

Die Einstellung für angewendet bestimmt, welche ECR -erstellten Repositorys mit dieser Vorlage erstellt werden. Die gültigen Werte sind `PULL_THROUGH_CACHE` und `REPLICATION`. Zum Beispiel, wenn Sie zum ersten Mal eine Pull-Through-Cache-Regel verwenden, um den Inhalt eines Upstream-Repositorys abzurufen und in Ihrer ECR privaten Amazon-Registrierung zu speichern. Wenn es keine Vorlage für die Erstellung eines Repositorys gibt, die Ihrer Pull-Through-Cache-Regel entspricht, ECR verwendet Amazon die Standardeinstellungen für das neue Repository.

Rolle beim Erstellen eines Repositor

Die Rolle zur Repository-Erstellung wird von Amazon übernommenARN, ECR wenn Repositorys mithilfe von Vorlagen zur Repository-Erstellung erstellt und konfiguriert werden. IAM Diese Rolle muss bei der Verwendung von Repository-Tags und/oder KMS in der Vorlage angegeben werden, andernfalls schlägt die Erstellung des Repositorys 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 weggelassen wird, MUTABLEwird die Standardeinstellung von verwendet, die das Überschreiben von Bild-Tags ermöglicht. 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 die zwischengespeicherten Bilder aktualisieren ECR kann, wenn die Tags identisch sind.

Wenn IMMUTABLEangegeben, sind alle Bild-Tags innerhalb des Repositorys unveränderlich, wodurch verhindert wird, dass sie überschrieben werden.

Verschlüsselungskonfiguration

Die Verschlüsselungskonfiguration, die für Repositorys verwendet werden soll, die mit der Vorlage erstellt wurden.

Wenn Sie den KMSVerschlüsselungstyp verwenden, wird der Inhalt des Repositorys mit serverseitiger Verschlüsselung verschlüsselt, wobei ein AWS Key Management Service Schlüssel gespeichert wird in. AWS KMS 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 Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung mit einem in AWS Key Management Service (SSE-KMS) gespeicherten AWS Key Management Service Schlüssel im Amazon Simple Storage Service-Benutzerhandbuch. Wenn Sie den KMSVerschlüsselungstyp und ihn mit regionsübergreifender Replikation verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Weitere Informationen finden Sie unter Erstellen einer KMS Schlüsselrichtlinie für die Replikation.

Wenn Sie den AES256Verschlüsselungstyp verwenden, ECR verwendet Amazon serverseitige Verschlüsselung mit von Amazon S3 verwalteten Verschlüsselungsschlüsseln, wodurch die Bilder im Repository mit einem AES -256-Verschlüsselungsalgorithmus verschlüsselt werden. Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung mit von Amazon SSE S3 verwalteten Verschlüsselungsschlüsseln (-S3) im Amazon Simple Storage Service-Benutzerhandbuch.

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 angeben, welche IAM Benutzer oder Rollen Zugriff auf ein Repository haben und welche Aktionen sie damit ausführen 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

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 definiertECR. 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 zur Erstellung eines Repositorys in Amazon erstellen ECR

Sie können eine Vorlage für die Repository-Erstellung erstellen, um die Einstellungen zu definieren, die für Repositorys verwendet werden sollen, die von Amazon in ECR Ihrem Namen während Pull-Through-Cache- oder Replikationsaktionen 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 beabsichtigen, KMS Schlüssel, Ressourcen-Tags oder eine Kombination aus beidem in einer oder mehreren Vorlagen zu verwenden, 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.

IAMBerechtigungen zum Erstellen von Vorlagen für die Erstellung von Repositorys

Die folgenden Berechtigungen sind erforderlich, damit ein IAM Principal Vorlagen für die Repository-Erstellung verwalten kann. Diese Berechtigungen müssen mithilfe einer identitätsbasierten IAM Richtlinie erteilt werden.

- `ecr:CreateRepositoryCreationTemplate` – Erteilt die Berechtigung zum Erstellen einer Repository-Erstellungsvorlage.
- `ecr:UpdateRepositoryCreationTemplate`— Erteilt die Erlaubnis, eine Vorlage zur Erstellung eines Repositorys 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 ECR Amazon-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.
- `ecr:TagResource` – Gewährt die Berechtigung zum Hinzufügen von Metadaten-Tags zu einer Ressource. Diese Berechtigung ist nur erforderlich, wenn die Repository-Erstellungsvorlage Ressourcen-Tags enthält.

Erstellen Sie eine benutzerdefinierte Richtlinie für Vorlagen zur Erstellung von Repositorys

Sie können die verwenden AWS Management Console , 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 Management Console

Um den JSON Policy-Editor zu verwenden, um eine benutzerdefinierte Richtlinie für Vorlagen zur Erstellung von Repositorys zu erstellen.

1. Melden Sie sich bei der an AWS Management Console 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 Create Policy (Richtlinie erstellen) aus.
4. Wählen Sie im Bereich Policy-Editor die JSONOption aus.
5. Geben Sie die folgende Richtlinie in das JSONFeld ein.

```
{
        "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 Richtlinienüberprüfung generiert wurden, und klicken Sie dann auf 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 Management Console um eine Rolle zu erstellen, die von Amazon verwendet werden kann, ECR wenn Sie die Rolle zur Repository-Erstellung in einer Repository-Erstellungsvorlage angeben, die Repository-Tags verwendet, oder KMS in einer Vorlage.

AWS Management Console

Um eine Rolle zu erstellen.

1. Melden Sie sich bei der an AWS Management Console 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:

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

```
}
```

5. Wählen Sie Weiter.
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 einer verwendet wirdARN, unterscheidet der Rollenname zwischen Groß- und Kleinschreibung. 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 Create Role (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 Management Console

So erstellen Sie eine Repository-Erstellungsvorlage (AWS Management Console)

1. Öffnen Sie die ECR Amazon-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 gesetzt. `ROOT`

6. Geben Sie unter Beantragt für an, für welche ECR Amazon-Workflows diese Vorlage gelten soll. Die Optionen sind `PULL_THROUGH_CACHE` 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 überschrieben werden ECR.

      Wenn Veränderlich ausgewählt ist, können Image-Tags überschrieben werden. Dies ist die empfohlene Einstellung für Vorlagen, die für Repositorys verwendet werden, die durch Replikationsaktionen erstellt wurden. Dadurch wird sichergestellt, dass Amazon die zwischengespeicherten Bilder aktualisieren ECR kann, wenn die Tags identisch sind.

      Wenn Unveränderlich ausgewählt ist, wird verhindert, dass Image-Tags überschrieben werden. Nachdem das Repository für unveränderliche Tags konfiguriert wurde, wird ein `ImageTagAlreadyExistsException`-Fehler zurückgegeben, wenn versucht wird, ein Image mit einem Tag zu übertragen, das sich bereits im Repository befindet. Wenn die Unveränderlichkeit von Tags für ein Repository aktiviert ist, wirkt sich dies auf alle Tags aus und Sie können einige Tags nicht unveränderlich machen, während andere dies nicht sind.

   b. Wählen Sie für die Verschlüsselungskonfiguration die zu verwendende Verschlüsselungseinstellung aus. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

Wenn AES-256 ausgewählt ist, ECR verwendet Amazon serverseitige Verschlüsselung mit vom Amazon Simple Storage Service verwalteten Verschlüsselungsschlüsseln, die Ihre Daten im Ruhezustand mit einem branchenüblichen AES -256-Verschlüsselungsalgorithmus verschlüsseln. Dies wird ohne zusätzliche Kosten angeboten.

Wenn diese Option ausgewählt AWS KMSist, ECR verwendet Amazon 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 verwaltet wirdECR, 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 Drop-down-Menü 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 Drop-down-Menü 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 aus. 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 sollen (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 and
 REPLICATION
    "prefix": "string",
    "description": "string",
    "imageTagMutability": "MUTABLE"|"IMMUTABLE",
    "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 `prod/*` mit einer Repository-Richtlinie erstellt wurde, die ALLOWs Bilder in future Repositorys überträgt und abruft, legt eine Lebenszyklusrichtlinie fest, nach der Bilder, die älter als zwei Wochen sind, ablaufen, und legt eine benutzerdefinierte Rolle fest, die den ECR Zugriff auf den KMS Schlüssel ermöglicht und das Ressourcen-Tag future Repositorys `examplekey` zuweist.

```
{
    "prefix": "prod",
    "description": "For repositories cached from my PTC rule and in my
 replication configuration that start with 'prod/'",
    "appliedFor": ["PULL_THROUGH_CACHE","REPLICATION"],
    "encryptionConfiguration": {
        "encryptionType": "KMS",
        "kmsKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-
cdef-example11111"
    },
    "resourceTags": [
        {
            "Key": "examplekey",
            "Value": "examplevalue"
        }
    ],
    "imageTagMutability": "MUTABLE",
    "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
```

Aktualisieren Sie eine Vorlage zur Erstellung eines Repositorys

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 Management Console

Um eine Vorlage für die Erstellung eines Repositorys zu bearbeiten (AWS Management Console)

1. Öffnen Sie die ECR Amazon-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der sich die zu bearbeitende Vorlage zur Repository-Erstellung 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 Befehl update-repository-creation-template.html, um eine bestehende 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"
  ```

  In der Ausgabe des Befehls werden die Details der aktualisierten Vorlage für die Repository-Erstellung angezeigt.

Löschen einer Vorlage zur Erstellung eines Repositorys 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 Management Console

So löschen Sie eine Repository-Erstellungsvorlage (AWS Management Console)

1. Öffnen Sie die ECR Amazon-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 bestehende Vorlage für die 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.

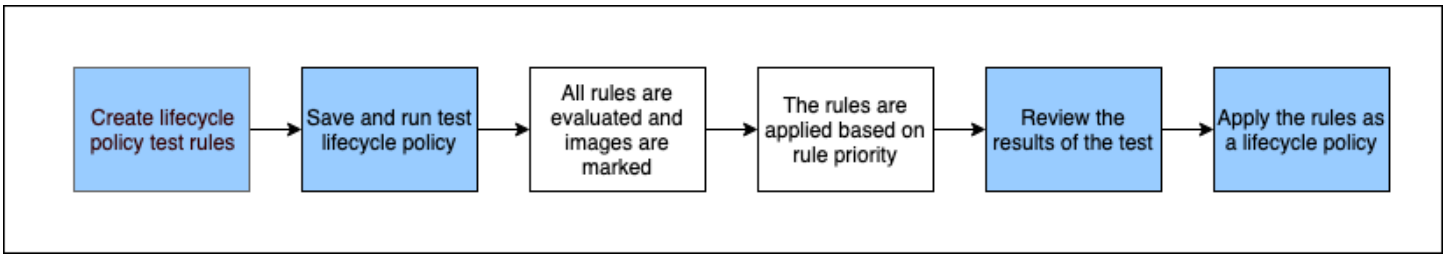
Automatisieren Sie die Bereinigung von Bildern mithilfe von Lebenszyklusrichtlinien in Amazon ECR

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. Basierend auf den Ablaufkriterien in der Lebenszyklusrichtlinie laufen Bilder je nach Alter oder Anzahl innerhalb von 24 Stunden ab. 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 ECRAmazon-Aktionen protokollieren 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 ECRAmazon-Aktionen protokollieren 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 Lebenszyklusrichtlinien-Evaluator wendet dann die Regeln auf der Grundlage der Regelpriorität an und zeigt an, welche Images im Repository als ablaufend gekennzeichnet sind.
5. Überprüfen Sie die Ergebnisse des Tests und vergewissern Sie sich, dass die Images, die als abgelaufen markiert sind, auch die gewünschten sind.
6. Wenden Sie die Testregeln als Lebenszyklusrichtlinie für das Repository an.
7. Sobald die Lebenszyklusrichtlinie erstellt ist, sollten Sie davon ausgehen, dass Images innerhalb von 24 Stunden, nachdem sie die Ablaufkriterien erfüllt haben, ablaufen.

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.

- Alle Regeln werden gleichzeitig ausgewertet, unabhängig von der Priorität der Regeln. Nachdem alle Regeln ausgewertet wurden, werden sie entsprechend der Regelpriorität angewendet.
- Ein Image läuft mit genau einer oder null Regeln ab.
- Ein Image, das mit den Markierungsanforderungen einer Regel übereinstimmt, kann nicht durch eine Regel mit niedrigerer Priorität ablaufen.
- Regeln können keine Images markieren, die mit Regeln höherer Priorität gekennzeichnet sind, aber sie können sie identifizieren, als wären sie nicht abgelaufen.
- Die Regelmenge muss eine eindeutige Menge an Tag-Präfixen enthalten.
- Es ist nur eine Regel zulässig, die nicht markierte Images auswählt.
- Wenn ein Image von einer Manifestliste referenziert wird, kann es nicht abgelaufen sein, ohne dass die Manifestliste zuvor gelöscht wurde.
- Das Ablaufen wird immer nach `pushed_at_time` sortiert, und ältere Images laufen immer vor neueren ab.
- 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.
- 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 der Filter `prod*` angewendet wird, werden Repositorys gefunden, deren Name mit `prod` beginnt, zum Beispiel `prod`, `prod1` oder `production-team1`. Wenn hingegen der Filter `*prod*` angewendet wird, werden auch Repositorys gefunden, deren Name `prod` enthält, zum Beispiel `repo-production` oder `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 der `tagPrefixList` stimmt ein Image erfolgreich überein, wenn alle Tags im `tagPrefixList`-Wert mit den Tags des Images übereinstimmen.
- Der `countUnit`-Parameter wird nur verwendet, wenn `countType` `sinceImagePushed` ist.
- Mit `countType = imageCountMoreThan` werden Images vom neuesten zum ältesten sortiert, basierend auf `pushed_at_time`, und anschließend laufen alle Images ab, die größer als der vorgegebene Zähler sind.
- Mit `countType = sinceImagePushed` laufen alle Images ab, deren `pushed_at_time` älter als die angegebene Anzahl an Tagen basierend auf `countNumber` ist.

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 Management Console)

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

    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.

    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 unter Suchkriterien entweder Seit Image-Push oder Image-Anzahl mehr als aus und geben Sie dann einen Wert an.

g. 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 davon ausgehen, dass nach Anwendung einer Lebenszyklusrichtlinie die betroffenen Images innerhalb von 24 Stunden ablaufen.
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, nach denen ungenutzte Repository-Images ablaufen. Nach der Erstellung einer Lebenszyklus-Richtlinie sind die betroffenen Images innerhalb von 24 Stunden abgelaufen.

> **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 gemäß Ihren Lebenszyklus-Policy-Regeln abgelaufenen Images Ihren Vorstellungen entsprechen. Anweisungen finden Sie unter Erstellen einer Lifecycle-Policy-Vorschau in Amazon ECR.

So erstellen Sie eine Lebenszyklusrichtlinie (AWS Management Console)

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

    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.

    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 unter Suchkriterien entweder Seit Image-Push oder Image-Anzahl mehr als aus und geben Sie dann einen Wert an.

g. 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>,
                    "countType": "imageCountMoreThan"|"sinceImagePushed",
                    "countUnit": "string",
                    "countNumber": integer
                },
                "action": {
                    "type": "expire"
                }
            }
        ]
    }
```

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.

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 usw. 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.

Beschreibung

`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 `tagged` angeben, müssen Sie auch einen `tagPrefixList`-Wert angeben. Wenn Sie `untagged` angeben, müssen Sie `tagPrefixList` weglassen.

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.

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 `countType` auf `sinceImagePushed` gesetzt ist, geben Sie auch `countUnit` und `countNumber` an, um eine zeitliche Obergrenze für die Images festzulegen, die in Ihrem Repository vorhanden sind.

Zähleinheit

`countUnit`

Typ: Zeichenkette

Erforderlich: ja, nur wenn `countType` auf `sinceImagePushed` gesetzt ist

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 `countType` `sinceImagePushed` ist. Es tritt ein Fehler auf, wenn Sie eine Zähleinheit angeben, wenn für `countType` ein anderer Wert angegeben ist.

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.

Aktion

`type`

Typ: Zeichenfolge

Erforderlich: Ja

Geben Sie einen Aktionstyp an. Der unterstützte Wert ist `expire`.

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. Weitere Informationen zu den Compliance-Programmen, die für Amazon geltenECR, finden Sie unter AWS Services in Scope by Compliance Program.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, 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 gemeinsamen Verantwortung bei der Nutzung von Amazon anwenden könnenECR. In den folgenden Themen erfahren Sie, wie Sie Amazon konfigurierenECR, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, mit denen Sie Ihre ECR Amazon-Ressourcen überwachen und sichern können.

Themen

Identity and Access Management für Amazon Elastic Container Registry

AWS Identity and Access Management (IAM) hilft einem Administrator AWS -Service , den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAMAdministratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um ECR Amazon-Ressourcen zu nutzen. IAMist eine AWS -Service , die Sie ohne zusätzliche Kosten verwenden können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie bei Amazon erledigenECR.

Servicebenutzer — Wenn Sie den ECR Amazon-Service für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Da Sie für Ihre Arbeit mehr ECR Amazon-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie auf eine Funktion in Amazon nicht zugreifen könnenECR, finden Sie weitere Informationen unterFehlerbehebung bei Amazon Elastic Container Registry - Identität und Zugriff.

Service-Administrator — Wenn Sie in Ihrem Unternehmen für die ECR Amazon-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf AmazonECR. Es ist Ihre Aufgabe,

zu bestimmen, auf welche ECR Amazon-Funktionen und -Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anfragen an Ihren IAM Administrator richten, um die Berechtigungen Ihrer Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die grundlegenden Konzepte von zu verstehenIAM. Weitere Informationen darüber, wie Ihr Unternehmen Amazon nutzen IAM kannECR, finden Sie unterSo funktioniert Amazon Elastic Container Registry mit IAM.

IAMAdministrator — Wenn Sie ein IAM Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf Amazon zu verwaltenECR. Beispiele für ECR identitätsbasierte Amazon-Richtlinien, die Sie in verwenden könnenIAM, finden Sie unter. 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 als IAM Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM Rolle übernehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAMIdentity Center-) Nutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als föderierte Identität anmelden, hat Ihr Administrator zuvor einen Identitätsverbund mithilfe von Rollen eingerichtet. IAM Wenn Sie AWS mithilfe eines Verbunds darauf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, mit der Sie Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch signieren können. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAMBenutzerhandbuch unter AWS API Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere

Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) AWS im IAM Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS -Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Der Zugriff erfolgt, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie im Benutzerhandbuch unter Aufgaben, für die Root-Benutzeranmeldedaten erforderlich sind. IAM

IAM-Benutzer und -Gruppen

Ein IAMBenutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wir empfehlen, sich nach Möglichkeit auf temporäre Anmeldeinformationen zu verlassen, anstatt IAM Benutzer mit langfristigen Anmeldeinformationen wie Passwörtern und Zugriffsschlüsseln zu erstellen. Wenn Sie jedoch spezielle Anwendungsfälle haben, für die langfristige Anmeldeinformationen von IAM Benutzern erforderlich sind, empfehlen wir, die Zugriffsschlüssel abwechselnd zu verwenden. Weitere Informationen finden Sie im Benutzerhandbuch unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, für die IAM langfristige Anmeldeinformationen erforderlich sind.

Eine IAMGruppe ist eine Identität, die eine Sammlung von IAM Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Wann sollte ein IAM Benutzer (statt einer Rolle) erstellt werden? im IAMBenutzerhandbuch.

IAMRollen

Eine IAMRolle ist eine Identität innerhalb von Ihnen AWS-Konto , für die bestimmte Berechtigungen gelten. Sie ähnelt einem IAM Benutzer, ist jedoch keiner bestimmten Person zugeordnet. Sie können vorübergehend eine IAM Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI AWS API OR-Operation aufrufen oder eine benutzerdefinierte Operation verwendenURL. Weitere Informationen zu Methoden zur Verwendung von Rollen finden Sie unter Verwenden von IAM Rollen im IAMBenutzerhandbuch.

IAMRollen mit temporären Anmeldeinformationen sind in den folgenden Situationen nützlich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle für einen externen Identitätsanbieter. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Um zu kontrollieren, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in. IAM Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM Benutzerberechtigungen — Ein IAM Benutzer oder eine Rolle kann eine IAM Rolle übernehmen, um vorübergehend verschiedene Berechtigungen für eine bestimmte Aufgabe zu übernehmen.
- Kontoübergreifender Zugriff — Sie können eine IAM Rolle verwenden, um einer Person (einem vertrauenswürdigen Principal) in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS -Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM
- Serviceübergreifender Zugriff — Einige AWS -Services verwenden Funktionen in anderen. AWS -Services Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Zugriffssitzungen weiterleiten (FAS) — Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services

könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FASverwendet die Berechtigungen des Prinzipals, der an aufruft AWS -Service, kombiniert mit der Anforderung, Anfragen AWS -Service an nachgelagerte Dienste zu stellen. FASAnfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS -Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien beim Stellen von FAS Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle — Eine Servicerolle ist eine IAMRolle, die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM Administrator kann eine Servicerolle von innen heraus erstellen, ändern und löschenIAM. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle zum Delegieren von Berechtigungen AWS -Service an eine.
- Dienstbezogene Rolle — Eine dienstverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS -Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Verwenden einer IAM Rolle zur Erteilung von Berechtigungen für Anwendungen, die auf EC2 Amazon-Instances ausgeführt werden.

Informationen darüber, ob Sie IAM Rollen oder IAM Benutzer verwenden sollten, finden Sie im Benutzerhandbuch unter Wann sollte eine IAM Rolle (anstelle eines IAM Benutzers) erstellt werden.

Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den

Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS Form von JSON Dokumenten gespeichert. Weitere Informationen zur Struktur und zum Inhalt von JSON Richtliniendokumenten finden Sie im IAMBenutzerhandbuch unter Überblick über JSON Richtlinien.

Administratoren können mithilfe von AWS JSON Richtlinien festlegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

IAMRichtlinien definieren Berechtigungen für eine Aktion, unabhängig von der Methode, mit der Sie den Vorgang ausführen. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen aus dem AWS Management Console AWS CLI, dem oder dem abrufen AWS API.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen IAM Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie unter IAMRichtlinien erstellen im Benutzerhandbuch. IAM

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können. AWS-Konto Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie oder einer Inline-Richtlinie wählen können, finden Sie im IAMBenutzerhandbuch unter Auswahl zwischen verwalteten Richtlinien und Inline-Richtlinien.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und

Amazon S3 S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS -Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien nicht IAM in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen — Eine Berechtigungsgrenze ist eine erweiterte Funktion, mit der Sie die maximalen Berechtigungen festlegen, die eine identitätsbasierte Richtlinie einer IAM Entität (IAMBenutzer oder Rolle) gewähren kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen zu Berechtigungsgrenzen finden Sie im IAMBenutzerhandbuch unter Berechtigungsgrenzen für IAM Entitäten.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen AWS Organizations. AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer AWS-Konten Unternehmenseigentümer. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und

der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Sitzungsrichtlinien.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAMBenutzerhandbuch unter Bewertungslogik für Richtlinien.

So funktioniert Amazon Elastic Container Registry mit IAM

Bevor Sie IAM den Zugriff auf Amazon verwaltenECR, sollten Sie sich darüber im Klaren sein, welche IAM Funktionen für Amazon verfügbar sindECR. Einen umfassenden Überblick darüber, wie Amazon ECR und andere AWS Dienste zusammenarbeitenIAM, finden Sie unter AWS Services That Work with IAM im IAMBenutzerhandbuch.

Themen

- ECRIdentity-basierte Richtlinien von Amazon
- ECRRessourcenbasierte Richtlinien von Amazon
- Autorisierung basierend auf ECR Amazon-Tags
- ECRIAMRollen bei Amazon

ECRIdentity-basierte Richtlinien von Amazon

Mit IAM identitätsbasierten Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zulässig oder verweigert werden. Amazon ECR unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Weitere Informationen zu allen Elementen, die Sie in einer JSON Richtlinie verwenden, finden Sie unter IAMJSONPolicy Elements Reference im IAMBenutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Action` Element einer JSON Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, für die nur eine Genehmigung erforderlich ist und für die es keinen entsprechenden Vorgang gibt. API Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Richtlinienaktionen in Amazon ECR verwenden vor der Aktion das folgende Präfix:`ecr:`. Um beispielsweise jemandem die Erlaubnis zu erteilen, ein ECR Amazon-Repository mit dem ECR `CreateRepository` API Amazon-Vorgang 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 eigene Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service ausführen 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 ECR Amazon-Aktionen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Elastic Container Registry im IAMBenutzerhandbuch.

Ressourcen

Administratoren können mithilfe von AWS JSON Richtlinien festlegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Resource` JSON Richtlinienelement gibt das Objekt oder die Objekte an, für die die Aktion gilt. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Es hat sich bewährt, eine Ressource mit ihrem Amazon-Ressourcennamen (ARN) anzugeben. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine ECR Amazon-Repository-Ressource hat FolgendesARN:

```
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 Ihrem Kontoauszug anzugeben, verwenden Sie Folgendes: 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 ARNs durch Kommas.

```
"Resource": [
      "resource1",
      "resource2"
```

Eine Liste der ECR Amazon-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von Amazon Elastic Container Registry definierte Ressourcen im IAMBenutzerhandbuch. Informationen

darüber, mit welchen Aktionen Sie die ARN einzelnen Ressourcen spezifizieren können, finden Sie unter Von Amazon Elastic Container Registry definierte Aktionen.

Bedingungsschlüssel

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. 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.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Sie können einem IAM Benutzer beispielsweise nur dann Zugriff auf eine Ressource gewähren, wenn sie mit seinem IAM Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMRichtlinienelemente: Variablen und Tags.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAMBenutzerhandbuch.

Amazon ECR definiert seinen eigenen Satz von Bedingungsschlüsseln und unterstützt auch die Verwendung einiger globaler Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAMBenutzerhandbuch.

Die meisten ECR Amazon-Aktionen unterstützen die Tasten `aws:ResourceTag` und `ecr:ResourceTag` Condition. Weitere Informationen finden Sie unter Verwenden Tag-basierter Zugriffskontrolle.

Eine Liste der ECR Amazon-Bedingungsschlüssel finden Sie unter Condition Keys Defined by Amazon Elastic Container Registry im IAMBenutzerhandbuch. Um zu erfahren, mit welchen Aktionen

und Ressourcen Sie einen Bedingungsschlüssel verwenden können, siehe Actions Defined by Amazon Elastic Container Registry.

Beispiele

Beispiele für ECR identitätsbasierte Richtlinien von Amazon finden Sie unter. Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien

ECRRessourcenbasierte Richtlinien von Amazon

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die festlegen, welche Aktionen ein bestimmter Principal unter welchen Bedingungen auf einer ECR Amazon-Ressource ausführen kann. Amazon ECR unterstützt ressourcenbasierte Berechtigungsrichtlinien für ECR Amazon-Repositorys. Ressourcenbasierte Richtlinien ermöglichen die Erteilung von Nutzungsberechtigungen für andere -Konten pro Ressource. Sie können auch eine ressourcenbasierte Richtlinie verwenden, um einem AWS Service den Zugriff auf Ihre ECR Amazon-Repositorys zu ermöglichen.

Um den kontoübergreifenden Zugriff zu ermöglichen, können Sie in einer ressourcenbasierten Richtlinie ein ganzes Konto oder IAM Entitäten in einem anderen Konto als Hauptbenutzer angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal 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, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Unterschiede zwischen IAM Rollen und ressourcenbasierten Richtlinien.

Der ECR Amazon-Service unterstützt nur eine Art von ressourcenbasierter Richtlinie, die als Repository-Richtlinie bezeichnet wird und an ein Repository angehängt ist. 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 ECR Amazon-Repository-Richtlinie `Sid` unterstützt das Policy-Element zusätzliche Zeichen und Leerzeichen, die in IAM Richtlinien nicht unterstützt werden.

Beispiele

Beispiele für ECR ressourcenbasierte Richtlinien von Amazon finden Sie unter Beispiele für Richtlinien für private Repositorien in Amazon ECR

Autorisierung basierend auf ECR Amazon-Tags

Sie können Tags an ECR Amazon-Ressourcen anhängen oder Tags in einer Anfrage an Amazon weitergebenECR. 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 Bedingung `aws:TagKeys` verwenden. Weitere Informationen zum Taggen von ECR Amazon-Ressourcen finden Sie unterKennzeichnen 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.

ECRIAMRollen bei Amazon

Eine IAMRolle ist eine Entität innerhalb Ihres AWS Kontos, die über bestimmte Berechtigungen verfügt.

Temporäre Anmeldeinformationen mit Amazon verwenden ECR

Sie können temporäre Anmeldeinformationen verwenden, um sich bei Federation anzumelden, eine IAM Rolle zu übernehmen oder eine kontoübergreifende 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 durchzuführen. Mit Diensten verknüpfte Rollen werden in Ihrem IAM Konto angezeigt und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.

Amazon ECR unterstützt servicebezogene Rollen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für Amazon ECR.

Beispiele für identitätsbasierte Amazon Elastic Container Service-Richtlinien

Standardmäßig sind Benutzer und Rollen nicht berechtigt, ECR Amazon-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe von AWS Management Console, AWS Command Line Interface (AWS CLI) oder ausführen AWS API. Um Benutzern die Berechtigung zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

Informationen zum Erstellen einer IAM identitätsbasierten Richtlinie anhand dieser JSON Beispieldokumente finden Sie unter IAMRichtlinien erstellen im IAMBenutzerhandbuch.

Einzelheiten zu den von Amazon definierten Aktionen und RessourcentypenECR, 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 zum Erstellen einer IAM identitätsbasierten Richtlinie anhand dieser JSON Beispieldokumente finden Sie im IAMBenutzerhandbuch unter Creating Policies on the JSON Tab.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand ECR Amazon-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder diese löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst 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 AWS im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien oder Verwaltete Richtlinien für Jobfunktionen.

- Berechtigungen mit den geringsten Rechten anwenden — Wenn Sie Berechtigungen mit IAM Richtlinien festlegen, gewähren Sie nur die Berechtigungen, die für die Ausführung 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 IAM zum Anwenden von Berechtigungen finden Sie IAMim Benutzerhandbuch unter Richtlinien und Berechtigungen. IAM
- Verwenden Sie Bedingungen in IAM Richtlinien, um den Zugriff weiter einzuschränken — Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen einzuschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um anzugeben, dass alle Anfragen über gesendet werden müssenSSL. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese über einen bestimmten Zweck verwendet werden AWS -Service, z. AWS CloudFormation B. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMJSONRichtlinienelemente: Bedingung.
- Verwenden Sie IAM Access Analyzer, um Ihre IAM Richtlinien zu validieren, um sichere und funktionale Berechtigungen zu gewährleisten. IAM Access Analyzer validiert neue und bestehende Richtlinien, sodass die Richtlinien der IAM Richtliniensprache (JSON) und den IAM bewährten Methoden entsprechen. IAMAccess Analyzer bietet mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen, um Sie bei der Erstellung sicherer und funktionaler Richtlinien zu unterstützen. Weitere Informationen finden Sie unter IAMAccess Analyzer-Richtlinienvalidierung im IAMBenutzerhandbuch.
- Multi-Faktor-Authentifizierung erforderlich (MFA) — Wenn Sie ein Szenario haben, in dem IAM Benutzer oder ein Root-Benutzer erforderlich sind AWS-Konto, aktivieren Sie die Option MFA für zusätzliche Sicherheit. Um festzulegen, MFA wann API Operationen aufgerufen werden, fügen Sie MFA Bedingungen zu Ihren Richtlinien hinzu. Weitere Informationen finden Sie unter Konfiguration des MFA -geschützten API Zugriffs im IAMBenutzerhandbuch.

Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Sicherheitsmethoden IAM im IAM Benutzerhandbuch. IAM

Verwenden der ECR Amazon-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 ECR Amazon-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 ECR Amazon-Konsole verwenden können, fügen Sie die `AmazonEC2ContainerRegistryReadOnly` AWS verwaltete Richtlinie zu den Entitäten hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen für einen IAM Benutzer im Benutzerhandbuch:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
                "ecr:DescribeImageScanFindings"
            ],
            "Resource": "*"
        }
    ]
}
```

Sie müssen Benutzern, die nur Anrufe an AWS CLI oder den tätigen, keine Mindestberechtigungen für die Konsole gewähren AWS API. Erlauben Sie stattdessen nur den Zugriff auf die Aktionen, die dem API Vorgang entsprechen, den Sie ausführen möchten.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es IAM Benutzern ermöglicht, die Inline- und verwalteten Richtlinien einzusehen, die mit ihrer Benutzeridentität verknüpft sind.

Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe von oder. AWS CLI AWS API

```
{
    "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 ECR Amazon-Repository

In diesem Beispiel möchten Sie einem Benutzer in Ihrem AWS Konto Zugriff auf eines Ihrer ECR Amazon-Repositorys gewähren. `my-repo` Sie möchten dem Benutzer auch erlauben, Images zu übertragen, abzurufen und aufzulisten.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListImagesInRepository",
         "Effect":"Allow",
         "Action":[
            "ecr:ListImages"
         ],
         "Resource":"arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
      },
      {
         "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 ECR CreateRepository API Amazon-Aktion können Sie beim Erstellen des Repositorys Tags angeben. 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 tagbasierte Zugriffskontrolle mithilfe von 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.

```
{
    "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 einem Benutzer den Zugriff auf alle Repositorys ermöglichen, sofern diese nicht als `key=environment,value=prod` gekennzeichnet sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ecr:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ecr:*",
            "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 wird AWS. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, 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 kundenverwaltete 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 im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien.

Amazon ECR bietet mehrere verwaltete Richtlinien, die Sie IAM Identitäten oder EC2 Amazon-Instances zuordnen können. Diese verwalteten Richtlinien ermöglichen ein unterschiedliches Maß an Kontrolle über den Zugriff auf ECR Ressourcen und API Abläufe von Amazon. Weitere Informationen zu den einzelnen API in diesen Richtlinien genannten Vorgängen finden Sie unter Aktionen in der Amazon Elastic Container Registry API Reference.

Themen

- AmazonEC2ContainerRegistryFullAccess
- AmazonEC2ContainerRegistryPowerUser
- AmazonEC2ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- ECRTemplateServiceRolePolicy
- ECRAktualisierungen der AWS verwalteten Richtlinien durch Amazon

`AmazonEC2ContainerRegistryFullAccess`

Sie können die `AmazonEC2ContainerRegistryFullAccess` Richtlinie an Ihre IAM Identitäten anhängen.

Sie können diese verwaltete Richtlinie als Ausgangspunkt verwenden, um Ihre eigene IAM Richtlinie auf der Grundlage Ihrer spezifischen Anforderungen zu erstellen. Sie können beispielsweise eine Richtlinie speziell dafür erstellen, einem Benutzer oder einer Rolle vollen Administratorzugriff zur Verwaltung der Nutzung von Amazon zu gewährenECR. Mit der Funktion Amazon ECR Lifecycle Policies können Sie das Lebenszyklusmanagement von Bildern in einem Repository spezifizieren.

Ereignisse im Rahmen von Lebenszyklusrichtlinien werden als CloudTrail Ereignisse gemeldet. Amazon ECR ist integriert, AWS CloudTrail sodass es Ihre Lifecycle-Policy-Ereignisse direkt in der ECR Amazon-Konsole anzeigen kann. Die `AmazonEC2ContainerRegistryFullAccess` verwaltete IAM Richtlinie beinhaltet die `cloudtrail:LookupEvents` Erlaubnis, dieses Verhalten zu ermöglichen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `ecr`— Ermöglicht Prinzipalen den vollen Zugriff auf alle Amazon ECRAPIs.
- `cloudtrail`— Ermöglicht Prinzipalen, nach Verwaltungsereignissen oder AWS CloudTrail Insights-Ereignissen zu suchen, die von erfasst wurden. CloudTrail

Die`AmazonEC2ContainerRegistryFullAccess` Politik ist wie folgt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "cloudtrail:LookupEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "replication.ecr.amazonaws.com"
                    ]
                }
            }
        }
    ]
```

```
}
```

AmazonEC2ContainerRegistryPowerUser

Sie können die `AmazonEC2ContainerRegistryPowerUser` Richtlinie an Ihre IAM Identitäten anhängen.

Diese Richtlinie gewährt Benutzern Administratorberechtigungen, die es IAM Benutzern ermöglichen, in Repositorys zu lesen und zu schreiben, erlaubt ihnen jedoch nicht, Repositorys zu löschen oder die für sie geltenden Richtliniendokumente zu ändern.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `ecr` – Ermöglicht Auftraggebern das Lesen und Schreiben auf Respositorys sowie das Lesen von Lebenszyklusrichtlinien. Auftraggeber sind nicht berechtigt, Repositorys zu löschen oder die auf sie angewandten Lebenszyklusrichtlinien zu ändern.

Die `AmazonEC2ContainerRegistryPowerUser`Politik ist wie folgt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
                "ecr:DescribeImageScanFindings",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
```

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

AmazonEC2ContainerRegistryReadOnly

Sie können die AmazonEC2ContainerRegistryReadOnly Richtlinie an Ihre IAM Identitäten anhängen.

Diese Richtlinie gewährt Amazon nur Leseberechtigungen. ECR Dazu gehört auch die Möglichkeit, Repositorys und Images innerhalb der Repositories aufzulisten. Es beinhaltet auch die Möglichkeit, ECR mit dem Docker CLI Bilder von Amazon abzurufen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen:

- ecr – Ermöglicht Auftraggebern das Lesen von Repositorys und deren jeweiligen Lebenszyklusrichtlinien.

Die AmazonEC2ContainerRegistryReadOnlyPolitik ist wie folgt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
```

```
                "ecr:DescribeImageScanFindings"
            ],
            "Resource": "*"
        }
    ]
}
```

AWSECRPullThroughCache_ServiceRolePolicy

Sie können die AWSECRPullThroughCache_ServiceRolePolicy verwaltete IAM Richtlinie nicht an Ihre IAM Entitäten anhängen. Diese Richtlinie ist mit einer serviceverknüpften Rolle verknüpft, die es Amazon ermöglicht, Bilder über ECR den Pull-Through-Cache-Workflow in Ihre Repositorys zu übertragen. Weitere Informationen finden Sie unter ECRService-verknüpfte Rolle mit Amazon für den Pull-Through-Cache.

ECRReplicationServiceRolePolicy

Sie können die ECRReplicationServiceRolePolicy verwaltete IAM Richtlinie nicht an Ihre IAM Entitäten anhängen. Diese Richtlinie ist mit einer dienstbezogenen Rolle verknüpft, die es Amazon ECR ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für Amazon ECR.

ECRTemplateServiceRolePolicy

Sie können die ECRTemplateServiceRolePolicy verwaltete IAM Richtlinie nicht an Ihre IAM Entitäten anhängen. Diese Richtlinie ist mit einer dienstbezogenen Rolle verknüpft, die es Amazon ECR ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für Amazon ECR.

ECRAktualisierungen der AWS verwalteten Richtlinien durch Amazon

Hier finden Sie Informationen zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon ECR seit Beginn der Nachverfolgung dieser Änderungen durch diesen Service. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS Feed auf der Amazon ECR Document-Verlaufsseite.

Änderung	Beschreibung	Datum
ECRTemplateService RolePolicy – Neue Richtlinie.	Amazon ECR hat eine neue Richtlinie hinzugefügt.	20. Juni 2024

Änderung	Beschreibung	Datum
	Diese Richtlinie ist mit der `ECRTemplateServiceRolePolicy` serviceverknüpften Rolle für die Funktion „Vorlage zur Erstellung von Repositorys“ verknüpft.	
AWSECRPullThroughCache_ServiceRolePolicy — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AWSECRPullThroughCache_ServiceRolePolicy` Richtlinie neue Berechtigungen hinzugefügt. Diese Berechtigungen ermöglichen AmazonECR, den verschlüsselten Inhalt eines Secrets Manager 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 erforderlich ist.	15. November 2023
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

Änderung	Beschreibung	Datum
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 EC2ContainerRegistryFullAccess — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryFullAccess` Richtlinie neue Berechtigungen hinzugefügt. Diese Berechtigungen ermöglichen es Principals, die mit dem ECR Amazon-Dienst verknüpfte Rolle zu erstellen.	4. Dezember 2020
Amazon EC2ContainerRegistryReadOnly — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryReadOnly` Richtlinie neue Berechtigungen hinzugefügt, die es Principals ermöglichen, Lebenszyklusrichtlinien zu lesen, Tags aufzulisten und die Scanergebnisse für Bilder zu beschreiben.	10. Dezember 2019

Änderung	Beschreibung	Datum
Amazon EC2ContainerRegistryPowerUser — 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 EC2ContainerRegistryFullAccess — 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
Amazon EC2ContainerRegistryReadOnly — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryReadOnly` Richtlinie neue Berechtigungen hinzugefügt. Sie ermöglichen es den Schulleitern, ECR Amazon-Bilder zu beschreiben.	11. Oktober 2016

Änderung	Beschreibung	Datum
Amazon EC2ContainerRegistryPowerUser — Aktualisierung einer bestehenden Richtlinie	Amazon ECR hat der `AmazonEC2ContainerRegistryPowerUser` Richtlinie neue Berechtigungen hinzugefügt. Sie ermöglichen es den Schulleitern, ECR Amazon-Bilder zu beschreiben.	11. Oktober 2016
Amazon EC2ContainerRegistryReadOnly — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt, die Amazon nur Leseberechtigungen gewährt. ECR Diese Berechtigungen umfassen die Möglichkeit, Repositories und Images innerhalb der Repositories aufzulisten. Sie beinhalten auch die Möglichkeit, ECR mit dem Docker CLI Bilder von Amazon abzurufen.	21. Dezember 2015
Amazon EC2ContainerRegistryPowerUser — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt, die Benutzern Administratorrechte gewährt, die es Benutzern ermöglichen, in Repositorys zu lesen und zu schreiben, es ihnen jedoch nicht erlaubt, Repositorys zu löschen oder die für sie geltenden Richtliniendokumente zu ändern.	21. Dezember 2015

Änderung	Beschreibung	Datum
Amazon EC2ContainerRegistryFullAccess — Neue Richtlinie	Amazon ECR hat eine neue Richtlinie hinzugefügt. Diese Richtlinie gewährt vollen Zugriff auf AmazonECR.	21. Dezember 2015
Amazon ECR hat begonnen, Änderungen zu verfolgen	Amazon ECR hat damit begonnen, Änderungen für AWS verwaltete Richtlinien nachzuverfolgen.	24. Juni 2021

Verwenden von serviceverknüpften Rollen für Amazon ECR

Amazon Elastic Container Registry (AmazonECR) 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 servicebezogene Rolle ist ein einzigartiger IAM Rollentyp, der 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 serviceverknüpfte Rolle erleichtert die Einrichtung der Replikation und den Pull-Through-Cache mit AmazonECR. Das liegt daran, dass Sie bei Verwendung dieser Rolle nicht alle erforderlichen Berechtigungen manuell hinzufügen müssen. Amazon ECR definiert die Berechtigungen seiner serviceverknüpften Rollen, und sofern nicht anders definiert, ECR kann nur Amazon seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Die Berechtigungsrichtlinie kann keiner anderen IAM Entität zugeordnet 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. Dadurch wird sichergestellt, dass Sie nicht versehentlich die Berechtigungen entfernen, die Amazon für diese Funktionen ECR benötigt.

Informationen zu anderen Diensten, die dienstbezogene Rollen unterstützen, finden Sie unter AWS Dienste, die mit Diensten funktionieren. IAM 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 Rollen im ECR Zusammenhang mit Amazon Services
- ECRServicebezogene Rolle mit Amazon für die Replikation
- ECRService-verknüpfte Rolle mit Amazon für den Pull-Through-Cache
- Mit dem Amazon ECR Service verknüpfte Rolle für Vorlagen zur Repository-Erstellung

Unterstützte Regionen für Rollen im ECR Zusammenhang mit Amazon Services

Amazon ECR unterstützt die Verwendung von servicebezogenen Rollen in allen Regionen, in denen der ECR Amazon-Service verfügbar ist. Weitere Informationen zur Verfügbarkeit ECR in der Amazon-Region finden Sie unter AWS Regionen und Endpunkte.

ECRServicebezogene Rolle mit Amazon für die Replikation

Amazon ECR verwendet eine servicebezogene Rolle mit dem Namen AWSServiceRoleForECRReplication, die es Amazon ermöglicht, Bilder über mehrere Konten hinweg ECR zu replizieren.

Servicebezogene Rollenberechtigungen für Amazon ECR

Die AWSServiceRoleForECRReplication dienstbezogene Rolle vertraut darauf, dass die folgenden Dienste die Rolle übernehmen:

- `replication.ecr.amazonaws.com`

Die folgende Richtlinie für `ECRReplicationServiceRolePolicy` Rollenberechtigungen ermöglicht AmazonECR, die folgenden Aktionen für Ressourcen zu verwenden:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository",
                "ecr:ReplicateImage"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> Das `ReplicateImage` ist ein internesAPI, das Amazon für die Replikation ECR verwendet und nicht direkt aufgerufen werden kann.

Sie müssen Berechtigungen konfigurieren, damit eine IAM Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstbezogene Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Berechtigungen für dienstverknüpfte Rollen im IAMBenutzerhandbuch.

Eine servicebezogene Rolle für Amazon erstellen ECR

Sie müssen die mit dem Amazon ECR Service verknüpfte Rolle nicht manuell erstellen. Wenn Sie die Replikationseinstellungen für Ihre Registrierung in der AWS Management Console, der AWS CLI oder der konfigurieren AWS API, ECR erstellt Amazon 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, ECR erstellt Amazon die serviceverknüpfte Rolle erneut für Sie.

Bearbeitung einer serviceverknüpften Rolle für Amazon ECR

Amazon erlaubt ECR es nicht, die AWSServiceRoleForECRReplication serviceverknüpfte 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 die Beschreibung der Rolle jedoch mithilfe IAM von bearbeiten. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Bearbeiten einer dienstbezogenen Rolle.

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 ECR Amazon-Service die Rollen noch verwendet, schlägt Ihre Löschaktion möglicherweise fehl. Wenn dies der Fall ist, warten Sie einige Minuten und versuchen Sie es erneut.

Um ECR Amazon-Ressourcen zu löschen, die verwendet werden von AWSServiceRoleForECRReplication

1. Öffnen Sie die ECR Amazon-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.

Um die mit dem Service verknüpfte Rolle manuell zu löschen, verwenden Sie IAM

Verwenden Sie die IAM Konsole, den oder AWS CLI, AWS API um die AWSServiceRoleForECRReplicationdienstverknüpfte Rolle zu löschen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Löschen einer dienstbezogenen Rolle.

ECRService-verknüpfte Rolle mit Amazon für den Pull-Through-Cache

Amazon ECR verwendet eine servicebezogene Rolle mit AWSServiceRoleForECRPullThroughCachedem Namen, die Amazon die Erlaubnis erteiltECR, in Ihrem Namen Aktionen durchzuführen, um Cache-Aktionen durchzuführen. Weitere Informationen zum Pull-Through-Cache finden Sie unter Vorlagen zur Steuerung von Repositorys, die während einer Pull-Through-Cache- oder Replikationsaktion erstellt wurden.

Servicebezogene Rollenberechtigungen für Amazon ECR

Die AWSServiceRoleForECRPullThroughCachedienstbezogene Rolle vertraut darauf, dass der folgende Dienst die Rolle übernimmt.

- `pullthroughcache.ecr.amazonaws.com`

Details zu Berechtigungen

Die Berechtigungsrichtlinie `AWSECRPullThroughCache_ServiceRolePolicy` ist mit der dienstverknüpften Rolle verbunden. Diese verwaltete Richtlinie erteilt Amazon die ECR Erlaubnis, die folgenden Aktionen durchzuführen. Weitere Informationen finden Sie unter `AWSECRPullThroughCache_ServiceRolePolicy`.

- `ecr`— Ermöglicht dem ECR Amazon-Service, Bilder in ein privates Repository zu übertragen.
- `secretsmanager:GetSecretValue`— Ermöglicht dem ECR Amazon-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 beinhaltet FolgendesJSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECR",
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManager",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
```

```
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecr-pullthroughcache/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine IAM Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstbezogene Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Berechtigungen für dienstbezogene Rollen im IAMBenutzerhandbuch.

Eine servicebezogene Rolle für Amazon erstellen ECR

Sie müssen die mit dem Amazon ECR Service verknüpfte Rolle für den Pull-Through-Cache nicht manuell erstellen. Wenn Sie eine Pull-Through-Cache-Regel für Ihre private Registrierung im AWS Management Console, dem AWS CLI oder dem erstellen AWS API, ECR erstellt Amazon 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, ECR erstellt Amazon die serviceverknüpfte Rolle erneut für Sie, sofern sie noch nicht vorhanden ist.

Bearbeitung einer serviceverknüpften Rolle für Amazon ECR

Amazon erlaubt ECR 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 die Beschreibung der Rolle jedoch mithilfe IAM von bearbeiten. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Bearbeiten einer dienstbezogenen Rolle.

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 ECR Amazon-Service die Rolle noch verwendet, schlägt Ihre Löschaktion möglicherweise fehl. Wenn dies der Fall ist, warten Sie einige Minuten und versuchen Sie es erneut.

Um ECR Amazon-Ressourcen zu löschen, die von der AWSServiceRoleForECRPullThroughCacheserviceverknüpften Rolle verwendet werden

1. Öffnen Sie die ECR Amazon-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.

Um die mit dem Service verknüpfte Rolle manuell zu löschen, verwenden Sie IAM

Verwenden Sie die IAM Konsole, den oder AWS CLI, AWS API um die AWSServiceRoleForECRPullThroughCachedienstverknüpfte Rolle zu löschen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Löschen einer dienstbezogenen Rolle.

Mit dem Amazon ECR Service verknüpfte Rolle für Vorlagen zur Repository-Erstellung

Amazon ECR verwendet eine servicebezogene Rolle mit dem Namen AWSServiceRoleForECRTemplate, die Amazon die Erlaubnis gibt, in Ihrem Namen Aktionen durchzuführen, ECR um Aktionen zur Erstellung von Repository-Vorlagen abzuschließen.

Servicebezogene Rollenberechtigungen für Amazon ECR

Die AWSServiceRoleForECRTemplatedienstbezogene Rolle vertraut darauf, dass der folgende Dienst die Rolle übernimmt.

- `ecr.amazonaws.com`

Details zu Berechtigungen

Die Berechtigungsrichtlinie `ECRTemplateServiceRolePolicy` ist mit der dienstverknüpften Rolle verbunden. Diese verwaltete Richtlinie erteilt Amazon die ECR Erlaubnis, in Ihrem Namen Aktionen zur Erstellung eines Repositorys durchzuführen.

Die `ECRTemplateServiceRolePolicy` Richtlinie umfasst FolgendesJSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
     "Sid": "CreateRepositoryWithTemplate",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*"
        }
    ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine IAM Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstbezogene Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Berechtigungen für dienstbezogene Rollen im IAMBenutzerhandbuch.

Eine servicebezogene Rolle für Amazon erstellen 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 im AWS Management Console AWS CLI, dem oder dem erstellen AWS API, ECR erstellt Amazon 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, ECR erstellt Amazon die serviceverknüpfte Rolle erneut für Sie, sofern sie noch nicht existiert.

Bearbeitung einer serviceverknüpften Rolle für Amazon ECR

Amazon erlaubt ECR 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 die Beschreibung der Rolle jedoch mithilfe IAM von bearbeiten. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Bearbeiten einer dienstbezogenen Rolle.

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 ECR Amazon-Service die Rolle noch verwendet, schlägt Ihre Löschaktion möglicherweise fehl. Wenn dies der Fall ist, warten Sie einige Minuten und versuchen Sie es erneut.

Um ECR Amazon-Ressourcen zu löschen, die von der AWSServiceRoleForECRTemplateserviceverknüpften Rolle verwendet werden

1. Öffnen Sie die ECR Amazon-Konsole unter https://console.aws.amazon.com/ecr/.
2. Wählen Sie in der Navigationsleiste die Region aus, in der Ihre Regeln für die Erstellung Ihres Repositorys 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.

Um die mit dem Service verknüpfte Rolle manuell zu löschen, verwenden Sie IAM

Verwenden Sie die IAM Konsole, den oder AWS CLI, AWS API um die AWSServiceRoleForECRTemplatedienstverknüpfte Rolle zu löschen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Löschen einer dienstbezogenen Rolle.

Fehlerbehebung bei Amazon Elastic Container Registry - Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon ECR und auftreten könnenIAM.

Themen

Ich bin nicht berechtigt, eine Aktion bei Amazon durchzuführen ECR

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 `mateojackson` IAM Benutzer versucht, die Konsole zu verwenden, um Details zu einer fiktiven *`my-example-widget`* Ressource anzuzeigen, aber nicht über die fiktiven `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 eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Amazon weitergeben könnenECR.

Einige AWS -Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder eine dienstbezogene 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 auszuführenECR. 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 ECR Amazon-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden 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:

- Informationen darüber, ob Amazon diese Funktionen ECR unterstützt, finden Sie unterSo funktioniert Amazon Elastic Container Registry mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM Benutzerhandbuch unter Gewähren des Zugriffs auf einen anderen IAMBenutzer AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAMBenutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie Zugriff über einen Identitätsverbund gewähren, finden Sie im Benutzerhandbuch unter Zugriff für extern authentifizierte Benutzer (Identitätsverbund). IAM

- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM

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 im Abschnitt Datenschutz FAQ. Informationen zum Datenschutz in Europa finden Sie im AWS Shared Responsibility Model und im GDPR Blogbeitrag im AWS Security Blog.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre 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 eine Multi-Faktor-Authentifizierung (MFA).
- Verwenden SieSSL/TLS, um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Einrichtung API und Protokollierung von Benutzeraktivitäten mit AWS CloudTrail.
- 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 FIPS 140-3 validierte kryptografische Module für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine benötigenAPI, verwenden Sie einen Endpunkt. FIPS Weitere Informationen zu den verfügbaren FIPS Endpunkten finden Sie unter Federal Information Processing Standard () 140-3. FIPS

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 zusammenarbeiten und die Konsole AWS -Services verwendenAPI, AWS CLI, oder 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 einem externen Server eine URL zur Verfügung stellen, empfehlen wir dringend, dass Sie keine Anmeldeinformationen angeben, URL um Ihre Anfrage an diesen Server zu validieren.

Themen

Verschlüsselung im Ruhezustand

Amazon ECR speichert Bilder in Amazon S3 S3-Buckets, die Amazon ECR verwaltet. Standardmäßig ECR verwendet Amazon serverseitige Verschlüsselung mit von Amazon S3 verwalteten Verschlüsselungsschlüsseln, wodurch Ihre Daten im Ruhezustand mit einem AES -256-Verschlüsselungsalgorithmus verschlüsselt werden. Dies erfordert kein Handeln Ihrerseits und wird ohne zusätzliche Kosten angeboten. Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung mit Amazon S3-Managed Encryption Keys (SSE-S3) im Amazon Simple Storage Service-Benutzerhandbuch.

Für mehr Kontrolle über die Verschlüsselung Ihrer ECR Amazon-Repositorys können Sie serverseitige Verschlüsselung mit in AWS Key Management Service ()AWS KMS gespeicherten KMS Schlüsseln verwenden. Wenn Sie Ihre Daten verschlüsseln, können Sie entweder den Standard verwenden Von AWS verwalteter Schlüssel, der von Amazon verwaltet wirdECR, oder Ihren eigenen KMS Schlüssel angeben (als vom Kunden verwalteter Schlüssel bezeichnet). AWS KMS Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung mit in AWS KMS (SSE-KMS) gespeicherten KMS Schlüsseln im Amazon Simple Storage Service-Benutzerhandbuch.

Jedes ECR Amazon-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 Ein privates Amazon ECR-Repository zum Speichern von Bildern erstellen.

Wenn ein Repository mit aktivierter AWS KMS Verschlüsselung erstellt wird, wird ein KMS Schlüssel verwendet, um den Inhalt des Repositorys zu verschlüsseln. Darüber hinaus ECR fügt Amazon dem KMS Schlüssel einen AWS KMS Zuschuss hinzu, wobei das ECR Amazon-Repository als Principal des Zuschussempfängers fungiert.

Im Folgenden erhalten Sie einen umfassenden Überblick darüber, wie Amazon bei der Ver- und Entschlüsselung Ihrer Repositorys integriert ECR ist: AWS KMS

1. Beim Erstellen eines Repositorys ECR sendet Amazon 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 an, AWS KMS Zuschüsse für den KMS Schlüssel zu gewähren, damit Amazon ECR Daten mithilfe des Datenschlüssels ver- und entschlüsseln kann.
3. Beim Pushen eines Images wird eine GenerateDataKeyAnfrage an gesendet, 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 Bildmanifest 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, ECR sendet Amazon zwei RetireGrantAnfragen an, AWS KMS um die für das Repository erstellten Zuschüsse zurückzuziehen.

Überlegungen

Die folgenden Punkte sollten bei der Verwendung von AWS KMS Verschlüsselung mit Amazon berücksichtigt werdenECR.

- Wenn Sie Ihr ECR Amazon-Repository mit KMS Verschlüsselung erstellen und keinen KMS Schlüssel angeben, ECR verwendet Amazon `aws/ecr` standardmäßig einen Von AWS verwalteter Schlüssel mit dem Alias. Dieser KMS Schlüssel wird in Ihrem Konto erstellt, wenn Sie zum ersten Mal ein Repository mit aktivierter KMS Verschlüsselung erstellen.
- Wenn Sie die KMS Verschlüsselung mit Ihrem eigenen KMS Schlüssel verwenden, muss der Schlüssel in derselben Region wie Ihr Repository existieren.

- Die Zuschüsse, die ECR Amazon in Ihrem Namen gewährt, sollten nicht widerrufen werden. Wenn Sie die Genehmigung widerrufen, die Amazon die ECR Erlaubnis erteilt, die AWS KMS Schlüssel in Ihrem Konto zu verwenden, ECR kann Amazon nicht auf diese Daten zugreifen, keine neuen Bilder verschlüsseln, die in das Repository übertragen werden, oder sie entschlüsseln, wenn sie abgerufen werden. Wenn Sie einen Zuschuss für Amazon widerrufenECR, wird die Änderung sofort wirksam. Um Zugriffsrechte zu widerrufen, sollten Sie das Repository löschen, anstatt die Gewährung zu widerrufen. Wenn ein Repository gelöscht wird, ECR zieht Amazon die Zuschüsse in Ihrem Namen zurück.
- Die Verwendung von AWS KMS Schlüsseln ist mit Kosten verbunden. Weitere Informationen finden Sie unter AWS Key Management Service Preise.

Erforderliche IAM Berechtigungen

Beim Erstellen oder Löschen eines ECR Amazon-Repositorys mit serverseitiger Verschlüsselung hängen die erforderlichen Berechtigungen von dem spezifischen KMS Schlüssel ab AWS KMS, den Sie verwenden.

Erforderliche IAM Berechtigungen bei der Verwendung von Von AWS verwalteter Schlüssel für Amazon ECR

Wenn die AWS KMS Verschlüsselung für ein ECR Amazon-Repository aktiviert ist, aber kein KMS Schlüssel angegeben ist, ECR wird standardmäßig der Von AWS verwalteter Schlüssel für Amazon verwendet. Wenn der AWS-managed KMS key for Amazon verwendet ECR wird, um ein Repository zu verschlüsseln, kann jeder Principal, der über die Berechtigung zum Erstellen eines Repositorys verfügt, auch die AWS KMS Verschlüsselung für das Repository aktivieren. Der IAM Principal, der das Repository löscht, muss jedoch über die entsprechende Genehmigung verfügen. `kms:RetireGrant` Dadurch können die Grants, die dem AWS KMS Schlüssel bei der Erstellung des Repositorys hinzugefügt wurden, zurückgezogen werden.

Die folgende IAM Beispielrichtlinie kann einem Benutzer als Inline-Richtlinie hinzugefügt werden, um sicherzustellen, dass er über die Mindestberechtigungen verfügt, die zum Löschen eines Repositorys mit aktivierter Verschlüsselung erforderlich sind. Der KMS Schlüssel, der zum Verschlüsseln des Repositorys verwendet wird, kann mithilfe des Resource-Parameters angegeben werden.

```
{
    "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 unter Verwendung eines vom Kunden verwalteten Schlüssels sind Berechtigungen sowohl für die KMS Schlüsselrichtlinie als auch für die IAM Richtlinie für den Benutzer oder die Rolle erforderlich, die das Repository erstellt.

Wenn Sie Ihren eigenen KMS Schlüssel erstellen, können Sie entweder den standardmäßigen Schlüssel verwenden, den die Richtlinie AWS KMS erstellt, oder Sie können Ihren eigenen Schlüssel 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. Zusätzliche bereichsspezifische Berechtigungen können der Schlüsselrichtlinie hinzugefügt werden, aber mindestens sollte der Root-Benutzer die Rechte zur Verwaltung des Schlüssels erhalten. KMS Damit der KMS Schlüssel nur für Anfragen verwendet werden kann, die von Amazon stammenECR, können Sie den ViaService Bedingungsschlüssel kms: mit dem `ecr.<region>.amazonaws.com` Wert verwenden.

Das folgende Beispiel für eine Schlüsselrichtlinie 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.

```
{
    "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, die Ihre Repositorys erstellt `kms:CreateGrantkms:RetireGrant`, muss zusätzlich zu den erforderlichen ECR Amazon-Berechtigungen über die Berechtigungen, und `kms:DescribeKey` verfügen.

> **Note**
>
> Die `kms:RetireGrant` Berechtigung muss der IAM Richtlinie des Benutzers oder der Rolle hinzugefügt werden, die das Repository erstellt. Die `kms:DescribeKey` Berechtigungen `kms:CreateGrant` und können entweder der Schlüsselrichtlinie für den Schlüssel oder der KMS IAM Richtlinie des Benutzers oder der Rolle hinzugefügt werden, die das Repository erstellt. Weitere Informationen zur Funktionsweise von AWS KMS Berechtigungen finden Sie in der Referenz „AWS KMS APIBerechtigungen: Aktionen und Ressourcen“ im AWS Key Management Service Entwicklerhandbuch.

Die folgende IAM Beispielrichtlinie kann einem Benutzer als Inline-Richtlinie 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 damit fertig ist. Der zur Verschlüsselung des Repositorys verwendete AWS KMS key -Schlüssel kann mit dem Ressourcen-Parameter angegeben werden.

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

Erlaubt einem Benutzer, beim Erstellen eines KMS Repositorys Schlüssel in der Konsole aufzulisten

Wenn Sie die ECR Amazon-Konsole zum Erstellen eines Repositorys verwenden, können Sie einem Benutzer Berechtigungen erteilen, damit er die vom Kunden verwalteten KMS Schlüssel in der Region auflisten kann, wenn Sie die Verschlüsselung für das Repository aktivieren. Das folgende IAM Richtlinienbeispiel zeigt die Berechtigungen, die erforderlich sind, um Ihre KMS Schlüssel und Aliase aufzulisten, wenn Sie die Konsole verwenden.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
}
```

Überwachung der ECR Amazon-Interaktion mit AWS KMS

Sie können AWS CloudTrail damit die Anfragen verfolgen, an die Amazon in AWS KMS Ihrem Namen ECR sendet. Die Protokolleinträge im CloudTrail Protokoll enthalten einen Verschlüsselungskontextschlüssel, um sie leichter identifizierbar zu machen.

ECRAmazon-Verschlüsselungskontext

Ein Verschlüsselungskontext ist ein Satz von Schlüssel-Wert-Paaren, der beliebige nicht geheime Daten enthält. Wenn Sie einen Verschlüsselungskontext in eine Anfrage zur Verschlüsselung von Daten aufnehmen, 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 ECR verwendet Amazon 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 ECR Amazon-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-Ressourcenname (ARN) des Amazon S3-Buckets.

  ```
  "aws:s3:arn": "ARN of an Amazon S3 bucket"
  ```

  Wenn der ARN des Buckets beispielsweise lautet`arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`, würde der Verschlüsselungskontext das folgende Paar beinhalten.

  ```
  "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-Ressourcennamen () des Repositorys. ARN Der Schlüssel lautet `aws:ecr:arn`. Der Wert entspricht dem des Repositorys. ARN

  ```
  "aws:ecr:arn": "ARN of an Amazon ECR repository"
  ```

Wenn der Wert ARN des Repositorys beispielsweise lautet`arn:aws:ecr:us-west-2:111122223333:repository/repository-name`, würde der Verschlüsselungskontext das folgende Paar beinhalten.

```
"aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
```

Fehlerbehebung

Wenn Sie ein ECR Amazon-Repository mit der Konsole löschen und das Repository erfolgreich gelöscht wurde, Amazon jedoch nicht in der Lage ECR ist, die 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 eine `EncryptionContextSubset` mit dem Amazon-Ressourcennamen (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 für das Repository hinzugefügt wurde, zurückziehen. Ersetzen Sie den Wert für *`GrantId`* mit der ID des Zuschusses 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. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen Anwendungen erstellen HIPAA können, die AWS für sie in Frage kommen.

  > **Note**
  >
  > Nicht alle sind berechtigt AWS -Services . HIPAA Weitere Informationen finden Sie in der Referenz für HIPAA qualifizierte Dienste.

- AWS Ressourcen zur AWS von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den

Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS -Services und die Leitlinien für Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zusammengefasst.

- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS -Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS -Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen zu erfüllen PCIDSS, z. B. durch die Erfüllung der Anforderungen zur Erkennung von Eindringlingen, die in bestimmten Compliance-Frameworks vorgeschrieben sind.
- AWS Audit Manager— Auf diese AWS -Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

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 Anrufe, um ECR über das Netzwerk auf Amazon zuzugreifen. Kunden müssen Folgendes unterstützen:

- Sicherheit auf Transportschicht (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Cipher-Suites mit perfekter Vorwärtsgeheimhaltung (PFS) wie (Ephemeral Diffie-Hellman) oder DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Darüber hinaus müssen Anfragen mithilfe einer Zugriffsschlüssel-ID und eines geheimen Zugriffsschlüssels, der einem Prinzipal zugeordnet ist, signiert werden. IAM Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Sie können diese API Operationen von jedem Netzwerkstandort aus aufrufen, Amazon ECR unterstützt jedoch ressourcenbasierte Zugriffsrichtlinien, die Einschränkungen auf der Grundlage der Quell-IP-Adresse beinhalten können. Sie können auch ECR Amazon-Richtlinien verwenden, um den Zugriff von bestimmten Amazon Virtual Private Cloud (AmazonVPC) -Endpunkten oder bestimmten VPCs zu kontrollieren. Dadurch wird der Netzwerkzugriff auf eine bestimmte ECR Amazon-Ressource effektiv nur von den spezifischen Ressourcen VPC innerhalb des AWS Netzwerks isoliert. Weitere Informationen finden Sie unter ECRVPCAmazon-Schnittstellenendpunkte ()AWS PrivateLink.

ECRVPCAmazon-Schnittstellenendpunkte ()AWS PrivateLink

Sie können die Sicherheitslage Ihres Unternehmens verbessern, VPC indem Sie Amazon so konfigurierenECR, dass es einen VPC Schnittstellenendpunkt verwendet. VPCEndgeräte werden von einer Technologie unterstützt AWS PrivateLink, mit der Sie privat ECR APIs über private IP-Adressen auf Amazon zugreifen können. AWS PrivateLink schränkt den gesamten Netzwerkverkehr zwischen Ihnen VPC und Amazon ECR auf das Amazon-Netzwerk ein. Sie benötigen kein Internet-Gateway, kein NAT Gerät oder ein virtuelles privates Gateway.

Weitere Informationen zu AWS PrivateLink VPC Endpunkten finden Sie unter VPCEndpoints im VPCAmazon-Benutzerhandbuch.

Überlegungen zu ECR VPC Amazon-Endpunkten

Bevor Sie VPC Endgeräte für Amazon konfigurierenECR, sollten Sie die folgenden Überlegungen beachten.

- Damit Ihre auf EC2 Amazon-Instances gehosteten ECS Amazon-Aufgaben private Images von Amazon abrufen könnenECR, erstellen Sie die VPC Schnittstellenendpunkte für AmazonECS. Weitere Informationen finden Sie unter Interface VPC Endpoints (AWS PrivateLink) im Amazon Elastic Container Service Developer Guide.
- Auf Fargate gehostete ECS Amazon-Aufgaben, die Container-Images von Amazon abrufen, ECR können den Zugriff auf die spezifischen, von VPC ihren Aufgaben verwendeten Aufgaben und auf den vom Service verwendeten VPC Endpunkt einschränken, indem der IAM Aufgabenausführungsrolle für die Aufgabe Bedingungsschlüssel hinzugefügt werden. Weitere Informationen finden Sie unter Optionale IAM Berechtigungen für Fargate-Aufgaben Pulling

Amazon ECR Images over Interface Endpoints im Amazon Elastic Container Service Developer Guide.

- Die mit dem VPC Endpunkt verbundene Sicherheitsgruppe muss eingehende Verbindungen über Port 443 aus dem privaten Subnetz von zulassen. VPC
- VPCEndpunkte unterstützen derzeit keine regionsübergreifenden Anfragen. Stellen Sie sicher, dass Sie Ihre VPC Endgeräte in derselben Region einrichten, in der Sie Ihre API Anrufe an Amazon ECR tätigen möchten.
- VPCEndgeräte unterstützen derzeit keine ECR öffentlichen Amazon Repositorys. Erwägen Sie die Verwendung einer Pull-Through-Cache-Regel, um das öffentliche Image in einem privaten Repository in derselben Region wie der VPC Endpunkt zu hosten. Weitere Informationen finden Sie unter Synchronisieren Sie eine Upstream-Registrierung mit einer ECR privaten Amazon-Registrierung.
- VPCSupport nur für Endgeräte, der DNS über Amazon Route 53 AWS bereitgestellt wird. Wenn Sie Ihre eigene verwenden möchtenDNS, können Sie die bedingte DNS Weiterleitung verwenden. Weitere Informationen finden Sie unter DHCPOptionssätze im VPCAmazon-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 ein neues, VPC das den Amazon S3 S3-Gateway-Endpunkt verwendet, und migrieren Sie dann Ihren ECS Amazon-Cluster und seine Container in den neuenVPC.
- Wenn ein Image zum ersten Mal mithilfe einer Pull-Through-Cache-Regel abgerufen wird und Sie Amazon ECR so konfiguriert haben, dass ein VPC Schnittstellenendpunkt verwendet AWS PrivateLink wirdVPC, müssen Sie in demselben ein öffentliches Subnetz mit einem NAT Gateway erstellen und dann den gesamten ausgehenden Datenverkehr von ihrem privaten Subnetz zum NAT Gateway weiterleiten, damit der Pull 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.

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 ECR Amazon-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 Endpoints weitere Schritte erforderlich, damit Ihre Container diese fremden Ebenen aus Azure abrufen können.

Es ist möglich, dieses Verhalten zu überschreiben, wenn Windows-Images an Amazon übertragen werden, ECR indem Sie das `--allow-nondistributable-artifacts` Flag im Docker-Daemon verwenden. Wenn diese Markierung aktiviert ist, werden die lizenzierten Ebenen an Amazon weitergeleitetECR, sodass diese Bilder ECR über den VPC Endpunkt von Amazon abgerufen werden können, ohne dass 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 das RepositoryURI, in das Sie Bilder übertragen.

```
{
    "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 Ebenengröße führt zu einer längeren Übertragungszeit und zusätzlichen Speicherkosten bei Amazon ECR für diese Bilder. 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. Beispielsweise ist das `mcr.microsoft.com/windows/servercore` Bild ungefähr 1,7 GiB groß, wenn es in Amazon komprimiert wirdECR.

Erstellen Sie die VPC Endpunkte für Amazon ECR

Um die VPC Endpunkte für den ECR Amazon-Service zu erstellen, verwenden Sie das Verfahren Creating an Interface Endpoint im VPCAmazon-Benutzerhandbuch.

ECSAmazon-Aufgaben, die auf EC2 Amazon-Instances gehostet werden, erfordern sowohl ECR Amazon-Endpunkte als auch den Amazon S3-Gateway-Endpunkt.

ECSAmazon-Aufgaben, die mit der Plattformversion `1.4.0` oder höher auf Fargate gehostet werden, erfordern sowohl ECR VPC Amazon-Endpunkte als auch die Amazon S3-Gateway-Endpunkte.

Auf Fargate gehostete ECS Amazon-Aufgaben, die die Plattformversion `1.3.0` oder eine frühere Version verwenden, benötigen nur die Datei com.amazonaws.***region***.ecr.dkr ECR VPC Amazon-Endpunkt und Amazon 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 Enable Private DNS Name in der VPC Amazon-Konsole ausgewählt ist, wenn Sie den VPC Endpunkt erstellen.

com.amazonaws.*region*.ecr.api

> **Note**
>
> Das angegebene *region* steht für die Regionskennung für eine AWS Region, die von Amazon unterstützt wirdECR, z. B. `us-east-2` für die Region USA Ost (Ohio).

Dieser Endpunkt wird für Anrufe an Amazon verwendet ECRAPI. APIAktionen wie `DescribeImages` und `CreateRepository` gehen zu diesem Endpunkt.

Wenn dieser Endpunkt erstellt wird, haben Sie die Möglichkeit, einen privaten DNS Hostnamen zu aktivieren. Aktivieren Sie diese Einstellung, indem Sie bei der Erstellung des VPC Endpunkts in der VPC Konsole die Option Privaten DNS Namen aktivieren auswählen. Wenn Sie einen privaten DNS Hostnamen für den VPC Endpunkt aktivieren, aktualisieren Sie Ihren SDK oder AWS CLI auf die neueste Version, sodass die Angabe eines Endpunkts URL bei der Verwendung von SDK oder AWS CLI nicht erforderlich ist.

Wenn Sie einen privaten DNS Hostnamen aktivieren und eine SDK AWS CLI OR-Version verwenden, die vor dem 24. Januar 2019 veröffentlicht wurde, müssen Sie den `--endpoint-url` Parameter verwenden, um die Schnittstellenendpunkte anzugeben. Das folgende Beispiel zeigt das Format für den 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, der die VPC Endpunkt-ID für den Schnittstellenendpunkt angibt. Das folgende Beispiel zeigt das Format für den EndpunktURL.

```
aws ecr create-repository --repository-name name --endpoint-url
 https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

Erstellen Sie den Amazon S3-Gateway-Endpunkt

Damit Ihre ECS Amazon-Aufgaben private Images von Amazon abrufen könnenECR, müssen Sie einen Gateway-Endpunkt für Amazon S3 erstellen. Der Gateway-Endpunkt ist erforderlich,

da Amazon S3 zum Speichern Ihrer Bildebenen ECR verwendet. Wenn Ihre Container Bilder von Amazon herunterladenECR, müssen sie auf Amazon zugreifen, ECR um das Image-Manifest abzurufen, und dann auf Amazon S3, um die eigentlichen Bildebenen herunterzuladen. Im Folgenden finden Sie den Amazon-Ressourcennamen (ARN) des Amazon S3-Buckets, der die Ebenen für jedes Docker-Image enthält.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

Verwenden Sie das Verfahren zum Erstellen eines Gateway-Endpunkts im VPCAmazon-Benutzerhandbuch, um den folgenden Amazon S3-Gateway-Endpunkt für Amazon zu erstellenECR. Achten Sie beim Erstellen des Endpunkts darauf, die Routing-Tabellen für Ihren auszuwählenVPC.

com.amazonaws.*region*.s3

> Der Amazon S3 S3-Gateway-Endpunkt verwendet ein IAM Richtliniendokument, um den Zugriff auf den Service einzuschränken. Die Richtlinie „Vollzugriff“ kann verwendet werden, da alle Einschränkungen, 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 S3-Bucket auf die für die Nutzung von Amazon erforderlichen Mindestberechtigungen beschränken möchtenECR, finden Sie weitere Informationen unterMindestberechtigungen für Amazon S3 S3-Bucket für Amazon ECR.

Mindestberechtigungen für Amazon S3 S3-Bucket für Amazon ECR

Der Amazon S3 S3-Gateway-Endpunkt verwendet ein IAM Richtliniendokument, um den Zugriff auf den Service einzuschränken. Um Amazon nur die Mindestberechtigungen für Amazon S3 S3-Bucket zuzulassenECR, beschränken Sie den Zugriff auf den Amazon S3 S3-Bucket, den Amazon ECR verwendet, wenn Sie das IAM Richtliniendokument für den Endpunkt erstellen.

In der folgenden Tabelle werden die von Amazon benötigten Amazon S3-Bucket-Policy-Berechtigungen beschriebenECR.

Berechtigung	Beschreibung
`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 die Regionskennung für eine AWS Region dar, die von Amazon unterstüt

Berechtigung	Beschreibung
	zt wirdECR, z. B. `us-east-2` für die Region USA Ost (Ohio).

Beispiel

Das folgende Beispiel zeigt, wie Sie Zugriff auf die Amazon S3 S3-Buckets gewähren, die für den ECR Betrieb von Amazon erforderlich sind.

```
{
  "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

Für ECS Amazon-Aufgaben, die den Starttyp Fargate verwenden und ein Gateway VPC ohne Internetanschluss verwenden, die auch den `awslogs` Protokolltreiber verwenden, um CloudWatch Protokollinformationen an Logs zu senden, müssen Sie die Datei com.amazonaws erstellen.***region***VPC.logs-Schnittstellenendpunkt für Logs. CloudWatch Weitere Informationen finden Sie unter Using CloudWatch Logs with interface VPC endpoints im Amazon CloudWatch Logs-Benutzerhandbuch.

Erstellen Sie eine Endpunktrichtlinie für Ihre ECR VPC Amazon-Endgeräte

Eine VPC Endpunktrichtlinie ist eine IAM Ressourcenrichtlinie, die Sie einem Endpunkt zuordnen, wenn Sie den Endpunkt erstellen oder ändern. Wenn Sie beim Erstellen eines Endpunkts keine Richtlinie anhängen, AWS fügt es eine Standardrichtlinie für Sie an, die vollen Zugriff auf den Dienst 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 geschrieben werden. Weitere Informationen finden Sie unter Controlling Access to Services with VPC Endpoints im VPCAmazon-Benutzerhandbuch.

Wir empfehlen, eine einzige IAM Ressourcenrichtlinie zu erstellen und sie an beide ECR VPC Amazon-Endgeräte anzuhängen.

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für AmazonECR. Diese Richtlinie ermöglicht es einer bestimmten IAM Rolle, Bilder von Amazon abzurufenECR.

```
{
 "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": "*"
  }
 ]
}
```

Um die VPC Endpunktrichtlinie für Amazon zu ändern ECR

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

2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Wenn Sie die VPC Endpunkte für Amazon noch nicht erstellt habenECR, finden Sie weitere Informationen unterErstellen Sie die VPC Endpunkte für Amazon ECR.
4. Wählen Sie den ECR VPC Amazon-Endpunkt aus, zu dem Sie eine Richtlinie hinzufügen möchten, und klicken Sie in der unteren Hälfte des Bildschirms auf die Registerkarte Richtlinie.
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

In Subnetzen, die mit Ihnen geteilt werden, können Sie keine VPC Endpunkte erstellen, beschreiben, ändern oder löschen. Sie können die VPC Endpunkte jedoch in Subnetzen verwenden, die mit Ihnen gemeinsam genutzt 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. In AWS kann ein dienstübergreifendes Identitätswechsels 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 Kontextschlüssel `aws:SourceArn`oder die `aws:SourceAccount`globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, die Amazon einem anderen Service für die Ressource ECR erteilt. 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, sich vor dem Problem mit dem verwirrten Stellvertreter zu schützen, besteht darin, den Kontextschlüssel für `aws:SourceArn` globale Bedingungen mit ARN der gesamten Ressource zu verwenden. Wenn Sie die gesamte ARN Ressource nicht kennen

oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den `aws:SourceArn` globalen Kontextbedingungsschlüssel mit Platzhalterzeichen (*) für die unbekannten Teile von. ARN Beispiel, `arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*`.

Wenn der `aws:SourceArn` Wert die Konto-ID nicht enthält, z. B. ein Amazon S3 S3-BucketARN, müssen Sie beide globalen Bedingungskontextschlüssel verwenden, um die 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 ECR Amazon-Repository-Richtlinie verwenden können, um den AWS CodeBuild Zugriff auf die ECR API Amazon-Aktionen zu ermöglichen, die für die Integration mit diesem Service erforderlich sind, und gleichzeitig das Problem des verwirrten Stellvertreters zu verhindern.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccess",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:BatchGetImage",
            "ecr:GetDownloadUrlForLayer"
         ],
         "Condition":{
            "ArnLike":{
               "aws:SourceArn":"arn:aws:codebuild:region:123456789012:project/project-
name"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

ECRAmazon-Überwachung

Sie können Ihre ECR API Amazon-Nutzung mit Amazon überwachen CloudWatch, das Rohdaten von Amazon sammelt und ECR zu lesbaren Kennzahlen 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. ECRAmazon-Metrikdaten werden automatisch innerhalb von CloudWatch einer Minute an gesendet. Weitere Informationen zu CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

Amazon ECR stellt auf Ihrer API Nutzung basierende Kennzahlen für Autorisierungs-, Image-Push- und Image-Pull-Aktionen bereit.

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 Ausgangsbasis für die normale ECR Amazon-Leistung in Ihrer Umgebung festzulegen, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Speichern Sie bei der Überwachung von Amazon historische ÜberwachungsdatenECR, damit Sie sie mit neuen Leistungsdaten vergleichen, normale Leistungsmuster und Leistungsanomalien identifizieren und Methoden zur Behebung von Problemen entwickeln können.

Themen

- Visualisierung Ihrer Service Quotas und Einstellung von Alarmen

- ECRAmazon-Nutzungsmetriken
- ECRAmazon-Nutzungsberichte
- ECRAmazon-Repository-Kennzahlen
- ECRAmazon-Veranstaltungen und EventBridge
- ECRAmazon-Aktionen protokollieren 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 QUOTA Ausdruck SERVICE_. 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 ALARM Status wechselt, wenn Ihre Nutzung 80 Prozent des Kontingents überschreitet.

c. Wählen Sie Weiter.

d. Wählen Sie auf der nächsten Seite ein SNS Amazon-Thema aus oder erstellen Sie ein neues. 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 Create alarm (Alarm erstellen).

ECRAmazon-Nutzungsmetriken

Sie können CloudWatch Nutzungsmetriken verwenden, um einen Überblick über die Ressourcennutzung Ihres Kontos zu erhalten. Verwenden Sie diese Metriken, um Ihre aktuelle Servicenutzung in CloudWatch Diagrammen und Dashboards zu visualisieren.

Die ECR Amazon-Nutzungskennzahlen entsprechen den AWS Servicekontingenten. Sie können Alarme konfigurieren, mit denen Sie benachrichtigt werden, wenn sich Ihre Nutzung einem Servicekontingent nähert. Weitere Informationen zu ECR Amazon-Servicekontingenten finden Sie unterAmazon ECR Service Quotas.

Amazon ECR veröffentlicht die folgenden Metriken im `AWS/Usage` Namespace.

Metrik	Beschreibung
`CallCount`	Die Anzahl der API Aktionsaufforderungen von Ihrem Konto aus. Die Ressourcen werden durch die Dimensionen definiert, die der Metrik zugeordnet sind.

Metrik	Beschreibung
	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 veröffentlichten Nutzungsmetriken zu verfeinernECR.

Dimension	Beschreibung
`Service`	Der Name des AWS Dienstes, der die Ressource enthält. Für ECR Amazon-Nutzungsmetriken ist der Wert für diese Dimension`ECR`.
`Type`	Der Typ von Entität, die gemeldet wird. Derzeit ist der einzig gültige Wert für ECR Amazon-Nutzungsmetriken`API`.
`Resource`	Der Typ der Ressource, die ausgeführt wird. Derzeit ECR sendet Amazon Informationen zu Ihrer API Nutzung für die folgenden API Aktionen zurück. • GetAuthorizationToken • BatchCheckLayerAvailability • InitiateLayerUpload • UploadLayerPart • CompleteLayerUpload • PutImage • BatchGetImage • GetDownloadUrlForLayer
`Class`	Die Klasse der nachverfolgten Ressource. Derzeit verwendet Amazon die Klassendimension ECR nicht.

ECRAmazon-Nutzungsberichte

AWS bietet ein kostenloses Berichtstool namens Cost Explorer, mit dem Sie die Kosten und die Nutzung Ihrer ECR Amazon-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 all Ihren ECR Amazon-Repositorys. 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.

ECRAmazon-Repository-Kennzahlen

Amazon ECR sendet Metriken zur Anzahl der Repository-Pulls an Amazon CloudWatch. ECRAmazon-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
- ECRAmazon-Metriken mit der CloudWatch Konsole anzeigen

CloudWatch Metriken aktivieren

Amazon ECR sendet automatisch Repository-Metriken 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 sendet CloudWatch.

ECRAmazon-Metriken

Amazon ECR stellt Ihnen Metriken zur Verfügung, 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 ECR Amazon-Metriken

ECRAmazon-Metriken verwenden den `AWS/ECR` Namespace und stellen Metriken für die folgenden Dimensionen bereit.

`RepositoryName`

Diese Dimension filtert die Daten, die Sie für alle Container-Images in einem bestimmten Repository anfordern.

ECRAmazon-Metriken mit der CloudWatch Konsole anzeigen

Sie können ECR Amazon-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.

Um Metriken in der CloudWatch Konsole anzuzeigen

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich Metrics (Metriken) All metrics (Alle Metriken) aus.
3. Wählen Sie auf der Registerkarte Durchsuchen unter AWS Namespaces die Option. ECR
4. Wählen Sie die Metriken, die angezeigt werden sollen. Repository-Metriken haben den Gültigkeitsbereich > Repository-Metriken. ECR

ECRAmazon-Veranstaltungen 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 nahezu EventBridge in Echtzeit zugestellt. 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
- Ein SNS Amazon-Thema oder eine SQS Amazon-Warteschlange benachrichtigen

Weitere Informationen finden Sie unter Erste Schritte mit Amazon EventBridge im EventBridge Amazon-Benutzerhandbuch.

Beispielereignisse von Amazon ECR

Im Folgenden finden Sie Beispielereignisse von AmazonECR. 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 ECR privates Amazon-Repository verschieben.

```
{
    "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 ECR privaten Amazon-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 scannen ECR.

> **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 gesendet wurden ECR.

```
{
    "version": "0",
    "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
    "detail-type": "ECR Image Scan",
    "source": "aws.ecr",
    "account": "123456789012",
    "time": "2019-10-29T02:36:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ecr:us-east-1:123456789012:repository/my-repository-name"
    ],
    "detail": {
        "scan-status": "COMPLETE",
        "repository-name": "my-repository-name",
        "finding-severity-counts": {
```

```
        "CRITICAL": 10,
        "MEDIUM": 9
      },
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "image-tags": []
    }
}
```

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 gesendet, ECR wenn es eine Änderung mit einer Ressource gibt, für die erweitertes 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 Software-Sicherheitslücken in Amazon scannen ECR.

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0ab8638973bf",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
  "action-type": "SCAN_FREQUENCY_CHANGE",
  "repositories": [{
    "repository-name": "repository-1",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
    "scan-frequency": "SCAN_ON_PUSH",
    "previous-scan-frequency": "MANUAL"
   },
   {
    "repository-name": "repository-2",
    "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
    "scan-frequency": "CONTINUOUS_SCAN",
    "previous-scan-frequency": "SCAN_ON_PUSH"
   },
```

```
    {
     "repository-name": "repository-3",
     "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
     "scan-frequency": "CONTINUOUS_SCAN",
     "previous-scan-frequency": "SCAN_ON_PUSH"
    }
   ],
   "resource-type": "REPOSITORY",
   "scan-type": "ENHANCED"
 }
}
```

Ereignis für eine Image-Löschung

Das folgende Ereignis wird gesendet, wenn ein Image gelöscht wird. Weitere Informationen finden Sie unter Ein Bild in Amazon löschen 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 abgeschlossene Image-Replikation

Das folgende Ereignis wird gesendet, wenn jede Image-Replikation abgeschlossen ist. Weitere Informationen finden Sie unter Replikation privater Bilder 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"
  }
}
```

ECRAmazon-Aktionen protokollieren 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 ausgeführt wurdenECR. CloudTrail erfasst die folgenden ECR Amazon-Aktionen als Ereignisse:

- Alle API Anrufe, auch Anrufe von der ECR Amazon-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 ECR sendet Amazon bei Aktionen mit Lebenszyklusrichtlinien, bei denen 10 oder mehr Bilder

abgelaufen sind, mehrere Ereignisse an CloudTrail. Darüber hinaus ECR enthält Amazon maximal 100 Tags pro Bild.

Wenn ein Trail erstellt wird, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für AmazonECR. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf einsehen. Anhand dieser Informationen können Sie die Anfrage, die an Amazon gestellt wurdeECR, die ursprüngliche IP-Adresse, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.

ECRAmazon-Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Amazon Aktivitäten auftretenECR, 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 der Ereignisse in Ihrem AWS Konto, einschließlich Veranstaltungen für AmazonECR, 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:

- Erstellen Sie einen Trail für Ihr AWS Konto
- AWS Serviceintegrationen mit Protokollen CloudTrail
- Konfiguration von SNS Amazon-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle ECR API Amazon-Aktionen werden von der Amazon Elastic Container Registry API Reference 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. 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 AWS Dienst gestellt wurde

Weitere Informationen finden Sie im CloudTrail`userIdentity`Element.

ECRAmazon-Protokolldateieinträge verstehen

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 gängige ECR Amazon-Aufgaben.

> **Note**
>
> 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 ECR Amazon-Eintrag beschränkt. In einer echten CloudTrail Protokolldatei sehen Sie Einträge und Ereignisse von mehreren AWS Diensten.

Themen

Beispiel: Repository-Aktion erstellen

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateRepository` Aktion demonstriert.


```
{
    "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 ECR Amazon-Repositorys

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die AWS KMS CreateGrant Aktion beim Erstellen eines ECR Amazon-Repositorys mit aktivierter KMS Verschlüsselung demonstriert. Für jedes Repository, das mit aktivierter KMS Verschlüsselung erstellt wurde, sollten Sie zwei CreateGrant Protokolleinträge in 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": "ecr.amazonaws.com",
 "userAgent": "ecr.amazonaws.com",
 "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 ein Bild abrufen und das Bild noch nicht lokal gespeichert haben, werden Ihnen auch `GetDownloadUrlForLayer` Verweise in den CloudTrail Protokollen angezeigt.

```
{
    "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.

> **Important**
>
> Aufgrund der Größenbeschränkungen einzelner CloudTrail Ereignisse ECR sendet Amazon bei Aktionen mit Lebenszyklusrichtlinien, bei denen 10 oder mehr Bilder abgelaufen sind, mehrere Ereignisse an CloudTrail. Darüber hinaus ECR enthält Amazon maximal 100 Tags pro Bild.

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

Verwenden von Amazon ECR mit einem AWS SDK

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eineAPI, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDKDokumentation	Codebeispiele
AWS SDK for C++	AWS SDK for C++ Code-Beispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK for Go	AWS SDK for Go Code-Beispiele
AWS SDK for Java	AWS SDK for Java Code-Beispiele
AWS SDK for JavaScript	AWS SDK for JavaScript Code-Beispiele
AWS SDK for Kotlin	AWS SDK for Kotlin Code-Beispiele
AWS SDK for .NET	AWS SDK for .NET Code-Beispiele
AWS SDK for PHP	AWS SDK for PHP Code-Beispiele
AWS Tools for PowerShell	Tools für PowerShell Codebeispiele
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) Codebeispiele
AWS SDK for Ruby	AWS SDK for Ruby Code-Beispiele
AWS SDK for Rust	AWS SDK for Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK for Swift	AWS SDK for Swift Code-Beispiele

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.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Erste Schritte

Hallo Amazon ECR

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon beginnen könnenECR.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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()));
    }
}
```

- APIEinzelheiten finden Sie listImagesunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
```

```
        }
    }
}
```

- APIEinzelheiten finden Sie listImagesin der AWS SDKAPIKotlin-Referenz.

Codebeispiele

- Aktionen für Amazon ECR mit AWS SDKs
  - Verwenden Sie CreateRepository mit einem AWS SDK oder CLI
  - Verwenden Sie es DeleteRepository mit einem AWS SDK oder CLI
  - Verwenden Sie es DescribeImages mit einem AWS SDK oder CLI
  - Verwenden Sie es DescribeRepositories mit einem AWS SDK oder CLI
  - Verwenden Sie es GetAuthorizationToken mit einem AWS SDK oder CLI
  - Verwenden Sie es GetRepositoryPolicy mit einem AWS SDK oder CLI
  - Verwenden Sie es ListImages mit einem AWS SDK oder CLI
  - Verwenden Sie es PushImageCmd mit einem AWS SDK oder CLI
  - Verwenden Sie es SetRepositoryPolicy mit einem AWS SDK oder CLI
  - Verwenden Sie es StartLifecyclePolicyPreview mit einem AWS SDK oder CLI
- Szenarien für die ECR Verwendung von Amazon AWS SDKs
  - Erfahren Sie mehr über die Kernoperationen ECR von Amazon mit einem AWS SDK

Aktionen für Amazon ECR mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne ECR Amazon-Aktionen mit ausführen können AWS SDKs. Diese Auszüge nennen Amazon ECR API und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden können.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Elastic Container Registry (AmazonECR) API Reference.

Beispiele

- Verwenden Sie CreateRepository mit einem AWS SDK oder CLI

- Verwenden Sie es DeleteRepository mit einem AWS SDK oder CLI
- Verwenden Sie es DescribeImages mit einem AWS SDK oder CLI
- Verwenden Sie es DescribeRepositories mit einem AWS SDK oder CLI
- Verwenden Sie es GetAuthorizationToken mit einem AWS SDK oder CLI
- Verwenden Sie es GetRepositoryPolicy mit einem AWS SDK oder CLI
- Verwenden Sie es ListImages mit einem AWS SDK oder CLI
- Verwenden Sie es PushImageCmd mit einem AWS SDK oder CLI
- Verwenden Sie es SetRepositoryPolicy mit einem AWS SDK oder CLI
- Verwenden Sie es StartLifecyclePolicyPreview mit einem AWS SDK oder CLI

Verwenden Sie **`CreateRepository`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`CreateRepository`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Beispiel 1: Um ein Repository zu erstellen

Im folgenden `create-repository` Beispiel wird ein Repository innerhalb des angegebenen Namespace in der Standardregistrierung für ein Konto erstellt.

```
aws ecr create-repository \
    --repository-name project-a/nginx-web-app
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "sample-repo",
```

```
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/nginx-web-app"
    }
}
```

Weitere Informationen finden Sie unter Creating a Repository im ECRAmazon-Benutzerhandbuch.

Beispiel 2: Um ein Repository zu erstellen, das mit der Unveränderlichkeit des Image-Tags konfiguriert ist

Im folgenden `create-repository` Beispiel wird in der Standardregistrierung für ein Konto ein Repository erstellt, das für die Unveränderlichkeit von Tags konfiguriert ist.

```
aws ecr create-repository \
    --repository-name sample-repo \
    --image-tag-mutability IMMUTABLE
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/sample-
repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Weitere Informationen finden Sie unter Image Tag Mutability im ECRAmazon-Benutzerhandbuch.

Beispiel 3: So erstellen Sie ein Repository, das mit einer Scan-Konfiguration konfiguriert ist

Im folgenden `create-repository` Beispiel wird ein Repository erstellt, das so konfiguriert ist, dass es beim Image-Push in der Standardregistrierung für ein Konto einen Schwachstellenscan durchführt.

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

```
    --image-scanning-configuration scanOnPush=true
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/sample-
repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Weitere Informationen finden Sie unter Scannen von Bildern im ECRAmazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie CreateRepositoryin der AWS CLI Befehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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);
           }
       }
   }
```

- APIEinzelheiten finden Sie CreateRepositoryunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { 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
```

```
        }
    }
```

- APIEinzelheiten finden Sie CreateRepositoryin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`DeleteRepository`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`DeleteRepository`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

So löschen Sie ein Repository

Das folgende `delete-repository` Beispiel mit Command Force löscht das angegebene Repository in der Standardregistrierung für ein Konto. Das `--force` Flag ist erforderlich, wenn das Repository Bilder 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 ECRAmazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie DeleteRepositoryin der AWS CLI Befehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
    }
```

- APIEinzelheiten finden Sie DeleteRepositoryunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- APIEinzelheiten finden Sie DeleteRepositoryin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`DescribeImages`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`DescribeImages`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Um ein Bild in einem Repository zu beschreiben

Im folgenden `describe-images` Beispiel werden Details zu einem Bild 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
        }
    ]
}
```

- APIEinzelheiten finden Sie unter DescribeImages AWS CLIBefehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
    }
```

- APIEinzelheiten finden Sie DescribeImagesunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { 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.")
            }
        }
    }
```

- APIEinzelheiten finden Sie DescribeImagesin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`DescribeRepositories`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`DescribeRepositories`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Um die Repositorys in einer Registrierung zu beschreiben

In diesem Beispiel werden die Repositorys in der Standardregistrierung 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"
        }
    ]
}
```

- APIEinzelheiten finden Sie DescribeRepositoriesin der AWS CLI Befehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
    }
```

- APIEinzelheiten finden Sie DescribeRepositoriesunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { 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 ""
            }
        }
    }
```

- APIEinzelheiten finden Sie DescribeRepositoriesin der AWS SDKAPIKotlin-Referenz.

Rust

SDKfür Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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(())
}
```

- APIEinzelheiten finden Sie DescribeRepositoriesin der AWS SDKAPIRust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`GetAuthorizationToken`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`GetAuthorizationToken`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Um ein Autorisierungstoken für Ihre Standardregistrierung zu erhalten

Mit dem folgenden `get-authorization-token` Beispielbefehl wird ein Autorisierungstoken für Ihre Standardregistrierung abgerufen.

```
aws ecr get-authorization-token
```

Ausgabe:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- APIEinzelheiten finden Sie GetAuthorizationTokenin der AWS CLI Befehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
   }
```

- APIEinzelheiten finden Sie GetAuthorizationTokenunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient { 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.")
            }
        }
    }
```

- APIEinzelheiten finden Sie GetAuthorizationTokenin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`GetRepositoryPolicy`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`GetRepositoryPolicy`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Um die Repository-Richtlinie für ein Repository abzurufen

Im folgenden `get-repository-policy` Beispiel 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}"
}
```

- APIEinzelheiten finden Sie GetRepositoryPolicyunter AWS CLI Befehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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;
   }
```

- APIEinzelheiten finden Sie GetRepositoryPolicyunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- APIEinzelheiten finden Sie GetRepositoryPolicyin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`ListImages`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`ListImages`.

CLI

AWS CLI

Um die Bilder in einem Repository aufzulisten

Im folgenden `list-images` Beispiel wird eine Liste der Bilder 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"
        }
    ]
}
```

- APIEinzelheiten finden Sie ListImagesin der AWS CLI Befehlsreferenz.

Rust

SDKfür Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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(())
}
```

- APIEinzelheiten finden Sie ListImagesin der AWS SDKAPIRust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **PushImageCmd** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutztPushImageCmd.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
    }
```

- APIEinzelheiten finden Sie PushImageCmdunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { 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)
            }
        }
    }
```

- APIEinzelheiten finden Sie PushImageCmdin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`SetRepositoryPolicy`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`SetRepositoryPolicy`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Um die Repository-Richtlinie für ein Repository festzulegen

Im folgenden `set-repository-policy` Beispiel 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" : "2008-10-17",
    "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}"
}
```

- APIEinzelheiten finden Sie unter SetRepositoryPolicy AWS CLIBefehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
    }
```

- APIEinzelheiten finden Sie SetRepositoryPolicyunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- APIEinzelheiten finden Sie SetRepositoryPolicyin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **`StartLifecyclePolicyPreview`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt`StartLifecyclePolicyPreview`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Lernen Sie die ECR Kernoperationen von Amazon kennen

CLI

AWS CLI

Um eine Lifecycle-Richtlinienvorschau zu erstellen

Im folgenden `start-lifecycle-policy-preview` Beispiel wird eine Lifecycle-Policy-Vorschau erstellt, die durch eine JSON Datei für das angegebene Repository definiert wird.

```
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"
}
```

- APIEinzelheiten finden Sie StartLifecyclePolicyPreviewunter AWS CLI Befehlsreferenz.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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();
    }
```

- APIEinzelheiten finden Sie StartLifecyclePolicyPreviewunter AWS SDK for Java 2.x APIReferenz.

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 { 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.")
            }
        }
    }
```

- APIEinzelheiten finden Sie StartLifecyclePolicyPreviewin der AWS SDKAPIKotlin-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Szenarien für die ECR Verwendung von Amazon AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon ECR mit implementieren AWS SDKs. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen

können, indem Sie mehrere Funktionen innerhalb von Amazon aufrufenECR. Jedes Szenario enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Beispiele

- Erfahren Sie mehr über die Kernoperationen ECR von Amazon mit einem AWS SDK

Erfahren Sie mehr über die Kernoperationen ECR von Amazon mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein ECR Amazon-Repository.
- Legen Sie Repository-Richtlinien fest.
- Repository abrufenURIs.
- Holen Sie sich ECR Amazon-Autorisierungstoken.
- Legen Sie Lebenszyklusrichtlinien für ECR Amazon-Repositorys fest.
- Pushen Sie ein Docker-Image in ein ECR Amazon-Repository.
- Überprüfen Sie die Existenz eines Images in einem ECR Amazon-Repository.
- Listen Sie ECR Amazon-Repositorys für Ihr Konto auf und informieren Sie sich über sie.
- Löschen Sie ECR Amazon-Repositorys.

Java

SDKfür Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem ECR Amazon-Funktionen demonstriert werden.

```
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:
 *
 * /getting_started_scenarios/ecr_scenario/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 ECR SDK Amazon-Methoden.

```
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;
        }
    }
}
```

- APIEinzelheiten finden Sie unter den folgenden Themen in der AWS SDK for Java 2.x APIReferenz.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Kotlin

SDKfür Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem ECR Amazon-Funktionen demonstriert werden.

```
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:
 *
 * /getting_started_scenarios/ecr_scenario/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 = "arn:aws:iam::814548047983:role/Admin"
    var localImageName: String
    var accountId = "814548047983"
    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 ECR SDK Amazon-Methoden.

```
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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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 { 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)
        }
    }
}
```

- APIEinzelheiten finden Sie in den folgenden Themen als AWS SDKKotlin-Referenz API.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden von Amazon ECR mit einem AWS SDK. 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	Beschreibung
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: 10.000	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.
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.

Name	Standard	Anpas	Beschreibung
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-Aktionen 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.
Rate der BatchCheckLayerAvailability-Anforderungen	Jede unterstützte Region: 1 000 pro Sekunde	Ja	Die maximale Anzahl von BatchCheckLayerAvailability-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen 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.

Name	Standard	Anpas	Beschreibung
Rate von BatchGetImage-Anforderungen	Jede unterstützte Region: 2.000 pro Sekunde	Ja	Die maximale Anzahl von BatchGetImage-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können. Bei einem Abrufen vom Image wird die BatchGetImage-API einmal aufgerufen, um das Image Manifest abzurufen. Wenn Sie eine Kontingenterhöhung für diese API beantragen, überprüfen Sie auch Ihre GetDownloadUrlForLayer-Nutzung.
Rate der CompleteLayerUpload-Anforderungen	Jede unterstützte Region: 100 pro Sekunde	Ja	Die maximale Anzahl von CompleteLayerUpload-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können. Bei einem Image-Push wird die CompleteLayerUpload-API einmal pro neuer Image-Ebene aufgerufen, um zu überprüfen, ob der Upload abgeschlossen ist.

Name	Standard	Anpas	Beschreibung
Rate der GetAuthorizationToken-Anforderungen	Jede unterstützte Region: 500 pro Sekunde	Ja	Die maximale Anzahl von GetAuthorizationToken-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können.
Rate der GetDownloadUrlForLayer-Anforderungen	Jede unterstützte Region: 3.000 pro Sekunde	Ja	Die maximale Anzahl von GetDownloadUrlForLayer-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können. Bei einem Abrufen vom Image wird die GetDownloadUrlForLayer-API einmal pro Image-Ebene aufgerufen, die noch nicht zwischengespeichert ist. Wenn Sie eine Kontingenterhöhung für diese API beantragen, überprüfen Sie auch Ihre BatchGetImage-Nutzung.

Name	Standard	Anpas	Beschreibung
Rate der InitiateLayerUpload-Anforderungen	Jede unterstützte Region: 100 pro Sekunde	Ja	Die maximale Anzahl von InitiateLayerUpload-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können. Bei einem Image-Push wird die InitiateLayerUpload-API einmal pro Image-Ebene aufgerufen, die noch nicht hochgeladen wurde. Ob eine Image-Ebene hochgeladen wurde, hängt von der BatchCheckLayerAvailability-API-Aktion ab.
Rate der PutImage-Anforderungen	Jede unterstützte Region: 10 pro Sekunde	Ja	Die maximale Anzahl von PutImage-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können. Wenn bei einem Image-Push alle neuen Image-Ebenen hochgeladen wurden, wird die PutImage-API einmal aufgerufen, um das Image Manifest und die mit dem Image verknüpften Tags zu erstellen oder zu aktualisieren.

Name	Standard	Anpas	Beschreibung
Rate von UploadLayerPart-Anforderungen	Jede unterstützte Region: 500 pro Sekunde	Ja	Die maximale Anzahl von UploadLayerPart-Anforderungen, die Sie pro Sekunde in der aktuellen Region vornehmen können. Bei einer Image-Übertragung wird jede neue Image-Ebene in Teilen hochgeladen und die UploadLayerPart-API wird einmal pro neuem Teil der Image-Ebene aufgerufen.
Rate der Image-Scans	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von Image-Scans pro Image und pro 24 Stunden.
Registered repositories (Registrierte Repositorys)	Jede unterstützte Region: 10.000	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

Name	Standard	Anpas	Beschreibung
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 Management Console

Amazon ECR ist mit Service Quotas integriert, einem AWS Service, der es Ihnen ermöglicht, Ihre Kontingente von einem zentralen Ort aus anzuzeigen und zu verwalten. 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 Management Console)

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 Service Quotas wird der Name der Service Quota, der angewendete Wert (falls verfügbar) und das AWS-Standardkontingent angezeigt. Zudem wird angezeigt, ob der Kontingentwert anpassbar ist.
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.

Erstellen eines CloudWatch-Alarms zur Überwachung von API-Nutzungsmetriken

Amazon ECR stellt CloudWatch-Nutzungsmetriken bereit, die den AWS-Service Quotas für jede der APIs entsprechen, die an den Aktionen Registrierungsauthentifizierung, 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 ECRAmazon-Nutzungsmetriken.

Führen Sie die folgenden Schritte aus, 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 Management Console)

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 Abschnitt Amazon CloudWatch Events alarms die Option Create.
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 Problemen im Zusammenhang mit Docker 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 Amazon EC2-Instance in einem privaten Subnetz ausführen, stellen Sie sicher, dass das Subnetz eine Route zum Internet hat. 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 Docker for Windows-Versionen nutzen ein Programm zur Verwaltung von Anmeldeinformationen mit der Bezeichnung `wincred`. Dieses Programm kann den über aws ecr get-login-password ausgegebenen Docker-Anmeldebefehl nicht ordnungsgemäß verarbeiten (weitere Informationen finden Sie unter https://github.com/docker/docker/issues/22910). 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-APIs finden Sie 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 für AWS SDKs und Tools. 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 unter Verwalten Ihrer Amazon ECR Service Quotas in der AWS Management Console.

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.

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.

Änderung	Beschreibung	Datum
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 Docker/OCI-Client-Endpunkten verknüpft sind. CloudTrail 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` dienstverknüpfte Rolle wurde hinzugefügt. Weitere Informationen finden Sie unter `ECRTemplateServiceRolePolicy`.	20. Juni 2024
Regions- und kontenübergreifende Replikation wurde für chinesische Regionen hinzugefügt.	Amazon ECR hat der Region China Unterstützung hinzugefügt, um zu filtern, welche Repositorys repliziert werden. Weitere Informationen finden Sie unter Replikation privater Bilder in Amazon ECR.	15. Mai 2024

Änderung	Beschreibung	Datum
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 ECR privaten Amazon-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- 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 ECR privaten Amazon-Registrierung.	15. November 2023

Änderung	Beschreibung	Datum
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
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 Signieren eines in einem ECR privaten Amazon-Repository gespeicherten Bildes.	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 ECR privaten Amazon-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 Anzahl der Repository-Abrufe an Amazon CloudWatch	Amazon ECR sendet Metriken zur Anzahl der Repository-Abrufe an Amazon CloudWatch. Weitere Informationen finden Sie unter ECRAmazon-Repository-Kennzahlen.	6. Januar 2022

Änderung	Beschreibung	Datum
Erweiterte Replikationsunterstützung	Amazon ECR hat die Unterstützung für die Filterung der Repositories, die repliziert werden, erweitert. Weitere Informationen finden Sie unter Replikation privater Bilder 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 kontoübergreifende 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 Ein Helm-Diagramm in ein ECR privates Amazon-Repository verschieben.	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

Änderung	Beschreibung	Datum
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 Ein Image mit mehreren Architekturen in ein ECR privates Amazon-Repository übertragen.	28. April 2020
Amazon ECR-Nutzungsmetriken	Amazon ECR hat CloudWatch Nutzungsmetriken hinzugefügt, die Aufschluss über die Ressourcennutzung Ihres Kontos geben. 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 ECRAmazon-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

Änderung	Beschreibung	Datum
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 CVE-Datenbank (Common Vulnerabilities and Exposures) des Open-Source-Projekts CoreOS Clair und liefert Ihnen eine Liste der Scanergebnisse. Weitere Informationen finden Sie unter Bilder auf Software-Sicherheitslücken in Amazon scannen ECR.	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 ECR VPC Amazon-Endgeräte.	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 Verhindern, dass Bild-Tags in Amazon überschrieben werden ECR.	25. Juli 2019
Schnittstellen-VPC -Endpunkte (AWS PrivateLink)	Unterstützung für die Konfiguration von VPC-Endpunkten 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 AWS Direct Connect. Weitere Informationen finden Sie unter ECRVPCAmazon-Schnittstellenendpunkte ()AWS PrivateLink.	25. Januar 2019

Änderung	Beschreibung	Datum
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 wurde umbenannt (vorher 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 für das Container-Image-Manifestformat 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.