aws

ユーザーガイド

Amazon ECR

API バージョン 2015-09-21

Amazon ECR: ユーザーガイド

Table of Contents

Amazon Elastic Container Registry とは

Amazon Elastic Container Registry (Amazon ECR) は、安全でスケーラブル、信頼性の高い AWS マネージドコンテナイメージレジストリサービスです。Amazon ECR は、 AWS IAM を使用したリソースベースのアクセス許可を持つプライベートリポジトリをサポートします。これは、指定されたユーザーまたは Amazon EC2 インスタンスがコンテナリポジトリとイメージにアクセスできるようにするためです。任意の CLI を使用して、Docker イメージ、Open Container Initiative (OCI) イメージ、および OCI 互換アーティファクトをプッシュ、プル、管理することが可能です。

> **Note**
>
> Amazon ECR は、パブリックコンテナイメージリポジトリもサポートしています。詳細については、Amazon ECR Public ユーザーガイドの「Amazon Elastic Container レジストリとは」を参照してください。

AWS コンテナサービスチームは、 でパブリックロードマップを維持しています GitHub。これには、チームが取り組んでいる内容に関する情報が含まれており、すべての AWS 顧客が直接フィードバックを提供できるようにします。詳細については、AWS Containers Roadmap を参照してください。

Amazon ECR のコンポーネント

Amazon ECR には、次のコンポーネントが含まれています。

レジストリ

Amazon ECR プライベートレジストリが各 AWS アカウントに提供されるため、レジストリに 1 つ以上のリポジトリを作成し、Docker イメージ、Open Container Initiative (OCI) イメージ、および OCI 互換アーティファクトを保存できます。詳細については、「Amazon ECR プライベートレジストリ」を参照してください。

認証トークン

クライアントがイメージをプッシュおよびプルするには、 AWS ユーザーとして Amazon ECR プライベートレジストリに対して認証する必要があります。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

リポジトリ

Amazon ECR リポジトリには、Docker イメージ、Open Container Initiative (OCI) イメージ、および OCI 互換アーティファクトが含まれます。詳しくは、「Amazon ECR プライベートリポジトリ」を参照してください。

リポジトリポリシー

リポジトリポリシーを使用して、リポジトリとリポジトリ内のコンテンツへのアクセス権を制御できます。詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。

イメージ

リポジトリには、コンテナイメージをプッシュおよびプルできます。開発システムでこれらのイメージは、ローカルに使用することや、Amazon ECS タスク定義と Amazon EKS ポッド仕様で使用することができます。詳細については、「Amazon ECS での Amazon ECR イメージの使用」および「Amazon EKS での Amazon ECR イメージの使用」を参照してください。

Amazon ECR の機能

Amazon ECR には次の機能があります。

- ライフサイクルポリシーを使用すると、リポジトリ内のイメージのライフサイクルを管理できます。未使用のイメージをクリーンアップするルールを定義します。ルールはリポジトリに適用する前にテストできます。詳しくは、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。
- イメージスキャンは、コンテナイメージ内のソフトウェアの脆弱性を特定するのに役立ちます。各リポジトリはプッシュ時にスキャンするように設定できます。その場合、リポジトリにプッシュされる新しい各イメージが確実にスキャンされます。その後、イメージスキャンの結果を取得できます。詳しくは、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。
- クロスリージョンおよびクロスアカウントレプリケーションを使用すると、必要な場所にイメージを簡単に作成できます。これは、レジストリ設定として、リージョンごとに構成されます。詳細については、「Amazon ECR のプライベートレジストリ設定」を参照してください。
- プルスルーキャッシュルールは、プライベート Amazon ECR レジストリのアップストリームレジストリ内のリポジトリのキャッシュ方法を提供します。プルスルーキャッシュルールを使用して、Amazon ECR は定期的にアップストリームレジストリに問い合わせて、Amazon ECR プライベートレジストリにキャッシュされたイメージが最新であることを確認します。詳細については、

「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。

Amazon ECR の開始方法

Amazon Elastic Container Service (Amazon ECS) または Amazon Elastic Kubernetes Service (Amazon EKS) を使用している場合、Amazon ECR は両方のサービスの拡張機能であるため、これらの 2 つのサービスのセットアップは Amazon ECR のセットアップと似ていることに注意してください。

Amazon ECR AWS Command Line Interface で を使用する場合は、最新の Amazon ECR 機能 AWS CLI をサポートする のバージョンを使用します。で Amazon ECR 機能のサポートが表示されない場合は AWS CLI、最新バージョンの にアップグレードします AWS CLI。の最新バージョンのインストールについては AWS CLI、「 ユーザーガイド」の「 の最新バージョンのインストールまたは更新 AWS CLIAWS Command Line Interface 」を参照してください。

および Docker を使用してコンテナイメージをプライベート Amazon ECR リポジトリに AWS CLI プッシュする方法については、「」を参照してくださいAmazon ECR でのライフサイクルを通じたイメージの移動。

Amazon ECR の料金表

Amazon ECR では、リポジトリに保存するデータの量と、イメージのプッシュとプルからのデータ転送に対してのみ料金が発生します。詳細については、Amazon ECR の料金を参照してください。

Amazon ECR でのライフサイクルを通じたイメージの移動

Amazon ECR を初めて使用する場合は、Docker CLI と で次の手順を使用してサンプルイメージ AWS CLI を作成し、デフォルトのレジストリで認証し、プライベートリポジトリを作成します。次に、 にイメージをプッシュし、プライベートリポジトリからイメージをプルします。サンプルイメージを使い終わったら、サンプルイメージとリポジトリを削除します。

AWS Management Console の代わりに を使用するには、 AWS CLI「」を参照してくださいthe section called “イメージを保存するリポジトリの作成”。

さまざまな AWS SDKs、IDE ツールキット、Windows PowerShell コマンドラインツールなど、AWS リソースの管理に使用できるその他のツールの詳細については、http://aws.amazon.com/tools/ を参照してください。

前提条件

最新の AWS CLI と Docker がインストールされておらず、すぐに使用できる状態でない場合は、次のステップに従ってこれらのツールの両方をインストールします。

のインストール AWS CLI

Amazon ECR AWS CLI で を使用するには、 AWS CLI 最新バージョンをインストールします。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interfaceのインストール」を参照してください。

Docker をインストールする

Docker は、Ubuntu のような最新の Linux ディストリビューションから macOS や Windows まで、さまざまなオペレーティングシステムで使用できます。特定のオペレーティングシステムに Docker をインストールする方法の詳細については、Docker インストールガイド を参照してください。

Docker を使用するには、ローカルの開発システムは必要ありません。Amazon EC2 をすでに使用している場合は、Amazon Linux 2023 インスタンスを起動し、Docker をインストールして開始できます。

Docker をインストール済みの場合は、この手順をスキップして「ステップ 1: Docker イメージを作成する」に進んでください。

Amazon Linux 2023 AMI を使用して Amazon EC2 インスタンスに Docker をインストールするには

1. 最新の Amazon Linux 2023 AMI を使用してインスタンスを起動します。詳細については、Amazon EC2 ユーザーガイド」の「インスタンスの起動」を参照してください。
2. インスタンスに接続します。詳細については、「Amazon EC2 ユーザーガイド」の「Linux インスタンスに接続する」を参照してください。 Amazon EC2
3. インスタンスでインストールされているパッケージとパッケージキャッシュを更新します。

```
sudo yum update -y
```

4. 最新の Docker Community Edition パッケージをインストールします。

```
sudo yum install docker
```

5. Docker サービスを開始します。

```
sudo service docker start
```

6. ec2-user を docker グループに追加すると、sudo を使用せずに Docker コマンドを実行できます。

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

7. ログアウトし、再びログインして、新しい docker グループアクセス権限を取得します。これは、現在の SSH ターミナルウィンドウを閉じて、新しいウィンドウでインスタンスに再接続することで達成できます。新しい SSH セッションには適切な docker グループ権限があります。
8. ec2-user が sudo を使用せずに Docker コマンドを実行できることを確認します。

```
docker info
```

> **Note**
>
> 場合によっては、Docker デーモンにアクセスするための ec2-user に対するアクセス権限を提供するため、インスタンスを再起動する必要があります。次のエラーが表示された場合は、インスタンスを再起動してください。

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

ステップ 1: Docker イメージを作成する

このステップでは、シンプルなウェブアプリケーションの Docker イメージを作成し、ローカルシステムまたは Amazon EC2 インスタンスでテストします。

シンプルなウェブアプリケーションの Docker イメージを作成するには

1. `Dockerfile` という名前のファイルを作成します。Dockerfile は、Docker イメージに使用する基本イメージと、そのイメージにインストールして実行するものを記述するマニフェストです。Dockerfile の詳細については、「Dockerfile リファレンス」を参照してください。

   ```
   touch Dockerfile
   ```

2. 前の手順で作成した `Dockerfile` を編集し、以下のコンテンツを追加します。

   ```
   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
   ```

この Dockerfile は、Amazon ECR パブリックでホストされているパブリック Amazon Linux 2 イメージを使用します。RUN の手順により、パッケージキャッシュが更新され、ウェブ サーバー用のいくつかのソフトウェアがインストールされてから、「Hello World!」 のコンテンツがウェブサーバーのドキュメントルートに書き込みされます。EXPOSE の命令はコンテナ上のポート 80 を公開し、CMD の命令はウェブサーバーを起動します。

3. Dockerfile から Docker イメージを作成します。

> **Note**
>
> Docker の一部のバージョンでは、下に示す相対パスの代わりに、次のコマンドで Dockerfile への完全パスが必要になる場合があります。

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

4. コンテナイメージを一覧表示します。

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

出力:

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

5. 新しく構築されたイメージを実行します。-p 80:80 オプションは、コンテナ上の公開されたポート 80 をホストシステム上のポート 80 にマッピングします。docker run の詳細については、「Docker run reference」を参照してください。

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

> **Note**
>
> Apache ウェブサーバーからの出力はターミナルウィンドウに表示されます。"Could not reliably determine the fully qualified domain name" メッセージは無視できます。

6. ブラウザーを開き、Docker を実行している、コンテナのホストサーバーを参照します。

   - EC2 インスタンスを使用している場合、これはサーバーの [Public DNS] 値であり、SSH でインスタンスに接続するときに使用するアドレスと同じです。インスタンスのセキュリティグループでポート 80 上の受信トラフィックを許可していることを確認します。
   - Docker をローカルに実行している場合は、ブラウザで http://localhost/ を参照します。
   - Windows または Mac コンピュータdocker-machineで を使用している場合は、 docker-machine ip コマンドで Docker をホストしている VirtualBox VM の IP アドレスを見つけ、*machine-name* を、使用している Docker マシンの名前に置き換えます。

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

   ウェブページに「Hello, World!」が確認できます。表示されます。

7. [Ctrl + C] キーを押して、Docker コンテナを停止します。

ステップ 2: デフォルトレジストリに対して認証する

をインストールして設定したら AWS CLI、デフォルトのレジストリに対して Docker CLI を認証します。これにより、docker コマンドは Amazon ECR を使用してイメージをプッシュおよびプルできます。は、認証プロセスを簡素化するget-login-passwordコマンド AWS CLI を提供します。

を使用して Amazon ECR レジストリに対して Docker を認証するには get-login-password、 aws ecr get-login-password コマンドを実行します。認証トークンを docker login コマンドに渡すとき、ユーザー名の AWS 値を使用し、認証先の Amazon ECR レジストリの URI を指定します。複数のレジストリに対して認証する場合は、レジストリごとにコマンドを繰り返す必要があります。

> **Important**
>
> エラーが発生した場合は、 AWS CLIの最新バージョンをインストールまたはアップグレードします。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interfaceのインストール」を参照してください。

- get-login-password (AWS CLI)

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

- Get-ECRLoginCommand (AWS Tools for Windows PowerShell)

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

ステップ 3: レポジトリを作成する

これで Amazon ECR にプッシュするイメージが用意できたので、それを保持するレポジトリを作成する必要があります。この例では、`hello-repository` イメージを後でプッシュする、`hello-world:latest` と呼ばれるレポジトリを作成します。レポジトリを作成するには、次のコマンドを実行します。

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

ステップ 4: イメージを Amazon ECR にプッシュする

ここで、前のセクションで作成した Amazon ECR リポジトリにイメージをプッシュできます。docker CLI を使用して、次の前提条件が満たされた後にイメージをプッシュします。

- の最小バージョンdockerは 1.7 です。
- Amazon ECR 認証トークンは で設定されていますdocker login。
- Amazon ECR リポジトリが存在し、リポジトリにプッシュするアクセス権がユーザーにある。

これらの前提条件が満たされたら、アカウントのデフォルトレジストリで新しく作成されたレポジトリにイメージをプッシュできます。

イメージにタグを付けて Amazon ECR にプッシュするには

1. ローカルに保存したイメージをリストし、タグを付けてプッシュするイメージを識別します。

```
docker images
```

出力:

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

2. リポジトリにプッシュするイメージにタグを付けます。

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

3. イメージをプッシュします。

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

出力:

```
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
```

ステップ 5: Amazon ECR からイメージをプルする

イメージが Amazon ECR リポジトリにプッシュされたら、他の場所からイメージをプルできます。docker CLI を使用して、次の前提条件が満たされた後にイメージをプルします。

- の最小バージョンdockerは 1.7 です。
- Amazon ECR 認証トークンは で設定されていますdocker login。
- Amazon ECR リポジトリが存在し、リポジトリからプルするアクセス許可がユーザーにある。

これらの前提条件が満たされたら、イメージをプルできます。Amazon ECR からサンプルイメージをプルするには、次のコマンドを実行します。

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

出力:

```
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
```

ステップ 6: イメージを削除する

いずれかのリポジトリにイメージが不要になった場合は、そのイメージを削除できます。イメージを削除するには、イメージが存在するリポジトリと、イメージの `imageTag`または `imageDigest`値を指定します。次の例では、イメージタグ を持つ`hello-repository`リポジトリ内のイメージを削除しますlatest。リポジトリからサンプルイメージを削除するには、次のコマンドを実行します。

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

ステップ 7: リポジトリを削除する

イメージのリポジトリ全体が不要になった場合は、リポジトリを削除できます。次の例では、 `--force`フラグを使用して、イメージを含むリポジトリを削除します。イメージ (およびその中に含まれるすべてのイメージ) を含むリポジトリを削除するには、次のコマンドを実行します。

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

Amazon ECR のパフォーマンスを最適化する

Amazon ECR を使用する際のパフォーマンスを最適化するために、設定と戦略に関する以下の推奨事項を使用できます。

レイヤーの同時アップロードを利用するには、Docker 1.10 以降を使用する

Docker イメージは、イメージの中間ビルドステージであるレイヤーで構成されます。Dockerfile の各行により、新しいレイヤーが作成されます。Docker 1.10 以降を使用する場合、Docker はデフォルトで可能な限り多くのレイヤーを Amazon ECR への同時アップロードとしてプッシュし、それによりアップロード時間が高速になります。

小さなベースイメージを使用する

Docker Hub を通じて利用できるデフォルトイメージには、アプリケーションが要求しない多くの依存関係が含まれている場合があります。Docker コミュニティで他のユーザーによって作成、管理される、より小さいイメージを使用するか、Docker の最小スクラッチイメージを使用して独自のベースイメージを構築することを検討します。詳細については、Docker のドキュメントの「ベースイメージの作成」を参照してください。

Dockerfile で、最も変更の少ない依存関係を前に配置する

Docker はレイヤーをキャッシュし、それによりビルド時間を高速化します。最後のビルド以降にレイヤーで何も変更されていない場合、Docker はレイヤーを再構築する代わりにキャッシュしたバージョンを使用します。ただし、各レイヤーは、それ以前のレイヤーに依存しています。レイヤーが変更された場合、Docker はそのレイヤーだけでなく、そのレイヤーの後のレイヤーも同様に再コンパイルします。

Docker ファイルの再構築と、レイヤーの再アップロードに必要な時間を最小化するため、最も変更を行わない依存関係を Dockerfile で前に配置します。頻繁に変更を行う依存関係 (アプリケーションのソースコードなど) はスタックで後に配置します。

コマンドをチェーン処理して不要なファイルの保存を避ける

レイヤーで作成された中間ファイルは、それ以降のレイヤーで削除された場合でも、そのレイヤーの一部として残ります。次の例を考えます。

```
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
```

この例では、最初と 2 番目の RUN コマンドによって作成されたレイヤーには、元の .tar.gz ファイルと解凍されていないそのすべてのコンテンツが含まれます。ただし、.tar.gz ファイルは 4 番目の RUN コマンドによって削除されます。これらのコマンドを 1 つの RUN ステートメントにチェーン処理して、不要なファイルが最終的な Docker イメージの一部とはならないようにできます。

```
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
```

最も近いリージョンのエンドポイントを使用する

アプリケーションが実行されている場所に最も近いリージョンのエンドポイントを使用することにより、Amazon ECR からのイメージのプルのレイテンシーを減らすことができます。アプリケーションが Amazon EC2 インスタンスで実行されている場合、次のシェルコードを使用して、インスタンスのアベイラビリティーゾーンからリージョンを取得できます。

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

リージョンは、 --regionパラメータを使用して AWS CLI コマンドに渡すことも、 aws configure コマンドを使用してプロファイルのデフォルトリージョンとして設定することもできます。
AWS SDK を使用して呼び出しを行うときにリージョンを設定することもできます。詳細については、ご使用の特定のプログラミング言語の SDK のドキュメントを参照してください。

Amazon ECR プライベートレジストリ

Amazon ECR プライベートレジストリは、可用性の高いスケーラブルなアーキテクチャでコンテナイメージをホストします。プライベートレジストリを使用して、Docker イメージ、Open Container Initiative (OCI) イメージ、アーティファクトで構成されるプライベートイメージリポジトリを管理できます。各 AWS アカウントには、デフォルトのプライベート Amazon ECR レジストリが提供されます。Amazon ECR パブリックレジストリの詳細については、Amazon Elastic Container Registry Public ユーザーガイドの「Public registries」を参照してください。

プライベートレジストリの概念

- デフォルトのプライベートレジストリの URL は `https://aws_account_id.dkr.ecr.us-west-2.amazonaws.com` です。
- デフォルトでは、アカウントにはプライベートリポジトリ内のリポジトリへの読み取りおよび書き込み許可があります。ただし、ユーザーには、Amazon ECR APIs を呼び出したり、プライベートリポジトリとの間でイメージをプッシュまたはプルしたりするためのアクセス許可が必要です。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するための管理ポリシーが複数用意されています。詳細については、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。
- プライベートレジストリに対して Docker クライアントを認証し、docker push コマンドと docker pull コマンドを使用してそのレジストリ内のリポジトリとの間でイメージをプッシュおよびプルできるようにする必要があります。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。
- プライベートリポジトリは、 ユーザーアクセスポリシーとリポジトリポリシーによって制御できます。リポジトリポリシーの詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。
- プライベートレジストリのリポジトリは、プライベートレジストリのレプリケーションを設定することで、プライベートレジストリ内および複数のアカウント間の複数のリージョンにわたってでレプリケートできます。詳細については、「Amazon ECR でのプライベートイメージのレプリケーション」を参照してください。

Amazon ECR でのプライベートレジストリ認証

AWS Management Console、、または AWS SDKs を使用して AWS CLI、プライベートリポジトリを作成および管理できます。また、これらの方法を使用して、イメージの一覧表示や削除などのいく

つかのアクションをイメージで実行できます。これらのクライアントは標準の AWS 認証方法を使用します。Amazon ECR API を使用してイメージをプッシュおよびプルできますが、Docker CLI または言語固有の Docker ライブラリの使用をお勧めします。

Docker CLI は、ネイティブの IAM 認証方法をサポートしていません。Amazon ECR が、Docker のプッシュ要求とプル要求を認証および認可できるようにするには、追加の手順を実行する必要があります。

以下のセクションで説明するレジストリ認証方法を使用できます。

Amazon ECR 認証情報ヘルパーを使用する

Amazon ECR には Docker 認証情報ヘルパーが用意されているので、Amazon ECR に対してイメージをプッシュおよびプルするときに、Docker 認証情報をより簡単に保存および使用できます。インストールおよび設定手順については、「Amazon ECR Docker 認証情報ヘルパー」を参照してください。

> **Note**
>
> Amazon ECR Docker 認証情報ヘルパーは、現在、多要素認証 (MFA) をサポートしていません。

認可トークンを使用する

認可トークンの許可スコープは、認証トークンの取得に使用される IAM プリンシパルの許可スコープと一致します。認証トークンは、IAM プリンシパルからアクセス可能で 12 時間有効な Amazon ECR レジストリにアクセスするために使用されます。認証トークンを取得するには、GetAuthorizationToken API オペレーションを使用して、ユーザー名AWSとエンコードされたパスワードを含む base64 でエンコードされた認証トークンを取得する必要があります。`get-login-password` コマンドは、 AWS CLI 認証トークンを取得してデコードすることで、これを簡略化します。このトークンをdocker loginコマンドにパイプして認証できます。

get-login を使用して Amazon ECR プライベートレジストリに対して Docker を認証するには

- を使用して Amazon ECR レジストリに対して Docker を認証するには get-login-password、aws ecr get-login-password コマンドを実行します。認証トークンを docker login コマンドに渡すとき、ユーザー名の AWS 値を使用し、認証先の Amazon ECR レジストリの URI を指定しま

す。複数のレジストリに対して認証する場合は、レジストリごとにコマンドを繰り返す必要があります。

> **Important**
>
> エラーが発生した場合は、 AWS CLIの最新バージョンをインストールまたはアップグレードします。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interfaceのインストール」を参照してください。

- get-login-password (AWS CLI)

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

- Get-ECRLoginCommand (AWS Tools for Windows PowerShell)

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

HTTP API 認証を使用する

Amazon ECR は Docker Registry HTTP API をサポートします。ただし、Amazon ECR はプライベートレジストリであるため、すべての HTTP リクエストで認可トークンを提供する必要があります。の -Hオプションを使用して HTTP 認証ヘッダーを追加curlし、 get-authorization-token AWS CLI コマンドによって提供される認証トークンを渡すことができます。

Amazon ECR HTTP API を使用して認証するには

1. で認証トークンを取得し AWS CLI 、環境変数に設定します。

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

2. API に対して認証するには、$TOKEN 変数を curl の -H オプションに渡します。たとえば、次のコマンドでは、Amazon ECR レポジトリでイメージタグを一覧表示します。詳細については、Docker レジストリ HTTP API リファレンスドキュメントを参照してください。

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

出力は次のとおりです。

```
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"]}
```

Amazon ECR のプライベートレジストリ設定

Amazon ECR はプライベートレジストリ設定を使用してレジストリレベルで機能を構成します。プライベートレジストリ設定は、リージョンごとに個別に構成されます。プライベートレジストリ設定を使用して、次の機能を設定できます。

- [レジストリ許可] — レジストリ許可ポリシーは、レプリケーション権限とプルスルーキャッシュ権限を制御します。詳細については、「Amazon ECR でのプライベートレジストリのアクセス許可」を参照してください。
- [プルスルーキャッシュルール] — プルスルーキャッシュルールを使用して、Amazon ECR プライベートレジストリのアップストリームレジストリからイメージをキャッシュします。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。
- [レプリケーション設定] — レプリケーション設定は、リポジトリを AWS リージョン間でコピーするのか、アカウントをまたいでコピーするのかを制御するために使用されます。詳細については、「Amazon ECR でのプライベートイメージのレプリケーション」を参照してください。
- [リポジトリ作成テンプレート] — リポジトリ作成テンプレートは、Amazon ECR がユーザーに代わって新しいリポジトリを作成したときに適用される標準設定を定義するために使用されます。例えば、プルスルーキャッシュアクションによって作成されたリポジトリなどです。詳細については、「プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート」を参照してください。

- [Scanning configuration] (スキャン設定) — デフォルトでは、レジストリでベーシックスキャンが有効です。オペレーティングシステムとプログラミング言語パッケージの脆弱性の両方をスキャンする自動連続スキャンモードを提供する拡張スキャンを有効にできます。詳細については、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。

Amazon ECR でのプライベートレジストリのアクセス許可

Amazon ECR はレジストリポリシーを使用して、プライベートレジストリレベルの AWS プリンシパルに許可を付与します。これらの許可は、レプリケーションへのアクセス範囲とキャッシュ機能のプルスルーに使用されます。

Amazon ECR では、プライベートレジストリレベルでのみ、以下の許可が適用されます。レジストリポリシーにアクションが追加されると、エラーが発生します。

- `ecr:ReplicateImage` – ソースレジストリと呼ばれる別のアカウントに、そのイメージをレジストリにレプリケートする許可を付与します。これは、クロスアカウントレプリケーションのみに使用されます。
- `ecr:BatchImportUpstreamImage` – 外部イメージを取得し、プライベートレジストリにインポートするアクセス許可を付与します。
- `ecr:CreateRepository` – プライベートレジストリにリポジトリを作成するアクセス許可を付与します。レプリケートまたはキャッシュされたイメージを保存するリポジトリがプライベートレジストリにまだ存在しない場合は、この許可が必要となります。

> **Note**
>
> プライベートレジストリ許可ポリシーに `ecr:*` アクションを追加することが可能である場合、ワイルドカードを使用するのではなく、使用している機能に基づいて必要な特定のアクションのみを追加することがベストプラクティスと考えられます。

トピック

- Amazon ECR のプライベートレジストリポリシーの例
- Amazon ECR でのクロスアカウントレプリケーションのためのレジストリ許可の付与
- Amazon ECR でのプルスルーキャッシュのレジストリ許可の付与

Amazon ECR のプライベートレジストリポリシーの例

以下の例では、Amazon ECR レジストリに対してユーザーが所有するアクセス許可を制御するために使用できるレジストリ許可ポリシーステートメントを示しています。

> **Note**
>
> 各例において、レジストリ許可ステートメントから `ecr:CreateRepository` アクションが削除されていても、レプリケーションは引き続き実行される場合があります。ただし、レプリケーションを成功させるには、アカウント内に同じ名前のリポジトリを作成する必要があります。

例: ソースアカウントのルートユーザーにすべてのリポジトリのレプリケーションを許可する

次のレジストリアクセス許可ポリシーは、ソースアカウントのルートユーザーがすべてのリポジトリをレプリケートすることを許可します。

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

例: 複数のアカウントのルートユーザーを許可する

次のレジストリアクセス許可ポリシーには、2 つのステートメントがあります。各ステートメントにより、ソースアカウントのルートユーザーはすべてのリポジトリをレプリケートできます。

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

例: ソースアカウントのルートユーザーに、プレフィックス **prod-** が付くすべてのリポジトリのレプリケーションを許可する

次のレジストリ許可ポリシーでは、ソースアカウントのルートユーザーが で始まるすべてのリポジトリをレプリケートできます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-*"
            ]
        }
    ]
}
```

Amazon ECR でのクロスアカウントレプリケーションのためのレジストリ許可の付与

クロスアカウントポリシータイプを使用して AWS プリンシパルに許可を付与し、ソースレジストリから指定レジストリへのレポジトリのレプリケーションを許可します。デフォルトでは、ユーザーには、独自のレジストリ内でクロスリージョンレプリケーションを設定する許可があります。レジストリにコンテンツをレプリケートする許可を別のアカウントに付与する場合、必要な操作は、レジストリポリシーを構成することだけです。

レジストリポリシーは、ecr:ReplicateImage API アクションに許可を付与する必要があります。この API は、複数のリージョンまたはアカウント間でイメージをレプリケートできる内部 Amazon ECR API です。ecr:CreateRepository の許可を付与することもできます。この場合、レポジトリが存在しない場合、Amazon ECR がレジストリ内にリポジトリを作成できるようになります。ecr:CreateRepository の許可が指定されていない場合、ソースリポジトリと同じ名前のリ

ポジトリを手動でレジストリに作成する必要があります。どちらも行われていない場合、レプリケーションは失敗します。失敗した CreateRepository または ReplicateImage API アクションは に表示されます CloudTrail。

レプリケーションの許可ポリシーを構成するには (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、レジストリポリシーを設定するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Registry permissions] (レジストリー許可) の順に選択します。
4. [Registry permissions] (レジストリー許可) ページで [Generate statement] (ステートメントを生成) を選択します。
5. ポリシージェネレータを使用してポリシーステートメントメントを定義するには、次の手順を実行します。

   a. [Policy type] (ポリシータイプ) で、[Cross account policy] (クロスアカウントポリシー) を選択します。

   b. [Statement ID] (ステートメント ID) に一意のステートメント ID を入力します。このフィールドは、レジストリポリシーの `Sid` として使用されます。

   c. [Accounts] (アカウント) に、許可を付与する各アカウントのアカウント ID を入力します。複数のアカウント ID を指定する場合は、カンマで区切ります。
6. [Preview policy statement] (ポリシーステートメントをプレビュー) セクションを展開して、レジストリの許可ポリシーステートメントを確認します。
7. ポリシーステートメントを確認したら、[Add to policy] (ポリシーに追加) を選択してレジストリにポリシーを保存します。

レプリケーションの許可ポリシーを構成するには (AWS CLI)

1. `registry_policy.json` という名前のファイルを作成し、レジストリポリシーの情報を入力します。

```
{
    "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. ポリシーファイルを使用してレジストリポリシーを作成します。

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

3. レジストリのポリシーを取得して確認します。

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

Amazon ECR でのプルスルーキャッシュのレジストリ許可の付与

Amazon ECR プライベートレジストリ許可は、プルスルーキャッシュを使用する個々の IAM エンティティのアクセス許可のスコープ設定に使用できます。レジストリ許可ポリシーによって付与されるアクセス許可よりも多くのアクセス許可が IAM ポリシーによって IAM エンティティに付与される場合、IAM ポリシーが優先されます。

プライベートレジストリの許可ポリシーを作成するには (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ許可ステートメントを設定するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Registry permissions] (レジストリー許可) の順に選択します。

4. [Registry permissions] (レジストリー許可) ページで [Generate statement] (ステートメントを生成) を選択します。
5. 作成するプルスルーキャッシュ許可ポリシーステートメントごとに、次の操作を行います。

   a. [ポリシータイプ] で、[プルスルーキャッシュポリシー] を選択します。
   b. [ステートメント ID] で、プルスルーキャッシュステートメントポリシーの名前を指定します。
   c. [IAM entities] (IAM エンティティ) で、ポリシーに含めるユーザー、グループ、またはロールを指定します。
   d. [リポジトリ名前空間] で、ポリシーを関連付けるプルスルーキャッシュルールを選択します。
   e. [リポジトリ名] で、ルールを適用するリポジトリベース名を指定します。たとえば、Amazon ECR パブリックで Amazon Linux リポジトリを指定する場合、リポジトリ名は `amazonlinux` になります。

Amazon ECR プライベートリポジトリ

Amazon ECR プライベートリポジトリには、Docker イメージ、Open Container Initiative (OCI) イメージ、および OCI 互換アーティファクトが含まれています。イメージリポジトリを作成、モニタリング、削除し、Amazon ECR API オペレーションまたは Amazon ECR コンソールのリポジトリセクションを使用して、イメージリポジトリにアクセスできるユーザーを制御するアクセス許可を設定できます。Amazon ECR は Docker CLI とも統合されているため、開発環境からリポジトリにイメージをプッシュおよびプルできます。

トピック

プライベートリポジトリの概念

- デフォルトでは、アカウントにはデフォルトリポジトリ (*aws_account_id*.dkr.ecr.*region*.amazonaws.com) 内のリポジトリへの読み取りおよび書き込みアクセス権があります。ただし、ユーザーには、Amazon ECR API への呼び出しと、リポジトリに対するイメージのプッシュまたはプルを行う許可が必要です。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するための管理ポリシーが複数用意されています。詳細については、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。
- リポジトリは、 ユーザーアクセスポリシーと個々のリポジトリポリシーによって制御できます。詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。
- リポジトリ名では、似たリポジトリをグループ化するのに使用できる名前空間をサポートできます。たとえば、同じレジストリを使用するチームが複数ある場合、チーム A が team-a 名前空間を使用し、チーム B が team-b 名前を使用することが可能です。こうすることで、各チームは、web-app という独自のイメージを持つことができます。各イメージの先頭には、チームの名前空間が付けられます。この設定により、各チームは干渉することなくイメージを同時に使用でき

ます。チーム A のイメージは `team-a/web-app` で、チーム B のイメージは `team-b/web-app` です。

- イメージは、独自のレジストリ内およびアカウント間でリージョン間で他のリポジトリにレプリケートできます。これを行うには、レジストリ設定でレプリケーション構成を指定します。詳細については、「Amazon ECR のプライベートレジストリ設定」を参照してください。

イメージを保存する Amazon ECR プライベートリポジトリの作成

Amazon ECR プライベートリポジトリを作成し、そのリポジトリを使用してコンテナイメージを保存します。 AWS Management Consoleを使用してプライベートリポジトリを作成する手順は、次のとおりです。を使用してリポジトリを作成する手順については、 AWS CLI「」を参照してくださいステップ 3: レポジトリを作成する。

リポジトリを作成するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、リポジトリを作成するリージョンを選択します。
3. リポジトリページで、プライベートリポジトリ を選択し、リポジトリの作成 を選択します。
4. [Visibility settings] (可視性の設定) で、[Private] (プライベート) が選択されていることを確認します。
5. [リポジトリ名] に、リポジトリの一意の名前を入力します。リポジトリ名は単独で指定できます (`nginx-web-app` など)。リポジトリ名に名前空間を付けて、リポジトリをカテゴリでグループ化することもできます (`project-a/nginx-web-app` など)。

   > **Note**
   >
   > リポジトリ名には最大 256 文字まで含めることができます。名前は英字で始まる必要があり、小文字、数字、複数のハイフン、複数のアンダースコア、ピリオド、複数のスラッシュのみを含めることができます。ダブルハイフン、ダブルアンダースコア、またはダブルフォワードスラッシュの使用はサポートされていません。

6. [Tag immutability] (タグの不変性) で、このリポジトリのタグの変更可能性の設定を選択します。タグがイミュータブルに設定されたリポジトリでは、イメージタグが上書きされるのを防ぐことができます。詳細については、「Amazon ECR でイメージタグが上書きされないようにする」を参照してください。

7. [Scan on push] (プッシュ時にスキャン) では、基本的なスキャンに対してはリポジトリレベルでのスキャン設定を指定できますが、プライベートレジストリレベルでスキャン設定を指定することがベストプラクティスとなります。プライベートレジストリでスキャン設定を指定することで、拡張スキャンまたはベーシックスキャンのいずれかを有効にすることができ、スキャンするリポジトリを指定するフィルターも定義できます。詳細については、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。
8. KMS 暗号化 では、 を使用してリポジトリ内のイメージの暗号化を有効にするかどうかを選択します AWS Key Management Service。デフォルトでは、KMS 暗号化が有効になっている場合、Amazon ECR はエイリアス で AWS マネージドキー (KMS キー) を使用しますaws/ecr。このキーは、KMS 暗号化を有効にしたリポジトリを初めて作成するときに、アカウントに作成されます。詳細については、「保管中の暗号化」を参照してください。
9. KMS 暗号化が有効になっている場合は、[Customer encryption settings (advanced)] (カスタマー暗号化設定 (高度)) をクリックして、独自の KMS キーを選択します。KMS キーは クラスターと同じリージョンにある必要があります。 AWS KMS キーの作成 を選択して AWS KMS コンソールに移動し、独自のキーを作成します。
10. [Create repository (リポジトリの作成)] を選択します。

次のステップ

イメージをリポジトリにプッシュする手順を表示するには、リポジトリを選択し、プッシュコマンドを表示 を選択します。リポジトリへのイメージのプッシュの詳細については、「Amazon ECR プライベートリポジトリへのイメージのプッシュ」を参照してください。

Amazon ECR でのプライベートリポジトリの内容と詳細の表示

プライベートリポジトリを作成したら、 でリポジトリの詳細を表示できます AWS Management Console。

- リポジトリに保存されているイメージ
- リポジトリに格納されている各イメージの詳細 (各イメージのサイズと SHA ダイジェストを含む)
- リポジトリのコンテンツに指定されたスキャン頻度
- リポジトリにアクティブなプルスルーキャッシュルールが関連付けられているかどうか
- リポジトリの暗号化設定

> **Note**
>
> Docker バージョン 1.9 以降、Docker クライアントは V2 Docker レジストリにプッシュする前にイメージレイヤーを圧縮します。docker images コマンドの出力は、圧縮されていないイメージのサイズを表示します。したがって、Docker は、 AWS Management Consoleで表示されているイメージよりも大きなイメージを返す可能性があることに注意してください。

リポジトリ情報を表示するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、表示するリポジトリを含むリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. [Repositories] (リポジトリ) ページで、[Private] (プライベート) タブを選択し、表示するレポジトリを選択します。
5. リポジトリの詳細ページでは、コンソールはデフォルトで [Images] (イメージ) ビューを表示します。ナビゲーションメニューを使用して、リポジトリに関するその他の情報を表示します。
   - [Summary] (概要) をクリックして、リポジトリの詳細とリポジトリのプルカウントデータを表示します。
   - [Images] (イメージ) を選択して、リポジトリ内のイメージタグに関する情報を表示します。イメージの詳細情報を表示するには、イメージタグを選択します。詳細については、「Amazon ECR でのイメージの詳細の表示」を参照してください。

     削除するイメージのうちタグが付いていないものがある場合、削除するリポジトリの左側のボックスを選択し、[Delete] (削除) を選択できます。詳細については、「Amazon ECR でのイメージの削除」を参照してください。
   - [Permissions (アクセス許可)] を選択し、リポジトリに適用されるリポジトリポリシーを表示します。詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。
   - [ライフサイクルポリシー] を選択し、リポジトリに適用されるライフサイクルポリシールールを表示します。ライフサイクルイベント履歴もここに表示されます。詳細については、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。
   - [タグ] を選択して、リポジトリに適用されているメタデータタグを表示します。

Amazon ECR でのプライベートリポジトリの削除

使用し終わったリポジトリは削除できます。でリポジトリを削除すると AWS Management Console、リポジトリに含まれるすべてのイメージも削除されます。これは元に戻すことはできません。

> **Important**
>
> 削除されたリポジトリ内のイメージも削除されます。このオペレーションは元に戻すことができません。

リポジトリを削除するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、削除するリポジトリを含むリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. [Repositories] (リポジトリ) ページで、[Private] (プライベート) タブを選択し、削除するレポジトリを選択して [Delete] (削除) を選びます。
5. [***repository_name*** の削除] ウィンドウで、選択されたリポジトリを削除することを確認し、[削除] を選択します。

Amazon ECR のプライベートリポジトリポリシー

Amazon ECR では、リソースベースのアクセス権限を使用してリポジトリへのアクセスを制御します。リソースベースのアクセス許可を使用すると、リポジトリにアクセスできるユーザーまたはロールと、リポジトリで実行できるアクションを指定できます。デフォルトでは、リポジトリを作成した AWS アカウントのみがリポジトリにアクセスできます。リポジトリへの追加のアクセスを許可するリポジトリポリシーを適用できます。

トピック

- レポジトリポリシーと IAM ポリシー
- Amazon ECR のプライベートリポジトリポリシーの例
- Amazon ECR でのプライベートリポジトリポリシーステートメントの設定

レポジトリポリシーと IAM ポリシー

Amazon ECR リポジトリポリシーは、個別の Amazon ECR リポジトリへのアクセスを制御することを目的として特に使用される IAM ポリシーのサブセットです。IAM ポリシーは、一般的に Amazon ECR サービス全体にアクセス権限を適用するために使用されますが、特定のリソースへのアクセスを制限するために使用することもできます。

Amazon ECR リポジトリポリシーと IAM ポリシーはどちらも、特定のユーザーまたはロールがリポジトリで実行できるアクションを決定するときに使用されます。ユーザーまたはロールがレポジトリから 1 つのアクションの実行を許可されていても、IAM ポリシーからアクセス権限を拒否される場合 (またはその逆の場合)、そのアクションは拒否されます。ユーザーあるいはロールは、レポジトリポリシーまたは IAM ポリシーのいずれかのみでアクションへのアクセスが許可されていることを必要とし、その両方でこのアクションが許可されている必要はありません。

> **Important**
>
> Amazon ECR では、ユーザーがレジストリへの認証を行って、Amazon ECR リポジトリに対するイメージのプッシュまたはプルを行う前に、IAM ポリシーを介して `ecr:GetAuthorizationToken` API への呼び出しを行う許可が必要です。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するいくつかのマネージド IAM ポリシーが用意されています。詳細については、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。

これらのいずれかのポリシータイプを使用して、次の例に示すようにレポジトリへのアクセスを制御します。

この例は、特定のユーザーがリポジトリ内でリポジトリとイメージを説明することを許可する Amazon ECR リポジトリポリシーを示しています。

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

```
            ]
        }
    ]
}
```

この例は、リソースパラメータを使用してポリシーで 1 つのレポジトリ (レポジトリの完全な ARN で指定) を対象とすることで、上記と同じ目的を達成する IAM ポリシーを示しています。Amazon リソースネーム (ARN) 形式の詳細については、「リソース」を参照してください。

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

Amazon ECR のプライベートリポジトリポリシーの例

> **Important**
>
> このページのリポジトリポリシーの例は、Amazon ECR プライベートリポジトリに適用するためのものです。Amazon ECR リポジトリをリソースとして指定するように変更しない限り、IAM プリンシパルと直接使用すると正しく動作しません。リポジトリポリシーの設定詳細については、「Amazon ECR でのプライベートリポジトリポリシーステートメントの設定」を参照してください。

Amazon ECR リポジトリポリシーは、個別の Amazon ECR リポジトリへのアクセスを制御することを目的として特に使用される IAM ポリシーのサブセットです。IAM ポリシーは、一般的に Amazon ECR サービス全体にアクセス権限を適用するために使用されますが、特定のリソースへのアクセスを制限するために使用することもできます。詳細については、「レポジトリポリシーと IAM ポリシー」を参照してください。

以下のリポジトリポリシーの例では、Amazon ECR プライベートリポジトリへのアクセスを制御するために使用できる許可ステートメントを示しています。

> **Important**
>
> Amazon ECR では、ユーザーがレジストリへの認証を行って、Amazon ECR リポジトリに対するイメージのプッシュまたはプルを行う前に、IAM ポリシーを介して `ecr:GetAuthorizationToken` API への呼び出しを行う許可が必要です。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するいくつかのマネージド IAM ポリシーが用意されています。詳細については、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。

例: 1 名以上の ユーザーを許可する

次のリポジトリポリシーでは、1 名以上の ユーザーがリポジトリに対してイメージをプッシュまたはプルできます。

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

```
}
```

例: 別のアカウントを許可する

次のリポジトリポリシーでは、特定のアカウントにイメージのプッシュが許可されます。

> **Important**
>
> アクセス権限を付与するアカウントには、リポジトリポリシーを作成するリージョンが有効になっている必要があります。有効でない場合、エラーが発生します。

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

次のリポジトリポリシーでは、一部のユーザーにイメージのプルが許可されますが (*pull-user-1* および *pull-user-2*)、別のユーザーにはフルアクセスが許可されます (*admin-user*)。

> **Note**
>
> で現在サポートされていないより複雑なリポジトリポリシーについては AWS Management Console、 set-repository-policy AWS CLI コマンドを使用してポリシーを適用できます。

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

例: すべて拒否する

次のリポジトリポリシーでは、すべてのアカウントのすべてのユーザーに対してイメージのプルが拒否されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPull",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
```

```
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

例: アクセスを特定の IP アドレスに制限する

次の例では、特定のアドレス範囲からリポジトリに適用された場合に、Amazon ECR オペレーションを実行するアクセス許可をすべてのユーザーに対して拒否します。

このステートメントの条件では、許可されたインターネットプロトコルバージョン 4 (IPv4) IP アドレスの `54.240.143.*` 範囲を識別します。

`Condition` ブロックは、 `NotIpAddress` 条件と、 AWS全体の`aws:SourceIp`条件キーである 条件キーを使用します。これらの条件キーの詳細については、「AWS グローバル条件コンテキストキー」を参照してください。`aws:sourceIp`IPv4 値は標準の CIDR 表記を使用します。詳細については、IAM ユーザーガイドの IP アドレス条件演算子 を参照してください。

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

例: AWS サービスを許可する

次のリポジトリポリシーは、そのサービスとの統合に必要な Amazon ECR API アクション AWS CodeBuild へのアクセスを許可します。次の例を使用する場合、`aws:SourceArn` 条件キーと

aws:SourceAccount 条件キーを使用して、これらのアクセス許可を引き受けることができるリソースの範囲を指定する必要があります。詳細については、「 ユーザーガイド」の「 の Amazon ECR サンプル CodeBuildAWS CodeBuild 」を参照してください。

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

Amazon ECR でのプライベートリポジトリポリシーステートメントの設定

以下の AWS Management Console 手順に従って、 のリポジトリにアクセスポリシーステートメントを追加できます。リポジトリごとに複数のポリシーステートメントを追加できます。エンドポイントポリシーの例については、「Amazon ECR のプライベートリポジトリポリシーの例」を参照してください。

> **Important**
>
> Amazon ECR では、ユーザーがレジストリへの認証を行って、Amazon ECR リポジトリに対するイメージのプッシュまたはプルを行う前に、IAM ポリシーを介して

`ecr:GetAuthorizationToken` API への呼び出しを行う許可が必要です。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するいくつかのマネージド IAM ポリシーが用意されています。詳細については、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。

リポジトリポリシーステートメントを設定するには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、ポリシーステートメントをオンに設定するリポジトリが含まれるリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. リポジトリページで、ポリシーステートメントを有効に設定するリポジトリを選択して、リポジトリの内容を表示します。
5. リポジトリイメージのリストビューのナビゲーションペインで、[アクセス権限]、[編集] を選択します。

> **Note**
>
> ナビゲーションペインに [Permissions] (アクセス許可) オプションが表示されない場合、画面がリポジトリイメージリストビューであることを確認します。

6. 許可を編集ページで [ステートメントを追加] を選択します。
7. [ステートメント名] に、ステートメントの名前を入力します。
8. [効果] で、ポリシーステートメントの結果を許可または明示的な拒否のどちらにするかを指定します。
9. [プリンシパル] で、ポリシーステートメントを適用する範囲を選択します。詳細については、IAM ユーザーガイドの「AWS JSON ポリシー要素: プリンシパル」を参照してください。

   - すべての認証済み AWS ユーザーにステートメントを適用するには、「全員」(*) チェックボックスをオンにします。
   - [サービスプリンシパル] で、特定のサービスにステートメントを適用するサービスプリンシパル名 (`ecs.amazonaws.com` など) を指定します。
   - AWS アカウント IDs には、 AWS アカウント番号 (など111122223333) を指定して、特定の AWS アカウントのすべてのユーザーにステートメントを適用します。カンマ区切りのリストを使用して複数のアカウントを指定できます。

> **Important**
>
> アクセス権限を付与するアカウントには、リポジトリポリシーを作成するリージョンが有効になっている必要があります。有効でない場合、エラーが発生します。

- IAM エンティティ で、ステートメントを適用する AWS アカウントのロールまたはユーザーを選択します。

> **Note**
>
> で現在サポートされていないより複雑なリポジトリポリシーについては AWS Management Console、 set-repository-policy AWS CLI コマンドを使用してポリシーを適用できます。

10. [アクション] で、個別の API オペレーションのリストとの間でポリシーステートメントを適用する Amazon ECR API オペレーションの範囲を選択します。
11. 完了したら、[Save] を選択してポリシーを設定します。
12. 追加する各レポジトリポリシーに対して、前のステップを繰り返します。

Amazon ECR でのプライベートリポジトリのタグ付け

Amazon ECR リポジトリの管理に役立つように、 AWS リソースタグ を使用して、新規または既存の Amazon ECR リポジトリに独自のメタデータを割り当てることができます。たとえば、アカウントの Amazon ECR リポジトリに対して各リポジトリの所有者を追跡しやすくするため、一連のタグを定義できます。

タグの基本

タグには、Amazon ECR に関連する意味はなく、完全に文字列として解釈されます。タグは自動的にはリソースに割り当てられません。タグのキーと値は編集でき、タグはリソースからいつでも削除できます。タグの値を空の文字列に設定することはできますが、タグの値を null に設定することはできません。特定のリソースについて既存のタグと同じキーを持つタグを追加した場合、以前の値は新しい値によって上書きされます。リソースを削除すると、リソースのタグも削除されます。

Amazon ECR コンソール、、 AWS CLIおよび Amazon ECR API を使用してタグを操作できます。

AWS Identity and Access Management (IAM) を使用すると、 AWS アカウント内のどのユーザーがタグを作成、編集、または削除するためのアクセス許可を持っているかを制御できます。IAM ポリシーのタグの詳細については、「」を参照してくださいthe section called “タグベースのアクセスコントロールを使用する”。

請求用のリソースにタグを付ける

Amazon ECR リポジトリに追加するタグは、コスト配分を有効にした後にコストと使用状況レポートでコスト配分を確認するときに便利です。詳しくは、「Amazon ECR 使用状況レポート」を参照してください。

リソースを組み合わせたコストを確認するには、同じタグキー値を持つリソースに基づいて、請求情報を整理します。例えば、複数のリソースに特定のアプリケーション名のタグを付け、請求情報を整理することで、複数のサービスを利用しているアプリケーションの合計コストを確認することができます。タグによるコスト配分レポートの設定の詳細については、AWS Billing ユーザーガイドの「毎月のコスト配分レポート」を参照してください。

> **Note**
>
> レポートを有効にすると、約 24 時間後に、今月のデータを表示できるようになります。

Amazon ECR のプライベートリポジトリへのタグの追加

プライベートリポジトリにタグを追加できます。

タグの名前とベストプラクティスの詳細については、「 リソースのタグ付けユーザーガイド」の「タグの命名制限と要件」と「ベストプラクティス」を参照してください。 AWS

リポジトリへのタグの追加 (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、使用するリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. [リポジトリ] ページで、タグ付けするリポジトリの横にあるチェックボックスを選択します。
5. [アクション] メニューから、[リポジトリタグ] を選択します。
6. [リポジトリタグ] ページで、[タグの追加]、[タグの追加] の順に選択します。
7. [リポジトリタグを編集] ページで、各タグのキーと値を指定してから [保存] を選択します。

リポジトリ (AWS CLI または API) へのタグの追加

または API を使用して、1 つ以上のタグを追加 AWS CLI または上書きできます。

- AWS CLI - タグリソース
- API アクション - TagResource

次の例は、 を使用してタグを追加する方法を示しています AWS CLI。

例 1: リポジトリにタグを付ける

次のコマンドは、リポジトリにタグを付けます。

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

例 2: リポジトリに複数のタグを付ける

次のコマンドは、リポジトリに 3 つのタグを追加します。

```
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
```

例 3: リポジトリのタグを一覧表示する

次のコマンドは、リポジトリに関連付けられているタグを一覧表示します。

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

例 4: リポジトリを作成してタグを追加する

次のコマンドでは、`test-repo` という名前のリポジトリを作成し、キーが `team` で値が `devs` のタグを追加します。

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

Amazon ECR のプライベートリポジトリからのタグの削除

プライベートリポジトリからタグを削除できます。

プライベートリポジトリからタグを削除するには (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、使用するリージョンを選択します。
3. [リポジトリ] ページで、タグを削除したいリポジトリの横にあるチェックボックスを選択します。
4. [アクション] メニューから、[リポジトリタグ] を選択します。
5. [リポジトリタグ] ページで、[編集] を選択します。
6. [リポジトリタグを編集] ページで、削除するタグごとに [削除] を選択してから [保存] を選択します。

プライベートリポジトリからタグを削除するには (AWS CLI)

または AWS CLI API を使用して、1 つ以上のタグを削除できます。

- AWS CLI - タグ解除リソース
- API アクション - UntagResource

次の例は、 を使用してリポジトリからタグを削除する方法を示しています AWS CLI。

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

Amazon ECR のプライベートイメージ

Amazon ECR は、Docker イメージ、Open Container Initiative (OCI) イメージ、および OCI 互換アーティファクトをプライベートリポジトリに保存します。Docker CLI またはその他のクライアントを使用して、リポジトリからイメージをプッシュおよびプルできます。

トピック

Amazon ECR プライベートリポジトリへのイメージのプッシュ

Docker イメージ、マニフェストリスト、Open Container Initiative (OCI) イメージと互換アーティファクトをプライベートリポジトリにプッシュできます。

Amazon ECR は、イメージを他のリポジトリにレプリケートする方法も提供します。プライベートレジストリ設定でレプリケーション設定を指定することで、独自のレジストリのリージョン間および異なるアカウント間でレプリケートできます。詳細については、「Amazon ECR のプライベートレジストリ設定」を参照してください。

トピック

- Docker イメージを Amazon ECR プライベートリポジトリにプッシュする
- マルチアーキテクチャイメージを Amazon ECR プライベートリポジトリにプッシュする
- Helm チャートを Amazon ECR プライベートリポジトリにプッシュする

Amazon ECR プライベートリポジトリにイメージをプッシュするための IAM アクセス許可

ユーザーには、Amazon ECR プライベートリポジトリにイメージをプッシュするための IAM アクセス許可が必要です。最小特権を付与するベストプラクティスに従って、特定のリポジトリへのアクセスを許可できます。すべてのリポジトリへのアクセスを許可することもできます。

ユーザーは、認可トークンをリクエストして、イメージをプッシュする各 Amazon ECR レジストリに対して認証を受ける必要があります。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するための AWS 管理ポリシーがいくつか用意されています。詳細については、「AWS Amazon Elastic Container Registry の マネージドポリシー」を参照してください。

独自の IAM ポリシーを作成することもできます。次の IAM ポリシーは、特定のリポジトリにイメージをプッシュするために必要なアクセス許可を付与します。リポジトリは、完全な Amazon リソースネーム (ARN) として指定する必要があります。

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

```
}
```

次の IAM ポリシーは、イメージをすべてのリポジトリにプッシュするために必要なアクセス許可を付与します。

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

Docker イメージを Amazon ECR プライベートリポジトリにプッシュする

docker pushコマンドを使用してコンテナイメージを Amazon ECR リポジトリにプッシュできます。

Amazon ECR は、マルチアーキテクチャイメージに使用される Docker マニフェストリストの作成とプッシュもサポートしています。詳細については、「マルチアーキテクチャイメージを Amazon ECR プライベートリポジトリにプッシュする」を参照してください。

Docker イメージを Amazon ECR リポジトリにプッシュするには

イメージをプッシュする前に、Amazon ECR リポジトリが存在している必要があります。詳細については、「the section called “イメージを保存するリポジトリの作成”」を参照してください。

1. イメージのプッシュ先となる Amazon ECR レジストリに対して Docker クライアントを認証します。認証トークンは、使用するレジストリごとに取得する必要があり、トークンは 12 時間有効です。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

Amazon ECR レジストリに対して Docker を認証するには、aws ecr get-login-password コマンドを実行します。認証トークンを docker login コマンドに渡すとき、ユーザー名の AWS 値を使用し、認証先の Amazon ECR レジストリの URI を指定します。複数のレジストリに対して認証する場合は、レジストリごとにコマンドを繰り返す必要があります。

> **Important**
>
> エラーが発生した場合は、 AWS CLIの最新バージョンをインストールまたはアップグレードします。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interfaceのインストール」を参照してください。

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

2. プッシュ先となるリポジトリにイメージリポジトリが存在しない場合は、作成します。詳細については、「イメージを保存する Amazon ECR プライベートリポジトリの作成」を参照してください。
3. プッシュするローカルイメージを識別します。docker images コマンドを実行し、システム上のコンテナイメージを一覧表示します。

   ```
   docker images
   ```

   イメージは、結果のコマンド出力で *repository:tag* の値またはイメージ ID によって識別できます。
4. Amazon ECR レジストリ、リポジトリ、およびオプションのイメージタグ名を組み合わせたタグをイメージに付与します。レジストリ形式は *aws_account_id*.dkr.ecr.*us-west-2*.amazonaws.com です。リポジトリ名は、イメージ用に作成したリポジトリと一致する必要があります。イメージタグを省略した場合、タグは latest と見なされます。

   次の例は、ID が *e9ae3c220b23* のローカルイメージに *aws_account_id*.dkr.ecr.*us-west-2*.amazonaws.com/my-repository:tag というタグを付けます。

   ```
   docker tag e9ae3c220b23 aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-
   repository:tag
   ```
5. docker push コマンドを使用してイメージをプッシュします。

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

6. (オプション) 追加のタグをイメージに適用し、Step 4 とStep 5 を繰り返して、それらのタグを Amazon ECR にプッシュします。

マルチアーキテクチャイメージを Amazon ECR プライベートリポジトリにプッシュする

Docker マニフェストリストを作成してプッシュすることで、マルチアーキテクチャイメージを Amazon ECR リポジトリにプッシュできます。マニフェストリストは、1 つ以上のイメージ名を指定して作成されるイメージのリストです。ほとんどの場合、マニフェストリストは同じ機能を提供するが、異なるオペレーティングシステムまたはアーキテクチャ用のイメージから作成されます。マニフェストリストは必須ではありません。詳細については、「Docker マニフェスト」を参照してください。

マニフェストリストは、他の Amazon ECR イメージと同様に、Amazon ECS タスク定義または Amazon EKS ポッド仕様でプルまたは参照できます。

前提条件

- Docker CLI で、実験的な機能を有効にします。実験的な機能の詳細については、Docker ドキュメントの「実験的な機能」を参照してください。
- イメージをプッシュする前に、Amazon ECR リポジトリが存在している必要があります。詳細については、「the section called “イメージを保存するリポジトリの作成”」を参照してください。
- Docker マニフェストを作成する前に、イメージをリポジトリにプッシュする必要があります。イメージをプッシュする方法については、「Docker イメージを Amazon ECR プライベートリポジトリにプッシュする」を参照してください。

マルチアーキテクチャ Docker イメージを Amazon ECR リポジトリにプッシュするには

1. イメージのプッシュ先となる Amazon ECR レジストリに対して Docker クライアントを認証します。認証トークンは、使用するレジストリごとに取得する必要があり、トークンは 12 時間有効です。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

   Amazon ECR レジストリに対して Docker を認証するには、aws ecr get-login-password コマンドを実行します。認証トークンを docker login コマンドに渡すとき、ユーザー名の AWS 値を使

用し、認証先の Amazon ECR レジストリの URI を指定します。複数のレジストリに対して認証する場合は、レジストリごとにコマンドを繰り返す必要があります。

> **Important**
>
> エラーが発生した場合は、 AWS CLIの最新バージョンをインストールまたはアップグレードします。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interfaceのインストール」を参照してください。

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

2. リポジトリ内のイメージをリストし、イメージタグを確認します。

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

3. Docker マニフェストリストを作成します。`manifest create` コマンドは、参照されたイメージがリポジトリにすでに存在することを確認し、マニフェストをローカルに作成します。

```
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. (オプション) Docker マニフェストリストを検査します。これにより、マニフェストリストで参照される各イメージマニフェストのサイズとダイジェストを確認できます。

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

5. Docker マニフェストリストを Amazon ECR リポジトリにプッシュします。

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

Helm チャートを Amazon ECR プライベートリポジトリにプッシュする

Open Container Initiative (OCI) アーティファクトを Amazon ECR リポジトリにプッシュできます。この機能の例を確認するには、次のステップを使用して Helm チャートを Amazon ECR にプッシュします。

Amazon ECR でホストされる Helm チャートを Amazon EKS で使用する方法については、「」を参照してくださいAmazon EKS クラスターへの Helm チャートのインストール。

Helm チャートを Amazon ECR リポジトリにプッシュするには

1. Helm クライアントの最新バージョンをインストールします。これらのステップは、Helm バージョン `3.8.2` を使用して作成されました。詳細については、「Installing Helm」を参照してください。
2. 次の手順に従って、テスト Helm チャートを作成します。詳細については、Helm ドキュメントの「Getting Started」を参照してください。

   a. `helm-test-chart` という名前の Helm チャートを作成し、`templates` ディレクトリの内容をクリアします。

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

   b. `templates` フォルダ ConfigMap に を作成します。

   ```
   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. チャートをパッケージ化します。出力には、Helm チャートをプッシュするときに使用する、パッケージ化されたチャートのファイル名が含まれます。

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

出力

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

4. Helm チャートを格納するリポジトリを作成します。リポジトリの名前は、ステップ 2 で Helm チャートを作成する場合に使用する名前と一致する必要があります。詳しくは、「イメージを保存する Amazon ECR プライベートリポジトリの作成」を参照してください。

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

5. Helm チャートのプッシュ先となる Amazon ECR レジストリに対して Helm クライアントを認証します。認証トークンは、使用するレジストリごとに取得する必要があり、トークンは 12 時間有効です。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

```
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. helm push コマンドを使用して Helm チャートをプッシュします。出力には、Amazon ECR リポジトリ URI と SHA ダイジェストが含まれているはずです。

```
helm push helm-test-chart-0.1.0.tgz oci://aws_account_id.dkr.ecr.us-
west-2.amazonaws.com/
```

7. Helm チャートを説明します。

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

出力で、artifactMediaType パラメータが適切なアーティファクトタイプを示していることを確認します。

```
{
    "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. (オプション) 追加のステップについては、Helm configmap をインストールして Amazon EKS の使用を開始してください。詳細については、「Amazon EKS クラスターへの Helm チャートのインストール」を参照してください。

Amazon ECR プライベートリポジトリに保存されているイメージの署名

Amazon ECR は と統合 AWS Signer され、コンテナイメージに署名する方法を提供します。コンテナイメージと署名の両方をプライベートのリポジトリに保存することができます。

考慮事項

Amazon ECR のイメージ署名を使用するときは、以下の点を考慮する必要があります。

- リポジトリに保存されている署名は、リポジトリあたりの最大イメージ数のサービスクォータカウントされます。詳細については、「Amazon ECR のサービスクォータ」を参照してください。
- Amazon ECR ライフサイクルポリシーを使用する場合、ルールによって OCI イメージインデックスを期限切れにしたり削除したりするアクションを実行すると、Amazon ECR は、そのイメージインデックスが参照している署名を 24 時間以内にすべて削除します。

前提条件

開始するには、以下の前提条件を満たしておく必要があります。

- AWS CLIの最新バージョンがインストールされ、設定されている。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI の最新バージョンをインストールまたは更新します。」を参照してください。
- Notation CLI と Notation 用 AWS Signer プラグインをインストールします。詳細については、「AWS Signer デベロッパーガイド」の「コンテナイメージの署名の前提条件」を参照してください。
- 署名するコンテナイメージを Amazon ECR プライベートリポジトリに保存します。詳細については、「Amazon ECR プライベートリポジトリへのイメージのプッシュ」を参照してください。

Notary クライアントの認証の設定

Notation CLI を使用して署名を作成するには、事前に Amazon ECR への認証が可能になるようにクライアントを設定する必要があります。Notation クライアントをインストールしたのと同じホストに Docker がインストールされている場合、Notation では Docker クライアントで使用されるのと同じ認証方法が再利用されます。Docker `login` コマンドと `logout` コマンドを使うと、Notation `sign` コマンドと `verify` コマンドで同じ認証情報を使用できるようになり、Notation を個別に認証する必要がなくなります。Notation クライアントの認証設定の詳細については、Notary Project ドキュメントの「OCI 準拠のレジストリによる認証」を参照してください。

Docker や Docker 認証情報を使用するその他のツールを使用していない場合は、認証情報ストアとして Amazon ECR Docker 認証情報ヘルパーを使用することをお勧めします。Amazon ECR 認証情報ヘルパーのインストールと設定方法の詳細については、「Amazon ECR Docker 認証情報ヘルパー」を参照してください。

イメージの署名

次のステップを実行すると、コンテナイメージの署名に必要なリソースを作成し、この署名を Amazon ECR プライベートリポジトリに保存することができます。Notation ではダイジェストを使用してイメージに署名します。

イメージに署名するには

1. AWS Signer 署名プラットフォームを使用して`Notation-OCI-SHA384-ECDSA`署名プロファイルを作成します。オプションで、`--signature-validity-period` パラメータを使用して署名の有効期間を指定することができます。この値は `DAYS`、`MONTHS`、`YEARS` のいずれかで指定できます。値を指定しない場合、デフォルトの値 (135 か月) が使用されます。

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

> **Note**
>
> 署名プロファイル名に使用できるのは、英数字とアンダースコア (_) のみです。

2. Notation のクライアントをデフォルトのレジストリに対して認証します。次の例では、 を使用して AWS CLI 、Amazon ECR プライベートレジストリに対して Notation CLI を認証します。

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

3. Notation CLI を使用してイメージに署名し、リポジトリ名と SHA ダイジェストを使用してイメージを指定します。これにより署名が作成され、署名対象のイメージと同じ Amazon ECR プライベートリポジトリにプッシュされます。

   次の例では、SHA ダイジェスト `sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE` を使用して `curl` リポジトリ内のイメージに署名しています。

```
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"
```

次のステップ

コンテナイメージに署名すると、署名をローカルで検証できます。イメージの検証手順については、「 デベロッパーガイド」の「署名後にイメージをローカルで検証する」を参照してください。AWS Signer

Amazon ECR プライベートリポジトリから署名を削除する

Amazon ECR プライベートリポジトリから署名を削除できます。Notation CLI を使用して署名を作成してプッシュすると、Amazon ECR リポジトリにも OCI イメージインデックスが作成されま

す。Amazon ECR API は、OCI イメージインデックスから参照されているアーティファクトやイメージの削除には対応していないため、これらのアーティファクトをクリーンアップする方法は次の方法を使用します。

- (推奨) ORAS CLI を使用してアーティファクトを削除すると、ORAS がイメージインデックスの更新または削除を処理します。
- Amazon ECR API またはコンソールを使用すると、まず OCI イメージインデックスを削除し、その後で署名など参照先のアーティファクトを削除できます。

ORAS クライアントを使用して署名やその他参照タイプのアーティファクトを削除する場合、OCI イメージインデックスは ORAS が管理します。ORAS は、まずアーティファクトの参照をインデックスから削除し、次にマニフェストを削除します。`oras manifest delete` コマンドは、署名のアーティファクトのインデックスを参照しながら使用することができます。

ORAS CLI を使用して署名を削除するには

1. ORAS クライアントをインストールして設定します。

   ORAS クライアントのインストールと設定の詳細については、ORAS ドキュメントの「インストール」を参照してください。
2. ORAS CLI を使用して署名を削除するには、次のコマンドを実行します。

```
oras manifest
 delete 111122223333.dkr.ecr.region.amazonaws.com/
repository_name@sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE
```

Amazon ECR でのイメージの詳細の表示

イメージをリポジトリにプッシュすると、そのイメージに関する情報を表示できます。詳細には、以下の情報が含まれます。

- イメージ URI
- イメージフラグ
- アーティファクトメディアのタイプ
- イメージマニフェストのタイプ
- スキャンステータス

- イメージのサイズ (MB)
- イメージがいつリポジトリにプッシュされたか
- レプリケーションステータス

イメージの詳細を表示するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、イメージを含むリポジトリが含まれるリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. リポジトリページで、表示するリポジトリを選択します。
5. リポジトリ: ***repository_name*** ページで、詳細を表示するイメージを選択します。

Amazon ECR プライベートリポジトリからローカル環境にイメージをプルする

Amazon ECR で利用可能な Docker イメージを実行する場合、docker pull コマンドを使用してローカル環境にプルします。これは、デフォルトのレジストリまたは別の AWS アカウントに関連付けられたレジストリから実行できます。

Amazon ECS タスク定義で Amazon ECR イメージを使用する場合は、「Amazon ECS での Amazon ECR イメージの使用」を参照してください。

> **Important**
>
> Amazon ECR では、ユーザーがレジストリへの認証を行って、Amazon ECR リポジトリに対するイメージのプッシュまたはプルを行う前に、IAM ポリシーを介して `ecr:GetAuthorizationToken` API への呼び出しを行う許可が必要です。Amazon ECR には、さまざまなレベルでユーザーアクセスを制御するための複数の AWS 管理ポリシーが用意されています。Amazon ECR の AWS マネージドポリシーの詳細については、「」を参照してくださいAWS Amazon Elastic Container Registry の マネージドポリシー。

Amazon ECR リポジトリから Docker イメージをプルするには

1. イメージのプル元になる Amazon ECR レジストリに対して Docker クライアントを認証します。認証トークンは、使用するレジストリごとに取得する必要があり、トークンは 12 時間有効

です。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

2. (オプション) プルするイメージを識別します。
   - レジストリ内のリポジトリは、aws ecr describe-repositories コマンドを使用してリストできます。

     ```
     aws ecr describe-repositories
     ```

     上のサンプルレジストリには、amazonlinux というリポジトリがあります。
   - リポジトリ内のイメージは、aws ecr describe-images コマンドで記述することができます。

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

     上記のリポジトリ例には、latest および 2016.09 というタグが付けられたイメージが、イメージダイジェスト sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807 とともにあります。
3. docker pull コマンドを使用してイメージをプルします。イメージ名の形式は、タグを使用してプルする場合は *registry*/*repository*[:*tag*]、ダイジェストを使用してプルする場合は *registry*/*repository*[@*digest*] とします。

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

   > **Important**
   >
   > *repository-url* not found: does not exist or no pull access エラーが表示された場合は、Amazon ECR で Docker クライアントを認証する必要があります。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

Amazon Linux コンテナイメージのプル

Amazon Linux コンテナイメージは、Amazon Linux AMI に含まれているのと同じソフトウェアコンポーネントから構築されます。Amazon Linux コンテナイメージは、Docker ワークロードのベース

イメージとして任意の環境で使用できます。Amazon EC2 のアプリケーションに Amazon Linux AMI を使用する場合は、Amazon Linux コンテナイメージを使用してアプリケーションをコンテナ化できます。

ローカル開発環境で Amazon Linux コンテナイメージを使用し、Amazon ECS AWS を使用してアプリケーションを にプッシュできます。詳細については、「Amazon ECS での Amazon ECR イメージの使用」を参照してください。

Amazon Linux コンテナイメージは、Amazon ECR Public および Docker Hub で入手できます。Amazon Linux コンテナイメージのサポートについては、AWS デベロッパーフォーラム を参照してください。

Amazon ECR Public から Amazon Linux コンテナイメージをプルするには

1. Amazon Linux Public レジストリで Docker クライアントを認証します。認証トークンは 12 時間有効です。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

   > **Note**
   >
   > ecr-public コマンドは、バージョン 1.18.1.187 以降の AWS CLI で使用できます。ただし、 AWS CLIの最新バージョンを使用することをお勧めします。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interfaceのインストール」を参照してください。

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

   出力は次のとおりです。

   ```
   Login succeeded
   ```

2. docker pull コマンドを使用して Amazon Linux コンテナイメージをプルします。Amazon ECR Public Gallery で Amazon Linux コンテナイメージを表示するには、「Amazon ECR Public Gallery - amazonlinux」を参照してください。

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

3. (オプション) コンテナをローカルに実行します。

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

Docker Hub から Amazon Linux コンテナイメージをプルするには

1. docker pull コマンドを使用して Amazon Linux コンテナイメージをプルします。

```
docker pull amazonlinux
```

2. (オプション) コンテナをローカルに実行します。

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

Amazon ECR でのイメージの削除

使用が終わったイメージはリポジトリから削除できます。リポジトリでの作業が終了したら、リポジトリ全体とその中のすべてのイメージを削除できます。詳細については、「Amazon ECR でのプライベートリポジトリの削除」を参照してください。

イメージを手動で削除する代わりに、リポジトリのライフサイクルポリシーを作成すると、リポジトリ内のイメージのライフサイクル管理をより詳細に制御できます。ライフサイクルポリシーを使用すると、このプロセスが自動化されます。詳細については、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。

イメージを削除するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、削除するイメージを含むリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. リポジトリページで、削除するイメージを含むリポジトリを選択します。
5. リポジトリ: ***repository_name*** ページで、削除するイメージの左側にあるボックスを選択し、[削除] を選択します。
6. [Delete image(s)] ダイアログボックスで、選択したイメージを削除することを確認し、[Delete] を選択します。

イメージを削除するには (AWS CLI)

1. リポジトリ内のイメージを一覧表示します。タグ付けされたイメージには、イメージダイジェストおよび関連するタグのリストの両方が含まれます。タグ付けされていないイメージには、イメージダイジェストのみが含まれます。

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

2. (オプション) 削除するイメージに関連付けられたタグを指定して、イメージの不要なタグを削除します。イメージの最後のタグを削除すると、イメージが削除されます。

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

3. イメージダイジェストを指定して、タグ付けされたイメージ、またはタグ付けされていないイメージを削除します。ダイジェストを参照してイメージを削除する場合、イメージとそのすべてのタグは削除されます。

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

   複数のイメージを削除するには、リクエストで複数のイメージタグまたはイメージダイジェストを指定します。

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

Amazon ECR でのイメージのタグ付け

Docker Image Manifest V2 Schema 2 のイメージでは、put-image コマンドの `--image-tag` オプションを使用して、既存のイメージにもう一度タグを付けることができます。Docker でイメージをプルまたはプッシュしなくても、もう一度タグを付けることができます。大きなイメージの場合、こ

のプロセスにより、イメージにもう一度タグを付けるために必要なネットワーク帯域幅と時間がかなり節約されます。

イメージにもう一度タグを付けるには (AWS CLI)

を使用してイメージにタグを付け直すには AWS CLI

1. batch-get-image コマンドを使用して、イメージを再タグ付けしてファイルに書き込むためのイメージマニフェストを取得します。この例では、リポジトリ *amazonlinux* 内の *latest* タグ付きイメージのマニフェストが *MANIFEST* という名前の環境変数に書き込まれます。

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

2. put-image コマンドの `--image-tag` オプションを使用して、新しいタグでイメージマニフェストを Amazon ECR に配置します。この例では、イメージには *2017.03* というタグが付きます。

> **Note**
>
> お使いのバージョンの で `--image-tag`オプションを使用できない場合は AWS CLI、最新バージョンにアップグレードしてください。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interfaceのインストール」を参照してください。

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

3. 新しいイメージタグがイメージにアタッチされていることを確認します。次の出力では、イメージに `latest` と `2017.03` のタグが付けられています。

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

出力は次のとおりです。

```
{
    "imageDetails": [
        {
```

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

イメージにもう一度タグを付けるには (AWS Tools for Windows PowerShell)

を使用してイメージにタグを付け直すには AWS Tools for Windows PowerShell

1. Get-ECRImageBatch コマンドレットを使用して、もう一度タグを付けるイメージの説明を取得し、環境変数にそれを書き込みます。この例では、リポジトリ *amazonlinux* 内の *latest* タグ付きイメージが環境変数 *$Image* に書き込まれます。

   > **Note**
   >
   > システムで Get-ECRImageBatch cmdlet が使用できない場合は、AWS Tools for Windows PowerShell ユーザーガイドの「AWS Tools for Windows PowerShellのインストール」を参照してください。

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

2. *$Manifest* 環境変数にイメージのマニフェストを書き込みます。

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

3. -ImageTag コマンドレットの Write-ECRImage オプションを使用して、イメージマニフェストを新しいタグで Amazon ECR に配置します。この例では、イメージには *2017.09* というタグが付きます。

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

4. 新しいイメージタグがイメージにアタッチされていることを確認します。次の出力では、イメージに latest と 2017.09 のタグが付けられています。

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

出力は次のとおりです。

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

Amazon ECR でイメージタグが上書きされないようにする

リポジトリでタグのイミュータビリティを有効にすると、イメージタグが上書きされるのを防ぐことができます。タグのイミュータビリティがオンになると、リポジトリに既に存在するタグを持つイメージをプッシュすると、ImageTagAlreadyExistsException エラーが返されます。タグのイミュータビリティはすべてのタグに影響します。一部のタグをイミュータブルにすることはできませんが、他のタグをイミュータブルにすることはできません。

AWS Management Console および AWS CLI ツールを使用して、新しいリポジトリまたは既存のリポジトリにイメージタグのミュータビリティを設定できます。コンソールのステップを使用してリポジトリを作成するには、「」を参照してくださいイメージを保存する Amazon ECR プライベートリポジトリの作成。

イメージタグのミュータビリティの設定 (AWS Management Console)

イメージタグのミュータビリティを設定するには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、編集するリポジトリを含むリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. [Repositories] (リポジトリ) ページで、[Private] (プライベート) タブを選択し、編集するレポジトリを選択して [Edit] (編集) を選びます。

5. [Tag immutability] (タグの不変性) で、このリポジトリのタグの変更可能性の設定を選択します。タグがイミュータブルに設定されたリポジトリでは、イメージタグが上書きされるのを防ぐことができます。詳細については、「Amazon ECR でイメージタグが上書きされないようにする」を参照してください。
6. [Image scan settings] (イメージスキャン設定) では、基本的なスキャンに対してはリポジトリレベルでのスキャン設定を指定できますが、プライベートレジストリレベルでスキャン設定を指定することがベストプラクティスとなります。プライベートレジストリでスキャン設定を指定することで、拡張スキャンまたはベーシックスキャンのいずれかを有効にすることができ、スキャンするリポジトリを指定するフィルターも定義できます。詳細については、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。
7. [Encryption settings] (暗号化設定) は表示専用フィールドであり、リポジトリの作成後にリポジトリの暗号化設定を変更することはできません。
8. [保存] を選択してリポジトリ設定を更新します。

イメージタグのミュータビリティの設定 (AWS CLI)

タグが変更不可に設定されたリポジトリを作成するには

次のいずれかのコマンドを使用して、タグが変更不可似せていされた新しいイメージリポジトリを作成します。

- create-repository (AWS CLI)

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

- New-ECRRepository (AWS Tools for Windows PowerShell)

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

リポジトリのイメージタグのミュータビリティ設定を更新するには

次のいずれかのコマンドを使用して、既存のリポジトリのイメージタグの変更可能性を更新します。

- put-image-tag-mutability (AWS CLI)

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

- Write-ECR ImageTagミュータビリティ (AWS Tools for Windows PowerShell)

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

Amazon ECR でのコンテナイメージマニフェスト形式のサポート

Amazon ECR は次のコンテナイメージマニフェスト形式をサポートします。

- Docker Image Manifest V2 Schema 1 (Docker バージョン 1.9 以前で使用)
- Docker Image Manifest V2 Schema 2 (Docker バージョン 1.10 以降で使用)
- Open Container Initiative (OCI) 仕様 (v1.0 以降)

Docker Image Manifest V2 Schema 2 のサポートでは、次の機能が提供されます。

- 単一のイメージに複数のタグを使用する機能。
- Windows コンテナイメージを保存するためのサポート。

Amazon ECR イメージマニフェストの変換

Amazon ECR に対してイメージをプッシュまたはプルする場合、コンテナエンジンクライアント (Docker など) はレジストリと通信し、クライアントとレジストリでイメージに対して使用することが理解されているマニフェスト形式について合意します。

Docker バージョン 1.9 以前で Amazon ECR にイメージをプッシュする場合、イメージマニフェストは Docker Image Manifest V2 Schema 1 形式で保存されます。Docker バージョン 1.10 以降で Amazon ECR にイメージをプッシュする場合、イメージマニフェストは Docker Image Manifest V2 Schema 2 形式で保存されます。

Amazon ECR からタグでイメージをプルすると、Amazon ECR はリポジトリに格納されているイメージマニフェスト形式を返します。その形式が返されるのは、その形式がクライアントによって理解される場合のみです。保存されているイメージマニフェスト形式がクライアントによって理

解されない場合、Amazon ECR はイメージマニフェストを理解される形式に変換します。たとえば、Docker 1.9 クライアントが Docker Image Manifest V2 Schema 2 で保存されているイメージマニフェストをリクエストすると、Amazon ECR は Docker Image Manifest V2 Schema 1 形式でマニフェストを返します。次の表は、タグ別にイメージをプルしたときに、Amazon ECR でサポートされる利用可能な変換を示します。

クライアントによってリクエストされたスキーマ	V2、スキーマ 1 形式で ECR にプッシュされる	V2、スキーマ 2 形式で ECR にプッシュされる	OCI 形式で ECR にプッシュされる
V2、スキーマ 1	変換は必要ありません	V2、スキーマ 1 に変換される	V2、スキーマ 1 に変換される
V2、スキーマ 2	利用可能な変換はなく、クライアントは V2、スキーマ 1 にフォールバックする	変換は必要ありません	V2、スキーマ 2 に変換される
OCI	変換は利用できません	OCI に変換される	変換は必要ありません

Important

ダイジェスト別にイメージをプルした場合、利用可能な翻訳はありません。クライアントは、Amazon ECR に保存されているイメージマニフェストの形式を理解する必要があります。Docker 1.9 以前のクライアントで、Docker Image Manifest V2 Schema 2 イメージをダイジェストを使用してリクエストする場合、イメージのプルは失敗します。詳細については、Docker ドキュメントの「Registry compatibility」を参照してください。
この例では、タグで同じイメージをリクエストした場合、Amazon ECR はクライアントが理解できる形式にイメージマニフェストを変換します。イメージのプルが成功します。

Amazon ECS での Amazon ECR イメージの使用

Amazon ECR プライベートリポジトリを使用して、Amazon ECS タスクがプルする可能性のあるコンテナイメージとアーティファクトをホストできます。これが機能するためには、Amazon ECS (または Fargate) コンテナエージェ

ントに、ecr:BatchGetImage、ecr:GetDownloadUrlForLayer、および ecr:GetAuthorizationToken API を作成する許可が必要です。

必要な IAM 許可

次の表は、タスクが Amazon ECR プライベートリポジトリからプルするために必要な許可を提供する、起動タイプごとに使用する IAM ロールを示しています。Amazon ECS には、必要なアクセス許可を含むマネージド IAM ポリシーが用意されています。

起動タイプ	IAM ロール	AWS マネージド IAM ポリシー
Amazon EC2 インスタンスの Amazon ECS	Amazon ECS クラスターに登録されている Amazon EC2 インスタンスに関連付けられているコンテナインスタンスの IAM ロールを使用します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「コンテナインスタンスの IAM ロール」を参照してください。	AmazonEC2ContainerServiceforEC2Role 詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「AmazonEC2ContainerServiceforEC2Role」を参照してください
Fargate の Amazon ECS	Amazon ECS タスク定義で参照するタスク実行 IAM ロールを使用します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「タスク実行 IAM ロール」を参照してください。	AmazonECSTaskExecutionRolePolicy 詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「AmazonECSTaskExecutionRolePolicy」を参照してください。
外部インスタンスの Amazon ECS	Amazon ECS クラスターに登録されているオンプレミスサーバーまたは仮想マシン (VM) に関連付けられているコンテナインスタンスの IAM	AmazonEC2ContainerServiceforEC2Role 詳細については、「Amazon Elastic Container Service

起動タイプ	IAM ロール	AWS マネージド IAM ポリシー
	ロールを使用します。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Container instance Amazon ECS role」(コンテナインスタンスの Amazon ECS ロール) を参照してください。	デベロッパーガイド」の「AmazonEC2ContainerServiceforEC2Role」を参照してください。

Important

AWS マネージド IAM ポリシーには、使用に不要な追加のアクセス許可が含まれています。この場合、これらは Amazon ECR プライベートリポジトリからプルするために最低限必要なアクセス許可です。

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

Amazon ECS タスク定義での Amazon ECR イメージの指定

Amazon ECS タスク定義を作成するときに、Amazon ECR プライベートリポジトリでホストされているコンテナイメージを指定できます。タスク定義では、Amazon ECR イメージ

用に完全な registry/repository:tag の名前を使用するようにしてください。例えば *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-repository*:*latest* です。

次のタスク定義スニペットは、Amazon ECS タスク定義の Amazon ECR でホストされるコンテナイメージを指定するために使用する構文を示しています。

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

Amazon EKS での Amazon ECR イメージの使用

Amazon ECR イメージは Amazon EKS で使用できます。

Amazon ECR からイメージを参照する場合は、イメージに registry/repository:tag の形式の完全な名前を使用する必要があります 例えば *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-repository*:*latest* です。

必要な IAM 許可

マネージドノード、セルフマネージドノード、または で Amazon EKS ワークロードをホストしている場合は AWS Fargate、以下を確認してください。

- マネージドノードまたはセルフマネージドノードでホストされている Amazon EKS ワークロード: Amazon EKS ワーカーノード IAM ロール (NodeInstanceRole) が必要です。Amazon EKS ワーカーノード IAM ロールには、Amazon ECR の次の IAM ポリシー許可が含まれている必要があります。

  ```
  {
  ```

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

> **Note**
>
> クラスターとワーカーノードグループの作成に `eksctl`または「Amazon EKS の開始方法」の AWS CloudFormation テンプレートを使用した場合、これらの IAM アクセス許可はデフォルトでワーカーノードの IAM ロールに適用されます。

- でホストされる Amazon EKS ワークロード AWS Fargate: Fargate ポッド実行ロールを使用します。これにより、ポッドにプライベート Amazon ECR リポジトリからイメージをプルするアクセス許可が付与されます。詳細については、「Fargate ポッド実行ロールの作成」を参照してください。

Amazon EKS クラスターへの Helm チャートのインストール

Amazon ECR でホストされている Helm チャートは、Amazon EKS クラスターにインストールできます。

前提条件

- Helm クライアントの最新バージョンをインストールします。これらのステップは、Helm バージョン `3.9.0` を使用して作成されました。詳細については、「Installing Helm」を参照してください。
- 少なくとも、 AWS CLI のバージョン `1.23.9` または `2.6.3` がコンピュータにインストールされています。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。

- Helm チャートを Amazon ECR リポジトリにプッシュします。詳細については、「Helm チャートを Amazon ECR プライベートリポジトリにプッシュする」を参照してください。
- Amazon EKS で使用する kubectl を設定します。詳細については、Amazon EKS ユーザーガイドの「Amazon EKS の kubeconfig を作成する」を参照してください。次のコマンドがクラスターに対して正常に実行された場合は、正しく設定されています。

```
kubectl get svc
```

Amazon EKS クラスターに Helm チャートをインストールするには

1. Helm チャートがホストされている Amazon ECR レジストリで Helm クライアントを認証します。認証トークンは、使用するレジストリごとに取得する必要があり、トークンは 12 時間有効です。詳細については、「Amazon ECR でのプライベートレジストリ認証」を参照してください。

```
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. グラフをインストールします。をリポジトリ*helm-test-chart*に、*0.1.0* を Helm チャートのタグに置き換えます。

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

出力は次のようになります。

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

3. チャートのインストールを検証します。

```
helm list -n default
```

出力例:

```
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. (オプション) インストールされている Helm チャート ConfigMap を確認します。

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

5. 終了したら、クラスターからチャートのリリースを削除できます。

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

Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする

基本的なスキャン機能が改善された Amazon ECR のプレビューリリースであり、変更される可能性があります。このパブリックプレビューでは、 のみを使用して AWS Management Console 、基本スキャンバージョンの改善をオプトインできます。

Amazon ECR イメージスキャンは、コンテナイメージ内のソフトウェアの脆弱性を特定するのに役立ちます。次のスキャンタイプが提供されています。

> **Important**
>
> 拡張スキャン、基本スキャン、および基本スキャンの改良バージョンを切り替えると、以前に確立されたスキャンが使用できなくなります。スキャンを再度設定する必要があります。ただし、以前のスキャンバージョンに戻すと、確立されたスキャンが使用可能になります。

- 拡張スキャン— Amazon ECR は Amazon Inspector と統合され、リポジトリの自動継続的なスキャンを提供します。コンテナイメージは、オペレーティングシステムとプログラミング言語パッケージの両方の脆弱性についてスキャンされます。新しい脆弱性が表示されると、スキャン結果が更新され、Amazon Inspector は にイベントを発行 EventBridge して通知します。拡張スキャンには、次の機能があります。
  - OS とプログラミング言語は脆弱性をパッケージ化します。
  - 2 つのスキャン頻度: プッシュスキャンと連続スキャン。
- 基本スキャン — Amazon ECR は、共通脆弱性識別子 (CVEs) データベースを使用する基本スキャンの 2 つのバージョンを提供します。オープンソースの Clair プロジェクトを使用する現在の GA バージョンと、 AWS ネイティブテクノロジーを使用する新しく改善された基本スキャン (プレビュー) バージョンです。ベーシックスキャンでは、プッシュ時にスキャンするようにリポジトリを設定します。手動スキャンを実行すると Amazon ECR によってスキャン結果のリストが提供されます。ベーシックスキャンには、次の機能があります。
  - OS スキャン。
  - 2 つのスキャン頻度: 手動とプッシュ時のスキャン。

> **Important**
>
> 新しいバージョンのベーシックスキャンでは、 `DescribeImages` API `imageScanStatus`の `imageScanFindingsSummary`および はサポートされていません。これらを表示するには、 `DescribeImageScanFindings` API を使用します。

Amazon ECR でスキャンするリポジトリを選択するフィルター

プライベートレジストリのイメージスキャンを設定するときに、フィルターを使用してスキャンするリポジトリを選択できます。

基本スキャンが使用されている場合は、プッシュフィルターでスキャンを指定して、新しいイメージがプッシュされたときにイメージスキャンを実行するように設定されるリポジトリを指定できます。プッシュフィルターの基本的スキャンのスキャンと一致しないリポジトリは、マニュアルスキャン頻度に設定されます。これは、スキャンを実行する場合、手動でスキャンをトリガーする必要があります。

強化スキャンが使用されている場合は、プッシュ時のスキャンと連続スキャン用に別々のフィルターを指定できます。拡張スキャンフィルターに一致しないリポジトリでは、スキャンが無効になります。拡張スキャンを使用し、複数のフィルターが同じリポジトリに一致するプッシュ時スキャンと連続スキャンに別々のフィルターを指定すると、Amazon ECR はそのリポジトリのスキャンオンプッシュフィルターに対して連続スキャンフィルターを適用します。

ワイルドカードをフィルタリングする

フィルターを指定すると、ワイルドカードを含まないフィルターは、そのフィルターを含むすべてのリポジトリ名と一致します。ワイルドカード (*) を含むフィルターは、リポジトリ名のゼロ文字以上の文字をワイルドカードで置き換える任意のリポジトリ名と一致します。

次の表に、リポジトリ名を横軸に表し、フィルターの例を縦軸に指定する例を示します。

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod	はい	はい	はい	はい	はい
*prod	はい	はい	いいえ	いいえ	いいえ

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod*	はい	いいえ	はい	いいえ	はい
prod	はい	はい	はい	はい	はい
prod*repo	いいえ	いいえ	はい	いいえ	はい

Amazon ECR で OS とプログラミング言語パッケージの脆弱性についてイメージをスキャンする

Amazon ECR 拡張スキャンは、コンテナイメージの脆弱性スキャンを提供する Amazon Inspector との統合です。コンテナイメージは、オペレーティングシステムとプログラミング言語パッケージの両方の脆弱性についてスキャンされます。スキャンの結果は、Amazon ECR と Amazon Inspector の両方で直接表示できます。Amazon Inspector の詳細については、Amazon Inspector ユーザーガイドの Scanning container images with Amazon Inspector を参照してください。

拡張スキャンでは、自動連続スキャン用に構成するリポジトリと、プッシュ時にスキャンするように構成するリポジトリを選択できます。これは、スキャンフィルターを設定することによって行われます。

拡張スキャンの考慮事項

Amazon ECR 拡張スキャンを有効にする前に、次の点を考慮してください。

- この機能を使用するために Amazon ECR に追加料金はかかりませんが、イメージをスキャンするために Amazon Inspector の料金がかかります。詳細については、「Amazon Inspector の料金」を参照してください。
- 次のリージョンでは、拡張スキャンはサポートされていません。
  - 中東 (アラブ首長国連邦) (`me-central-1`)
  - アジアパシフィック (ハイデラバード) (`ap-south-2`)
  - イスラエル (テルアビブ) (`il-central-1`)
  - アジアパシフィック (メルボルン) (`ap-southeast-4`)
  - 欧州 (スペイン) (`eu-south-2`)

- Amazon Inspector では特定のオペレーティングシステムのスキャンがサポートされます。完全なリストについては、Amazon Inspector ユーザーガイドの「サポートされているオペレーティングシステム - Amazon ECR スキャン」を参照してください。
- Amazon Inspector は、サービスにリンクされた IAM ロールを使用します。このロールは、リポジトリに対する拡張スキャンを行うのに必要なアクセス許可を提供します。プライベートレジストリで拡張スキャンがオンになっている場合、サービスにリンクされた IAM ロールは Amazon Inspector によって自動的に作成されます。詳細については、「Amazon Inspector ユーザーガイド」の「Amazon Inspector でのサービスにリンクされたロールの使用」を参照してください。
- プライベートレジストリの拡張スキャンを最初に有効にすると、Amazon Inspector は、イメージプッシュのタイムスタンプに基づいて、過去 30 日間に Amazon ECR にプッシュされたイメージ、または過去 90 日間にプルされたイメージのみを認識します。古い画像は `SCAN_ELIGIBILITY_EXPIRED` スキャンステータスになります。これらの画像を Amazon Inspector でスキャンしたい場合は、リポジトリに再度プッシュする必要があります。
- 拡張スキャンをオンにした後に Amazon ECR にプッシュされたすべての画像は、設定された期間、継続してスキャンされます。デフォルトでは、有効期間は [一生] です。この設定は、Amazon Inspector コンソールを使用して設定できます。詳細については、「Amazon Inspector でのイメージの拡張スキャン期間の変更」を参照してください。
- Amazon ECR プライベートレジストリで拡張スキャンがオンになっている場合、スキャンフィルターに一致するリポジトリは、拡張スキャンのみを使用してスキャンされます。フィルターと一致しないリポジトリのスキャン頻度は `Off` であり、スキャンされません。拡張スキャンを使用した手動スキャンはサポートされていません。詳細については、「Amazon ECR でスキャンするリポジトリを選択するフィルター」を参照してください。
- 複数のフィルターが同じリポジトリに一致するプッシュ時スキャンと連続スキャンに別々のフィルターを指定すると、Amazon ECR はそのリポジトリのスキャンオンプッシュフィルターに対して連続スキャンフィルターを適用します。
- 拡張スキャンがオンの場合、Amazon ECR はリポジトリのスキャン頻度が変更され EventBridge ると にイベントを送信します。Amazon Inspector は、最初のスキャンが完了した EventBridge とき、およびイメージスキャンの結果が作成、更新、または閉じられたときに、 にイベントを発行します。

Amazon ECR での拡張スキャンに必要な IAM アクセス許可

Amazon ECR 拡張スキャンには、Amazon Inspector サービスにリンクされた IAM ロールが必要です。また、拡張スキャンを有効化および使用する IAM プリンシパルには、スキャンで必要な Amazon Inspector API を呼び出すアクセス許可が必要です。プライベートレジストリで拡張スキャ

ンがオンになっている場合、Amazon Inspector サービスにリンクされた IAM ロールは Amazon Inspector によって自動的に作成されます。詳細については、Amazon Inspector ユーザーガイドの Amazon Inspector でのサービスにリンクされたロールの使用 を参照してください。

次の IAM ポリシーは、拡張スキャンの有効化と使用に必要なアクセス許可を付与します。これには、Amazon Inspector がサービスにリンクされた IAM ロールを作成するために必要なアクセス許可、および拡張スキャンをオンまたはオフにして、およびスキャン結果の取得に必要な Amazon Inspector API のアクセス許可が含まれます。

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

Amazon ECR でのイメージの拡張スキャンの設定

プライベートレジストリのリージョンごとに拡張スキャンを設定します。

拡張スキャンを設定するための適切な IAM アクセス許可があることを確認します。詳細については、「Amazon ECR での拡張スキャンに必要な IAM アクセス許可」を参照してください。

AWS Management Console

プライベートレジストリの拡張スキャンを有効にするには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、スキャン設定を設定するリージョンを選択します。
3. ナビゲーションペインで、プライベートレジストリ 、設定 、スキャン を選択します。
4. [Scanning configuration] (スキャン設定) ページの [Scan type] (スキャンタイプ) で、[Enhanced scanning] (拡張スキャン) を選択します。

   デフォルトでは、拡張スキャンを選択すると、すべてのリポジトリが継続的にスキャンされます。
5. 連続スキャンする特定のリポジトリを選択するには、すべてのリポジトリの連続スキャンボックスをクリアしてから、フィルターを定義します。

   > **Important**
   >
   > ワイルドカードを含まないフィルターは、そのフィルターを含むすべてのリポジトリ名と一致します。ワイルドカード (*) を含むフィルターは、リポジトリ名のゼロ文字以上の文字がワイルドカードで置き換えられるリポジトリ名と一致します。フィルターの動作の例については、「」を参照してくださいthe section called “ワイルドカードをフィルタリングする”。

   a. リポジトリ名に基づいてフィルターを入力し、フィルターの追加 を選択します。
   b. イメージがプッシュされたときにスキャンするリポジトリを決定します。

      - プッシュ時にすべてのリポジトリをスキャンするには、すべてのリポジトリをプッシュ時にスキャン を選択します。
      - プッシュ時にスキャンする特定のリポジトリを選択するには、リポジトリ名に基づいてフィルターを入力し、フィルターの追加 を選択します。
6. [保存] を選択します。
7. 拡張スキャンをオンにするリージョンごとに、これらの手順を繰り返します。

AWS CLI

次の AWS CLI コマンドを使用して、 を使用してプライベートレジストリの拡張スキャンを有効にします AWS CLI。スキャンフィルターは、`rules` オブジェクトを使用して指定します。

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

  次の例では、プライベートレジストリの拡張スキャンをオンにします。デフォルトでは、`rules` が指定されていない場合、Amazon ECR はすべてのリポジトリのスキャン設定を継続スキャンに設定します。

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

  次の例では、プライベートレジストリの拡張スキャンオンにして、スキャンフィルターを指定します。この例のスキャンフィルターは、名前に `prod` が含まれるすべてのリポジトリの継続スキャンをオンにします。

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

  次の例では、プライベートレジストリの拡張スキャンをオンにして、複数のスキャンフィルターを指定します。この例のスキャンフィルターは、名前に `prod` が含まれるすべてのリポジトリの継続スキャンをオンにして、他のすべてのリポジトリに対してはプッシュ時のみスキャンをオンにします。

```
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
```

Amazon Inspector でのイメージの拡張スキャン期間の変更

Amazon Inspector が Amazon ECR プライベートリポジトリ内のイメージを継続的にスキャンする日数を変更できます。デフォルトでは、Amazon ECR プライベートレジストリで拡張スキャンがオンになっている場合、Amazon Inspector サービスは、イメージが削除されるか、拡張スキャンが無効になるまで、リポジトリを継続的に監視します。Amazon Inspector がイメージをスキャンする期間は、Amazon Inspector の設定を使用して変更できます。利用可能なスキャン期間は、[Lifetime (デフォルト)] (有効期間)、[180 day] (180 日間)、および [30 day] (30 日間) です。リポジトリのスキャン期間が経過すると、スキャンの脆弱性を一覧表示するときに `SCAN_ELIGIBILITY_EXPIRED` のスキャンステータスが表示されます。詳細については、Amazon Inspector ユーザーガイドの「Changing the Amazon ECR automated re-scan duration」(Amazon ECR の自動化された再スキャン期間の変更) を参照してください。

拡張スキャン時間の設定を変更するには

1. https://console.aws.amazon.com/inspector/v2/home で Amazon Inspector コンソールを開きます。
2. 左側のナビゲーションで [設定] を展開して [全般] を選択します。
3. [設定] ページの [ECR re-scan duration] (ECR 再スキャン期間) [保存] を選択します。

EventBridge Amazon ECR で拡張スキャンのために送信されるイベント

拡張スキャンがオンの場合、Amazon ECR はリポジトリのスキャン頻度が変更され EventBridge ると にイベントを送信します。Amazon Inspector は、最初のスキャンが完了した EventBridge とき、およびイメージスキャンの結果が作成、更新、または閉じられたときに、 にイベントを送信します。

リポジトリのスキャン頻度の変更に関するイベント

レジストリで拡張スキャンがオンになっている場合、拡張スキャンがオンになっているリソースで変更すると、Amazon ECR から次のイベントが送信されます。これには、新しいリポジトリの作成、リポジトリのスキャン頻度の変更、または拡張スキャンがオンになっているリポジトリでのイメージの作成または削除が含まれます。詳細については、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。

```
{
 "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"
 }
}
```

初期イメージスキャンのイベント (拡張スキャン)

レジストリで拡張スキャンがオンになっている場合、初期イメージスキャンが完了すると、Amazon Inspector によって以下のイベントが送信されます。`finding-severity-counts` パラメータは、重要度レベルが存在する場合にのみ、その値を返します。たとえば、イメージに `CRITICAL` レベルの結果が含まれていない場合、重要度のカウントは返されません。詳細については、「Amazon ECR で OS とプログラミング言語パッケージの脆弱性についてイメージをスキャンする」を参照してください。

イベントパターン:

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

出力例:

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

イメージスキャン結果の更新のイベント (拡張スキャン)

レジストリで拡張スキャンがオンになっている場合、イメージスキャン結果が作成、更新、または閉じられると、Amazon Inspector によって以下のイベントが送信されます。詳細については、「Amazon ECR で OS とプログラミング言語パッケージの脆弱性についてイメージをスキャンする」を参照してください。

イベントパターン:

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

出力例:

```
{
    "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"
    }
}
```


Amazon ECR での拡張スキャンの検出結果の取得

最後に完了した拡張イメージスキャンのスキャン結果を取得し、Amazon Inspector で結果を開いて詳細を確認できます。検出されたソフトウェアの脆弱性は、共通脆弱性識別子 (CVEs) データベースに基づいて重要度別に一覧表示されます。

イメージをスキャンする際の一般的な問題のトラブルシューティングの詳細については、「Amazon ECR でのイメージスキャンのトラブルシューティング」を参照してください。

AWS Management Console

AWS Management Consoleを使用してイメージスキャンの結果を取得する手順は、次のとおりです。

イメージスキャンの検出結果を取得するには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、リポジトリがあるリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. リポジトリページで、スキャン結果を取得するイメージを含むリポジトリを選択します。
5. [イメージ] ページの [Vulnerabilities] (脆弱性) 列で、スキャン結果を取得するイメージの [See findings] (結果の表示) を選択します。
6. Amazon Inspector コンソールで詳細を表示するには、名前列で脆弱性名を選択します。

AWS CLI

次の AWS CLI コマンドを使用して、 を使用してイメージスキャンの結果を取得します AWS CLI。`imageTag` または `imageDigest` を使用してイメージを指定できます。どちらのイメージも list-images CLI コマンドを使用して取得できます。

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

  次の例では、イメージタグを使用しています。

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

  次の例では、イメージダイジェストを使用しています。

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

Amazon ECR で OS の脆弱性がないかイメージをスキャンする

基本的なスキャン機能が改善された Amazon ECR のプレビューリリースであり、変更される可能性があります。このパブリックプレビューでは、 のみを使用して AWS Management Console 、基本スキャンバージョンの改善をオプトインできます。

Amazon ECR には、共通脆弱性識別子 (CVEs) データベースを使用する 2 つのバージョンのベーシックスキャンが用意されています。

- オープンソースの Clair プロジェクトを使用する現在の GA バージョン。Clair の詳細については、「」の「 Clair」を参照してください GitHub。
- AWS ネイティブテクノロジーを使用するベーシックスキャン (プレビュー) の新しく改善されたバージョン。

Amazon ECR は、利用可能な場合、アップストリームディストリビューションソースからの CVE の重要度を使用します。それ以外の場合は、共通脆弱性評価システム (CVSS) スコアが使用されます。CVSS スコアは、NVD 脆弱性の重大度評価を取得するために使用できます。詳細については、「NVD 脆弱性の重大度」を参照してください。

どちらのバージョンの Amazon ECR ベーシックスキャンでも、プッシュ時にスキャンするリポジトリを指定するためのフィルターがサポートされています。プッシュ時のスキャンフィルターに一致しないリポジトリは、手動スキャン頻度に設定されます。つまり、スキャンを手動で開始する必要があります。イメージは 24 時間に 1 回スキャンできます。24 時間には、設定されている場合はプッシュ時の最初のスキャンと手動スキャンが含まれます。

最後に完了したイメージスキャンの結果は、各イメージに対して取得できます。イメージスキャンが完了すると、Amazon ECR はイベントを Amazon に送信します EventBridge。詳細については、「Amazon ECR イベントと EventBridge」を参照してください。

基本スキャンを改善するためのリージョンのサポート

基本スキャンの改良バージョンは、以下のリージョンでサポートされています。

- アジアパシフィック (香港) (`ap-east-1`)
- 欧州 (ストックホルム) (`eu-north-1`)

- 中東 (バーレーン) (me-south-1)
- アジアパシフィック (ムンバイ) (ap-south-1)
- 欧州 (パリ) (eu-west-3)
- AWS GovCloud （ 米国東部) (us-gov-east-1)
- アフリカ (ケープタウン) (af-south-1)
- アジアパシフィック (ジャカルタ) (ap-southeast-3)
- 欧州 (フランクフルト) (eu-central-1)
- 欧州 (アイルランド) (eu-west-1)
- 南米 (サンパウロ) (sa-east-1)
- 米国東部 (オハイオ) (us-east-2)
- AWS GovCloud （ 米国西部) (us-gov-west-1)
- アジアパシフィック (東京) (ap-northeast-1)
- アジアパシフィック (ソウル) (ap-northeast-2)
- アジアパシフィック (大阪) (ap-northeast-3)
- 欧州 (ミラノ) (eu-south-1)
- 欧州 (ロンドン) (eu-west-2)
- 米国東部 (バージニア北部) (us-east-1)
- アジアパシフィック (シンガポール) (ap-southeast-1)
- アジアパシフィック (シドニー) (ap-southeast-2)
- カナダ (中部) (ca-central-1)
- 米国西部 (北カリフォルニア) (us-west-1)
- 米国西部 (オレゴン) (us-west-2)
- 欧州 (チューリッヒ) (eu-central-2)

オペレーティングシステムのベーシックスキャンのサポートとベーシックスキャンの改善

セキュリティのベストプラクティスとして、またカバレッジを継続するために、サポートされているバージョンのオペレーティングシステムを引き続き使用することをお勧めします。ベンダーポリシー

に従って、廃止されたオペレーティングシステムはパッチで更新されなくなり、多くの場合、新しいセキュリティアドバイザリはリリースされなくなります。さらに、影響を受けるオペレーティングシステムが標準サポートが終了すると、既存のセキュリティアドバイザリと検出情報をフィードから削除するベンダーもあります。ディストリビューションがベンダーからサポートを失った後、Amazon ECR は脆弱性のスキャンをサポートしなくなる可能性があります。Amazon ECR が廃止されたオペレーティングシステムに対して生成した検出結果は、情報提供のみを目的として使用してください。現在サポートされているオペレーティングシステムとバージョンを以下に示します。

オペレーティングシステム	Version
Alpine Linux (Alpine)	3.19
Alpine Linux (Alpine)	3.18
Alpine Linux (Alpine)	3.17
Alpine Linux (Alpine)	3.16
Amazon Linux 2 (AL2)	AL2
Amazon Linux 2023 (AL2023)	AL2023
CentOS Linux (CentOS)	7
Debian サーバー (Bookworm)	12
Debian サーバー (Bullseye)	11
Debian サーバー (Buster)	10
Oracle Linux (Oracle)	9
Oracle Linux (Oracle)	8
Oracle Linux (Oracle)	7
Ubuntu (ルーナー)	23.04
Ubuntu (Jammy)	22.04 (LTS)
Ubuntu (Focal)	20.04 (LTS)

オペレーティングシステム	Version
Ubuntu (Bionic)	18.04 (ESM)
Ubuntu (Xenial)	16.04 (ESM)
Ubuntu (Trusty)	14.04 (ESM)
Red Hat Enterprise Linux (RHEL)	7
Red Hat Enterprise Linux (RHEL)	8
Red Hat Enterprise Linux (RHEL)	9

Amazon ECR でのイメージのベーシックスキャンの改善の設定

Amazon ECR ベーシックスキャンの改善バージョンがプレビューで利用可能になりました。ベーシックスキャンが改善され、 AWS ネイティブテクノロジーを使用できるようになりました。

プライベートリポジトリのリージョンごとに改善されたベーシックスキャンを設定します。基本スキャンの改善をサポートするリージョンのリストについては、「」を参照してください基本スキャンを改善するためのリージョンのサポート。

プライベートレジストリのベーシックスキャンの改善を有効にするには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、スキャン設定を設定するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Scanning] (スキャン) の順に選択します。
4. スキャン設定ページで、スキャンタイプ で、基本スキャンの改善 (プレビュー中) - 新しい を選択します。
5. デフォルトでは、すべてのリポジトリは [Manual] (手動) スキャンに設定されます。オプションで、[プッシュ時にスキャンするフィルター] を指定して、プッシュ時のスキャンを設定できます。すべてのリポジトリまたは個々のリポジトリに対して、プッシュ時のスキャンを設定できます。詳細については、「Amazon ECR でスキャンするリポジトリを選択するフィルター」を参照してください。

Amazon ECR でのイメージのベーシックスキャンの設定

デフォルトでは、Amazon ECR はすべてのプライベートレジストリのベーシックスキャンを有効にします。そのため、プライベートレジストリのスキャン設定を変更しない限り、ベーシックスキャンを有効にする必要はありません。ベーシックスキャンでは、オープンソースの Clair プロジェクトを使用します。

プッシュフィルターで 1 つ以上のスキャンを定義するには、次のステップを使用します。

プライベートレジストリのベーシックスキャンを有効にするには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、スキャン設定を設定するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Scanning] (スキャン) の順に選択します。
4. [Scanning configuration] (スキャン設定) ページの [Scan type] (スキャンタイプ) で [Basic scanning] (ベーシックスキャン) を選択します。
5. デフォルトでは、すべてのリポジトリは [Manual] (手動) スキャンに設定されます。オプションで、[プッシュ時にスキャンするフィルター] を指定して、プッシュ時のスキャンを設定できます。すべてのリポジトリまたは個々のリポジトリに対して、プッシュ時のスキャンを設定できます。詳細については、「Amazon ECR でスキャンするリポジトリを選択するフィルター」を参照してください。

Amazon ECR で OS の脆弱性がないかイメージを手動でスキャンする

リポジトリがプッシュ時にスキャンするように設定されていない場合は、イメージスキャンを手動で開始できます。イメージは 24 時間に 1 回スキャンできます。24 時間には、設定されている場合はプッシュ時の最初のスキャンと手動スキャンが含まれます。

イメージをスキャンする際の一般的な問題のトラブルシューティングの詳細については、「Amazon ECR でのイメージスキャンのトラブルシューティング」を参照してください。

AWS Management Console

AWS Management Consoleを使用して手動イメージスキャンを開始するには、次の手順を実行します。

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。

2. ナビゲーションバーから、リポジトリを作成するリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. リポジトリページで、スキャンするイメージを含むリポジトリを選択します。
5. イメージページで、スキャンするイメージを選択し、[スキャン] を選択します。

AWS CLI

- start-image-scan (AWS CLI)

  次の例では、イメージタグを使用しています。

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

  次の例では、イメージダイジェストを使用しています。

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

AWS Tools for Windows PowerShell

- Get-ECR ImageScanの検出結果 (AWS Tools for Windows PowerShell)

  次の例では、イメージタグを使用しています。

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

  次の例では、イメージダイジェストを使用しています。

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

Amazon ECR でのベーシックスキャンの検出結果の取得

最後に完了した基本イメージスキャンのスキャン結果を取得できます。検出されたソフトウェアの脆弱性は、共通脆弱性識別子 (CVEs) データベースに基づいて重要度別に一覧表示されます。

イメージをスキャンする際の一般的な問題のトラブルシューティングの詳細については、「Amazon ECR でのイメージスキャンのトラブルシューティング」を参照してください。

AWS Management Console

AWS Management Consoleを使用してイメージスキャンの結果を取得する手順は、次のとおりです。

イメージスキャンの検出結果を取得するには

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、リポジトリを作成するリージョンを選択します。
3. ナビゲーションペインで、[Repositories] を選択します。
4. リポジトリページで、スキャン結果を取得するイメージを含むリポジトリを選択します。
5. イメージページの [Vulnerabilities (脆弱性)] 列で、スキャン結果を取得するイメージの [Details (詳細)] を選択します。

AWS CLI

次の AWS CLI コマンドを使用して、 を使用してイメージスキャンの結果を取得します AWS CLI。`imageTag` または `imageDigest` を使用してイメージを指定できます。どちらのイメージも list-images CLI コマンドを使用して取得できます。

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

  次の例では、イメージタグを使用しています。

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

  次の例では、イメージダイジェストを使用しています。

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

AWS Tools for Windows PowerShell

- Get-ECR ImageScanの検出結果 (AWS Tools for Windows PowerShell)

  次の例では、イメージタグを使用しています。

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

  次の例では、イメージダイジェストを使用しています。

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

Amazon ECR でのイメージスキャンのトラブルシューティング

以下は、一般的なイメージスキャンの失敗です。このようなエラーは、Amazon ECR コンソールでイメージの詳細を表示するか、 API または `DescribeImageScanFindings` API AWS CLI を使用して表示できます。

UnsupportedImageエラー

Amazon ECR でのベーシックイメージスキャニングがサポートされていないオペレーティングシステムを使用して構築されたイメージに対してベーシックスキャンを実行しようとすると、`UnsupportedImageError` エラーが発生することがあります。Amazon ECR は、Amazon Linux、Amazon Linux 2、Debian、Ubuntu、CentOS、Oracle Linux、Alpine、RHEL Linux ディストリビューションのメジャーバージョンのパッケージ脆弱性スキャニングをサポートします。ディストリビューションでベンダーからのサポートがなくなると、Amazon ECR は脆弱性のスキャンをサポートしなくなる可能性があります。Amazon ECRは、Docker スクラッチイメージから構築されたイメージのスキャンをサポートしません。

> **Important**
>
> 拡張スキャンを使用する場合、Amazon Inspector では特定のオペレーティングシステムおよびメディアタイプのスキャンがサポートされます。完全なリストについては、Amazon Inspector ユーザーガイドの「サポートされているオペレーティングシステムおよびメディアタイプ」を参照してください。

UNDEFINED 深刻度が返される

深刻度が UNDEFINED のスキャン結果が返される場合があります。この問題の一般的な原因は以下のとおりです。

- この脆弱性に、CVE ソースによって優先度が割り当てられていなかった。
- この脆弱性に、Amazon ECR が認識しない優先度が割り当てられていた。

脆弱性の深刻度と説明を判断するには、ソースから直接 CVE を表示できます。

スキャンステータス SCAN_ELIGIBILITY_EXPIRED を理解する

プライベートレジストリに対して Amazon Inspector を使用した拡張スキャンが有効になっている場合にスキャンの脆弱性を表示すると、SCAN_ELIGIBILITY_EXPIRED のスキャンステータスが表示されることがあります。この問題の最も一般的な原因は以下のとおりです。

- プライベートレジストリの拡張スキャンを最初にオンにすると、Amazon Inspector は、画像プッシュのタイムスタンプに基づいて、過去 30 日以内に Amazon ECR にプッシュされた画像のみを認識します。古い画像は SCAN_ELIGIBILITY_EXPIRED スキャンステータスになります。これらの画像を Amazon Inspector でスキャンしたい場合は、リポジトリに再度プッシュする必要があります。
- Amazon Inspector コンソールで ECR 再スキャン時間を変更し、その時間が経過すると、イメージのスキャンステータスは expired の理由コードで inactive に変更され、イメージに関するすべての結果はクローズされるようにスケジュールされます。その結果、Amazon ECR コンソールにスキャンステータスが SCAN_ELIGIBILITY_EXPIRED として一覧表示されます。

アップストリームレジストリと Amazon ECR プライベートレジストリを同期する

プルスルーキャッシュルールを使用すると、アップストリームレジストリの内容を Amazon ECR プライベートレジストリと同期できます。

Amazon ECR は現在、次のアップストリームレジストリのプルスルーキャッシュルールの作成をサポートしています。

- Docker Hub、Microsoft Azure コンテナレジストリ、 GitHub コンテナレジストリ、 GitLab コンテナレジストリ (認証が必要)
- Amazon ECR パブリック、Kubernetes コンテナイメージレジストリ、Quay (認証は不要)

GitLab Container Registry の場合、Amazon ECR は GitLab software-as-a-service offering, GitLab.com でのみプルスルーキャッシュをサポートします。

認証が必要なアップストリームレジストリの場合は、認証情報を AWS Secrets Manager シークレットに保存する必要があります。Amazon ECR コンソールでは、認証された各アップストリームレジストリの Secrets Manager シークレットを簡単に作成できます。Secrets Manager コンソールを使用して Secrets Manager シークレットを作成する方法の詳細については、「」を参照してくださいシーク AWS Secrets Manager レットへのアップストリームリポジトリ認証情報の保存。

アップストリームレジストリのプルスルーキャッシュルールを作成したら、Amazon ECR プライベートレジストリ URI を使用して、そのアップストリームレジストリからイメージをプルするだけです。次に Amazon ECR はリポジトリを作成し、そのイメージをプライベートレジストリにキャッシュします。特定のタグを持つキャッシュされたイメージのその後のプルリクエストで、Amazon ECR はアップストリームレジストリをチェックして、その特定のタグを持つイメージの新しいバージョンがあるかどうかを確認し、少なくとも 24 時間に 1 回プライベートレジストリでイメージを更新しようとします。

リポジトリ作成テンプレート

Amazon ECR は、現在プレビュー中のリポジトリ作成テンプレートのサポートを追加しました。これにより、プルスルーキャッシュルールを使用して Amazon ECR がユーザーに代わって作成した新しいリポジトリの初期設定を指定できるようになります。各テンプレートには、新しいリポジトリを特定のテンプレートと一致させるために使用されるリポジトリ名前空間プレフィックスが含まれて

います。テンプレートでは、リソースベースのアクセスポリシー、タグのイミュータビリティ、暗号化、ライフサイクルポリシーなど、すべてのリポジトリ設定の設定を指定できます。リポジトリ作成テンプレートの設定はリポジトリの作成時にのみ適用され、既存のリポジトリや他の方法で作成されたリポジトリには影響しません。詳細については、「プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート」を参照してください。

プルスルーキャッシュルールを使用する際の考慮事項

Amazon ECR プルスルーキャッシュルールを使用する場合は、次の点を考慮してください。

- 次のリージョンでは、プルスルーキャッシュルールの作成はサポートされていません。
  - 中国 (北京) (cn-north-1)
  - 中国 (寧夏) (cn-northwest-1)
  - AWS GovCloud （ 米国東部) (us-gov-east-1)
  - AWS GovCloud （ 米国西部) (us-gov-west-1)
- AWS Lambda は、プルスルーキャッシュルールを使用した Amazon ECR からのコンテナイメージのプルをサポートしていません。
- プルスルーキャッシュを使用してイメージをプルする場合、イメージを最初にプルするときに Amazon ECR FIPS サービスエンドポイントはサポートされません。ただし、Amazon ECR FIPS サービスエンドポイントは、その後のプルでも使用できます。
- キャッシュされたイメージが Amazon ECR プライベートレジストリ URI を介してプルされると、イメージのプルは AWS IP アドレスによって開始されます。これにより、アップストリームレジストリが実行するプルレートクォータに対して、イメージのプルがカウントされないようになります。
- キャッシュされたイメージが Amazon ECR プライベートレジストリ URI を介してプルされると、Amazon ECR はアップストリームリポジトリを少なくとも 24 時間に 1 回チェックして、キャッシュされたイメージが最新バージョンであるかどうかを確認します。アップストリームレジストリに新しいイメージがある場合、Amazon ECR はキャッシュされたイメージの更新を試みます。このタイマーは、キャッシュされたイメージの最後のプルに基づいています。
- Amazon ECR が何らかの理由でアップストリームレジストリからイメージを更新できず、イメージがプルされた場合でも、最後にキャッシュされたイメージがプルされます。
- アップストリームのレジストリ認証情報を含む Secrets Manager シークレットを作成する場合、シークレット名には ecr-pullthroughcache/ プレフィックスを使用する必要があります。シークレットは、プルスルーキャッシュルールが作成されたのと同じアカウントとリージョンにある必要もあります。

- プルスルーキャッシュルールを使用してマルチアーキテクチャイメージをプルすると、マニフェストリストとマニフェストリストで参照されている各イメージが Amazon ECR リポジトリにプルされます。特定のアーキテクチャのみをプルする場合は、マニフェストリストに関連付けられたタグではなく、アーキテクチャに関連付けられたイメージダイジェストまたはタグを使用してイメージをプルできます。
- Amazon ECR は、サービスにリンクされた IAM ロールを使用します。このロールは、Amazon ECR がユーザーに代わってキャッシュされたイメージのリポジトリを作成し、認証のために Secret Manager シークレット値を取得し、キャッシュされたイメージをプッシュするために必要なアクセス許可を提供します。プルスルーキャッシュルールを作成すると、サービスにリンクされた IAM ロールが自動的に作成されます。詳細については、「プルスルーキャッシュの Amazon ECR サービスにリンクされたロール」を参照してください。
- デフォルトで、キャッシュされたイメージをプルする IAM プリンシパルには、IAM ポリシーによってアクセス許可が付与されています。Amazon ECR プライベートレジストリアクセス許可ポリシーを使用して、IAM エンティティのアクセス許可のスコープをさらに設定できます。詳細については、「レジストリ許可の使用」を参照してください。
- プルスルーキャッシュワークフローを使用して作成された Amazon ECR リポジトリは、他の Amazon ECR リポジトリと同様に処理されます。レプリケーションやイメージスキャンなど、すべてのリポジトリ機能がサポートされています。
- Amazon ECR がプルスルーキャッシュアクションを使用してユーザーに代わって新しいリポジトリを作成すると、一致するリポジトリ作成テンプレートがない限り、次のデフォルト設定がリポジトリに適用されます。リポジトリ作成テンプレートを使用して、Amazon ECR がユーザーに代わって作成したリポジトリに適用される設定を定義できます。詳細については、「プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート」を参照してください。
  - タグのイミュータビリティ — オフにすると、タグは変更可能になり上書きできます。
  - 暗号化 — デフォルトの AES256 暗号化が使用されます。
  - リポジトリ権限 — 省略。リポジトリ権限ポリシーは適用されません。
  - ライフサイクルポリシー — 省略。ライフサイクルポリシーは適用されません。
  - リソースタグ — 省略。リソースタグは適用されません。
- プルスルーキャッシュルールを使用してリポジトリのイメージタグのイミュータビリティを有効にすると、Amazon ECR が同じタグを使用してイメージを更新できなくなります。
- プルスルーキャッシュルールを使用してイメージをプルすると、インターネットへのルートが初めて必要になる場合があります。インターネットへのルートが必要な状況がいくつかあるため、障害を避けるためにルートを設定するのが最善です。したがって、 を使用してインターフェイス VPC エンドポイントを使用するように Amazon ECR を設定している場合は AWS PrivateLink 、最初の

プルにインターネットへのルートがあることを確認する必要があります。これを行う 1 つの方法は、インターネットゲートウェイを使用して同じ VPC にパブリックサブネットを作成し、プライベートサブネットからパブリックサブネットにすべてのアウトバウンドトラフィックをインターネットにルーティングすることです。プルスルーキャッシュルールを使用した後続のイメージプルでは、これは必要ありません。詳細については、Amazon Virtual Private Cloud ユーザーガイドの「ルートオプションの例」を参照してください。

アップストリームレジストリと Amazon ECR プライベートレジストリを同期するために必要な IAM アクセス許可

プルスルーキャッシュルールを効果的に使用するためには、プライベートレジストリへの認証とイメージのプッシュとプルに必要となる Amazon ECR API のアクセス許可に加えて、次のアクセス許可も更に必要となります。

- `ecr:CreatePullThroughCacheRule` – プルスルーキャッシュルールを作成するアクセス許可を付与します。このアクセス許可は、アイデンティティに基づく IAM ポリシーを介して付与する必要があります。
- `ecr:BatchImportUpstreamImage` – 外部イメージを取得し、プライベートレジストリにインポートするアクセス許可を付与します。このアクセス許可は、プライベートレジストリアクセス許可ポリシー、ID に基づく IAM ポリシー、またはリソースに基づくリポジトリアクセス許可ポリシーを使用して付与できます。レポジトリアクセス許可の使用に関する詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。
- `ecr:CreateRepository` – プライベートレジストリにリポジトリを作成するアクセス許可を付与します。キャッシュされたイメージを格納するリポジトリがまだ存在しない場合には、この許可が必要となります。このアクセス許可は、アイデンティティに基づく IAM ポリシーまたはプライベートレジストリアクセス許可のいずれかによって付与できます。
- `ecr:TagResource` – Amazon ECR リソースにメタデータタグを追加する許可を付与します。この許可が必要なのは、プルスルーキャッシュルールを使用するイメージに、リポジトリにリソースタグを追加するように設定されたリポジトリ作成テンプレートが関連付けられている場合のみです。このアクセス許可は、アイデンティティに基づく IAM ポリシーを介して付与する必要があります。

レジストリ許可の使用

Amazon ECR プライベートレジストリ許可は、プルスルーキャッシュを使用する個々の IAM エンティティのアクセス許可のスコープ設定に使用できます。レジストリ許可ポリシーによって付与されるアクセス許可よりも多くのアクセス許可が IAM ポリシーによって IAM エンティティに付与される場合、IAM ポリシーが優先されます。例えば、 ユーザーに `ecr:*` 許可が付与されている場合には、レジストリレベルで更に許可する必要はありません。

プライベートレジストリの許可ポリシーを作成するには (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ許可ステートメントを設定するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Registry permissions] (レジストリー許可) の順に選択します。
4. [Registry permissions] (レジストリー許可) ページで [Generate statement] (ステートメントを生成) を選択します。
5. 作成するプルスルーキャッシュ許可ポリシーステートメントごとに、次の操作を行います。

    a. [ポリシータイプ] で、[プルスルーキャッシュポリシー] を選択します。
    b. [ステートメント ID] で、プルスルーキャッシュステートメントポリシーの名前を指定します。
    c. [IAM entities] (IAM エンティティ) で、ポリシーに含めるユーザー、グループ、またはロールを指定します。
    d. [リポジトリ名前空間] で、ポリシーを関連付けるプルスルーキャッシュルールを選択します。
    e. [リポジトリ名] で、ルールを適用するリポジトリベース名を指定します。たとえば、Amazon ECR パブリックで Amazon Linux リポジトリを指定する場合、リポジトリ名は `amazonlinux` になります。

プライベートレジストリの許可ポリシーを作成するには (AWS CLI)

次の AWS CLI コマンドを使用して、 を使用してプライベートレジストリのアクセス許可を指定します AWS CLI。

1. レジストリポリシーのコンテンツが含まれる `ptc-registry-policy.json` という名のローカルファイルを作成します。次の例では、リポジトリを作成し、以前に作成したプルスルーキャッ

シュルールに関連付けられたアップストリームソースである Amazon ECR Public からイメージをプルする ecr-pull-through-cache-user 許可が付与されています。

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

> **Important**
>
> ecr-CreateRepository 許可は、キャッシュされたイメージを格納するリポジトリがまだ存在しない場合にのみ必要となります。たとえば、リポジトリ作成アクションとイメージをプルするアクションが、管理者や開発者などの別の IAM プリンシパルによって実行されている場合などです。

2. レジストリポリシーを設定するには、put-registry-policy コマンドを使用します。

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

次のステップ

プルスルーキャッシュルールを使用する準備ができたら、次のステップは以下のとおりです。

- プルスルーキャッシュルールを作成します。詳細については、「Amazon ECR でのプルスルーキャッシュルールの作成」を参照してください。
- リポジトリ作成テンプレートを作成します。リポジトリ作成テンプレートは、プルスルーキャッシュアクションで Amazon ECR がユーザーに代わって作成した新しいリポジトリに使用する設定を定義することができます。詳細については、「プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート」を参照してください。

Amazon ECR でのプルスルーキャッシュルールの作成

Amazon ECR プライベートレジストリにキャッシュするイメージを含むアップストリームレジストリごとに、プルスルーキャッシュルールを作成する必要があります。

認証が必要なアップストリームレジストリの場合は、認証情報を Secrets Manager シークレットに保存する必要があります。既存の シークレットを使用するか、新しいシークレットを作成できます。Secrets Manager シークレットは、Amazon ECR コンソールまたは Secrets Manager コンソールで作成できます。Amazon ECR コンソールの代わりに Secrets Manager コンソールを使用して Secrets Manager シークレットを作成するには、「」を参照してくださいシーク AWS Secrets Manager レットへのアップストリームリポジトリ認証情報の保存。

前提条件

- プルスルーキャッシュルールを作成するための適切な IAM アクセス許可があることを確認します。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期するために必要な IAM アクセス許可」を参照してください。
- 認証が必要なアップストリームレジストリの場合: 既存のシークレットを使用する場合は、Secrets Manager シークレットが次の要件を満たしていることを確認します。
  - シークレットの名前は で始まりますecr-pullthroughcache/。は、 AWS Management Console ecr-pullthroughcache/ プレフィックスを持つ Secrets Manager シークレットのみを表示します。
  - シークレットがあるアカウントとリージョンは、プルスルーキャッシュルールがあるアカウントとリージョンと一致する必要があります。

プルスルーキャッシュルールの作成 (AWS Management Console)

以下のステップは、Amazon ECR コンソールを使用してプルスルーキャッシュルールと Secrets Manager シークレットを作成する方法を示しています。Secrets Manager コンソールを使用してシークレットを作成するには、「」を参照してくださいシーク AWS Secrets Manager レットへのアップストリームリポジトリ認証情報の保存。

Amazon ECR パブリック、Kubernetes コンテナレジストリ、または Quay の場合

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ設定を構成するリージョンを選択します。

3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Pull through cache] (プルスルーキャッシュ) の順に選択します。
4. [Pull through cache configuration] (プルスルーキャッシュの設定) ページで、[Add rule] (ルールの追加) を選択します。
5. [ステップ 1: ソースを指定] ページの [レジストリ] で、アップストリームレジストリのリストから Amazon ECR パブリック、Kubernetes、または Quay のいずれかを選択し、[次へ] を選択します。
6. [ステップ 2: 宛先を指定する] ページの [Amazon ECR リポジトリプレフィックス] で、ソースパブリックレジストリから取得したイメージをキャッシュするときに使用するリポジトリ名前空間プレフィックスを指定し、[次へ] を選択します。デフォルトでは、名前空間は設定されていますが、カスタム名前空間も指定できます。
7. [ステップ 3: 確認と作成] ページで、プルスルーキャッシュルールの設定を確認し、[作成] を選択します。
8. 作成する各プルスルーキャッシュに対して、前のステップを繰り返します。プルスルーキャッシュルールは、リージョンごとに個別に作成されます。

Docker Hub の場合

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ設定を構成するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Pull through cache] (プルスルーキャッシュ) の順に選択します。
4. [Pull through cache configuration] (プルスルーキャッシュの設定) ページで、[Add rule] (ルールの追加) を選択します。
5. [ステップ 1: ソースを指定] ページの [レジストリ] で [Docker Hub] を選択し、[次へ] を選択します。
6. [ステップ 2: 認証の設定] ページの [アップストリームの認証情報] では、Docker Hub の認証認証情報を AWS Secrets Manager シークレットに保存する必要があります。既存のシークレットを指定するか、Amazon ECR コンソールを使用して新しいシークレットを作成できます。

    a. 既存のシークレットを使用するには、既存のシーク AWS レットを使用する を選択します。[シークレット名] では、ドロップダウンを使用して既存のシークレットを選択し、[次へ] を選択します。

> **Note**
>
> は、 `ecr-pullthroughcache/`プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみを表示します。シークレットは、プルスルーキャッシュルールが作成されたのと同じアカウントとリージョンにある必要もあります。

b. 新しいシークレットを作成するには、[AWS シークレットを作成する] を選択し、次の操作を行って、[次へ] を選択します。

   i. [シークレット名] には、シークレットのわかりやすい名前を指定します。シークレット名は 1～512 文字の Unicode 文字で構成されます。

   ii. [Docker ハブのユーザー名] には、Docker Hub のユーザー名を指定します。

   iii. [Docker Hub のアクセストークン]には、Docker Hub アクセストークンを指定します。Docker Hub アクセストークンの作成について詳しくは、Docker ドキュメントの「Create and manage access tokens」を参照してください。

7. [ステップ 3: 宛先を指定する] ページの [Amazon ECR リポジトリプレフィックス] で、ソースパブリックレジストリから取得したイメージをキャッシュするときに使用するリポジトリ名前空間を指定し、[次へ] を選択します。

   デフォルトでは、名前空間は設定されていますが、カスタム名前空間も指定できます。

8. [ステップ 4: 確認と作成] ページで、プルスルーキャッシュルールの設定を確認し、[作成] を選択します。

9. 作成する各プルスルーキャッシュに対して、前のステップを繰り返します。プルスルーキャッシュルールは、リージョンごとに個別に作成されます。

GitHub コンテナレジストリの場合

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ設定を構成するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Pull through cache] (プルスルーキャッシュ) の順に選択します。
4. [Pull through cache configuration] (プルスルーキャッシュの設定) ページで、[Add rule] (ルールの追加) を選択します。

5. ステップ 1: ソースページを指定し、レジストリ で、GitHub コンテナレジストリ 、次へ を選択します。
6. ステップ 2: 認証を設定するページで、アップストリーム認証情報 の場合、 GitHub コンテナレジストリの認証情報を シークレットに保存 AWS Secrets Manager する必要があります。既存のシークレットを指定するか、Amazon ECR コンソールを使用して新しいシークレットを作成できます。

   a. 既存のシークレットを使用するには、既存のシーク AWS レットを使用する を選択します。[シークレット名] では、ドロップダウンを使用して既存のシークレットを選択し、[次へ] を選択します。

      > **Note**
      >
      > は、 `ecr-pullthroughcache/`プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみを表示します。シークレットは、プルスルーキャッシュルールが作成されたのと同じアカウントとリージョンにある必要もあります。

   b. 新しいシークレットを作成するには、[AWS シークレットを作成する] を選択し、次の操作を行って、[次へ] を選択します。

      i. [シークレット名] には、シークレットのわかりやすい名前を指定します。シークレット名は 1～512 文字の Unicode 文字で構成されます。

      ii. GitHub コンテナレジストリのユーザー名 には、 GitHub コンテナレジストリのユーザー名を指定します。

      iii. GitHub コンテナレジストリアクセストークン で、 GitHub コンテナレジストリアクセストークンを指定します。 GitHub アクセストークンの作成の詳細については、GitHub ドキュメントの「個人用アクセストークンの管理」を参照してください。
7. [ステップ 3: 宛先を指定する] ページの [Amazon ECR リポジトリプレフィックス] で、ソースパブリックレジストリから取得したイメージをキャッシュするときに使用するリポジトリ名前空間を指定し、[次へ] を選択します。

   デフォルトでは、名前空間は設定されていますが、カスタム名前空間も指定できます。
8. [ステップ 4: 確認と作成] ページで、プルスルーキャッシュルールの設定を確認し、[作成] を選択します。
9. 作成する各プルスルーキャッシュに対して、前のステップを繰り返します。プルスルーキャッシュルールは、リージョンごとに個別に作成されます。

Microsoft Azure コンテナレジストリの場合

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ設定を構成するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Pull through cache] (プルスルーキャッシュ) の順に選択します。
4. [Pull through cache configuration] (プルスルーキャッシュの設定) ページで、[Add rule] (ルールの追加) を選択します。
5. [ステップ 1:ソースの指定] ページで、以下の操作を行います。

   a. [レジストリ] には、[Microsoft Azure コンテナレジストリ] を選択します。

   b. [ソースレジストリ URL] に Microsoft Azure コンテナレジストリの名前を指定し、[次へ] を選択します。

   > **Important**
   >
   > `.azurecr.io` サフィックスはユーザーに代わって入力されるため、プレフィックスを指定するだけで済みます。

6. [ステップ 2: 認証の設定] ページの [アップストリームの認証情報] では、Microsoft Azure コンテナレジストリ の認証認証情報を AWS Secrets Manager シークレットに保存する必要があります。既存のシークレットを指定するか、Amazon ECR コンソールを使用して新しいシークレットを作成できます。

   a. 既存のシークレットを使用するには、既存のシーク AWS レットを使用する を選択します。[シークレット名] では、ドロップダウンを使用して既存のシークレットを選択し、[次へ] を選択します。

   > **Note**
   >
   > は、 `ecr-pullthroughcache/`プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみを表示します。シークレットは、プルスルーキャッシュルールが作成されたのと同じアカウントとリージョンにある必要もあります。

   b. 新しいシークレットを作成するには、[AWS シークレットを作成する] を選択し、次の操作を行って、[次へ] を選択します。

i. [シークレット名] には、シークレットのわかりやすい名前を指定します。シークレット名は 1～512 文字の Unicode 文字で構成されます。

ii. [Microsoft Azure コンテナレジストリのユーザー名] には、Microsoft Azure コンテナーレジストリのユーザー名を指定します。

iii. [Microsoft Azure コンテナレジストリのアクセストークン] には、Microsoft Azure コンテナレジストリのアクセストークンを指定します。Microsoft Azure コンテナレジストリアクストークンの作成について詳しくは、Microsoft Azure ドキュメントの「トークンを作成する - ポータル」を参照してください。

7. [ステップ 3: 宛先を指定する] ページの [Amazon ECR リポジトリプレフィックス] で、ソースパブリックレジストリから取得したイメージをキャッシュするときに使用するリポジトリ名前空間を指定し、[次へ] を選択します。

   デフォルトでは、名前空間は設定されていますが、カスタム名前空間も指定できます。

8. [ステップ 4: 確認と作成] ページで、プルスルーキャッシュルールの設定を確認し、[作成] を選択します。

9. 作成する各プルスルーキャッシュに対して、前のステップを繰り返します。プルスルーキャッシュルールは、リージョンごとに個別に作成されます。

GitLab コンテナレジストリの場合

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プライベートレジストリ設定を構成するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Pull through cache] (プルスルーキャッシュ) の順に選択します。
4. [Pull through cache configuration] (プルスルーキャッシュの設定) ページで、[Add rule] (ルールの追加) を選択します。
5. ステップ 1: ソースを指定するページの Registry で、 GitLab Container Registry, Next を選択します。
6. ステップ 2: 認証を設定するページで、アップストリーム認証情報 の場合、 GitLab コンテナレジストリの認証情報を シークレットに保存 AWS Secrets Manager する必要があります。既存のシークレットを指定するか、Amazon ECR コンソールを使用して新しいシークレットを作成できます。

a. 既存のシークレットを使用するには、既存のシーク AWS レットを使用する を選択します。[シークレット名] では、ドロップダウンを使用して既存のシークレットを選択し、[次へ] を選択します。Secrets Manager コンソールを使用して Secrets Manager シークレットを作成する方法の詳細については、シーク AWS Secrets Manager レットへのアップストリームリポジトリ認証情報の保存 を参照してください。

> **Note**
>
> は、 `ecr-pullthroughcache/`プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみを表示します。シークレットは、プルスルーキャッシュルールが作成されたのと同じアカウントとリージョンにある必要もあります。

b. 新しいシークレットを作成するには、[AWS シークレットを作成する] を選択し、次の操作を行って、[次へ] を選択します。

   i. [シークレット名] には、シークレットのわかりやすい名前を指定します。シークレット名は 1～512 文字の Unicode 文字で構成されます。

   ii. GitLab コンテナレジストリのユーザー名 には、 GitLab コンテナレジストリのユーザー名を指定します。

   iii. GitLab コンテナレジストリアクセストークン には、 GitLab コンテナレジストリアクセストークンを指定します。 GitLab コンテナレジストリアクセストークンの作成の詳細については、 GitLab ドキュメントの「個人用アクセストークン」、「グループアクセストークン」、または「プロジェクトアクセストークン」を参照してください。

7. [ステップ 3: 宛先を指定する] ページの [Amazon ECR リポジトリプレフィックス] で、ソースパブリックレジストリから取得したイメージをキャッシュするときに使用するリポジトリ名前空間を指定し、[次へ] を選択します。

   デフォルトでは、名前空間は設定されていますが、カスタム名前空間も指定できます。

8. [ステップ 4: 確認と作成] ページで、プルスルーキャッシュルールの設定を確認し、[作成] を選択します。

9. 作成する各プルスルーキャッシュに対して、前のステップを繰り返します。プルスルーキャッシュルールは、リージョンごとに個別に作成されます。

プルスルーキャッシュルールを作成するには (AWS CLI)

create-pull-through-cache-rule AWS CLI コマンドを使用して、Amazon ECR プライベートレジストリのプルスルーキャッシュルールを作成します。認証が必要なアップストリームレジストリでは、認証情報を Secrets Manager シークレットに保存する必要があります。Secrets Manager コンソールを使用してシークレットを作成するには、「」を参照してくださいシーク AWS Secrets Manager レットへのアップストリームリポジトリ認証情報の保存。

以下の例は、サポートされている各アップストリームレジストリについて提供されています。

Amazon ECR Public の場合

次の例では、Amazon ECR パブリックレジストリのプルスルーキャッシュルールを作成します。リポジトリプレフィックス `ecr-public` を指定します。この結果、プルスルーキャッシュルールを使用して作成された各リポジトリは命名スキーム `ecr-public/`*`upstream-repository-name`* を持ちます。

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

Kubernetes コンテナレジストリの場合

次の例では、Kubernetes パブリックレジストリのプルスルーキャッシュルールを作成します。リポジトリプレフィックス `kubernetes` を指定します。この結果、プルスルーキャッシュルールを使用して作成された各リポジトリは命名スキーム `kubernetes/`*`upstream-repository-name`* を持ちます。

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix kubernetes \
     --upstream-registry-url registry.k8s.io \
     --region us-east-2
```

Quay の場合

次の例では、Quay パブリックレジストリのプルスルーキャッシュルールを作成します。リポジトリプレフィックス `quay` を指定します。この結果、プルスルーキャッシュルールを使用して作成された各リポジトリは命名スキーム `quay/`*`upstream-repository-name`* を持ちます。

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

Docker Hub の場合

次の例では、Docker Hub レジストリのプルスルーキャッシュルールを作成します。リポジトリプレフィックス docker-hub を指定します。この結果、プルスルーキャッシュルールを使用して作成された各リポジトリは命名スキーム docker-hub/*upstream-repository-name* を持ちます。Docker Hub の認証情報が含まれているシークレットの完全な Amazon リソースネーム (ARN) を指定する必要があります。

```
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
```

GitHub コンテナレジストリの場合

次の例では、 GitHub コンテナレジストリのプルスルーキャッシュルールを作成します。リポジトリプレフィックス docker-hub を指定します。この結果、プルスルーキャッシュルールを使用して作成された各リポジトリは命名スキーム github/*upstream-repository-name* を持ちます。GitHub コンテナレジストリの認証情報を含むシークレットの完全な Amazon リソースネーム (ARN) を指定する必要があります。

```
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
```

Microsoft Azure コンテナレジストリの場合

次の例では、Microsoft Azure Container Registry のプルスルーキャッシュルールを作成します。リポジトリプレフィックス azure を指定します。この結果、プルスルーキャッシュルールを使

用して作成された各リポジトリは命名スキーム azure/*upstream-repository-name* を持ちます。Microsoft Azure コンテナレジストリの認証情報が含まれているシークレットの完全な Amazon リソースネーム (ARN) を指定する必要があります。

```
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
```

GitLab コンテナレジストリの場合

次の例では、 GitLab コンテナレジストリのプルスルーキャッシュルールを作成します。リポジトリプレフィックス gitlab を指定します。この結果、プルスルーキャッシュルールを使用して作成された各リポジトリは命名スキーム gitlab/*upstream-repository-name* を持ちます。 GitLab コンテナレジストリの認証情報を含むシークレットの完全な Amazon リソースネーム (ARN) を指定する必要があります。

```
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
```

次のステップ

プルスルーキャッシュルールを作成した後、次のステップは次のとおりです。

- リポジトリ作成テンプレートを作成します。リポジトリ作成テンプレートは、プルスルーキャッシュアクションで Amazon ECR がユーザーに代わって作成した新しいリポジトリに使用する設定を定義することができます。詳細については、「プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート」を参照してください。
- プルスルーキャッシュルールを検証します。プルスルーキャッシュルールを検証する際、Amazon ECR はアップストリームレジストリとのネットワーク接続を確立し、アップストリームレジストリの認証情報を含む Secrets Manager シークレットにアクセスできること、および認証が成功したことを確認します。詳細については、「Amazon ECR でのプルスルーキャッシュルールの検証」を参照してください。

- プルスルーキャッシュルールの使用を開始します。詳細については、「Amazon ECR のプルスルーキャッシュルールを使用してイメージをプルする」を参照してください。

プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート

リポジトリ作成テンプレートの機能は Amazon ECR のプレビュー版に含まれているもので、変更される可能性があります。このパブリックプレビューでは、リポジトリ作成テンプレートの管理に使用できるの AWS Management Console は のみです。

Amazon ECR リポジトリ作成テンプレートを使用して、プルスルーキャッシュアクション中にユーザーに代わって Amazon ECR によって作成されたリポジトリの設定を定義します。リポジトリ作成テンプレートの設定はリポジトリの作成時にのみ適用され、既存のリポジトリや他の方法で作成されたリポジトリには影響しません。

リポジトリ作成テンプレートは、次のリージョンではサポートされていません。

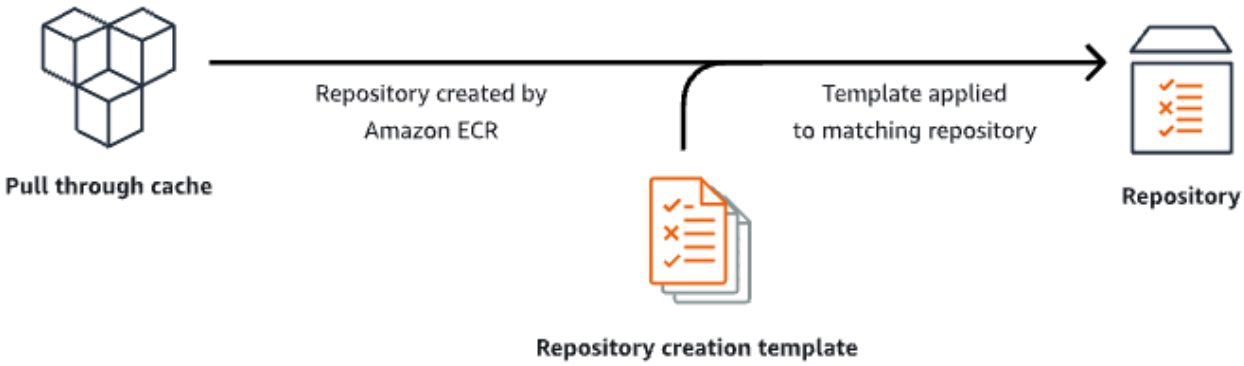
- 中国 (北京) (`cn-north-1`)
- 中国 (寧夏) (`cn-northwest-1`)
- AWS GovCloud （米国東部) (`us-gov-east-1`)
- AWS GovCloud （米国西部) (`us-gov-west-1`)

リポジトリ作成テンプレートの仕組み

Amazon ECR がユーザーに代わって新しいプライベートリポジトリを作成する必要がある場合があります。例えば、プルスルーキャッシュルールを初めて使用して、アップストリームリポジトリのコンテンツを取得し、Amazon ECR プライベートレジストリに保存します。プルスルーキャッシュルールに一致するリポジトリ作成テンプレートがない場合、Amazon ECR は新しいリポジトリのデフォルト設定を使用します。これらのデフォルト設定には、タグの不変性をオフにする、AES-256 暗号化を使用する、リポジトリやライフサイクルポリシーを一切適用しないなどが含まれます。

プルスルーキャッシュルールに一致するプレフィックスを付けたリポジトリ作成テンプレートを使用すると、プルスルーキャッシュアクションによって作成された新しいリポジトリに Amazon ECR が適用する設定を定義できます。新しいリポジトリのタグ不変性、暗号化設定、リポジトリ権限、ライフサイクルポリシー、リソースタグを定義できます。

次の図は、リポジトリ作成テンプレートを使用するときに Amazon ECR が使用するワークフローを示しています。



以下では、リポジトリ作成テンプレートの各パラメータについて詳しく説明します。

プレフィックス

[プレフィックス] は、テンプレートに関連付けるリポジトリ名前空間のプレフィックスです。このプレフィックスを使用して作成されたすべてのリポジトリには、このテンプレートで定義されている設定が適用されます。例えば、`prod` というプレフィックスは、`prod/` で始まるすべてのリポジトリに適用されます。同様に、`prod/team` というプレフィックスは、`prod/team/` で始まるすべてのリポジトリに適用されます。

作成テンプレートが関連付けられていないレジストリ内のすべてのリポジトリにテンプレートを適用するには、プレフィックスとして `ROOT` を使用できます。

> **Important**
>
> プレフィックスの末尾には常に `/` が適用されると想定されます。`ecr-public` をプレフィックスとして指定すると、Amazon ECR はそれを `ecr-public/` として扱います。プルスルーキャッシュルールを使用する場合、ルール作成時に指定するリポジトリプレフィックスは、リポジトリ作成テンプレートのプレフィックスとしても指定する必要があります。

説明

このテンプレートの説明は省略可能で、リポジトリ作成テンプレートの目的を説明するために使用されます。

[テンプレートのバージョン]

使用するリポジトリ作成テンプレートのバージョンです。現在サポートされているテンプレートのバージョンは TV1 のみです。

設定バージョン

使用するテンプレートのリポジトリ設定バージョンです。各テンプレートにはリポジトリ設定が含まれている必要があります。デフォルトの設定バージョンは CV1 で、イメージタグの変更可能性、リポジトリポリシー、およびライフサイクルポリシー設定で構成されます。

イメージタグの変更可能性

テンプレートを使用して作成されたリポジトリに使用するタグの変更可能性設定です。このパラメータを省略すると、MUTABLE のデフォルト設定が使用されます。デフォルト設定では、イメージタグの上書きが許可されます。これは、プルスルーキャッシュアクションによって作成されたリポジトリに使用されるテンプレートに使用することを推奨する設定です。これにより、タグが同じ場合でも Amazon ECR はキャッシュされたイメージを更新できます。

IMMUTABLE を指定すると、リポジトリ内のすべてのイメージタグは不変となり、上書きが禁止されます。

暗号化設定

テンプレートを使用して作成されたリポジトリに使用する暗号化設定です。

KMS 暗号化タイプを使用する場合、リポジトリのコンテンツは、 AWS KMSに保存されている AWS Key Management Service キーにより、サーバー側の暗号化を使用して暗号化されます。AWS KMS を使用してデータを暗号化する場合、Amazon ECR のデフォルトの AWS マネージド AWS KMS キーを使用するか、既に作成した独自の AWS KMS キーを指定できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の AWS Key Management Service 「 (SSE-KMS) に保存されている AWS Key Management Service キーによるサーバー側の暗号化を使用したデータの保護」を参照してください。

AES256 の暗号化タイプを使用する場合、Amazon ECR は Amazon S3 で管理された暗号化キーによりサーバー側の暗号化キーを使用して、AES-256 暗号化アルゴリズムを使用するリポジトリ内のイメージを暗号化します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3)」を参照してください。

リポジトリ権限

テンプレートを使用して作成されたリポジトリに適用するリポジトリポリシーです。リポジトリポリシーは、リソースベースのアクセス権限を使用してリポジトリへのアクセスを制御します。リソースベースのアクセス権限により、どの IAM ユーザーあるいはロールがリポジトリにアクセスでき、どのようなアクションを実行できるかを指定できます。デフォルトでは、リポジトリを作成した AWS アカウントのみがリポジトリにアクセスできます。リポジトリへの追加のアクセス許可を付与または拒否するポリシードキュメントを適用できます。詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。

リポジトリライフサイクルポリシー

テンプレートを使用して作成されたリポジトリに使用するライフサイクルポリシーです。ライフサイクルポリシーを使用すると、プライベートリポジトリ内のイメージのライフサイクル管理をより詳細に制御できます。ライフサイクルポリシーは 1 つまたは複数のルールで、各ルールでは Amazon ECR へのアクションが定義されています。時間やカウント数に基づいてイメージの有効期限を設定することによってコンテナイメージを自動的にクリーンアップできます。詳細については、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。

リソースタグ

リソースタグは、リポジトリに適用し、分類と整理に役立つメタデータです。タグはそれぞれ、1 つのキーとオプションの 1 つの値で設定されており、どちらもお客様側が定義します。

リポジトリ作成テンプレートを作成するための IAM アクセス許可

IAM プリンシパルがリポジトリ作成テンプレートを管理するには、以下のアクセス許可が必要です。これらのアクセス許可は、アイデンティティに基づく IAM ポリシーを使用して付与する必要があります。

- `ecr:CreateRepositoryCreationTemplate` — リポジトリ作成テンプレートを作成するアクセス許可を付与します。
- `ecr:DeleteRepositoryCreationTemplate` — リポジトリ作成テンプレートを削除するアクセス許可を付与します。
- `ecr:PutLifecyclePolicy` — ライフサイクルポリシーを作成し、それをリポジトリに適用するアクセス許可を付与します。このアクセス許可は、リポジトリ作成テンプレートにライフサイクルポリシーが含まれている場合にのみ必要です。

- `ecr:SetRepositoryPolicy` — リポジトリのアクセス許可ポリシーを作成するアクセス許可を付与します。このアクセス許可は、リポジトリ作成テンプレートにリポジトリポリシーが含まれている場合にのみ必要です。
- `ecr:TagResource` – リソースにメタデータタグを追加する許可を付与します。このアクセス許可は、リポジトリ作成テンプレートにリソースタグが含まれている場合にのみ必要です。

Amazon ECR でのリポジトリ作成テンプレートの作成

リポジトリ作成テンプレートを作成して、プルスルーキャッシュアクションで Amazon ECR がユーザーに代わって作成したリポジトリに使用する設定を定義することができます。リポジトリ作成テンプレートが作成されると、プルスルーキャッシュアクション中に作成されたすべての新しいリポジトリに 設定が適用されます。これは以前に作成されたリポジトリには影響しません。

リポジトリ作成テンプレートを作成するには (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、リポジトリ作成テンプレートを作成するリージョンを選択します。
3. ナビゲーションペインで、[プライベートレジストリ]、[リポジトリ作成テンプレート] を選択します。
4. [リポジトリ作成テンプレート] ページで、[テンプレートを作成] を選択します。
5. [ステップ 1: テンプレートの定義] ページの [テンプレートの詳細] で、[特定のプレフィックス] を選択してテンプレートを特定のリポジトリ名前空間のプレフィックスに適用するか、[ECR レジストリの任意のプレフィックス] を選択して、リージョン内の他のテンプレートと一致しないすべてのリポジトリにテンプレートを適用します。

   a. [特定のプレフィックス] を選択した場合は、[プレフィックス] にテンプレートを適用するリポジトリ名前空間プレフィックスを指定します。プレフィックスの末尾には常に / が適用されると想定されます。例えば、`prod` というプレフィックスは、`prod/` で始まるすべてのリポジトリに適用されます。同様に、`prod/team` というプレフィックスは、`prod/team/` で始まるすべてのリポジトリに適用されます。

   b. [ECR レジストリの任意のプレフィックス] を選択すると、[プレフィックス] は `ROOT` に設定されます。
6. [テンプレートの説明] には、テンプレートの説明をオプションで指定し、[次へ] を選択します。
7. [ステップ 2: リポジトリ作成設定の追加] ページで、テンプレートを使用して作成されたリポジトリに適用するリポジトリ設定設定を指定します。

a. [イメージタグのミュータビリティ] で、このリポジトリのタグの変更可能性の設定を選択します。詳細については、「Amazon ECR でイメージタグが上書きされないようにする」を参照してください。

   ミュータブルを選択すると、イメージタグを上書きできます。これは、プルスルーキャッシュアクションによって作成されたリポジトリに使用されるテンプレートに使用することを推奨する設定です。これにより、タグが同じ場合でも Amazon ECR はキャッシュされたイメージを更新できます。

   イミュータブルを選択すると、イメージタグが上書きされるのを防ぐことができます。リポジトリをイミュータブルタグ用に設定した後、リポジトリに既に存在しているタグ付きイメージをプッシュする試行が実行されると、`ImageTagAlreadyExistsException` エラーが返されます。リポジトリでタグのイミュータビリティがオンになっている場合、これはすべてのタグに影響し、一部のタグをイミュータブルにすることはできません。

b. [暗号化設定] では、使用する暗号化設定を選択します。詳細については、「保管中の暗号化」を参照してください。

   [AES-256] が選択されていると、Amazon ECR は、Amazon Simple Storage Service が管理する暗号化キーによるサーバー側暗号化を使用し、保管中のデータが業界標準の AES-256 暗号化アルゴリズムで暗号化されます。これは追加コストなしで提供されます。

   AWS KMS を選択すると、Amazon ECR は AWS Key Management Service () に保存されているキーによるサーバー側の暗号化を使用しますAWS KMS。 AWS KMS を使用してデータを暗号化する場合、Amazon ECR によって管理されるデフォルトの AWS マネージドキーを使用するか、カスタマーマネージド AWS KMS キー と呼ばれる独自のキーを指定できます。

   > **Note**
   >
   > 一度リポジトリを作成すると、リポジトリの暗号化設定は変更できません。

c. [リポジトリ権限] については、このテンプレートを使用して作成されたリポジトリに適用するリポジトリ権限ポリシーを指定します。オプションで、ドロップダウンを使用して、最も一般的なユースケース用の JSON サンプルを 1 つ選択できます。詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。

d. [リポジトリライフサイクルポリシー] では、このテンプレートを使用して作成されたリポジトリに適用するリポジトリライフサイクルポリシーを指定します。オプションで、ドロップ

ダウンを使用して、最も一般的なユースケース用の JSON サンプルを 1 つ選択できます。詳細については、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。

e. リポジトリ AWS タグ では、メタデータをキーと値のペアの形式で指定し、このテンプレートを使用して作成されたリポジトリに関連付けてから、次へ を選択します。詳細については、「Amazon ECR でのプライベートリポジトリのタグ付け」を参照してください。

8. [ステップ 3: 確認と作成] ページで、リポジトリ作成テンプレートに指定した設定を確認します。[編集] を選択して、変更を加えます。完了したら、[作成] を選択します。

Amazon ECR でのリポジトリ作成テンプレートの削除

不要になったリポジトリ作成テンプレートを削除できます。リポジトリ作成テンプレートを削除すると、プルスルーキャッシュアクション中に作成されたリポジトリにはデフォルト設定が適用されます。

リポジトリ作成テンプレートを削除するには (AWS Management Console)

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、削除するリポジトリ作成テンプレートがあるリージョンを選択します。
3. ナビゲーションペインで、[プライベートレジストリ]、[リポジトリ作成テンプレート] を選択します。
4. [リポジトリ作成テンプレート] ページで、削除するリポジトリ作成テンプレートを選択します。
5. [アクション] ドロップダウンメニューから [削除] を選択します。

Amazon ECR でのプルスルーキャッシュルールの検証

プルスルーキャッシュルールを作成した後、認証を必要とするアップストリームレジストリでは、ルールが正しく動作することを検証できます。プルスルーキャッシュルールを検証する場合、Amazon ECR はアップストリームレジストリとのネットワーク接続を行い、アップストリームレジストリの認証情報を含む Secrets Manager シークレットにアクセスできることを確認し、認証が成功したことを確認します。

プルスルーキャッシュルールの使用を開始する前に、適切な IAM アクセス許可があることを確認してください。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期するために必要な IAM アクセス許可」を参照してください。

プルスルーキャッシュルールを検証するには (AWS Management Console)

以下のステップは、Amazon ECR コンソールを使用してプルスルーキャッシュルールを検証する方法を示しています。

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、検証するプルスルーキャッシュルールが含まれているリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ)、[Pull through cache] (プルスルーキャッシュ) の順に選択します。
4. [プルスルーキャッシュ設定] ページで、検証するプルスルーキャッシュルールを選択します。次に、[アクション] ドロップダウンメニューを使用して [詳細を表示] を選択します。
5. プルスルーキャッシュルールの詳細ページで、[アクション] ドロップダウンメニューを使用して [認証を確認] を選択します。Amazon ECR は結果を示すバナーを表示します。
6. 検証する各プルスルーキャッシュルールに対して、これらのステップを繰り返します。

プルスルーキャッシュルールを検証するには (AWS CLI)

validate-pull-through-cache-rule AWS CLI コマンドは、Amazon ECR プライベートレジストリのプルスルーキャッシュルールを検証するために使用されます。次の例では `ecr-public` 名前空間プレフィックスを使用しています。その値を、検証するプルスルーキャッシュルールのプレフィックス値に置き換えます。

```
aws ecr validate-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --region us-east-2
```

レスポンスで、`isValid` パラメータは検証が成功したかどうかを示します。`true` の場合、Amazon ECR はアップストリームレジストリにアクセスでき、認証は成功しました。`false` の場合、問題が発生し、検証が失敗しました。`failure` パラメータは原因を示します。

Amazon ECR のプルスルーキャッシュルールを使用してイメージをプルする

以下の例は、プルスルーキャッシュルールを使用してイメージをプルするときに使用するコマンド構文を示しています。プルスルーキャッシュルールを使用してアップストリームイメージをプルする際

にエラーが発生した場合は、Amazon ECR でのプルスルーキャッシュの問題のトラブルシューティング を参照してください。最も一般的なエラーと、それらを解決する方法が記載されています。

プルスルーキャッシュルールの使用を開始する前に、適切な IAM アクセス許可があることを確認してください。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期するために必要な IAM アクセス許可」を参照してください。

> **Note**
>
> 次の例では、 が使用するデフォルトの Amazon ECR リポジトリ名前空間値 AWS Management Console を使用します。設定した Amazon ECR プライベートリポジトリ URI を使用していることを確認します。

Amazon ECR Public の場合

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/
image_name:tag
```

Kubernetes コンテナレジストリ

```
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

Docker Hub の公式イメージの場合:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

> **Note**
>
> Docker Hub 公式イメージの場合は、/library プレフィックスを含める必要があります。その他すべての Docker Hub リポジトリでは、/library プレフィックスを省略する必要があります。

その他すべての Docker Hub イメージの場合:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

GitHub コンテナレジストリ

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/
image_name:tag
```

Microsoft Azure コンテナレジストリ

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/
image_name:tag
```

GitLab コンテナレジストリ

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

シーク AWS Secrets Manager レットへのアップストリームリポジトリ認証情報の保存

認証を必要とするアップストリームリポジトリのプルスルーキャッシュルールを作成する場合、認証情報を Secrets Manager シークレットに保存する必要があります。Secrets Manager シークレットを使用するとコストがかかる場合があります。詳細については、「AWS Secrets Manager 料金表」を参照してください。

以下の手順では、サポートされている各アップストリームリポジトリの Secrets Manager シークレットを作成する方法を示します。Secrets Manager コンソールを使用してシークレットを作成する

代わりに、オプションで Amazon ECR コンソールのプルスルーキャッシュルール作成ワークフローを使用してシークレットを作成できます。詳細については、「Amazon ECR でのプルスルーキャッシュルールの作成」を参照してください。

Docker Hub

Docker Hub 認証情報の Secrets Manager シークレットを作成するには (AWS Management Console)

1. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [シークレットのタイプを選択] ページで、次の操作を行います。
   a. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
   b. [キー/値のペア] では、Docker Hub 認証情報用に 2 行を作成します。シークレットには最大 65536 バイトまで保存できます。
      i. 最初のキーと値のペアでは、`username` をキーとして指定し、値として Docker Hub のユーザー名を指定します。
      ii. 2 番目のキーと値のペアでは、`accessToken` をキーとして指定し、値として Docker Hub のアクセストークンを指定します。Docker Hub アクセストークンの作成について詳しくは、Docker ドキュメントの「Create and manage access tokens」を参照してください。
   c. [暗号化キー] にはデフォルト aws/secretsmanager AWS KMS key 値を選択した状態で、[次へ] を選択します。このキーを使用してもコストは発生しません。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager でのシークレットの暗号化と復号化」を参照してください。

   > **Important**
   >
   > シークレットを暗号化するには、デフォルトの `aws/secretsmanager` 暗号化キーを使用する必要があります。Amazon ECR では、このためのカスタマーマネージドキー (CMK) の使用はサポートされていません。

4. [シークレットを設定] ページで、次の操作を行います。

a. わかりやすいシークレット名と説明を入力します。シークレット名は 1～512 文字の Unicode 文字を含み、`ecr-pullthroughcache/` をプレフィックスとする必要があります。

> **Important**
>
> Amazon ECR には、 `ecr-pullthroughcache/` プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみが表示されます。

b. (オプション) [Tags] (タグ) セクションで、タグをシークレットに追加します。タグ付け方法については、AWS Secrets Manager ユーザーガイドの「Tag Secrets Manager シークレット」を参照してください。機密情報は暗号化されていないため、タグに保存しないでください。

c. (オプション) [Resource permissions] (リソースに対するアクセス許可) でリソースポリシーをシークレットに追加するには、[Edit permissions] (アクセス許可の編集) をクリックします。詳細については、AWS Secrets Manager ユーザーガイドの「アクセス許可ポリシーを Secrets Manager シークレットにアタッチする」を参照してください。

d. (オプション) シークレットのレプリケート で、シークレットを別の にレプリケートするには AWS リージョン、シークレットのレプリケート を選択します。シークレットのレプリケーションは、この段階で実行することも、後に戻ってきて実行することもできます。詳細については、AWS Secrets Manager ユーザーガイドの「シークレットを他のリージョンにレプリケートする」を参照してください。

e. [次へ] をクリックします。

5. (オプション) [Configure rotation] (ローテーションを設定する) ページで、シークレットの自動ローテーションを有効にできます。ローテーションをオフにしておいて、後でオンにすることもできます。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager シークレットのローテーション」を参照してください。[Next] を選択します。

6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

   Secrets Manager はシークレットのリストに戻ります。新しいシークレットが表示されない場合は、更新ボタンを選択します。

GitHub Container Registry

GitHub コンテナレジストリ認証情報の Secrets Manager シークレットを作成するには (AWS Management Console)

1. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [シークレットのタイプを選択] ページで、次の操作を行います。
   a. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。
   b. キーと値のペア で、 GitHub 認証情報の 2 行を作成します。シークレットには最大 65536 バイトまで保存できます。
      i. 最初のキーと値のペアでは、キー`username`として を指定し GitHub、値としてユーザー名を指定します。
      ii. 2 番目のキーと値のペアでは、キー`accessToken`として を指定し、値として GitHub アクセストークンを指定します。 GitHub アクセストークンの作成の詳細については、 GitHub ドキュメントの「個人用アクセストークンの管理」を参照してください。
   c. [暗号化キー] にはデフォルト aws/secretsmanager AWS KMS key 値を選択した状態で、[次へ] を選択します。このキーを使用してもコストは発生しません。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager でのシークレットの暗号化と復号化」を参照してください。

      > **Important**
      > シークレットを暗号化するには、デフォルトの `aws/secretsmanager` 暗号化キーを使用する必要があります。Amazon ECR では、このためのカスタマーマネージドキー (CMK) の使用はサポートされていません。

4. [Configure secret] (シークレットを設定する) ページで、次の操作を行います。
   a. わかりやすいシークレット名と説明を入力します。シークレット名は 1～512 文字の Unicode 文字を含み、`ecr-pullthroughcache/` をプレフィックスとする必要があります。

> **Important**
>
> Amazon ECR には、 `ecr-pullthroughcache/` プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみが表示されます。

b. (オプション) [Tags] (タグ) セクションで、タグをシークレットに追加します。タグ付け方法については、AWS Secrets Manager ユーザーガイドの「Tag Secrets Manager シークレット」を参照してください。機密情報は暗号化されていないため、タグに保存しないでください。

c. (オプション) [Resource permissions] (リソースに対するアクセス許可) でリソースポリシーをシークレットに追加するには、[Edit permissions] (アクセス許可の編集) をクリックします。詳細については、AWS Secrets Manager ユーザーガイドの「アクセス許可ポリシーを Secrets Manager シークレットにアタッチする」を参照してください。

d. (オプション) シークレットのレプリケート で、シークレットを別の にレプリケートするには AWS リージョン、シークレットのレプリケート を選択します。シークレットのレプリケーションは、この段階で実行することも、後に戻ってきて実行することもできます。詳細については、AWS Secrets Manager ユーザーガイドの「シークレットを他のリージョンにレプリケートする」を参照してください。

e. [次へ] をクリックします。

5. (オプション) [Configure rotation] (ローテーションを設定する) ページで、シークレットの自動ローテーションを有効にできます。ローテーションをオフにしておいて、後でオンにすることもできます。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager シークレットのローテーション」を参照してください。[Next] を選択します。

6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

   Secrets Manager はシークレットのリストに戻ります。新しいシークレットが表示されない場合は、更新ボタンを選択します。

Microsoft Azure Container Registry

Microsoft Azure コンテナレジストリの認証情報用の Secrets Manager シークレットを作成するには (AWS Management Console)

1. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。

2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [シークレットのタイプを選択] ページで、次の操作を行います。

   a. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。

   b. [キー/値のペア] では、Microsoft Azure 認証情報用に 2 行を作成します。シークレットには最大 65536 バイトまで保存できます。

      i. 最初のキーと値のペアでは、`username` をキーとして指定し、値として Microsoft Azure コンテナレジストリのユーザー名を指定します。

      ii. 2 番目のキーと値のペアでは、`accessToken` をキーとして指定し、値として Microsoft Azure コンテナレジストリのアクセストークンを指定します。Microsoft Azure アクセストークンの作成について詳しくは、Microsoft Azure ドキュメントの「トークンを作成する - ポータル」を参照してください。

   c. [暗号化キー] にはデフォルト aws/secretsmanager AWS KMS key 値を選択した状態で、[次へ] を選択します。このキーを使用してもコストは発生しません。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager でのシークレットの暗号化と復号化」を参照してください。

   > **Important**
   >
   > シークレットを暗号化するには、デフォルトの `aws/secretsmanager` 暗号化キーを使用する必要があります。Amazon ECR では、このためのカスタマーマネージドキー (CMK) の使用はサポートされていません。

4. [Configure secret] (シークレットを設定する) ページで、次の操作を行います。

   a. わかりやすいシークレット名と説明を入力します。シークレット名は 1～512 文字の Unicode 文字を含み、`ecr-pullthroughcache/` をプレフィックスとする必要があります。

   > **Important**
   >
   > Amazon ECR には、 `ecr-pullthroughcache/` プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみが表示されます。

b. (オプション) [Tags] (タグ) セクションで、タグをシークレットに追加します。タグ付け方法については、AWS Secrets Manager ユーザーガイドの「Tag Secrets Manager シークレット」を参照してください。機密情報は暗号化されていないため、タグに保存しないでください。

c. (オプション) [Resource permissions] (リソースに対するアクセス許可) でリソースポリシーをシークレットに追加するには、[Edit permissions] (アクセス許可の編集) をクリックします。詳細については、AWS Secrets Manager ユーザーガイドの「アクセス許可ポリシーを Secrets Manager シークレットにアタッチする」を参照してください。

d. (オプション) シークレットのレプリケート で、シークレットを別の にレプリケートするには AWS リージョン、シークレットのレプリケート を選択します。シークレットのレプリケーションは、この段階で実行することも、後に戻ってきて実行することもできます。詳細については、AWS Secrets Manager ユーザーガイドの「シークレットを他のリージョンにレプリケートする」を参照してください。

e. [次へ] をクリックします。

5. (オプション) [Configure rotation] (ローテーションを設定する) ページで、シークレットの自動ローテーションを有効にできます。ローテーションをオフにしておいて、後でオンにすることもできます。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager シークレットのローテーション」を参照してください。[Next] を選択します。

6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

   Secrets Manager はシークレットのリストに戻ります。新しいシークレットが表示されない場合は、更新ボタンを選択します。

GitLab Container Registry

GitLab コンテナレジストリ認証情報の Secrets Manager シークレットを作成するには (AWS Management Console)

1. Secrets Manager のコンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。
2. [Store a new secret] (新しいシークレットを保存する) を選択します。
3. [シークレットのタイプを選択] ページで、次の操作を行います。

   a. [Secret type] (シークレットタイプ) で、[Other type of secret] (他の種類のシークレット) を選択します。

b. キーと値のペア で、 GitLab 認証情報用に 2 行を作成します。シークレットには最大 65536 バイトまで保存できます。

   i. 最初のキーと値のペアでは、キーusernameとして を指定し、値として GitLab コンテナレジストリのユーザー名を指定します。

   ii. 2 番目のキーと値のペアでは、キーaccessTokenとして を指定し、値として GitLab コンテナレジストリアクセストークンを指定します。 GitLab コンテナレジストリアクセストークンの作成の詳細については、 GitLab ドキュメントの「個人用アクセストークン」、「グループアクセストークン」、または「プロジェクトアクセストークン」を参照してください。

c. [暗号化キー] にはデフォルト aws/secretsmanager AWS KMS key 値を選択した状態で、[次へ] を選択します。このキーを使用してもコストは発生しません。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager でのシークレットの暗号化と復号化」を参照してください。

> **Important**
>
> シークレットを暗号化するには、デフォルトの aws/secretsmanager 暗号化キーを使用する必要があります。Amazon ECR では、このためのカスタマーマネージドキー (CMK) の使用はサポートされていません。

4. [Configure secret] (シークレットを設定する) ページで、次の操作を行います。

a. わかりやすいシークレット名と説明を入力します。シークレット名は 1～512 文字の Unicode 文字を含み、ecr-pullthroughcache/ をプレフィックスとする必要があります。

> **Important**
>
> Amazon ECR には、 ecr-pullthroughcache/ プレフィックスを使用する名前の Secrets Manager シークレット AWS Management Console のみが表示されます。

b. (オプション) [Tags] (タグ) セクションで、タグをシークレットに追加します。タグ付け方法については、AWS Secrets Manager ユーザーガイドの「Tag Secrets Manager シークレット」を参照してください。機密情報は暗号化されていないため、タグに保存しないでください。

c. (オプション) [Resource permissions] (リソースに対するアクセス許可) でリソースポリシーをシークレットに追加するには、[Edit permissions] (アクセス許可の編集) をクリックします。詳細については、AWS Secrets Manager ユーザーガイドの「アクセス許可ポリシーを Secrets Manager シークレットにアタッチする」を参照してください。

d. (オプション) シークレットのレプリケート で、シークレットを別の にレプリケートするには AWS リージョン、シークレットのレプリケート を選択します。シークレットのレプリケーションは、この段階で実行することも、後に戻ってきて実行することもできます。詳細については、AWS Secrets Manager ユーザーガイドの「シークレットを他のリージョンにレプリケートする」を参照してください。

e. [次へ] をクリックします。

5. (オプション) [Configure rotation] (ローテーションを設定する) ページで、シークレットの自動ローテーションを有効にできます。ローテーションをオフにしておいて、後でオンにすることもできます。詳細については、AWS Secrets Manager ユーザーガイドの「Secrets Manager シークレットのローテーション」を参照してください。[Next] を選択します。

6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

   Secrets Manager はシークレットのリストに戻ります。新しいシークレットが表示されない場合は、更新ボタンを選択します。

Amazon ECR でのプルスルーキャッシュの問題のトラブルシューティング

プルスルーキャッシュルールを使用してアップストリームイメージをプルする際に、最もよく発生する可能性のある一般的なエラーは次のとおりです。

リポジトリが存在しません

リポジトリが存在しないことを示すエラーは、ほとんどの場合、Amazon ECR プライベートレジストリにリポジトリが存在しないか、アップストリームイメージをプルする IAM プリンシパルに `ecr:CreateRepository` アクセス許可が付与されていないのが原因です。このエラーを解決するには、プルコマンドのリポジトリ URI が正しいこと、必要となる IAM アクセス許可がアップストリームイメージをプルする IAM プリンシパルに付与されていること、またはアップストリームのイメージをプルする前に、プッシュされるアップストリームイメージのリポジトリが Amazon ECR プライベートレジストリに作成されていることを確認します。必要となる IAM アクセス許可の詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期するために必要な IAM アクセス許可」を参照してください。

このエラーの例を以下に示します。

```
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'
```

リクエストされたイメージが見つかりません

イメージが見つからないことを示すエラーは、ほとんどの場合、アップストリームレジストリにイメージが存在しないか、アップストリームイメージをプルする IAM プリンシパルに `ecr:BatchImportUpstreamImage` アクセス許可が付与されておらず、Amazon ECR プライベートレジストリ内にレポジトリがすでに作成されているのが原因です。このエラーを解決するには、アップストリームイメージとイメージタグ名が正しく、それが存在し、アップストリームイメージをプルする IAM プリンシパルに必要な IAM アクセス許可が付与されていることを確認する必要があります。必要となる IAM アクセス許可の詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期するために必要な IAM アクセス許可」を参照してください。

このエラーの例を以下に示します。

```
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
```

Docker Hub リポジトリからプルする場合の 403 Forbidden

Docker Official Image としてタグ付けされている Docker Hub リポジトリからプルする場合は、使用する URI に `/library/` を含める必要があります。例えば *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/docker-hub/`**`library/`***`image_name:tag`* です。Docker Hub Official イメージの `/library/` を省略した場合、プルスルーキャッシュルールを使用してイメージをプルしようとすると `403 Forbidden` エラーが返されます。詳細については、「Amazon ECR のプルスルーキャッシュルールを使用してイメージをプルする」を参照してください。

このエラーの例を以下に示します。

```
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
```

Amazon ECR でのプライベートイメージのレプリケーション

リポジトリのレプリケーションをサポートするために、Amazon ECR プライベートレジストリを構成できます。Amazon ECR は、クロスリージョンレプリケーションおよびクロスアカウントレプリケーションの両方をサポートしています。クロスアカウントレプリケーションを実行するには、ターゲットアカウントで、ソースレジストリからのレプリケーションを許可するレジストリのアクセス許可ポリシーを設定する必要があります。詳しくは、「Amazon ECR でのプライベートレジストリのアクセス許可」を参照してください。

トピック

- プライベートイメージのレプリケーションに関する考慮事項
- Amazon ECR のプライベートイメージレプリケーションの例
- Amazon ECR でのプライベートイメージレプリケーションの設定

プライベートイメージのレプリケーションに関する考慮事項

プライベートイメージのレプリケーションを使用する際には、以下の点を考慮する必要があります。

- 複製されるのは、レプリケーションの構成後にリポジトリにプッシュされたリポジトリコンテンツのみです。リポジトリ内の既存のコンテンツはいずれも複製されません。リポジトリにレプリケーションが構成されると、Amazon ECR は宛先とソースの同期を継続します。
- レプリケーションが行われても、リポジトリ名はリージョンとアカウント間で同じままです。Amazon ECR はレプリケーション中のリポジトリ名の変更をサポートしていません。
- レプリケーション用にプライベートレジストリを初めて設定すると、Amazon ECR は、お客様に代わってサービスにリンクされた IAM ロールを作成します。サービスにリンクされた IAM ロールは、レジストリでレポジトリの作成とイメージのレプリケートを行うために必要なアクセス許可を Amazon ECR レプリケーションサービスに付与します。詳細については、「Amazon ECR でのサービスにリンクされたロールの使用」を参照してください。
- クロスアカウントレプリケーションを実行するには、プライベートレジストリのレプリケート先が、ソースレジストリによるそのイメージのレプリケートを許可する必要があります。これは、プライベートレジストリの許可ポリシーを設定することによって許可できます。詳細については、「Amazon ECR でのプライベートレジストリのアクセス許可」を参照してください。

- プライベートレジストリの許可ポリシーが許可を取り消すように変更された場合でも、以前に許可された進行中のレプリケーションは完了することができます。
- クロスリージョンレプリケーションを行うには、そのリージョン内またはリージョンに対してレプリケーションアクションが発生する前に、ソースアカウントとターゲットアカウントの両方がリージョンにオプトインされている必要があります。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS リージョンの管理」を参照してください。
- クロスリージョンレプリケーションは、 AWS パーティション間ではサポートされていません。たとえば、us-west-2 のリポジトリは cn-north-1 にレプリケートできません。 AWS パーティションの詳細については、「 AWS 全般のリファレンス」の「ARN 形式」を参照してください。
- プライベートレジストリのレプリケーション設定には、すべてのルール (合計で最大 10 個) 全体で最大 25 個の一意のレプリケート先を含めることができます。各ルールには、最大 100 個のフィルターを含めることができます。これは、例えば本番環境やテスト用に使用されるイメージが含まれるリポジトリに対して個別のルールを指定することを可能にします。
- レプリケーション設定は、リポジトリプレフィックスを指定することによって、プライベートレジストリ内のレプリケートされるリポジトリのフィルタリングをサポートします。例については、「例: リポジトリフィルターを使用したクロスリージョンレプリケーションの設定」を参照してください。
- レプリケーションアクションは、イメージプッシュごとに 1 回のみ実行されます。たとえば、us-west-2 から us-east-1 および us-east-1 から us-east-2 へのクロスリージョンレプリケーションを設定した場合、us-west-2 にプッシュされたイメージは us-east-1 にのみレプリケートされ、us-east-2 には再度レプリk-トされません。この動作は、クロスリージョンおよびクロスアカウントのレプリケーションの両方に適用されます。
- 大部分のイメージは 30 分以内にレプリケートされますが、まれにレプリケーションに時間がかかることがあります。
- レジストリレプリケーションでは、削除アクションは実行されません。レプリケートされたイメージとリポジトリは、使用しなくなったときに手動で削除できます。
- IAM ポリシーやライフサイクルポリシーなどのリポジトリポリシーはレプリケートされず、定義されているリポジトリ以外には影響しません。
- リポジトリ設定はレプリケートされません。タグのイミュータブル性、イメージスキャニング、および KMS 暗号化の設定は、レプリケーションアクションに起因して作成されたすべてのリポジトリでデフォルトで無効になっています。タグのイミュータブル性とイメージスキャニングの設定は、リポジトリの作成後に変更できます。ただし、この設定は、設定が変更された後にプッシュされたイメージにのみ適用されます。

- リポジトリでタグのイミュータブル性が有効で、既存のイメージと同じタグを使用するイメージをレプリケートする場合、イメージはレプリケートされますが、重複したタグは含まれません。その結果、イメージのタグ付けが解除される可能性があります。

Amazon ECR のプライベートイメージレプリケーションの例

以下の例は、プライベートのイメージレプリケーションの一般的な使用事例を示しています。を使用してレプリケーションを設定する場合は AWS CLI、JSON ファイルを作成する際の出発点として JSON の例を使用できます。を使用してレプリケーションを設定すると AWS Management Console、レビューと送信ページでレプリケーションルールを確認すると、同様の JSON が表示されます。

例: 単一の送信先リージョンへのクロスリージョンレプリケーションの設定

次に、単一のレジストリ内でのクロスリージョンレプリケーション設定の例を示します。この例では、アカウント ID が 111122223333 であること、およびこのレプリケーション設定が us-west-2 以外のリージョンで指定されていることを前提とします。

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

例: リポジトリフィルターを使用したクロスリージョンレプリケーションの設定

以下は、プレフィックス名の値と一致するリポジトリのクロスリージョンレプリケーションを設定する例です。この例は、アカウント ID が 111122223333 であること、us-west-1 以外のリージョンでこのレプリケーション設定を指定していること、およびプレフィックスが prod のリポジトリがあることを前提としています。

```
{
 "rules": [{
  "destinations": [{
   "region": "us-west-1",
   "registryId": "111122223333"
  }],
  "repositoryFilters": [{
   "filter": "prod",
   "filterType": "PREFIX_MATCH"
  }]
 }]
}
```

例: 複数の送信先リージョンへのクロスリージョンレプリケーションの設定

次に、単一のレジストリ内でのクロスリージョンレプリケーション設定の例を示します。この例では、アカウント ID が 111122223333 であること、およびこのレプリケーション設定が us-west-1 および us-west-2 以外のリージョンで指定されていることを前提とします。

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-1",
                    "registryId": "111122223333"
                },
                {
                    "region": "us-west-2",
                    "registryId": "111122223333"
                }
            ]
        }
    ]
}
```

例: クロスアカウントレプリケーションの設定

次に、レジストリのクロスアカウントレプリケーション設定の例を示します。この例では、444455556666 アカウントと us-west-2 リージョンも対するレプリケーションを設定します。

> **Important**
>
> クロスアカウントレプリケーションを実行するには、ターゲットアカウントで、レプリケーションを許可するレジストリのアクセス許可ポリシーを設定する必要があります。詳細については、「Amazon ECR でのプライベートレジストリのアクセス許可」を参照してください。

```
{
    "rules": [
        {
            "destinations": [
                {
                    "region": "us-west-2",
                    "registryId": "444455556666"
                }
            ]
        }
    ]
}
```

例: 1 つの設定内での複数のルールの指定

以下は、レジストリに複数のレプリケーションルールを設定する例です。この例は、プレフィックスが prod のリポジトリを us-west-2 リージョンにレプリケートし、プレフィックスが test のリポジトリを us-east-2 リージョンにレプリケートする 1 つのルールで *111122223333* アカウントのレプリケーションを設定します。レプリケーション設定には最大 10 個のルールを含めることができ、各ルールが最大 25 個の送信先を指定します。

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

Amazon ECR でのプライベートイメージレプリケーションの設定

プライベートレジストリのリージョンごとにレプリケーションを設定します。クロスリージョンレプリケーションまたはクロスアカウントレプリケーションを設定できます。

レプリケーションの一般的な使用方法の例については、Amazon ECR のプライベートイメージレプリケーションの例 を参照してください。

レジストリのレプリケーション設定を構成するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、レジストリのレプリケーション設定を構成するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ) を選択します。
4. [Private registry] (プライベートレジストリ) ページの [Replication] (レプリケーション) セクションで、[Edit] (編集) をクリックします。
5. [Replication] (レプリケーション) ページで [Add replication rule] (レプリケーションルールを追加) をクリックします。
6. [Destination types] (送信先タイプ) ページで、クロスリージョンレプリケーションもしくはクロスアカウントレプリケーションのいずれか、またはそれら両方を有効にしてから、[Next] (次へ) をクリックします。
7. クロスリージョンレプリケーションを有効にした場合は、[Configure destination regions] (送信先リージョンを設定) で、1 つ、または複数の [Destination regions] (送信先リージョン) を選択し、[Next] (次へ) をクリックします。

8. クロスアカウントレプリケーションを有効にした場合は、[Cross-account replication] (クロスアカウントレプリケーション) で、レジストリのクロスアカウントレプリケーション設定を選択します。[Destination account] (送信先アカウント) には送信先アカウントのアカウント ID を入力し、レプリケート先にする 1 つ、または複数の [Destination regions] (送信先リージョン) を選択します。レプリケーションの送信先として追加のアカウントを設定するには、[Destination account +] (送信先アカウント +) をクリックします。

   > **Important**
   >
   > クロスアカウントレプリケーションを実行するには、ターゲットアカウントで、レプリケーションを許可するレジストリのアクセス許可ポリシーを設定する必要があります。詳細については、「Amazon ECR でのプライベートレジストリのアクセス許可」を参照してください。

9. (オプション) [Add filters] (フィルターを追加) ページで、レプリケーションルールに対して 1 つ、または複数のフィルターを指定してから、[Add] (追加) をクリックします。レプリケーションアクションに関連付けるフィルターごとに、このステップを繰り返します。フィルターはリポジトリ名のプレフィックスとして指定する必要があります。フィルターを追加しない場合、すべてのリポジトリの内容が複製されます。[Next] (次へ) をクリックすると、すべてのフィルターが追加されます。

10. [Review and submit] (確認して送信) ページでレプリケーションルールの設定を確認してから、[Submit rule] (ルールを送信) をクリックします。

レジストリのレプリケーション設定を構成するには (AWS CLI)

1. レジストリ用に定義するレプリケーションルールが含まれた JSON ファイルを作成します。レプリケーション設定には最大 10 個のルールを含めることができ、各ルールが最大 25 個の送信先と 100 個のフィルターを指定します。独自のアカウント内でクロスリージョンレプリケーションを設定するには、独自のアカウント ID を指定します。その他の例については、「Amazon ECR のプライベートイメージレプリケーションの例」を参照してください。

```
{
 "rules": [{
  "destinations": [{
   "region": "destination_region",
   "registryId": "destination_accountId"
  }],
  "repositoryFilters": [{
```

```
    "filter": "repository_prefix_name",
    "filterType": "PREFIX_MATCH"
   }]
  }]
}
```

2. レジストリのレプリケーション設定を作成します。

```
aws ecr put-replication-configuration \
     --replication-configuration file://replication-settings.json \
     --region us-west-2
```

3. レジストリ設定を確認します。

```
aws ecr describe-registry \
     --region us-west-2
```

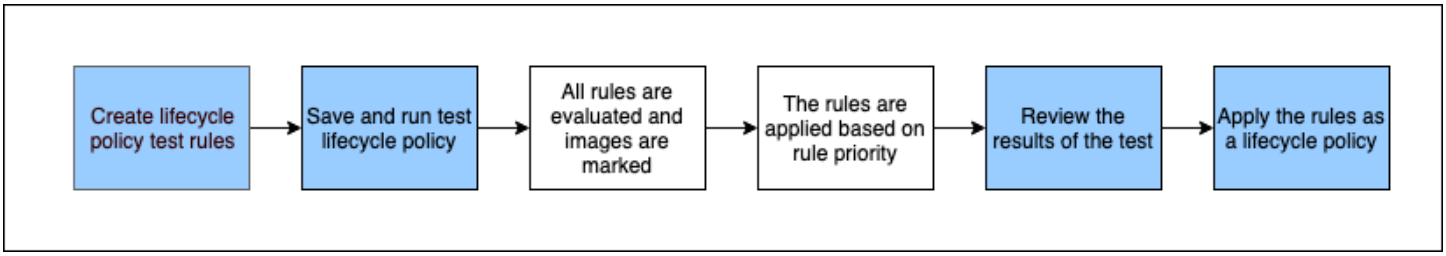
Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する

Amazon ECR ライフサイクルポリシーを使用すると、プライベートリポジトリ内のイメージのライフサイクル管理をより詳細に制御できます。ライフサイクルポリシーには 1 つ以上のルールが含まれており、各ルールは Amazon ECR のアクションを定義します。ライフサイクルポリシーの有効期限条件に基づいて、イメージは経過時間または 24 時間以内の数に基づいて期限切れになります。Amazon ECR がライフサイクルポリシーに基づいてアクションを実行すると、このアクションは のイベントとしてキャプチャされます AWS CloudTrail。詳細については、「を使用した Amazon ECR アクションのログ記録 AWS CloudTrail」を参照してください。

ライフサイクルポリシーの機能

ライフサイクルポリシーは、リポジトリ内のどのイメージに有効期限切れにするかを決定する 1 つ以上のルールで構成されています。ライフサイクルポリシーの使用を検討する場合、リポジトリに適用する前に、ライフサイクルポリシーのプレビューでライフサイクルポリシーが有効期限切れになったイメージを確認することが重要です。ライフサイクルポリシーがリポジトリに適用されると、イメージが有効期限の基準を満たしてから 24 時間以内に期限切れになることが予想されます。Amazon ECR がライフサイクルポリシーに基づいてアクションを実行すると、これはイベントとして AWS CloudTrailにキャプチャされます。詳細については、「を使用した Amazon ECR アクションのログ記録 AWS CloudTrail」を参照してください。

次の図表は、ライフサイクルポリシーのワークフローを示します。



1. テストルールを 1 つ以上作成します。
2. テストルールを保存し、プレビューを実行します。
3. ライフサイクルポリシーエバリュエーターは、すべてのルールを評価し、各ルールが影響するイメージにマークを付けます。
4. 次に、ライフサイクルポリシーエバリュエーターは、ルールの優先度に基づいてルールを適用し、リポジトリ内で有効期限切れに設定されるイメージを表示します。

5. テストの結果を確認し、有効期限切れとマークされたイメージが意図したとおりのものであることを確認します。
6. テストルールをリポジトリのライフサイクルポリシーとして適用します。
7. ライフサイクル ポリシーが作成されると、イメージが有効期限の基準を満たしてから 24 時間以内に期限切れになることが予想されます。

ライフサイクルポリシーの評価ルール

ライフサイクルポリシーエバリュエーターは、ライフサイクルポリシーのプレーンテキスト JSON を解析して、すべてのルールを評価し、ルールの優先順位に基づいてリポジトリ内のイメージに適用します。次に、ライフサイクルポリシーエバリュエーターのロジックについて詳しく説明します。例については、「Amazon ECR のライフサイクルポリシーの例」を参照してください。

- 優先順位に関係なく、すべてのルールが同時に評価されます。すべてのルールが評価された後、優先度に基づいてルールが適用されます。
- イメージは 1 または ゼロのルールで正確に期限切れとなります。
- ルールのタグ付け要件に一致するイメージは、優先度がより低いルールによって期限切れにはなりません。
- ルールが、より優先度の高いルールでマークされたイメージをマークすることはありませんが、期限切れになっていないかのように識別されることがあります。
- 一連のルールには、一意の一連のタグプレフィックスを含める必要があります。
- タグが付いていないイメージを選択できるのは 1 つのルールのみです。
- 画像がマニフェストリストによって参照されている場合、初めにマニフェストリストを削除しないと有効期限切れになりません。
- 期限切れは常に pushed_at_time の順に並べられ、より古いイメージが新しいものよりも先に期限切れとなります。
- ライフサイクルポリシールールでは、tagPatternList または tagPrefixList のいずれかを指定できますが、両方は指定できません。ただし、ライフサイクルポリシーにはパターンリストとプレフィックスリストの両方を使用する複数のルールが含まれる場合があります。
- tagPatternList または tagPrefixList パラメータは、tagStatus が tagged の場合にのみ使用できます。
- tagPatternList を使用する場合、ワイルドカードフィルターに一致すればイメージがマークされます。例えば、prod* のフィルターを適用すると、prod、prod1 または production-team1 など、prod で始まる名前のリポジトリと一致します。同様に、*prod* のフィルターを適用す

ると、repo-production や prod-team など、名前に prod が含まれるリポジトリと一致します。

> **Important**
>
> 1 文字列あたりのワイルドカード (*) の上限は 4 つです。例えば、["test*1*2*3*4*5*6"] は有効ですが ["*test*1*2*3", "test*1*2*3*"] は無効です。

- tagPrefixList を使用すると、tagPrefixList 値のすべてのタグが、イメージのタグのいずれかに一致した場合、そのイメージがマークされます。
- countUnit パラメータは、countType が sinceImagePushed の場合のみ使用されます。
- countType = imageCountMoreThan では、イメージは期間の新しいものから始めて最も古いものへと pushed_at_time に基づいて順に並べられた後、指定したカウントより大きいイメージはすべて期限切れとなります。
- countType = sinceImagePushed では、countNumber に基づき、pushed_at_time が指定された日数より古いすべてのイメージは期限切れとなります。

Amazon ECR でのライフサイクルポリシーのプレビューの作成

ライフサイクルポリシーのプレビューを使用して、ライフサイクルポリシーを適用する前にイメージリポジトリへの影響を確認できます。ベストプラクティスとして、リポジトリにライフサイクルポリシーを適用する前に、プレビューを実行することをお勧めします。

> **Note**
>
> Amazon ECR レプリケーションを使用して異なるリージョンまたはアカウント間でリポジトリのコピーを作成する場合、ライフサイクルポリシーは、作成されたリージョンのリポジトリに対してのみアクションを実行できることに注意してください。したがって、レプリケーションをオンにしている場合は、リポジトリを複製する各リージョンとアカウントにライフサイクルポリシーを作成してください。

ライフサイクルポリシーのプレビューを作成するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。

2. ナビゲーションバーから、ライフサイクルポリシーのプレビューを実行するリポジトリを含むリージョンを選択します。
3. ナビゲーションペインで、[プライベートレジストリ] の下で、[リポジトリ] を選択します。
4. [プライベートリポジトリ] ページでリポジトリを選択し、[アクション] ドロップダウンを使用して [ライフサイクルポリシー] を選択します。
5. リポジトリのライフサイクルポリシーページで、[テストルールの編集]、[ルールの作成] の順に選択します。
6. ライフサイクルポリシーのルールに以下の詳細を指定します。

   a. [ルールの優先順位] で、ルールの優先順位を数字で入力します。ルールの優先順位によって、ライフサイクルポリシールールが適用される順序が決まります。

   b. [ルールの説明] で、ライフサイクルポリシーのルールの説明を入力します。

   c. [イメージのステータス] では、[タグ付き (ワイルドカードマッチング)]、[タグ付き (プレフィックスマッチング)]、[タグ付けなし]、または [任意] を選択します。

   d. [イメージのステータス] で [タグ付き (ワイルドカードマッチング)] を選択した場合、[ワイルドカードマッチング用のタグを指定] では、ライフサイクルポリシーに基づいてアクションを実行するためのワイルドカード (*) 付きのイメージタグのリストを指定できます。例えば、イメージに `prod`、`prod1`、`prod2` というようにタグが付いている場合、すべてのアクションを実行する `prod*` を指定します。複数のタグを指定する場合、指定されたすべてのタグが付いているイメージのみが選択されます。

   > **Important**
   >
   > 1 文字列あたりのワイルドカード (*) の上限は 4 つです。例えば、`["test*1*2*3*4*5*6"]` は有効ですが `["*test*1*2*3", "test*1*2*3*"]` は無効です。

   e. [イメージのステータス] で [タグ付き (プレフィックスマッチング)] を選択した場合、[ワイルドカードマッチング用のタグを指定] では、ライフサイクルポリシーに基づいてアクションを実行するためのイメージタグのリストを指定できます。

   f. [一致条件] で、[イメージをプッシュしてから] または [次の数値を超えるイメージ数] を選択し、値を指定します。

   g. [保存] を選択します。

7. ステップ 5 ~ 7 を繰り返すことにより、追加のライフサイクルポリシーのルールを作成します。

8. ライフサイクルポリシーのプレビューを実行するには、[テストの保存と実行] を選択します。
9. [イメージがテストライフサイクルルールに一致しました] で、ライフサイクルポリシーのプレビューの影響を確認します。
10. プレビューの結果に満足したら、[Apply as lifecycle policy] を選択して指定したルールでライフサイクルポリシーを作成します。ライフサイクルポリシーの適用すると、影響を受けるイメージが 24 時間以内に有効期限切れになります。
11. プレビュー結果に満足できない場合は、1 つ以上のテストライフサイクルルールを削除し、1 つ以上のルールを作成して置き換え、テストを繰り返すことができます。

Amazon ECR でのリポジトリのライフサイクルポリシーの作成

ライフサイクルポリシーを使用して、未使用のリポジトリイメージを期限切れにする一連のルールを作成します。ライフサイクルポリシーを作成すると、影響を受けるイメージは 24 時間以内に期限切れになります。

> **Note**
>
> Amazon ECR レプリケーションを使用して異なるリージョンまたはアカウント間でリポジトリのコピーを作成する場合、ライフサイクルポリシーは、作成されたリージョンのリポジトリに対してのみアクションを実行できることに注意してください。したがって、レプリケーションをオンにしている場合は、リポジトリを複製する各リージョンとアカウントにライフサイクルポリシーを作成してください。

前提条件

ベストプラクティス： ライフサイクルポリシーのプレビューを作成して、ライフサイクルポリシールールによって期限切れになったイメージが意図したとおりであることを確認します。手順については、「Amazon ECR でのライフサイクルポリシーのプレビューの作成」を参照してください。

ライフサイクルポリシーを作成するには (AWS Management Console)

1. https://console.aws.amazon.com/ecr/repositories で Amazon ECR コンソールを開きます。
2. ナビゲーションバーから、ライフサイクルポリシーを作成するリポジトリを含むリージョンを選択します。
3. ナビゲーションペインで、[プライベートレジストリ] の下で、[リポジトリ] を選択します。

4. [プライベートリポジトリ] ページでリポジトリを選択し、[アクション] ドロップダウンを使用して [ライフサイクルポリシー] を選択します。
5. リポジトリのライフサイクルポリシーで、[ルールを作成] を選択します。
6. ライフサイクルポリシーのルールに以下の詳細を入力します。

   a. [ルールの優先順位] で、ルールの優先順位を数字で入力します。ルールの優先順位によって、ライフサイクルポリシールールが適用される順序が決まります。

   b. [ルールの説明] で、ライフサイクルポリシーのルールの説明を入力します。

   c. [イメージのステータス] では、[タグ付き (ワイルドカードマッチング)]、[タグ付き (プレフィックスマッチング)]、[タグ付けなし]、または [任意] を選択します。

   d. [イメージのステータス] で [タグ付き (ワイルドカードマッチング)] を選択した場合、[ワイルドカードマッチング用のタグを指定] では、ライフサイクルポリシーに基づいてアクションを実行するためのワイルドカード (*) 付きのイメージタグのリストを指定できます。例えば、イメージに `prod`、`prod1`、`prod2` というようにタグが付いている場合、すべてのアクションを実行する `prod*` を指定します。複数のタグを指定する場合、指定されたすべてのタグが付いているイメージのみが選択されます。

   > **Important**
   >
   > 1 文字列あたりのワイルドカード (*) の上限は 4 つです。例えば、`["test*1*2*3*4*5*6"]` は有効ですが `["*test*1*2*3", "test*1*2*3*"]` は無効です。

   e. [イメージのステータス] で [タグ付き (プレフィックスマッチング)] を選択した場合、[ワイルドカードマッチング用のタグを指定] では、ライフサイクルポリシーに基づいてアクションを実行するためのイメージタグのリストを指定できます。

   f. [一致条件] で、[イメージをプッシュしてから] または [次の数値を超えるイメージ数] を選択し、値を指定します。

   g. [保存] を選択します。

7. ステップ 5 ~ 7 を繰り返すことにより、追加のライフサイクルポリシーのルールを作成します。

ライフサイクルポリシーを作成するには (AWS CLI)

1. ライフサイクルポリシーを作成するリポジトリの名前を取得します。

```
aws ecr describe-repositories
```

2. ライフサイクルポリシーの内容を指定して、policy.json というローカルファイル名を作成します。ライフサイクルポリシーの例については、「Amazon ECR のライフサイクルポリシーの例」を参照してください。
3. リポジトリ名を指定してライフサイクルポリシーを作成し、作成したライフサイクルポリシー JSON ファイルを参照します。

```
aws ecr put-lifecycle-policy \
      --repository-name repository-name \
      --lifecycle-policy-text file://policy.json
```

Amazon ECR のライフサイクルポリシーの例

構文を示すライフサイクルポリシーの例を次に示します。

ポリシープロパティの詳細については、「」を参照してくださいAmazon ECR のライフサイクルポリシープロパティ。を使用してライフサイクルポリシーを作成する手順については、 AWS CLI「」を参照してくださいライフサイクルポリシーを作成するには (AWS CLI)。

ライフサイクルポリシーのテンプレート

ライフサイクルポリシーの内容は、リポジトリに関連付ける前に評価されます。以下に示しているのは、ライフサイクルポリシーの JSON 構文テンプレートです。

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

イメージの経過日数によるフィルタリング

次の例は、prod* の tagPatternList が同様に 14 日より古いものを使用することにより、prod で始まるタグでイメージを有効期限切れにするポリシーのライフサイクルポリシー構文を示しています。

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["prod*"],
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

イメージ数によるフィルタリング

次の例で、タグ付けされていないイメージを 1 つだけ保持して残りはすべて期限切れにするポリシーのライフサイクルポリシーの構文を示します。

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
```

```
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

複数のルールによるフィルタリング

以下は、ライフサイクルポリシーで複数のルールを使用する例です。サンプルのリポジトリとライフサイクルポリシーが結果の説明とともに示されています。

例 A

リポジトリのコンテンツ

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: 10 days ago
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: 9 days ago
- Image C, Taglist: ["beta-3"], Pushed: 8 days ago

ライフサイクルポリシーのテキスト

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

このライフサイクルポリシーのロジックは次のようになります。

- ルール 1 は、prod というプレフィックスでタグ付けされたイメージを特定します。最も古いイメージから始めて、一致するイメージが 1 つか数個になるまでマークし続けます。イメージ A が期限切れとしてマークされます。
- ルール 2 は、beta というプレフィックスでタグ付けされたイメージを特定します。最も古いイメージから始めて、一致するイメージが 1 つか数個になるまでマークし続けます。イメージ A とイメージ B の両方が期限切れとしてマークされます。ただし、イメージ A はルール 1 ですでに確認されていて、もしイメージ B が期限切れであれば、ルール 1 に違反してしまうので、スキップされます。
- 結果: イメージ A は期限切れです。

例 B

これは前の例と同じリポジトリですが、結果を説明するためにルールの優先順位が変更されています。

リポジトリのコンテンツ

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: 10 days ago
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: 9 days ago
- Image C, Taglist: ["beta-3"], Pushed: 8 days ago

ライフサイクルポリシーのテキスト

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

このライフサイクルポリシーのロジックは次のようになります。

- ルール 1 は、beta というプレフィックスでタグ付けされたイメージを特定します。最も古いイメージから始めて、一致するイメージが 1 つか数個になるまでマークし続けます。3 つのすべてのイメージが確認され、イメージ A とイメージ B が期限切れとしてマークされます。
- ルール 2 は、prod というプレフィックスでタグ付けされたイメージを特定します。最も古いイメージから始めて、一致するイメージが 1 つか数個になるまでマークし続けます。今回は、使用可能なイメージはすべてルール 1 で確認済みのため、確認できるイメージがありません。そのため、追加でイメージをマークすることはありません。

- 結果: イメージ A と B は期限切れです。

単一のルールで複数のタグをフィルタリングする

次の例で、1 つのルールでの複数のタグパターンのライフサイクルポリシーの構文を指定します。サンプルのリポジトリとライフサイクルポリシーが結果の説明とともに示されています。

例 A

1 つのルールで複数のタグパターンが指定されたときは、イメージはすべてのリストされたタグパターンに一致する必要があります。

リポジトリのコンテンツ

- Image A, Taglist: ["alpha-1"], Pushed: 12 days ago
- Image B, Taglist: ["beta-1"], Pushed: 11 days ago
- Image C, Taglist: ["alpha-2", "beta-2"], Pushed: 10 days ago
- Image D, Taglist: ["alpha-3"], Pushed: 4 days ago
- Image E, Taglist: ["beta-3"], Pushed: 3 days ago
- Image F, Taglist: ["alpha-4", "beta-4"], Pushed: 2 days ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
            },
            "action": {
                "type": "expire"
            }
        }
    ]
```

```
}
```

このライフサイクルポリシーのロジックは次のようになります。

- ルール 1 は、alpha と beta というプレフィックスでタグ付けされたイメージを特定します。イメージ C と F が確認されます。5 日より古いイメージをマークするので、イメージ C がマークされます。
- 結果: イメージ C は期限切れです。

例 B

次の例では、タグは排他的ではないことを説明します。

リポジトリのコンテンツ

- Image A, Taglist: ["alpha-1", "beta-1", "gamma-1"], Pushed: 10 days ago
- Image B, Taglist: ["alpha-2", "beta-2"], Pushed: 9 days ago
- Image C, Taglist: ["alpha-3", "beta-3", "gamma-2"], Pushed: 8 days ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPatternList": ["alpha*", "beta*"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

このライフサイクルポリシーのロジックは次のようになります。

- ルール 1 は、alpha と beta というプレフィックスでタグ付けされたイメージを特定します。すべてのイメージが確認されます。最も古いイメージから始めて、一致するイメージが 1 つか数個になるまでマークし続けます。イメージ A と B が期限切れとしてマークされます。
- 結果: イメージ A と B は期限切れです。

すべてのイメージでのフィルタリング

次のライフサイクルポリシーの例では、異なるフィルタですべてのイメージを指定します。サンプルのリポジトリとライフサイクルポリシーが結果の説明とともに示されています。

例 A

次に、すべてのルールを適用する一方、イメージを 1 つだけ保持して残りはすべて期限切れにするポリシーのライフサイクルポリシーの構文を示します。

リポジトリのコンテンツ

- Image A, Taglist: ["alpha-1"], Pushed: 4 days ago
- Image B, Taglist: ["beta-1"], Pushed: 3 days ago
- Image C, Taglist: [], Pushed: 2 days ago
- Image D, Taglist: ["alpha-2"], Pushed: 1 day ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

このライフサイクルポリシーのロジックは次のようになります。

- ルール 1 は、すべてのイメージを特定します。イメージ A、B、C、D が確認されます。最も新しいもの以外のすべてのイメージが期限切れとされます。イメージ A、B、C が期限切れとしてマークされます。
- 結果: イメージ A、B、C は期限切れです。

例 B

以下の例は、単一のポリシーのすべてのルールのタイプを組み合わせるライフサイクルポリシーを示しています。

リポジトリのコンテンツ

- Image A, Taglist: ["alpha-", "beta-1", "-1"], Pushed: 4 days ago
- Image B, Taglist: [], Pushed: 3 days ago
- Image C, Taglist: ["alpha-2"], Pushed: 2 days ago
- Image D, Taglist: ["git hash"], Pushed: 1 day ago
- Image E, Taglist: [], Pushed: 1 day ago

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

このライフサイクルポリシーの口ジックは次のようになります。

- ルール 1 は、alpha というプレフィックスでタグ付けされたイメージを特定します。イメージ A と C を識別します。最も新しいイメージを保持し、残りは期限切れとしてマークされます。イメージ A が期限切れとしてマークされます。
- ルール 2 は、タグ付けされていないイメージを特定します。イメージ B と E を識別します。有効期限が 1 日より古いすべてのイメージがマークされます。イメージ B が期限切れとしてマークされます。
- ルール 3 は、すべてのイメージを特定します。イメージ A、B、C、D、E を識別します。最も新しいイメージを保持し、残りは期限切れとしてマークされます。ただし、優先度がより高いルールにより識別されるため、イメージ A、B、C、または E をマークすることはできません。イメージ D が期限切れとしてマークされます。
- 結果: イメージ A、B、D は期限切れです。

Amazon ECR のライフサイクルポリシープロパティ

ライフサイクルポリシーには、次のプロパティがあります。

ライフサイクルポリシーの例については、「」を参照してくださいAmazon ECR のライフサイクルポリシーの例。を使用してライフサイクルポリシーを作成する手順については、 AWS CLI「」を参照してくださいライフサイクルポリシーを作成するには (AWS CLI)。

ルールの優先順位

`rulePriority`

タイプ: 整数

必須: はい

ルールを適用する順序を低いものから高いものの順に設定します。優先度が のライフサイクルポリシールール1が最初に適用され、優先度が のルール2が次に適用されます。ライフサイクルポリシーにルールを追加するときは、それぞれに `rulePriority` の一意の値を付ける必要があります。ポリシーのルール間で値をシーケンシャルにする必要はありません。any の `tagStatus` を持つルールは、`rulePriority` の最大値を持ち、最後に評価される必要があります。

説明

`description`

型: 文字列

必須: いいえ

(オプション) ライフサイクルポリシー内のルールの目的について説明します。

タグステータス

`tagStatus`

型: 文字列

必須: はい

追加するライフサイクルポリシーのルールがイメージのタグを指定するかどうかを決定します。使用できるオプションは、tagged、untagged、または any です。any を指定する場合は、すべてのイメージに対してルールが評価されます。tagged を指定する場合は、tagPrefixList 値も指定する必要があります。untagged を指定する場合は、tagPrefixList を省略する必要があります。

タグパターンリスト

`tagPatternList`

タイプ: list[string]

必須:はい、tagPrefixList がタグ付きに設定されていて、tagStatus が指定されていない場合

タグ付きイメージのライフサイクルポリシーを作成するときは、tagPatternList を使用してタグの有効期限を指定するのがベストプラクティスです。ライフサイクルポリシーでアクションを実行するときワイルドカード (*) を含む可能性のあるイメージタグパターンのカンマ区切りリストを指定する必要があります。例えば、イメージに prod、prod1、prod2 というようにタグが付いている場合、すべてを指定するためにタグパターンリスト prod* を使用します。複数のタグを指定する場合、指定されたすべてのタグが付いているイメージのみが選択されます。

> **Important**
>
> 1 文字列あたりのワイルドカード (*) の上限は 4 つです。例えば、["test*1*2*3*4*5*6"] は有効ですが ["*test*1*2*3", "test*1*2*3*"] は無効です。

タグプレフィックスリスト

`tagPrefixList`

タイプ: list[string]

必須:はい、tagPatternList がタグ付きに設定されていて、tagStatus が指定されていない場合

"tagStatus": "tagged" を指定し、tagPatternList を指定していない場合にのみ使用されます。ライフサイクルポリシーでアクションを実行するための、カンマ区切りのイメージタグプレフィックスのリストを指定する必要があります。たとえば、イメージに prod、prod1、prod2 というようにタグが付いている場合、すべてを指定するためにタグプレフィックス prod を使用します。複数のタグを指定する場合、指定されたすべてのタグが付いているイメージのみが選択されます。

カウントタイプ

countType

型: 文字列

必須: はい

イメージに適用するカウントタイプを指定します。

countType が imageCountMoreThan に設定してある場合は、countNumber も指定して、リポジトリに存在するイメージ数の制限を設定するルールを作成します。countType が sinceImagePushed に設定してある場合は、countUnit および countNumber も指定して、リポジトリに存在するイメージの時間制限を指定します。

カウント単位

countUnit

型: 文字列

必須: countType が sinceImagePushed に設定されている場合のみ、必須

日数を表す countNumber に加えて、カウント単位の days も時間単位として指定します。

これを指定するのは countType が sinceImagePushed である場合に限ります。countType が他の値である場合にカウント単位を指定すると、エラーが発生します。

カウント数

countNumber

タイプ: 整数

必須: はい

カウント数を指定します。許容値は正の整数です (0 は許容値ではありません)。

使用している countType が imageCountMoreThan である場合、この値はリポジトリに維持するイメージの最大数です。使用している countType が sinceImagePushed である場合、この値はイメージの最大期限です。

アクション

type

型: 文字列

必須: はい

アクションタイプを指定します。サポート対象の値は expire です。

Amazon Elastic Container Registry のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ — クラウドで AWS サービスを実行するインフラストラクチャを保護する責任 AWS は にあります AWS 。 AWS また、 は、安全に使用できるサービスも提供します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon ECR に適用されるコンプライアンスプログラムについては、「コンプライアンスプログラムによるAWS 対象範囲内のサービス」を参照してください。
- クラウド内のセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon ECR 使用時における責任共有モデルの適用法を理解するのに役立ちます。以下のトピックで、セキュリティおよびコンプライアンスの目的を満たすように、Amazon ECR を設定する方法について説明します。また、Amazon ECR リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

Amazon Elastic Container Registry の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインインを許可) し、誰に Amazon ECR リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon ECR で行う作業によって異なります。

サービスユーザー – ジョブを実行するために Amazon ECR サービスを使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。さらに多くの Amazon ECR 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Amazon ECR の機能にアクセスできない場合は、「Amazon Elastic Container Registry の アイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の Amazon ECR リソースを担当しているユーザーには、通常、Amazon ECR へのフルアクセスがあります。サービスのユーザーがどの Amazon ECR 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解

してください。会社で Amazon ECR を使用して IAM を利用する方法の詳細については、「Amazon Elastic Container Registry と IAM が連動するしくみ」を参照してください。

IAM 管理者 – IAM 管理者は、Amazon ECR へのアクセスを管理するポリシーの作成方法の詳細を理解することが推奨されます。IAM で使用できる Amazon ECR アイデンティティベースのポリシーの例を表示するには、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーション ID の例です。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『IAM ユーザーガイド』の「AWSにおける多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドの「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロール

を引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「サードパーティーアイデンティティプロバイダー向けロールの作成」 を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスにリンクされたロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するため

のアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

Amazon Elastic Container Registry と IAM が連動するしくみ

IAM を使用して Amazon ECR へのアクセスを管理する前に、Amazon ECR で使用できる IAM 機能について理解しておく必要があります。Amazon ECR およびその他の AWS のサービスが IAM と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

トピック

Amazon ECR のアイデンティティベースのポリシー

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、アクションを許可または拒否する条件を指定できます。Amazon ECR は、特定のアクション、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

アクション

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon ECR のポリシーアクションは、アクションの前にプレフィックス `ecr:` を使用します。たとえば、Amazon ECR `CreateRepository` API オペレーションを使用して Amazon ECR リポジトリを作成するアクセス許可を付与するには、ポリシーに `ecr:CreateRepository` アクションを含

めます。ポリシーステートメントには、Action または NotAction エレメントを含める必要があります。Amazon ECR は、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一ステートメントに複数アクションを指定するには、次のようにカンマで区切ります:

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "ecr:Describe*"
```

Amazon ECR アクションのリストについては、IAM ユーザーガイドの「Amazon Elastic Container Registry のアクション、リソース、および条件キー」を参照してください。

リソース

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

Amazon ECR リポジトリリソースには、次の ARN があります。

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

ARN の形式の詳細については、「Amazon リソースネーム (ARNs AWS 「サービス名前空間」を参照してください。

たとえば、ステートメントの us-east-1 リージョン で my-repo リポジトリを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

特定のアカウントに属するすべての リポジトリを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

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

Amazon ECR リソースタイプとその ARN のリストを表示するには、IAM ユーザーガイドの「Amazon Elastic Container Registry で定義されるリソースタイプ」を参照してください。各リソースの ARN を指定できるアクションについては、「Amazon Elastic Container Registry で定義されるアクション」を参照してください。

条件キー

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定するか、1 つの Condition 要素に複数のキーを指定すると、 AWS は AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細

については、『IAM ユーザーガイド』の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

Amazon ECR では独自の条件キーが定義されており、また一部のグローバル条件キーの使用がサポートされています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイドAWS 」の「グローバル条件コンテキストキー」を参照してください。

ほとんどの Amazon ECR アクションは、`aws:ResourceTag` と `ecr:ResourceTag` の条件キーをサポートします。詳細については、「タグベースのアクセスコントロールを使用する」を参照してください。

Amazon ECR 条件キーのリストについては、IAM ユーザーガイドの「Amazon Elastic Container Registry の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Elastic Container Registry で定義されるアクション」を参照してください。

例

Amazon ECR でのアイデンティティベースのポリシーの例は、「Amazon Elastic Container Registry のアイデンティティベースのポリシーの例」を参照してください。

Amazon ECR のリソースベースのポリシー

リソースベースのポリシーとは、Amazon ECR リソース上で指定するプリンシパルとしてのどのアクションをどの条件で実行できるかを指定する JSON ポリシードキュメントです。Amazon ECR は、Amazon ECR リポジトリのリソースベースのアクセス許可ポリシーをサポートします。リソースベースのポリシーでは、リソースごとに他の アカウントに使用許可を付与することができます。リソースベースのポリシーを使用して、Amazon ECR リポジトリへのアクセスを AWS サービスに許可することもできます。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウントにある場合は、プリンシパルエンティティにリソースへのアクセス許可も付与する必要があります。アクセス許可は、アイデンティティベースのポリシーをエンティティにアタッチすることで付与します。ただし、リソースベースのポリ

シーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、ID ベースのポリシーをさらに付与する必要はありません。詳細については、IAM ユーザーガイドの「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon ECR サービスでは、リポジトリポリシーと呼ばれるリソースベースのポリシーのタイプを 1 つのみサポートしており、これがリポジトリにアタッチされます。このポリシーは、リポジトリに対してアクションを実行できるプリンシパルエンティティ (アカウント、ユーザー、ロール、フェデレーティッドユーザー) を定義します。リソースベースのポリシーをリポジトリにアタッチする方法については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。

> **Note**
>
> Amazon ECR リポジトリポリシーでは、ポリシー要素 `Sid` は IAM ポリシーではサポートされていない追加の文字とスペースをサポートします。

例

Amazon ECR リソースベースのポリシーの例を表示するには、「Amazon ECR のプライベートリポジトリポリシーの例」を参照してください。

Amazon ECR タグに基づく認可

タグは、Amazon ECR リソースにアタッチするか、Amazon ECR へのリクエストで渡すことができます。タグに基づいてアクセスを管理するには、`ecr:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。Amazon ECR リソースのタグ付けの詳細については、「Amazon ECR でのプライベートリポジトリのタグ付け」を参照してください。

リソースのタグに基づいてリソースへのアクセスを制限するためのアイデンティティベースポリシーの例を表示するには、「タグベースのアクセスコントロールを使用する」を参照してください。

Amazon ECR のIAM ロール

IAM ロールは、特定のアクセス許可を持つ AWS アカウント内のエンティティです。

Amazon ECR での一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインインする、IAM ロールを引き受ける、またはクロスアカウントロールを引き受けることができます。一時的なセキュリティ認証情報を取得す

るには、 AssumeRoleやGetFederationトークン などの AWS STS API オペレーションを呼び出します。

Amazon ECR では、一時認証情報が使用できます。

サービスリンクロール

サービスにリンクされたロールを使用すると、 AWS サービスは他の サービスのリソースにアクセスして、ユーザーに代わってアクションを実行できます。サービスリンクロールは IAM アカウント内に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

Amazon ECR は、サービスにリンクされたロールをサポートしています。詳細については、「Amazon ECR でのサービスにリンクされたロールの使用」を参照してください。

Amazon Elastic Container Registry のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーおよびロールには Amazon ECR リソースを作成または変更するアクセス許可がありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

Amazon ECR が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認証リファレンス」の「Amazon Elastic Container Registry のアクション、リソース、および条件キー」を参照してください。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースポリシーを作成する方法については、「IAM ユーザーガイド」の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーのベストプラクティス

- Amazon ECR コンソールの使用
- ユーザーが自分のアクセス許可を表示できるようにする
- 1 つの Amazon ECR リポジトリにアクセスする

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon ECR リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して権限を適用する方法の詳細については、『IAM ユーザーガイド』の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレー

ションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon ECR コンソールの使用

Amazon Elastic Container Registry コンソールにアクセスするには、最小限の許可のセットが必要です。これらのアクセス許可により、 AWS アカウント内の Amazon ECR リソースの詳細を一覧表示および表示できます。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

これらのエンティティが引き続き Amazon ECR コンソールを使用できるようにするには、エンティティに AmazonEC2ContainerRegistryReadOnly AWS 管理ポリシーを追加します。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

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

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーが自分のアクセス許可を表示できるようにする

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

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

```
}
```

1 つの Amazon ECR リポジトリにアクセスする

この例では、 AWS アカウント内のユーザーに Amazon ECR リポジトリの 1 つである へのアクセス権を付与しますmy-repo。また、ユーザーがイメージをプッシュ、プル、リスト表示できるようにします。

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

タグベースのアクセスコントロールを使用する

Amazon ECR CreateRepository API アクションを使用すると、リポジトリの作成時にタグを指定できます。詳細については、「Amazon ECR でのプライベートリポジトリのタグ付け」を参照してください。

作成時にユーザーがリポジトリにタグ付けするには、そのリソースを作成するアクションを使用するためのアクセス許可が必要です (例: `ecr:CreateRepository`)。タグがリソース作成アクションで指定されている場合、Amazon は `ecr:CreateRepository` アクションで追加の認可を実行してユーザーがタグを作成するアクセス許可を持っているかどうかを確認します。

タグベースのアクセスコントロールは、IAM ポリシーを介して使用できます。以下は例です。

次のポリシーによって、ユーザーは、`key=environment,value=dev` としてリポジトリを作成またはタグ付けすることのみ許可されます。

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

次のポリシーによって、すべてのリポジトリ (key=environment,value=prod としてタグ付けされていない限り) へのユーザーアクセスが許可されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ecr:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ecr:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/environment": "prod"
                }
            }
        }
    ]
}
```

AWS Amazon Elastic Container Registry の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、多くの一般的なユースケースにアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) が更新されます。 は、新しい AWS のサービス が起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、 AWS 管理ポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

Amazon ECR には、IAM ID または Amazon EC2 インスタンスにアタッチできる複数の管理ポリシーが用意されています。これらの管理ポリシーにより、Amazon ECR リソースと API オペレーションへのアクセスをさまざまなレベルで制御できます。これらのポリシーに記載されている各 API オペレーションの詳細については、Amazon Elastic Container Registry API リファレンスの「アクション」を参照してください。

トピック

- AmazonEC2ContainerRegistryFullAccess
- AmazonEC2ContainerRegistryPowerUser
- AmazonEC2ContainerRegistryReadOnly
- AWSECRPullThroughCache_ServiceRolePolicy
- ECRReplicationServiceRolePolicy
- AWS マネージドポリシーに対する Amazon ECR の更新

AmazonEC2ContainerRegistryFullAccess

`AmazonEC2ContainerRegistryFullAccess` ポリシーは IAM ID にアタッチできます。

この管理ポリシーを開始点として使用して、特定の要件に基づいて独自の IAM ポリシーを作成できます。例えば、Amazon ECR の使用を管理するための完全な管理者アクセスをユーザーまたはロールに提供するのに特化したポリシーを作成できます。Amazon ECR ライフサイクルポリシー機能を使用すると、リポジトリ内のイメージのライフサイクル管理を指定できます。ライフサイクルポリシーイベントは CloudTrail イベントとして報告されます。Amazon ECR は と統合 AWS CloudTrail されているため、ライフサイクルポリシーイベントを Amazon ECR コンソールに直接表示できま

す。AmazonEC2ContainerRegistryFullAccess マネージド IAM ポリシーには、この動作を容易にするための cloudtrail:LookupEvents アクセス許可が含まれています。

許可の詳細

このポリシーには以下のアクセス許可が含まれています。

- ecr — プリンシパルにすべての Amazon ECR API へのフルアクセスを許可します。
- cloudtrail – プリンシパルが、 によってキャプチャされた管理イベントまたは AWS CloudTrail Insights イベントを検索できるようにします CloudTrail。

AmazonEC2ContainerRegistryFullAccess ポリシーを以下に示します。

```
{
    "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

AmazonEC2ContainerRegistryPowerUser ポリシーは IAM ID にアタッチできます。

このポリシーは、IAM ユーザーがリポジトリに対する読み取りと書き込みを行う管理権限を付与しますが、IAM ユーザーは、リポジトリを削除することや、適用されるポリシードキュメントを変更することはできません。

許可の詳細

このポリシーには以下のアクセス許可が含まれています。

- ecr – リポジトリへの読み取りと書き込み、ライフサイクルポリシーの読み取りをプリンシパルに許可します。プリンシパルには、リポジトリを削除するアクセス許可およびリポジトリに適用されるライフサイクルポリシーを変更するアクセス許可は付与されません。

AmazonEC2ContainerRegistryPowerUser ポリシーを以下に示します。

```
{
    "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

AmazonEC2ContainerRegistryReadOnly ポリシーは IAM ID にアタッチできます。

このポリシーでは、Amazon ECR に対する読み取り専用アクセスを付与します。リポジトリおよびリポジトリ内のイメージをリストすることができます。また、Docker CLI を使用して Amazon ECR からイメージをプルすることもできます。

許可の詳細

このポリシーには以下のアクセス許可が含まれています。

- ecr — リポジトリとそれぞれのライフサイクルポリシーの読み取りをプリンシパルに許可します。

AmazonEC2ContainerRegistryReadOnly ポリシーを以下に示します。

```
{
    "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

AWSECRPullThroughCache_ServiceRolePolicy マネージド IAM ポリシーを IAM エンティティにアタッチすることはできません。このポリシーは、プルスルーキャッシュワークフローを通じて Amazon ECR がイメージをリポジトリにプッシュすることを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「プルスルーキャッシュの Amazon ECR サービスにリンクされたロール」を参照してください。

ECRReplicationServiceRolePolicy

ECRReplicationServiceRolePolicy マネージド IAM ポリシーを IAM エンティティにアタッチすることはできません。このポリシーは、サービスにリンクされたロールにアタッチされ、ユーザーに代わって Amazon ECR がアクションを実行することを許可します。詳細については、「Amazon ECR でのサービスにリンクされたロールの使用」を参照してください。

AWS マネージドポリシーに対する Amazon ECR の更新

Amazon ECR の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した時点から表示します。このページの変更に関する自動通知を有効にするには、Amazon ECR ドキュメントの履歴ページから RSS フィードをサブスクライブしてください。

変更	説明	日付
AWSECRPullThroughCache_ServiceRolePolicy - 既存ポリシーへの更新	Amazon ECR は、新しいアクセス許可を AWSECRPullThroughCache_ServiceRolePolicy ポリシーに追加しました。これらの権限により、Amazon ECR は Secrets Manager シークレットの暗号化されたコンテンツを取得できます。これは、プルスルーキャッシュルールを使用して、認証が必要なアップストリームレジストリから	2023 年 11 月 15 日

変更	説明	日付
	イメージをキャッシュする場合に必要です。	
AWSECRPullThroughCache_ServiceRolePolicy - 新しいポリシー	Amazon ECR は新しいポリシーを追加しました。このポリシーは、プルスルーキャッシュ機能の AWSServiceRoleForECRPullThroughCache サービスにリンクされたロールに関連付けられます。	2021 年 11 月 29 日
ECRReplicationServiceRolePolicy – 新しいポリシー	Amazon ECR は新しいポリシーを追加しました。このポリシーは、レプリケーション機能の AWSServiceRoleForECRReplication サービスにリンクされたロールに関連付けられます。	2020 年 12 月 4 日
AmazonEC2ContainerRegistryFullAccess – 既存のポリシーの更新	Amazon ECR は、新しいアクセス許可を AmazonEC2ContainerRegistryFullAccess ポリシーに追加しました。これらのアクセス許可により、プリンシパルは Amazon ECR サービスにリンクされたロールを作成できるようになります。	2020 年 12 月 4 日

変更	説明	日付
AmazonEC2ContainerRegistryReadOnly – 既存のポリシーの更新	Amazon ECR は、ライフサイクルポリシーの読み取り、タグの一覧表示、イメージのスキャン結果の記述をプリンシパルに許可する新しいアクセス許可を AmazonEC2ContainerRegistryReadOnly ポリシーに追加しました。	2019 年 12 月 10 日
AmazonEC2ContainerRegistryPowerUser – 既存のポリシーの更新	Amazon ECR は、新しいアクセス許可を AmazonEC2ContainerRegistryPowerUser ポリシーに追加しました。これにより、プリンシパルは、ライフサイクルポリシーの読み取り、タグの一覧表示、イメージのスキャン結果の記述を行うことができるようになります。	2019 年 12 月 10 日
AmazonEC2ContainerRegistryFullAccess – 既存のポリシーの更新	Amazon ECR は、新しいアクセス許可を AmazonEC2ContainerRegistryFullAccess ポリシーに追加しました。これにより、プリンシパルは によってキャプチャされた管理イベントまたは AWS CloudTrail Insights イベントを検索できます CloudTrail。	2017 年 11 月 10 日

変更	説明	日付
AmazonEC2ContainerRegistryReadOnly — 既存のポリシーの更新	Amazon ECR は、新しいアクセス許可を AmazonEC2ContainerRegistryReadOnly ポリシーに追加しました。これにより、プリンシパルは Amazon ECR イメージを記述できるようになります。	2016 年 10 月 11 日
AmazonEC2ContainerRegistryPowerUser — 既存のポリシーの更新	Amazon ECR は、新しいアクセス許可を AmazonEC2ContainerRegistryPowerUser ポリシーに追加しました。これにより、プリンシパルは Amazon ECR イメージを記述できるようになります。	2016 年 10 月 11 日
AmazonEC2ContainerRegistryReadOnly – 新しいポリシー	Amazon ECR は、Amazon ECR に読み取り専用アクセス権限を付与する新しいポリシーを追加しました。これらのアクセス許可には、リポジトリおよびリポジトリ内のイメージをリストすることが含まれます。また、Docker CLI を使用して Amazon ECR からイメージをプルすることもできます。	2015 年 12 月 21 日

変更	説明	日付
AmazonEC2ContainerRegistryPowerUser – 新しいポリシー	Amazon ECR は、ユーザーがリポジトリに対する読み取りと書き込みを行える管理権限を付与する新しいポリシーを追加しました。ただし、ユーザーは、リポジトリを削除することや、適用されるポリシードキュメントを変更することはできません。	2015 年 12 月 21 日
AmazonEC2ContainerRegistryFullAccess – 新しいポリシー	Amazon ECR は新しいポリシーを追加しました。このポリシーは、Amazon ECR へのフルアクセスを付与します。	2015 年 12 月 21 日
Amazon ECR が変更の追跡を開始しました。	Amazon ECR が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 6 月 24 日

Amazon ECR でのサービスにリンクされたロールの使用

Amazon Elastic Container Registry (Amazon ECR) は、 AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用して、レプリケーションとプルスルーキャッシュ機能を使用するために必要なアクセス許可を提供します。サービスにリンクされたロールは、Amazon ECR に直接リンクされた特殊な IAM ロールです。サービスにリンクされたロールは、Amazon ECR で事前定義されています。これには、サービスがプライベートレジストリのレプリケーションとプルスルーキャッシュ機能をサポートするために必要なすべてのアクセス許可が含まれます。レジストリのレプリケーションまたはプルスルーキャッシュを構成すると、サービスにリンクされたロールが自動的に作成されます。詳細については、「Amazon ECR のプライベートレジストリ設定」を参照してください。

サービスにリンクされたロールを使用すると、Amazon ECR を使用したレプリケーションとプルスルーキャッシュの設定が簡単になります。この理由は、このロールを使用することにより、必要なアクセス許可をすべて手動で追加する必要がなくなるためです。サービスにリンクされたロールのアクセス許可は、Amazon ECR により定義されます。特に指定されている場合を除き、そのロールを引

き受けることができるのは Amazon ECR のみです。定義された許可には、信頼ポリシーと許可ポリシーが含まれます。アクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。

対応するサービスにリンクされたロールを削除できるのは、先にレジストリでレプリケーションまたはプルスルーキャッシュのいずれかを無効にした後のみです。これにより、これらの機能に対して Amazon ECR で必要なアクセス許可を誤って削除することがなくなります。

サービスにリンクされたロールをサポートするその他のサービスの詳細については、「IAM と連携するAWS のサービス」を参照してください。このリンク先のページで、「サービスにリンクされたロール」列が「あり」になっているサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[あり] リンクを選択します。

トピック

- Amazon ECR のサービスにリンクされたロールがサポートされるリージョン
- レプリケーション用の Amazon ECR サービスにリンクされたロール
- プルスルーキャッシュの Amazon ECR サービスにリンクされたロール

Amazon ECR のサービスにリンクされたロールがサポートされるリージョン

Amazon ECR では、このサービスを利用できるすべてのリージョンで、サービスにリンクされたロールの使用がサポートされています。Amazon ECR リージョンの可用性については、「AWS リージョンとエンドポイント」を参照してください。

レプリケーション用の Amazon ECR サービスにリンクされたロール

Amazon ECR は、Amazon ECR AWSServiceRoleForECRReplicationが複数のアカウントにイメージをレプリケートできるようにする という名前のサービスにリンクされたロールを使用します。

Amazon ECR でのサービスにリンクされたロールのアクセス許可

AWSServiceRoleForECRReplication サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `replication.ecr.amazonaws.com`

以下の `ECRReplicationServiceRolePolicy` ロールのアクセス許可ポリシーは、リソースに対して以下のアクションを使用することを Amazon ECR に許可します。

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

> **Note**
>
> `ReplicateImage` は、Amazon ECR がレプリケーションに使用する内部 API で、直接呼び出すことはできません。

アクセス許可を設定して、IAM エンティティ (ユーザー、グループ、ロールなど) がサービスリンクロールの作成、編集、削除を行うことを許可する必要があります。詳細については、IAM ユーザーガイドの「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon ECR でのサービスにリンクされたロールの作成

Amazon ECR サービスにリンクされたロールを手動で作成する必要はありません。、 AWS Management Console、 AWS CLIまたは AWS API でレジストリのレプリケーション設定を構成すると、Amazon ECR によってサービスにリンクされたロールが作成されます。

このサービスにリンクされたロールを削除した後に、そのロールを再作成する必要がある場合は、同じプロセスを使用してアカウントでロールを再作成することができます。レジストリのレプリケーション設定を構成すると、サービスにリンクされたロールAmazon ECR が Amazon ECR によって再度作成されます。

Amazon ECR でのサービスにリンクされたロールの編集

Amazon ECR では、 AWSServiceRoleForECRReplication サービスにリンクされたロールを手動で編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用し

たロール記述の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

Amazon ECR でのサービスにリンクされたロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、アクティブにモニタリングまたはメンテナンスされない未使用のエンティティがなくなります。ただし、サービスにリンクされたロールを手動で削除するには、すべてのリージョンでレジストリのレプリケーション設定を削除する必要があります。

> **Note**
>
> Amazon ECR サービスがロールを使用しているときにリソースを削除しようとすると、削除アクションが失敗することがあります。その場合は、数分待ってから再試行してください。

で使用される Amazon ECR リソースを削除するには AWSServiceRoleForECRReplication

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、レプリケーション設定が有効なリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ) を選択します。
4. [Private registry] (プライベートレジストリ) ページの [Replication configuration] (レプリケーション設定) セクションで、[Edit] (編集) をクリックします。
5. すべてのレプリケーションルールを削除するには、[Delete all] (すべて削除) を選択します。このステップには確認が必要です。

サービスにリンクされたロールを IAM で手動削除するには

IAM コンソール、、または AWS API を使用して AWS CLI、AWSServiceRoleForECRReplicationサービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの削除」を参照してください。

プルスルーキャッシュの Amazon ECR サービスにリンクされたロール

Amazon ECR は、 という名前のサービスにリンクされたロールを使用します。AWSServiceRoleForECRPullThroughCacheこのロールは、Amazon ECR がユーザーに代わってアクションを実行し、プルスルーキャッシュアクションを完了するアクセス許可を付与します。プル

スルーの詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。

Amazon ECR でのサービスにリンクされたロールのアクセス許可

AWSServiceRoleForECRPullThroughCache サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- pullthroughcache.ecr.amazonaws.com

許可の詳細

AWSECRPullThroughCache_ServiceRolePolicy 許可ポリシーは、サービスにリンクされたロールにアタッチされます。この管理ポリシーは Amazon ECR に以下のアクションを実行する許可を付与します。詳細については、「AWSECRPullThroughCache_ServiceRolePolicy」を参照してください。

- ecr — Amazon ECR サービスがプライベートリポジトリにイメージをプッシュできるようにします。
- secretsmanager:GetSecretValue — Amazon ECR サービスが AWS Secrets Manager シークレットの暗号化されたコンテンツを取得できるようにします。これは、プルスルーキャッシュルールを使用して、プライベートレジストリの認証が必要なアップストリームレジストリからイメージをキャッシュする場合に必要です。この許可は、ecr-pullthroughcache/ という名前のプレフィックスが付いたシークレットのみに適用されます。

AWSECRPullThroughCache_ServiceRolePolicy ポリシーには、次の JSON が含まれます。

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

アクセス許可を設定して、IAM エンティティ (ユーザー、グループ、ロールなど) がサービスリンクロールの作成、編集、削除を行うことを許可する必要があります。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールのアクセス許可」を参照してください。

Amazon ECR でのサービスにリンクされたロールの作成

プルスルーキャッシュの Amazon ECR サービスにリンクされたロールを手動で作成する必要はありません。、 AWS Management Console、 AWS CLIまたは AWS API でプライベートレジストリのプルスルーキャッシュルールを作成すると、Amazon ECR によってサービスにリンクされたロールが作成されます。

このサービスにリンクされたロールを削除した後に、そのロールを再作成する必要がある場合は、同じプロセスを使用してアカウントでロールを再作成することができます。プライベートレジストリーのプルスルーキャッシュルールを作成すると、サービスにリンクされたロールがまだ存在しない場合は、Amazon ECR によって自動的に作成されます。

Amazon ECR でのサービスにリンクされたロールの編集

Amazon ECR では、AWSServiceRoleForECRPullThroughCacheサービスにリンクされたロールを手動で編集することはできません。サービスにリンクされたロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

Amazon ECR でのサービスにリンクされたロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、アクティブにモニタリングまたはメンテナンスされない未使用のエンティティがなくなります。ただし、サービスにリンクされたロールを手動で削除するには、すべてのリージョンでレジストリのプルスルーキャッシュルールを削除する必要があります。

> **Note**
>
> Amazon ECR サービスがロールをまだ使用しているときにリソースを削除しようとすると、削除アクションが失敗することがあります。その場合は、数分待ってから再試行してください。

AWSServiceRoleForECRPullThroughCache サービスにリンクされたロールによって使用される Amazon ECR リソースを削除するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. ナビゲーションバーから、プルスルーキャッシュルールを作成するリージョンを選択します。
3. ナビゲーションペインで、[Private registry] (プライベートレジストリ) を選択します。
4. [Private registry] (プライベートレジストリ) ページの [Pull through cache configuration] (プルスルーキャッシュの設定) セクションで、[Edit] (編集) をクリックします。
5. 作成したプルスルーキャッシュルールごとに、ルールを選択し、[Delete rule] (ルールを削除) を選択します。

サービスにリンクされたロールを IAM で手動削除するには

IAM コンソール、、または AWS API を使用して AWS CLI、AWSServiceRoleForECRPullThroughCacheサービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの削除」を参照してください。

Amazon Elastic Container Registry の アイデンティティとアクセスのトラブルシューティング

次の情報は、Amazon ECR と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- Amazon ECR でアクションを実行する権限がない
- iam を実行する権限がありません。PassRole
- 自分の 以外のユーザーに Amazon ECR リソース AWS アカウント へのアクセスを許可したい

Amazon ECR でアクションを実行する権限がない

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なecr:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

この場合、ecr:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon ECR にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して Amazon ECR でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon ECR リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon ECR がこれらの機能をサポートしているかどうかについては、「Amazon Elastic Container Registry と IAM が連動するしくみ」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセス権限」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon ECR でのデータ保護

AWS 責任共有モデル、Amazon Elastic Container Service でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ

設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon ECS AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

トピック

- 保管中の暗号化

保管中の暗号化

Amazon ECR は、Amazon ECR が管理する Amazon S3 バケットにイメージを保存します。デフォルトでは、Amazon ECR は、Amazon S3 が管理する暗号化キーによるサーバー側暗号化を使用し、保管中のデータが AES-256 暗号化アルゴリズムで暗号化されます。これは、お客様によるアクションを必要とせず、追加料金なしで提供されます。詳細については、Amazon Simple Storage Service

ユーザーガイドの「Amazon S3 が管理する暗号化キーによるサーバー側の暗号化 (SSE-S3) を使用したデータの保護」を参照してください。

Amazon ECR リポジトリの暗号化をより詳細に制御するには、 AWS Key Management Service () に保存されている KMS キーによるサーバー側の暗号化を使用できますAWS KMS。を使用して AWS KMS データを暗号化する場合、Amazon ECR によって管理 AWS マネージドキーされるデフォルトの を使用するか、独自の KMS キー (カスタマーマネージドキーと呼ばれます) を指定できます。詳細については、Amazon Simple Storage Service ユーザーガイドの AWS KMS 「 (SSE-KMS) に保存されている KMS キーによるサーバー側の暗号化を使用したデータの保護」を参照してください。

各 Amazon ECR リポジトリには、リポジトリの作成時に設定される暗号化設定があります。リポジトリごとに異なる暗号化設定を使用できます。詳細については、「イメージを保存する Amazon ECR プライベートリポジトリの作成」を参照してください。

AWS KMS 暗号化を有効にしてリポジトリを作成すると、KMS キーを使用してリポジトリの内容を暗号化します。さらに、Amazon ECR は、Amazon ECR リポジトリを AWS KMS 被付与者プリンシパルとして KMS キーに権限を追加します。

Amazon ECR が AWS KMS と統合してレポジトリを暗号化および復号化する方法の高レベルの概要を以下に示します。

1. リポジトリを作成すると、Amazon ECR は へのDescribeKey呼び出しを送信 AWS KMS して、暗号化設定で指定された KMS キーの Amazon リソースネーム (ARN) を検証および取得します。
2. Amazon ECR は に 2 つのCreateGrantリクエストを送信 AWS KMS して KMS キーに許可を作成し、Amazon ECR がデータキーを使用してデータを暗号化および復号できるようにします。
3. イメージをプッシュすると、GenerateDataイメージレイヤーとマニフェストの暗号化に使用する KMS キー AWS KMS を指定するキーリクエストが に対して行われます。
4. AWS KMS は新しいデータキーを生成し、指定された KMS キーで暗号化し、暗号化されたデータキーを送信して、イメージレイヤーメタデータとイメージマニフェストとともに保存します。
5. イメージをプルすると、暗号化されたデータキーを指定して AWS KMS、Decrypt リクエストが に対して行われます。
6. AWS KMS は、暗号化されたデータキーを復号し、復号されたデータキーを Amazon S3 に送信します。
7. データキーは、イメージレイヤーをプルする前にイメージレイヤーを復号化するために使用されます。
8. リポジトリが削除されると、Amazon ECR はリポジトリ用に作成された許可を廃止 AWS KMS するための 2 つのRetireGrantリクエストを に送信します。

考慮事項

Amazon ECR で AWS KMS 暗号化を使用する場合は、次の点を考慮する必要があります。

- KMS 暗号化を使用して Amazon ECR リポジトリを作成し、KMS キーを指定しない場合、Amazon ECR は`aws/ecr`デフォルトでエイリアス AWS マネージドキー を持つ を使用します。この KMS キーは、KMS 暗号化を有効にしたリポジトリを初めて作成するときに、アカウントに作成されます。
- 独自の KMS キーで KMS 暗号化を使用する場合、そのキーはリポジトリと同じリージョンに存在する必要があります。
- お客様の代わりに Amazon ECR が作成する許可は取り消さないでください。アカウント内の AWS KMS キーを使用するアクセス許可を Amazon ECR に付与する許可を取り消すと、Amazon ECR はこのデータにアクセスしたり、リポジトリにプッシュされた新しいイメージを暗号化したり、プル時に復号化したりすることはできません。Amazon ECR の許可を取り消すと、変更がすぐに適用されます。アクセス権限を取り消すには、許可を取り消すのではなく、リポジトリを削除します。リポジトリを削除すると、Amazon ECR がユーザーに代わって許可を取り消します。
- AWS KMS キーの使用にはコストがかかります。詳細については、「AWS Key Management Service 料金表」を参照してください。

必要な IAM 許可

AWS KMSを使用するサーバー側の暗号化が有効な Amazon ECR リポジトリを作成または削除する場合、必要なアクセス許可は、使用する特定の KMS キーに応じて異なります。

Amazon ECR の を使用する場合 AWS マネージドキー に必要な IAM アクセス許可

デフォルトでは、Amazon ECR リポジトリで AWS KMS 暗号化が有効になっていても KMS キーが指定されていない場合、Amazon ECR AWS マネージドキー の が使用されます。Amazon ECR の AWS管理の KMS キーを使用してリポジトリを暗号化する場合、リポジトリを作成するアクセス許可を持つプリンシパルは、リポジトリで AWS KMS 暗号化を有効にすることもできます。ただし、リポジトリを削除する IAM プリンシパルには、`kms:RetireGrant` アクセス許可が必要です。これにより、リポジトリの作成時に AWS KMS キーに追加された許可の廃止が可能になります。

次の IAM ポリシーの例をインラインポリシーとしてユーザーに追加すると、暗号化が有効なリポジトリを削除するのに必要な最小限のアクセス許可がユーザーに付与されます。リポジトリの暗号化に使用する KMS キーは、リソースパラメータを使用して指定できます。

```
{
```

```
    "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"
        }
    ]
}
```

カスタマーマネージドキーを使用する場合に必要な IAM アクセス許可

カスタマーマネージドキーを使用して AWS KMS 暗号化を有効にしたリポジトリを作成する場合、リポジトリを作成するユーザーまたはロールの KMS キーポリシーと IAM ポリシーの両方に必要なアクセス許可があります。

独自の KMS キーを作成する場合、 AWS KMS が作成するデフォルトのキーポリシーを使用するか、独自のキーポリシーを指定することができます。アカウント所有者がカスタマーマネージドキーを管理できるように、KMS キーのキーポリシーでは、アカウントのルートユーザーのすべての AWS KMS アクションを許可する必要があります。追加のスコープ付きアクセス許可をキーポリシーに追加することもできますが、少なくともルートユーザーには KMS キーを管理するためのアクセス許可が必要です。Amazon ECR から送信されるリクエストにのみ KMS キーを使用できるようにするには、 `ecr.<region>.amazonaws.com`値で kms:ViaService condition キーを使用できます。

次のキーポリシーの例では、KMS キーを所有する AWS アカウント (ルートユーザー) に KMS キーへのフルアクセスを許可します。このキーポリシーの例の詳細については、「 デベロッパーガイド」の AWS 「アカウントへのアクセスを許可し、IAM ポリシーを有効にする」を参照してください。 AWS Key Management Service

```
{
    "Version": "2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

リポジトリを作成する IAM ユーザー、IAM ロール、または AWS アカウントには`kms:CreateGrant`、必要な Amazon ECR アクセス`kms:DescribeKey`許可に加えて`kms:RetireGrant`、、、および アクセス許可が必要です。

> **Note**
>
> リポジトリを作成するユーザーまたはロールの IAM ポリシーに `kms:RetireGrant` アクセス許可を追加する必要があります。`kms:CreateGrant` と `kms:DescribeKey` のアクセス許可は、KMS キーのキーポリシー、またはリポジトリを作成するユーザーまたはロールの IAM ポリシーに追加できます。アクセス AWS KMS 許可の仕組みの詳細については、「AWS Key Management Service デベロッパーガイド」のAWS KMS 「API アクセス許可: アクションとリソースのリファレンス」を参照してください。

次の IAM ポリシーの例をインラインポリシーとしてユーザーに追加すると、暗号化が有効なリポジトリの作成、および作業が終了したときにリポジトリの削除を行うのに必要な最小限のアクセス許可がユーザーに付与されます。リポジトリの暗号化に使用する AWS KMS key は、リソースパラメータを使用して指定できます。

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

リポジトリの作成時にコンソールに KMS キーを一覧表示できるようにする

Amazon ECR コンソールを使用してリポジトリを作成すると、リポジトリの暗号化を有効にするときに、ユーザーがリージョンでカスタマーマネージド KMS キーを一覧表示するアクセス許可を付与できます。次の IAM ポリシーの例は、コンソールを使用するときに KMS キーとエイリアスを一覧表示するために必要なアクセス許可を示しています。

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

AWS KMSと Amazon ECR のインタラクションのモニタリング

を使用して AWS CloudTrail 、Amazon ECR が AWS KMS ユーザーに代わって に送信するリクエストを追跡できます。ログの CloudTrail ログエントリには、識別しやすくするための暗号化コンテキストキーが含まれています。

Amazon ECR 暗号化コンテキスト

暗号化コンテキスト は、任意非シークレットデータを含むキーと値のペアのセットです。データを暗号化するリクエストに暗号化コンテキストを含めると、 は暗号化コンテキストを暗号化されたデータに AWS KMS 暗号的にバインドします。データを復号するには、同じ暗号化コンテキストに渡す必要があります。

へのGenerateDataキーリクエストと復号リクエストでは AWS KMS、Amazon ECR は、使用するリポジトリと Amazon S3 バケットを識別する 2 つの名前と値のペアを持つ暗号化コンテキストを使用

します。以下の例ではこれを示しています。名前は変わりませんが、組み合わされた暗号化コンテキストの値は、値ごとに異なります。

```
"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"
}
```

暗号化コンテキストを使用して、 AWS CloudTrailや Amazon CloudWatch Logs などの監査レコードやログでこれらの暗号化オペレーションを識別し、ポリシーや許可での承認の条件として識別できます。

Amazon ECR 暗号化コンテキストは、2 つの名前と値のペアで構成されます。

- aws:s3:arn – バケットを識別する最初の名前と値のペア。キーは、aws:s3:arn です。値は、Amazon S3 バケットの Amazon リソースネーム (ARN)です。

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

  たとえば、バケットの ARN が arn:aws:s3:::*us-west-2*-starport-manifest-bucket/*EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1*/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df の場合、暗号化コンテキストには次のペアが含まれます。

  ```
  "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
  ```

- aws:ecr:arn – リポジトリの Amazon リソースネーム (ARN) を識別する 2 番目の名前と値のペア。キーは、aws:ecr:arn です。値は、リポジトリの ARNです。

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

  たとえば、リポジトリの ARN が arn:aws:ecr:*us-west-2*:*111122223333*:repository/*repository-name* の場合、暗号化コンテキストには次のペアが含まれます。

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

トラブルシューティング

コンソールで Amazon ECR リポジトリを削除するときに、リポジトリが正常に削除されても、Amazon ECR がリポジトリの KMS キーに追加された許可を使用停止できない場合、次のエラーが表示されます。

```
The repository [{repository-name}] has been deleted successfully but the grants created
 by the kmsKey [{kms_key}] failed to be retired
```

この場合、リポジトリの AWS KMS 許可を自分で廃止できます。

リポジトリの AWS KMS 許可を手動で廃止するには

1. リポジトリに使用される AWS KMS キーの許可を一覧表示します。key-id 値は、コンソールに表示されるエラーに含まれています。list-keys コマンドを使用して、 アカウント内の特定のリージョンにある AWS マネージドキー とカスタマーマネージド KMS キーの両方を一覧表示することもできます。

   ```
   aws kms list-grants \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
        --region us-west-2
   ```

   出力には、リポジトリの Amazon リソースネーム (ARN) を含む EncryptionContextSubset が含まれます。これを使用して、使用停止にするのがキーに追加されたどの許可かを特定できます。GrantId 値は、次のステップで許可を使用停止するときに使用します。

2. リポジトリに追加された AWS KMS キーの各許可を廃止します。の値を、前のステップの出力からの許可の ID *GrantId*に置き換えます。

   ```
   aws kms retire-grant \
        --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
        --grant-id GrantId \
        --region us-west-2
   ```

Amazon Elastic Container Registry のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照

し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド — コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。

- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Amazon Elastic Container Registry のインフラストラクチャセキュリティ

マネージドサービスである Amazon Elastic Container Registry は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「 Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Amazon ECR にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

これらの API オペレーションは任意のネットワークの場所から呼び出すことができますが、Amazon ECR ではリソースベースのアクセスポリシーがサポートされているため、ソース IP アドレスに基づく制限を含めることができます。また、Amazon ECR ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを管理することもできます。これにより、実質的にネットワーク内の特定の VPC からのみ、特定の Amazon ECR リ

ソースへの AWS ネットワークアクセスが分離されます。詳細については、「Amazon ECR インターフェイス VPC エンドポイント (AWS PrivateLink) 」を参照してください。

Amazon ECR インターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを使用するように Amazon ECR を設定することで、VPC のセキュリティ体制を強化できます。VPC エンドポイントは AWS PrivateLink、プライベート IP アドレスを介して Amazon ECR APIs にプライベートにアクセスできるテクノロジーである を利用しています。 AWS PrivateLink は、VPC と Amazon ECR 間のすべてのネットワークトラフィックを Amazon ネットワークに制限します。インターネットゲートウェイ、NAT デバイス、または仮想プライベートゲートウェイは必要ありません。

AWS PrivateLink および VPC エンドポイントの詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイント」を参照してください。

Amazon ECR VPC エンドポイントに関する考慮事項

Amazon ECR の VPC エンドポイントを設定する前に、以下の考慮事項に注意してください。

- Amazon EC2 インスタンスでホストされる Amazon ECS タスクで Amazon ECR からプライベートイメージをプルするには、Amazon ECS 用のインターフェイス VPC エンドポイントも作成する必要があります。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink) 」を参照してください。

  > **Important**
  >
  > Fargate でホストされる Amazon ECS タスクでは、Amazon ECS インターフェイス VPC エンドポイントは必要ありません。

- Linux プラットフォームバージョン `1.3.0` 以前を使用する Fargate でホストされる Amazon ECS タスクの場合、この機能を活用するために必要なのは、com.amazonaws.***region***.ecr.dkr Amazon ECR VPC エンドポイントと Amazon S3 ゲートウェイエンドポイントのみです。
- Linux プラットフォームバージョン `1.4.0` 以降を使用する Fargate でホストされる Amazon ECS タスクの場合、この機能を活用するには、com.amazonaws.***region***.ecr.dkr と com.amazonaws.***region***.ecr.api の両方の Amazon ECR VPC エンドポイントに加えて、Amazon S3 ゲートウェイエンドポイントが必要です。
- Windows プラットフォームバージョン `1.0.0` 以降を使用する Fargate でホストされる Amazon ECS タスクの場合、この機能を活用するには、com.amazonaws.***region***.ecr.dkr と

com.amazonaws.*region*.ecr.api の両方の Amazon ECR VPC エンドポイントに加えて、Amazon S3 ゲートウェイエンドポイントが必要です。

- Amazon ECR からコンテナイメージをプルする Fargate でホストされる Amazon ECS タスクでは、条件キーをタスクのタスク実行 IAM ロールに追加することによって、タスクが使用する特定の VPC へのアクセス、およびサービスが使用する VPC エンドポイントへのアクセスを制限することができます。詳細については、Amazon Elastic Container Service デベロッパーガイドの「インターフェイスエンドポイントを介して Amazon ECR をプルする Fargate タスクの最適な IAM アクセス許可」を参照してください。
- `awslogs` ログドライバーを使用してログ情報を Logs に送信する Amazon ECR からコンテナイメージをプルする Fargate でホストされる Amazon ECS CloudWatch タスクには、Logs VPC CloudWatch エンドポイントが必要です。詳細については、「Logs CloudWatch エンドポイントを作成する」を参照してください。
- VPCエンドポイントにアタッチされたセキュリティグループでは、VPCのプライベートサブネットから、ポート 443 で着信接続を許可する必要があります。
- 現在、VPC エンドポイントはクロスリージョンリクエストをサポートしていません。Amazon ECR に対して API コールを発行するリージョンと同じリージョンに VPC エンドポイントを作成してください。
- 現在、VPC エンドポイントは Amazon ECR パブリックリポジトリをサポートしていません。プルスルーキャッシュルールを使用して、VPC エンドポイントと同じリージョンにあるプライベートリポジトリでパブリックイメージをホストすることを検討してください。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。
- VPC エンドポイントは、Amazon Route 53 を通じて AWS 提供される DNS のみをサポートします。独自の DNS を使用したい場合は、条件付き DNS 転送を使用できます。詳細については、Amazon VPC ユーザーガイドのDHCP Options Setsを参照してください。
- コンテナに Amazon S3 への既存の接続がある場合、Amazon S3 ゲートウェイエンドポイントを追加すると接続が一時的に中断される場合があります。この中断を回避するには、Amazon S3 ゲートウェイエンドポイントを使用する新しい VPC を作成してから、Amazon ECS クラスターとそのコンテナを新しい VPC に移行します。
- 初めてプルスルーキャッシュルールを使用してイメージをプルするとき、 AWS PrivateLink を使って、インターフェイス VPC エンドポイントを使用するように Amazon ECR を設定した場合、NAT ゲートウェイを使用して、同じ VPC 内にパブリックサブネットを作成し、プルが機能するように、プライベートサブネットから NAT ゲートウェイへのすべてのアウトバウンドトラフィックをインターネットにルーティングする必要があります。その後のイメージプルでは、これ

は必要ありません。詳細については、Amazon Virtual Private Cloud ユーザーガイドの「シナリオ: プライベートサブネットからインターネットにアクセスする」を参照してください。

Windows イメージに関する考慮事項

Windows オペレーティングシステムに基づくイメージには、ライセンスによって配布が制限されているアーティファクトが含まれます。デフォルトでは、Windows イメージを Amazon ECR リポジトリにプッシュすると、これらのアーティファクトを含むレイヤーは外部レイヤーと見なされるため、プッシュされません。アーティファクトが Microsoft によって提供されている場合、外部レイヤーは Microsoft Azure インフラストラクチャから取得されます。このため、コンテナがこれらの外部レイヤーを Azure からプルできるようにするには、VPC エンドポイントを作成する以外に、追加のステップが必要です。

Docker デーモンの `--allow-nondistributable-artifacts` フラグを使用して、Windowsイメージを Amazon ECR にプッシュするときに、この動作をオーバーライドすることができます。有効にすると、このフラグはライセンスされたレイヤーを Amazon ECR にプッシュします。これにより、Azure への追加アクセスを必要とすることなく、これらのイメージを VPC エンドポイント経由で Amazon ECR からプルすることができます。

> **Important**
>
> この `--allow-nondistributable-artifacts` フラグを使用しても、Windows コンテナベースイメージライセンスの条項に従う義務が排除されるわけではありません。したがって、パブリックまたはサードパーティーによる再配布のために Windows コンテンツを投稿することはできません。お客様自身の環境内での使用は許可されています。

Docker インストールでこのフラグを使用できるようにするには、Docker デーモン設定ファイルを変更する必要があります。通常は Docker インストールに応じて、[Docker エンジン] セクションの設定または環境設定メニューで設定するか、直接 `C:\ProgramData\docker\config\daemon.json` ファイルを編集できます。

以下に示しているのは、必要な設定の例です。イメージをプッシュするリポジトリ URI に値を置き換えます。

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
```

```
    ]
}
```

Docker デーモン設定ファイルを変更したら、イメージをプッシュする前に Docker デーモンを再起動する必要があります。ベースレイヤーがリポジトリにプッシュされたことを確認して、プッシュが成功したことを確認します。

> **Note**
>
> Windows イメージのベースレイヤーは大きくなります。レイヤーサイズにより、プッシュ時間が長くなり、Amazon ECR でのこれらのイメージのストレージコストが増大します。これらの理由から、このオプションは、構築時間と継続的なストレージコストを削減することが厳密に要求される場合にのみ使用することをお勧めします。例えば、`mcr.microsoft.com/windows/servercore` イメージを Amazon ECR で圧縮すると、約 1.7 GiB のサイズになります。

Amazon ECR の VPC エンドポイントを作成する

Amazon ECR サービスの VPC エンドポイントを作成するには、Amazon VPC ユーザーガイドのインターフェイスエンドポイント作成の手順を使用します。

Amazon EC2 インスタンスでホストされる Amazon ECS タスクの場合、Amazon ECR エンドポイントと Amazon S3 ゲートウェイエンドポイントの両方が必要です。

プラットフォームバージョン `1.4.0` 以降を使用する Fargate でホストされる Amazon ECS タスクの場合、Amazon ECR VPC エンドポイントと Amazon S3 ゲートウェイエンドポイントの両方が必要です。

プラットフォームバージョン `1.3.0` 以前を使用する Fargate でホストされる Amazon ECS タスクの場合、必要なのは com.amazonaws.***region***.ecr.dkr Amazon ECR VPC エンドポイントと Amazon S3 ゲートウェイエンドポイントのみです。

> **Note**
>
> エンドポイントが作成される順序は重要ではありません。

com.amazonaws.*region*.ecr.dkr

このエンドポイントは、Docker Registry API に使用されます。push や pull などの Docker クライアントコマンドでは、このエンドポイントが使用されます。

このエンドポイントを作成する際に、プライベート DNS ホスト名を有効にする必要があります。 これを行うには、VPC エンドポイントを作成するときに、Amazon VPC コンソールで [プライベート DNS 名を有効にする] オプションが選択されていることを確認します。

com.amazonaws.*region*.ecr.api

> **Note**
>
> 指定されたリージョンは、米国東部 (オハイオ) リージョンの など、Amazon ECR で AWS サポートされているリージョンus-east-2のリージョン識別子を表します。

このエンドポイントは、Amazon ECR API への呼び出しに使用されます。DescribeImages や CreateRepository などの API アクションは、このエンドポイントに移動します。

このエンドポイントを作成すると、プライベート DNS ホスト名を有効にするオプションが使用可能になります。VPC エンドポイントの作成時に VPC コンソールで [プライベート DNS 名を有効にする] を選択して、この設定名を有効にします。VPC エンドポイントのプライベート DNS ホスト名を有効にする場合は、SDK または を最新バージョン AWS CLI に更新して、SDK または AWS CLI を使用するときにエンドポイント URL を指定する必要はありません。

プライベート DNS ホスト名を有効にし、2019 年 1 月 24 日より前にリリースされた SDK または AWS CLI バージョンを使用している場合は、 --endpoint-urlパラメータを使用してインターフェイスエンドポイントを指定する必要があります。次の例は、エンドポイント URL の形式を示しています。

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

VPC エンドポイントでプライベート DNS ホスト名を有効にしない場合は、インターフェイスエンドポイントで VPC エンドポイント ID を指定する --endpoint-url パラメータを使用する必要があります。次の例は、エンドポイント URL の形式を示しています。

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

Amazon S3 ゲートウェイエンドポイントを作成する

Amazon ECS タスクで Amazon ECR からプライベートイメージをプルするには、Amazon S3 のゲートウェイエンドポイントを作成する必要があります。Amazon ECR は Amazon S3 を使用してイメージレイヤーを保存するため、ゲートウェイエンドポイントが必要です。Amazon ECR からイメージをダウンロードするコンテナは、Amazon ECR にアクセスしてイメージマニフェストを取得してから Amazon S3 にアクセスして実際のイメージレイヤーをダウンロードする必要があります。各 Docker イメージのレイヤーを含む Amazon S3 バケットの Amazon リソースネーム (ARN) を以下に示します。

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

Amazon ECR に以下の Amazon S3 ゲートウェイエンドポイントを作成するには、Amazon VPC ユーザーガイドのゲートウェイエンドポイントの作成手順を使用します。エンドポイントを作成するときは、必ず VPC のルートテーブルを選択してください。

com.amazonaws.*region*.s3

Amazon S3 ゲートウェイエンドポイントは IAM ポリシードキュメントを使用してサービスへのアクセスを制限します。 このポリシーには、タスクの IAM ロールまたはその他の IAM ユーザーポリシーに設定された制限が引き続き優先して適用されるので、フルアクセスポリシーを使用できます。Amazon S3 バケットアクセスを Amazon ECR を使用するための最小限のアクセス許可に制限する場合は、「Amazon ECR の最小 Amazon S3 バケットアクセス許可」を参照してください。

Amazon ECR の最小 Amazon S3 バケットアクセス許可

Amazon S3 ゲートウェイエンドポイントは IAM ポリシードキュメントを使用してサービスへのアクセスを制限します。 Amazon ECR に最低限の Amazon S3 バケットアクセス許可のみを許可するには、エンドポイントの IAM ポリシードキュメントを作成するときに Amazon ECR が使用する Amazon S3 バケットへのアクセスを制限します。

次の表は、Amazon ECR に必要な Amazon S3 バケットポリシーアクセス許可を示しています。

アクセス許可	説明
`arn:aws:s3:::prod-region-starport-layer-bucket/*`	各 Docker イメージのレイヤーを含む Amazon S3 バケットへのアクセスを提供します。米国東部 (オハイオ) リージョンの `us-east-2` の

アクセス許可	説明
	ように、Amazon ECR でサポートされている AWS リージョンのリージョン識別子を表します。

例

以下の例は、Amazon ECR オペレーションに必要な Amazon S3 バケットへのアクセスを提供する方法を示しています。

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

Logs CloudWatch エンドポイントを作成する

awslogs ログドライバーを使用してログ情報を Logs に送信するインターネットゲートウェイのない VPC を使用する Fargate 起動タイプを使用する Amazon ECS CloudWatch タスクでは、ログの com.amazonaws.*region* .logs インターフェイス VPC CloudWatch エンドポイントを作成する必要があります。詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「インターフェイス VPC エンドポイントでのログの使用」を参照してください。 CloudWatch

Amazon ECR VPC エンドポイントのエンドポイントポリシーを作成する

VPC エンドポイントポリシーは、エンドポイントの作成時または変更時にエンドポイントに加える国際機械技術者協会 (IAM) のリソースポリシーです。エンドポイントの作成時にポリシーをアタッチしない場合、 はサービスへのフルアクセスを許可するデフォルトのポリシーをア AWS タッチします。エンドポイントポリシーは、 ユーザーポリシーやサービス固有のポリシーを上書き、または置き換えません。これは、評価項目から指定されたサービスへのアクセスを制御するための別の

ポリシーです。評価項目のポリシーは、JSON形式で記載する必要があります。詳細については、「Amazon VPCユーザーガイド」の「VPC評価項目によるサービスのアクセス制御」を参照してください。

1 つの IAM リソースポリシーを作成し、両方の Amazon ECR VPC エンドポイントにアタッチすることをお勧めします。

Amazon ECR API のエンドポイントポリシーの例を次に示します。このポリシーは、特定の IAM ロールが Amazon ECR からイメージをプルできるようにします。

```
{
 "Statement": [{
  "Sid": "AllowPull",
  "Principal": {
   "AWS": "arn:aws:iam::1234567890:role/role_name"
  },
  "Action": [
   "ecr:BatchGetImage",
   "ecr:GetDownloadUrlForLayer",
                  "ecr:GetAuthorizationToken"
  ],
  "Effect": "Allow",
  "Resource": "*"
 }]
}
```

次のエンドポイントポリシーの例では、指定されたリポジトリが削除されないようにしています。

```
{
 "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"
  }
```

```
 ]
}
```

次のエンドポイントポリシーの例では、前述の 2 つの例を 1 つのポリシーにまとめています。

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

Amazon ECR の VPC エンドポイントポリシーを変更するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. ナビゲーションペインで、[Endpoints] (エンドポイント) を選択します。
3. Amazon ECR の VPC エンドポイントをまだ作成していない場合は、「Amazon ECR の VPC エンドポイントを作成する」を参照してください。

4. ポリシーを追加する Amazon ECR VPC エンドポイントを選択し、画面の下部にある [ポリシー] タブを選択します。
5. [ポリシーの編集] を選択してポリシーを変更します。
6. [保存] を選択してポリシーを保存します。

共有サブネット

自分と共有されているサブネットで VPC エンドポイントを作成、説明、変更、または削除することはできません。ただし、VPC エンドポイントを使用することはできます。

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐために、 AWS には、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールが用意されています。

リソースポリシー内では `aws:SourceArn` または `aws:SourceAccount` のグローバル条件コンテキストキーを使用して、Amazon ECR が別のサービスに付与する、リソースへのアクセス許可を制限することをお勧めします。クロスサービスアクセスにリソースを 1 つだけ関連付けたい場合は、`aws:SourceArn` を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、`aws:SourceAccount` を使用します。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、`aws:SourceArn` グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー `aws:SourceArn` で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、`arn:aws:`*`servicename`*`:`*`region`*`:`*`123456789012`*`:*` です。

`aws:SourceArn` の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

`aws:SourceArn` の値は ResourceDescription である必要があります。

次の例は、Amazon ECR リポジトリポリシーで aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、そのサービスとの統合に必要な Amazon ECR API アクション AWS CodeBuild へのアクセスを許可する方法と、混乱した代理問題を回避する方法を示しています。

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

Amazon ECR のモニタリング

Amazon ECR API の使用状況を Amazon でモニタリングできます。Amazon は CloudWatch、Amazon ECR から raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 2 週間記録されるため、履歴情報にアクセスして API の使用状況を把握できます。Amazon ECR メトリクスデータは 1 分 CloudWatch 間隔で に自動的に送信されます。の詳細については CloudWatch、「Amazon ユーザーガイド CloudWatch 」を参照してください。

Amazon ECR には、認可、イメージプッシュ、およびイメージプルアクションの API の使用状況に基づいたメトリクスが用意されています。

モニタリングは、Amazon ECR および AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、 AWS ソリューションを構成するリソースからモニタリングデータを収集することをお勧めします。ただし、Amazon ECR のモニタリングを開始する前に、以下の質問に対する回答を反映したモニタリング計画を作成する必要があります。

- モニタリングの目的は何ですか?
- どのリソースをモニタリングしますか?
- どのくらいの頻度でこれらのリソースをモニタリングしますか?
- どのモニタリングツールを利用しますか?
- 誰がモニタリングタスクを実行しますか?
- 問題が発生したときに誰が通知を受け取りますか?

次のステップでは、さまざまなタイミングと負荷条件でパフォーマンスを測定することにより、お客様の環境で通常の Amazon ECR のパフォーマンスのベースラインを確定します。Amazon ECR のモニタリングでは、過去のモニタリングデータを保存し、新しいパフォーマンスデータと比較することで、パフォーマンスの通常パターンと異常パターンを特定し、問題に対処する方法を考案できます。

トピック

- サービスクォータの可視化とアラームの設定
- Amazon ECR 使用状況メトリクス
- Amazon ECR 使用状況レポート

- Amazon ECR リポジトリメトリクス
- Amazon ECR イベントと EventBridge
- を使用した Amazon ECR アクションの口グ記録 AWS CloudTrail

サービスクォータの可視化とアラームの設定

CloudWatch コンソールを使用してサービスクォータを視覚化し、現在の使用状況とサービスクォータの比較を確認できます。クォータに近づいたときに通知されるようにアラームを設定することもできます。

サービスクォータを可視化し、オプションでアラームを設定するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで メトリクスを選択します。
3. [All metrics] (すべてのメトリクス) タブで [Usage] (使用状況) を選択し、[By AWS Resource] (AWS リソース別) を選択します。

   サービスクォータ使用状況メトリクスのリストが表示されます。
4. いずれかのメトリクスの横にあるチェックボックスをオンにします。

   グラフには、その AWS リソースの現在の使用状況が表示されます。
5. サービスクォータをグラフに追加するには、次の手順を実行します。

   a. [グラフ化したメトリクス] タブを選択します。

   b. [Math expression (数式)]、[Start with an empty expression (空の式で開始)] の順に選択します。次に、新しい行の [Details (詳細)] に「**SERVICE_QUOTA(m1)**」と入力します。

      グラフに新しい行が追加され、メトリクスで表されるリソースのサービスクォータが表示されます。
6. 現在の使用状況をクォータの割合として表示するには、新しい式を追加するか、現在の [SERVICE_QUOTA] 式を変更します。新しい式には、**m1/60/SERVICE_QUOTA(m1)*100** を使用します。
7. （オプション）サービスクォータに近づいた場合に通知するアラームを設定するには、次の手順を実行します。

   a. **m1/60/SERVICE_QUOTA(m1)*100** 行の [Actions (アクション)] で、アラームアイコンを選択します。それはベルのように見えます。

アラーム作成ページが表示されます。

b. [Conditions (条件)] で、[Threshold type (しきい値の種類)] が [Static (静的)] で、[Whenever Expression1 is (式 1)] が [Greater (大きい)] に設定されていることを確認します。[than (以上)] に「**80**」と入力します。これにより、使用量がクォータの 80% を超えたときに ALARM 状態になるアラームが作成されます。

c. [次へ] をクリックします。

d. 次のページで、Amazon SNS トピックを選択するか、新しいトピックを作成します。このトピックは、アラームが ALARM 状態になったときに通知されます。次いで、[次へ] を選択します。

e. 次のページで、アラームの名前と説明を入力し、[Next] (次へ) を選択します。

f. [アラームを作成] を選択します。

Amazon ECR 使用状況メトリクス

CloudWatch 使用状況メトリクスを使用して、アカウントのリソース使用状況を可視化できます。これらのメトリクスを使用して、 CloudWatch グラフとダッシュボードで現在のサービス使用状況を視覚化します。

Amazon ECR 使用状況メトリクスは、 AWS サービスクォータに対応しています。使用量がサービスクォータに近づいたときに警告するアラームを設定することもできます。Amazon ECR のサービスのクォータの詳細については、「Amazon ECR のサービスクォータ」を参照してください。

Amazon ECR は、`AWS/Usage` 名前空間に以下のメトリクスを公開します。

メトリクス	説明
`CallCount`	アカウントからの API アクションの呼び出し回数。リソースは、メトリクスに関連付けられたディメンションによって定義されます。 このメトリクスの最も便利な統計は `SUM` であり、定義した期間中のすべての寄稿者からの値の合計を表します。

次のディメンションは、Amazon ECR によって発行される使用状況メトリクスを絞り込むために使用されます。

ディメンション	説明
`Service`	リソースを含む AWS サービスの名前。Amazon ECR 使用状況メトリクスの場合、このディメンションの値は `ECR` です。
`Type`	報告されるエンティティタイプ。現在、Amazon ECR 使用状況メトリクスの有効な値は `API` のみです。
`Resource`	実行中のリソースタイプ。現在、Amazon ECR は以下の API アクションの API の使用状況に関する情報を返します。 • GetAuthorizationToken • BatchCheckLayerAvailability • InitiateLayerUpload • UploadLayerPart • CompleteLayerUpload • PutImage • BatchGetImage • GetDownloadUrlForLayer
`Class`	追跡されているリソースのクラス。現在、Amazon ECR はクラスディメンションを使用していません。

Amazon ECR 使用状況レポート

AWS には、Amazon ECR リソースのコストと使用状況を分析できる Cost Explorer と呼ばれる無料のレポートツールが用意されています。

Cost Explorer を使用して、使用状況とコストのグラフを表示します。過去 13 か月からデータを表示でき、また次の 3 か月間にどのくらい使用する可能性があるかを予測します。時間の経過に伴う AWS リソースの使用量パターンを確認して、さらに照会が必要な分野を識別し、コストの把握に役立つ傾向を確認するには、Cost Explorer を使用します。データの時間範囲を指定したり、時間データを日または月ごとに表示したりもできます。

コストおよび使用状況レポートの計測データには、すべての Amazon ECR リポジトリの使用状況が示されます。詳細については、「請求用のリソースにタグを付ける」を参照してください。

AWS コストと使用状況レポートの作成の詳細については、「 ユーザーガイド」のAWS 「コストと使用状況レポートAWS Billing 」を参照してください。

Amazon ECR リポジトリメトリクス

Amazon ECR は、リポジトリのプルカウントメトリクスを Amazon に送信します CloudWatch。Amazon ECR メトリクスデータは 1 分 CloudWatch 間隔で自動的に に送信されます。の詳細については CloudWatch、「Amazon ユーザーガイド CloudWatch 」を参照してください。

トピック

- CloudWatch メトリクスの有効化
- 使用できるメトリクスとディメンション
- コンソール CloudWatchを使用した Amazon ECR メトリクスの表示

CloudWatch メトリクスの有効化

Amazon ECR は、すべてのリポジトリのリポジトリメトリクスを自動的に送信します。手動の手順を実行する必要はありません。

使用できるメトリクスとディメンション

以下のセクションでは、Amazon ECR が Amazon に送信するメトリクスとディメンションを一覧表示します CloudWatch。

Amazon ECR メトリクス

Amazon ECR には、リポジトリをモニタリングするためのメトリクスが用意されています。プルカウントを測定できます。

AWS/ECR 名前空間には、次のメトリクスが含まれます。

`RepositoryPullCount`

リポジトリ内のイメージのプルの総数。

有効なディメンション: `RepositoryName`。

有効な統計: 平均、最小、最大、合計、サンプル数。最も有用な統計は Sum です。

単位: 整数。

Amazon ECR メトリクスのディメンション

Amazon ECR メトリクスは AWS/ECR 名前空間を使用し、以下のディメンションのメトリクスを提供しています。

RepositoryName

このディメンションは、指定したリポジトリのすべてのコンテナイメージに対してリクエストしたデータをフィルターします。

コンソール CloudWatchを使用した Amazon ECR メトリクスの表示

Amazon ECR リポジトリのメトリクスは、 CloudWatch コンソールで表示できます。 CloudWatch コンソールには、 リソースをきめ細かくカスタマイズして表示できます。詳細については、「Amazon ユーザーガイド CloudWatch 」を参照してください。

CloudWatch コンソールでメトリクスを表示するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[メトリクス]、[すべてのメトリクス] を選択します。
3. リポジトリの [参照] タブの [AWS 名前空間] で [ECR] を選択します。
4. 表示するメトリクスを選択します。リポジトリメトリクスは、[ECR > リポジトリメトリクス] としてスコープされます。

Amazon ECR イベントと EventBridge

Amazon EventBridge では、 AWS サービスを自動化し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。 AWS サービスからのイベントは、ほぼリアルタイムで EventBridge に配信されます。簡単なルールを作成し、ルールで対象とするイベントを指定し、イベントがルールに一致した場合に自動的に実行するアクションを含めることができます。自動的にトリガーできるオペレーションには、以下が含まれます。

- CloudWatch Logs でのロググループへのイベントの追加
- AWS Lambda 関数の呼び出し
- Amazon EC2 Run Command の呼び出し

- Amazon Kinesis Data Streams へのイベントの中継
- AWS Step Functions ステートマシンのアクティブ化
- Amazon SNS トピックまたは Amazon SQS キューの通知

詳細については、「Amazon ユーザーガイド」の「Amazon の開始 EventBridge方法」を参照してください。 EventBridge

Amazon ECR のサンプルイベント

以下に、Amazon ECR のイベントの例を示します。イベントは、ベストエフォートベースで発生します。

完了したイメージプッシュのイベント

各イメージプッシュが完了すると、以下のイベントが送信されます。詳細については、「Docker イメージを Amazon ECR プライベートリポジトリにプッシュする」を参照してください。

```
{
    "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"
    }
}
```

プルスルーキャッシュアクションのイベント

プルスルーキャッシュアクションが試行されると、次のイベントが送信されます。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。

```
{
    "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",
    }
}
```

完了したイメージスキャンのイベント (ベーシックスキャン)

レジストリでベーシックスキャンを有効にすると、各イメージスキャンが完了した際、以下のイベントが送信されます。finding-severity-counts パラメータは、重要度レベルが存在する場合にのみ、その値を返します。たとえば、イメージに CRITICAL レベルの結果が含まれていない場合、重要度のカウントは返されません。詳細については、「Amazon ECR で OS の脆弱性がないかイメージをスキャンする」を参照してください。

> **Note**
>
> 拡張スキャンが有効なときに Amazon Inspector によって発行されるイベントの詳細については、EventBridge Amazon ECR で拡張スキャンのために送信されるイベント を参照してください。

```
{
    "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": []
    }
}
```

拡張スキャンが有効なリソースについての変更通知のイベント (拡張スキャン)

レジストリで拡張スキャンが有効になっている場合、拡張スキャンが有効になっているリソースで変更があった際、Amazon ECR から次のイベントが送信されます。これには、新しいリポジトリの作成、リポジトリのスキャン頻度の変更、または拡張スキャンが有効になっているリポジトリでのイメージの作成または削除が含まれます。詳しくは、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。

```
{
 "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"
 }
}
```

イメージ削除のイベント

イメージが削除されると、以下のイベントが送信されます。詳細については、「Amazon ECR でのイメージの削除」を参照してください。

```
{
    "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"
    }
```

```
}
```

を使用した Amazon ECR アクションのログ記録 AWS CloudTrail

Amazon ECR は AWS CloudTrail、Amazon ECR のユーザー、ロール、または サービスによって実行されたアクションを記録する AWS サービスである と統合されています。 は、次の Amazon ECR アクションをイベントとして CloudTrail キャプチャします。

- Amazon ECR コンソールからのコールを始めとするすべての API コール
- リポジトリの暗号化設定によって実行されたすべてのアクション
- 成功アクションと失敗アクションの両方を含む、ライフサイクルポリシールールによって実行されるすべてのアクション

> **Important**
>
> 個々の CloudTrail イベントのサイズ制限により、10 個以上のイメージの有効期限が切れているライフサイクルポリシーアクションの場合、Amazon ECR は複数のイベントを に送信します CloudTrail。また、Amazon ECR には、イメージごとに最大 100 個のタグが含まれます。

証跡が作成されると、Amazon ECR の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。 Amazon S3 証跡を設定しない場合でも、 CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。この情報を使用して、Amazon ECR に対して行われたリクエスト、リクエストの作成元の IP アドレス、リクエストの実行者、リクエストの実行日時などの詳細を特定できます。

詳細については、『AWS CloudTrail ユーザーガイド』を参照してください。

の Amazon ECR 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。Amazon ECR でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴を使用した CloudTrail イベントの表示」を参照してください。

Amazon ECR のイベントなど、 AWS アカウント内のイベントの継続的な記録については、証跡を作成します。証跡により CloudTrail 、 はログファイルを Amazon S3 バケットに配信できます。コンソールで証跡を作成する場合、証跡を単一のリージョンまたはすべてのリージョンに適用できます。証跡は、 AWS パーティション内のイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベントデータを分析して処理するように他の AWS サービスを設定できます。詳細については、以下を参照してください。

- AWS アカウントの証跡の作成
- AWS CloudTrail ログと サービスの統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

すべての Amazon ECR API アクションは によってログに記録 CloudTrail され、「Amazon Elastic Container Registry API Reference」に記載されています。一般的なタスクを実行すると、そのタスクの一部である各 API アクションの CloudTrail ログファイルにセクションが生成されます。例えば、リポジトリを作成すると、`GetAuthorizationToken`、`CreateRepository`および `SetRepositoryPolicy`セクションが CloudTrail ログファイルに生成されます。イメージをリポジトリにプッシュすると、`InitiateLayerUpload`、`UploadLayerPart`、`CompleteLayerUpload`、`PutImage` のセクションが生成されます。イメージをプルすると、`GetDownloadUrlForLayer` と `BatchGetImage` のセクションが生成されます。これらの一般的なタスクの例については、「CloudTrail ログエントリの例」を参照してください。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます:

- リクエストが、ルートと ユーザー認証情報のどちらを使用して送信されたか
- リクエストが、ロールとフェデレーティッドユーザーのどちらの一時的なセキュリティ認証情報を使用して送信されたか
- リクエストが別の AWS サービスによって行われたかどうか

詳細については、「 CloudTrail要素`userIdentity`」を参照してください。

Amazon ECR ログファイルエントリの理解

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータ、およびその他の情報が含まれます。 CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

CloudTrail ログエントリの例

以下は、いくつかの一般的な Amazon ECR タスクの CloudTrail ログエントリの例です。

> **Note**
>
> これらの例は、読みやすくするために書式設定されています。 CloudTrail ログファイルでは、すべてのエントリとイベントが 1 行に連結されます。さらに、この例は 1 つの Amazon ECR エントリに限定しています。実際の CloudTrail ログファイルには、複数の AWS サービスからのエントリとイベントが表示されます。

トピック

例: リポジトリの作成アクション

次の例は、 `CreateRepository`アクションを示す CloudTrail ログエントリを示しています。

```
{
    "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"
}
```

例： AWS KMS Amazon ECR リポジトリを作成するときの CreateGrant API アクション

次の例は、KMS 暗号化を有効にして Amazon ECR リポジトリを作成するときの CreateGrant アクションを示す CloudTrail AWS KMS ログエントリを示しています。KMS 暗号化を有効にして作成されたリポジトリごとに、 に 2 つの CreateGrant ログエントリが表示されます 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"
}
```

例: イメージプッシュアクション

次の例は、 PutImageアクションを使用するイメージプッシュを示す CloudTrail ログエントリを示しています。

> **Note**
>
> イメージをプッシュすると、 CloudTrail ログに InitiateLayerUpload、UploadLayerPart、および CompleteLayerUploadリファレンスも表示されます。

```
{
    "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": "203.0.113.12",
 "userAgent": "console.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"
}
```

例: イメージプルアクション

次の例は、 BatchGetImageアクションを使用するイメージプルを示す CloudTrail ログエントリを示しています。

> **Note**
>
> イメージをプルするときに、イメージがローカルにまだない場合は、 CloudTrail ログにもGetDownloadUrlForLayer参照が表示されます。


```
{
    "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": "203.0.113.12",
 "userAgent": "console.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"
}
```

例: イメージライフサイクルポリシーアクション

次の例は、ライフサイクルポリシールールが原因でイメージの有効期限が切れたときを示す CloudTrail ログエントリを示しています。このイベントタイプは、イベント名フィールドの `PolicyExecutionEvent` をフィルタリングすることによって検索できます。

> **Important**
>
> 個々の CloudTrail イベントのサイズ制限により、10 個以上のイメージの有効期限が切れているライフサイクルポリシーアクションの場合、Amazon ECR は複数のイベントを に送信します CloudTrail。また、Amazon ECR には、イメージごとに最大 100 個のタグが含まれます。

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

AWS SDK での Amazon ECR の使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

AWS SDKsコード例

次のコード例は、 AWS Software Development Kit (SDK) で Amazon ECR を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon ECR の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- AWS SDKsアクション
  - AWS SDK または CLI DescribeRepositoriesで を使用する
  - AWS SDK または CLI ListImagesで を使用する

AWS SDKsアクション

次のコード例は、 AWS SDKs を使用して個々の Amazon ECR アクションを実行する方法を示しています。これらの抜粋は Amazon ECR API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコードの抜粋です。各例には GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「Amazon Elastic Container Registry (Amazon ECR) API Reference」を参照してください。

例

- AWS SDK または CLI DescribeRepositoriesで を使用する
- AWS SDK または CLI ListImagesで を使用する

AWS SDK または CLI **DescribeRepositories**で を使用する

以下のコード例は、`DescribeRepositories` の使用方法を示しています。

CLI

AWS CLI

レジストリ内のリポジトリを記述するには

この例は、アカウントのデフォルトレジストリ内のリポジトリを記述します。

コマンド:

```
aws ecr describe-repositories
```

出力:

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

- API の詳細については、「 コマンドリファレンスDescribeRepositories」の「」を参照してください。 AWS CLI

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、 DescribeRepositories AWS SDK for Rust API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon ECR の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListImages**で を使用する

以下のコード例は、ListImages の使用方法を示しています。

CLI

AWS CLI

リポジトリ内のイメージを一覧表示するには

次の list-images の例は、cluster-autoscaler リポジトリ内のイメージのリストを表示します。

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

出力:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
        }
    ]
}
```

- API の詳細については、「 コマンドリファレンスListImages」の「」を参照してください。 AWS CLI

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、 ListImages AWS SDK for Rust API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon ECR の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon ECR のサービスクォータ

次の表に、Amazon Elastic Container Registry (Amazon ECR) のデフォルトのサービスクォータを示します。

名前	デフォルト	引き上げ可能	説明
レプリケーション設定のルールごとのフィルター数	サポートされている各リージョン: 100	いいえ	レプリケーション設定のルールごとのフィルターの最大数。
リポジトリあたりのイメージ数	サポートされている各リージョン: 10,000	はい	リポジトリあたりのイメージの最大数。
レイヤーパート	サポートされている各リージョン: 4,200	いいえ	レイヤーパートの最大数これは、Amazon ECR API アクションを直接使用して、イメージプッシュオペレーションのマルチパートアップロードを開始している場合にのみ適用されます。
ライフサイクルポリシーの長さ	サポートされている各リージョン: 30,720	いいえ	ライフサイクルポリシーの最大文字数。
レイヤーパートの最大サイズ	サポートされている各リージョン: 10	いいえ	レイヤーパートの最大サイズ (MiB)。これは、Amazon ECR API アクションを直接使用して、

名前	デフォルト	引き上げ可能	説明
			イメージプッシュオペレーションのマルチパートアップロードを開始している場合にのみ適用されます。
レイヤーの最大サイズ	サポートされている各リージョン: 52,000	いいえ	レイヤーの最大サイズ (MiB)。
レイヤーパートの最小サイズ	サポートされている各リージョン : 5	いいえ	レイヤーパートの最小サイズ (MiB)。これは、Amazon ECR API アクションを直接使用して、イメージプッシュオペレーションのマルチパートアップロードを開始している場合にのみ適用されます。
レジストリーごとのプルスルーキャッシュルール	サポートされている各リージョン: 50	いいえ	プルスルーキャッシュルールの最大数。

名前	デフォルト	引き上げ可能	説明
BatchCheckLayerAvailability リクエストのレート	サポートされている各リージョン: 1,000/秒	はい	現在のリージョンで 1 秒あたりに実行できる BatchCheckLayerAvailability リクエストの最大数。イメージがリポジトリにプッシュされると、イメージレイヤーごとに以前にアップロードされたかどうかが確認されます。アップロードされている場合、そのイメージレイヤーはスキップされます。
BatchGetImage リクエストのレート	サポートされている各リージョン: 2,000/秒	はい	現在のリージョンで 1 秒あたりに実行できる BatchGetImage リクエストの最大数。イメージがプルされると、BatchGetImage API が 1 回呼び出され、イメージマニフェストが取得されます。この API のクォータの引き上げをリクエストする場合、GetDownloadUrlForLayer の使用状況も確認してください。

名前	デフォルト	引き上げ可能	説明
CompleteLayerUpload リクエストのレート	サポートされている各リージョン: 100/秒	はい	現在のリージョンで 1 秒あたりに実行できる CompleteLayerUpload リクエストの最大数。イメージがプッシュされると、新しいイメージレイヤーごとに CompleteLayerUpload API が 1 回呼び出されて、アップロードが完了したことが確認されます。
GetAuthorizationToken リクエストのレート	サポートされている各リージョン: 500/秒	はい	現在のリージョンで 1 秒あたりに実行できる GetAuthorizationToken リクエストの最大数。

名前	デフォルト	引き上げ可能	説明
GetDownloadUrlForLayer リクエストのレート	サポートされている各リージョン: 3,000/秒	はい	現在のリージョンで 1 秒あたりに実行できる GetDownloadUrlForLayer リクエストの最大数。イメージがプルされると、まだキャッシュされていないイメージレイヤーごとに GetDownloadUrlForLayer API が 1 回呼び出されます。この API のクォータの引き上げをリクエストする場合、BatchGetImage の使用状況も確認してください。

名前	デフォルト	引き上げ可能	説明
InitiateLayerUpload リクエストのレート	サポートされている各リージョン: 100/秒	はい	現在のリージョンで 1 秒あたりに実行できる InitiateLayerUpload リクエストの最大数。イメージがプッシュされると、まだアップロードされていないイメージレイヤーごとに InitiateLayerUpload API が 1 回呼び出されます。イメージレイヤーがアップロードされたかどうかは、BatchCheckLayerAvailability API アクションによって決定されます。
PutImage リクエストのレート	サポートされている各リージョン: 10/秒	はい	現在のリージョンで 1 秒あたりに実行できる PutImage リクエストの最大数。イメージがプッシュされ、すべての新しいイメージレイヤーがアップロードされると、PutImage API が 1 回呼び出され、イメージマニフェスト、およびイメージに関連付けられたタグが作成または更新されます。

名前	デフォルト	引き上げ可能	説明
UploadLayerPart リクエストのレート	サポートされている各リージョン: 500/秒	はい	現在のリージョンで 1 秒あたりに実行できる UploadLayerPart リクエストの最大数。イメージがプッシュされると、新しい各イメージレイヤーがいくつかに分けてアップロードされ、UploadLayerPart API が新しいイメージレイヤーパートごとに 1 回呼び出されます。
イメージスキャンのレート	サポートされている各リージョン: 1	いいえ	24 時間あたりのイメージスキャンの最大数。
登録済みリポジトリ	サポートされている各リージョン: 10,000	はい	このアカウントで現在のリージョンに作成できるリポジトリの最大数。
ライフサイクルポリシーあたりのルール	サポートされている各リージョン: 50	いいえ	ライフサイクルポリシーのルールの最大数
レプリケーション設定ごとのルール	サポートされている各リージョン: 10	いいえ	レプリケーション設定のルールの最大数。

名前	デフォルト	引き上げ可能	説明
イメージあたりのタグ	サポートされている各リージョン: 1,000	いいえ	イメージあたりのタグの最大数。
レプリケーション設定のすべてのルール全体における一意のレプリケート先	サポートされている各リージョン: 25	いいえ	レプリケーション設定のすべてのルール全体における一意のレプリケート先の最大数。

AWS Management Console での Amazon ECR サービスクォータの管理

Amazon ECR は、Service Quotas と統合されています。Service Quotas は、クォータを一元的な場所から表示および管理できる AWS サービスです。詳細については、「Service Quotas ユーザーガイド」の「Service Quotas とは」を参照してください。

Service Quotas を使用すると、すべての Amazon ECR サービスクォータの値を簡単に検索できます。

Amazon ECR のサービスクォータを表示するには (AWS Management Console)

1. https://console.aws.amazon.com/servicequotas/ で Service Quotas コンソールを開きます。
2. ナビゲーションペインで、[AWS サービス] を選択します。
3. [AWS のサービス] リストから [Amazon Elastic Container Registry (Amazon ECR)] を選択します。

   [Service Quotas] の一覧には、サービスクォータ名、適用された値 (使用可能な場合)、AWS デフォルトのクォータ、クォータ値が調整可能かどうかが表示されます。
4. 説明など、Service Quotas に関する追加情報を表示するには、クォータ名を選択します。

クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト を参照してください。

API 使用量メトリクスをモニタリングするための CloudWatch アラームの作成

Amazon ECR には、レジストリ認証、イメージプッシュ、イメージプルアクションに関連する各 API の AWS の サービスクォータに対応する CloudWatch 使用量メトリクスが用意されています。Service Quotas コンソールでは、使用状況をグラフで可視化し、使用量がサービスクォータに近づくと警告するアラームを設定できます。詳細については、「Amazon ECR 使用状況メトリクス」を参照してください。

以下の手順を使用して、Amazon ECR API 使用量メトリクスのいずれかに基づいて CloudWatch アラームを作成します。

Amazon ECR 使用量クォータに基づいてアラームを作成するには (AWS Management Console)

1. https://console.aws.amazon.com/servicequotas/ で Service Quotas コンソールを開きます。
2. ナビゲーションペインで、[AWS サービス] を選択します。
3. [AWS のサービス] リストから [Amazon Elastic Container Registry (Amazon ECR)] を選択します。
4. [Service quotas (サービスクォータ)] リストで、アラームを作成する Amazon ECR 使用量クォータを選択します。
5. Amazon CloudWatch Events アラームのセクションで [Create] (作成) を選択します。
6. [アラームのしきい値] で、適用されたクォータ値からアラーム値として設定する値の割合を選択します。
7. [アラーム名] にアラームの名前を入力し、[Create (作成)] を選択します。

Amazon ECR のトラブルシューティング

この章では、Amazon ECR の診断情報を検索し、一般的な問題とエラーメッセージのトラブルシューティング手順について説明します。

トピック

- Amazon ECR を使用する際の Docker コマンドと問題のトラブルシューティング
- Amazon ECR エラーメッセージのトラブルシューティング

Amazon ECR を使用する際の Docker コマンドと問題のトラブルシューティング

場合によっては、Amazon ECR に対して Docker コマンドを実行すると、エラーメッセージが表示されることがあります。一般的なエラーメッセージと考えられる解決策を以下に説明します。

トピック

- Docker ログに予想されるエラーメッセージが含まれていない
- Amazon ECR レポジトリからイメージをプルするときのエラー: "ファイルシステムの検証に失敗しました" または "404: イメージが見つかりません"
- Amazon ECR からイメージをプルする際のエラー: "ファイルシステムレイヤーの検証に失敗しました"
- レポジトリへのプッシュの際の HTTP 403 エラー、または "基本的な認証情報がありません" エラー

Docker ログに予想されるエラーメッセージが含まれていない

Docker 関連の問題のデバッグを開始するには、まずホストインスタンスで実行されている Docker デーモンで Docker デバッグ出力をオンにします。Amazon ECS コンテナインスタンスで Amazon ECR からプルされたイメージを使用している場合は、「Amazon Elastic Container Service デベロッパーガイド」の「Docker デーモンからの詳細な出力の設定」を参照してください。

Amazon ECR レポジトリからイメージをプルするときのエラー: "ファイルシステムの検証に失敗しました" または "404: イメージが見つかりません"

docker pull コマンドを使用すると、Docker 1.9 以降で Amazon ECR レポジトリからイメージをプルするときにエラー `Filesystem verification failed` が表示されます。Docker 1.9 より前のバージョンを使用するときはエラー `404: Image not found` が表示される場合があります。

考えられる理由とその説明を以下に示します。

ローカルディスクがいっぱいである

docker pull を実行しているローカルディスクがいっぱいである場合、ローカルファイルで計算された SHA-1 ハッシュは、Amazon ECR で計算されたものとは異なる可能性があります。ローカルディスクに、プルしている Docker イメージを保存するのに十分な空き容量があることを確認します。新しいイメージの容量を確保するために、古いイメージを削除することもできます。docker images コマンドを使用して、ローカルにダウンロードしたすべての Docker イメージのリストとそのサイズを表示します。

ネットワークエラーにより、クライアントがリモートリポジトリに接続できない

Amazon ECR レポジトリへの呼び出しでは、インターネットへの機能している接続が必要です。ネットワーク設定を確認し、他のツールやアプリケーションがインターネット上のリソースにアクセスできることを確認します。プライベートサブネットの Amazon EC2 インスタンスで docker pull を実行している場合は、そのサブネットにインターネットへのルートがあることを確認します。ネットワークアドレス変換 (NAT) サーバーまたはマネージド NAT ゲートウェイを使用します。

現時点では、Amazon ECR リポジトリへの呼び出しでは、会社のファイアウォールから Amazon Simple Storage Service (Amazon S3) へのネットワークアクセスも必要です。組織で、サービスエンドポイントを許可するファイアウォールソフトウェアまたは NAT デバイスを使用している場合、現在のリージョンの Amazon S3 サービスエンドポイントが許可されていることを確認します。

HTTP プロキシの背後で Docker を使用している場合は、適切なプロキシ設定を使用して Docker を設定できます。詳細については、Docker ドキュメントの「HTTP プロキシ」を参照してください。

Amazon ECR からイメージをプルする際のエラー: "ファイルシステムレイヤーの検証に失敗しました"

docker pull コマンドを使用してイメージをプルするときに、エラー `image image-name not found` が表示される場合があります。Docker のログを調べると、次のようなエラーが見つかる場合があります。

```
filesystem layer verification failed for digest sha256:2b96f...
```

このエラーは、イメージの 1 つ以上のレイヤーがダウンロードに失敗したことを示します。考えられる理由とその説明を以下に示します。

古いバージョンの Docker 使用している

> このエラーは、Docker 1.10 より前のバージョンを使用している場合に、まれに発生することがあります。Docker クライアントを 1.10 以降にアップグレードします。

クライアントでネットワークエラーまたはディスクエラーが発生した

> `Filesystem verification failed` メッセージについて前に説明したように、ディスクがいっぱいであるか、ネットワークの問題により、1 つ以上のレイヤーをダウンロードできない可能性があります。上記の推奨事項に従って、ファイルシステムがいっぱいでないこと、およびネットワーク内から Amazon S3 へのアクセスを有効にしたことを確認します。

レポジトリへのプッシュの際の HTTP 403 エラー、または "基本的な認証情報がありません" エラー

aws ecr get-login-password コマンドを使用して Docker に対して正常に認証されても、`HTTP 403 (Forbidden)` エラーが発生したり、docker push コマンドまたは docker pull コマンドからのエラーメッセージ `no basic auth credentials` が表示されたりする場合があります。この問題の既知の原因をいくつか次に示します。

別のリージョンに対して認証されている

> 認証リクエストは特定のリージョンに関連付けられていて、リージョン間では使用できません。たとえば、米国西部 (オレゴン)リージョン から認証トークンを取得する場合、これを使用して、米国東部 (バージニア北部) リージョンのリポジトリに対して認証を行うことはできません。この問題を解決するには、リポジトリが存在する同じリージョンから認証トークンを取得したことを

確認してください。詳細については、「the section called “レジストリの認証”」を参照してください。

プッシュ先として認証したリポジトリに対するアクセス許可がない

プッシュ先のリポジトリに対するアクセス許可がありません。詳細については、「Amazon ECR のプライベートリポジトリポリシー」を参照してください。

トークンの有効期限が切れた。

GetAuthorizationToken オペレーションを使用して取得した認証トークンのデフォルトの有効期限は 12 時間です。

wincred 認証情報マネージャーのバグ

一部のバージョンの Docker for Windows では、wincred という認証情報マネージャーを使用します。この認証情報マネージャーは、aws ecr get-login-password によって生成された Docker ログインコマンドを正しく処理しません (詳細については、https://github.com/docker/docker/issues/22910 を参照)。出力される Docker ログインコマンドは実行できますが、イメージをプッシュまたはプルしようとすると、コマンドは失敗します。これに対処するには、aws ecr get-login-password から出力される Docker ログインコマンドのレジストリ引数から https:// スキームを削除します。HTTPS スキームのない Docker ログインコマンドの例を次に示します。

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Amazon ECR エラーメッセージのトラブルシューティング

場合によっては、Amazon ECR コンソールまたは を通じて開始した API コールが、エラーメッセージで AWS CLI 終了します。一般的なエラーメッセージと考えられる解決策を以下に説明します。

HTTP 429: リクエストが多すぎるまたは ThrottleException

1 つ以上の Amazon ECR アクション429: Too Many Requestsまたは API コールからエラーまたはThrottleExceptionエラーが表示される場合があります。これは、短い間隔で Amazon ECR の単一のエンドポイントを繰り返し呼び出していて、リクエストがスロットリングされていることを示します。スロットリングは、1 人のユーザーから単一のエンドポイントへの呼び出しが、期間中に特定のしきい値を超えたときに発生します。

Amazon ECR の各 API オペレーションには、レートスロットリングが関連付けられています。たとえば、GetAuthorizationToken アクションのスロットリングは 20 TPS (トランザクショ

ン/秒) で、最大 200 TPS のバーストが可能です。各リージョンで、各アカウントには、最大 200 GetAuthorizationToken クレジットを保存できるバケットが割り当てられ、1 秒あたり 20 クレジットの割合で補充されます。バケットに 200 クレジットがある場合、1 秒あたり 200 GetAuthorizationToken API トランザクションに達すると、1 秒あたり 20 トランザクションが無期限に維持されます。Amazon ECR APIs「」を参照してくださいAmazon ECR のサービスクォータ。

スロットリングエラーを処理するには、増分バックオフをコードに含めて再試行関数を実装します。詳細については、 SDK およびツールリファレンスガイドの「再試行動作」を参照してください。AWS SDKs もう 1 つのオプションは、Service Quotas コンソールを使用してレート制限の引き上げをリクエストすることです。 詳細については、AWS Management Console での Amazon ECR サービスクォータの管理を参照してください。

HTTP 403: "ユーザー [arn] は [operation] の実行を許可されていません"

Amazon ECR を使用してアクションを実行しようとすると、次のエラーを受け取ることがあります。

```
$ 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: *
```

これは、Amazon ECR を使用するアクセス権限がユーザーに付与されていないか、それらのアクセス権限が正しく設定されていないことを示します。特に、Amazon ECR レポジトリに対してアクションを実行している場合は、そのリポジトリにアクセスする権限がユーザーに付与されていることを確認します。Amazon ECR の許可の作成と検証の詳細については、「Amazon Elastic Container Registry の Identity and Access Management」を参照してください。

HTTP 404: "レポジトリは存在しません" エラー

現在存在しない Docker Hub リポジトリを指定すると、Docker Hub はそのレポジトリを自動的に作成します。Amazon ECR では、レポジトリを使用するには、その前に新しいリポジトリを明示的に作成する必要があります。これにより、(タイプミスなどの原因で) 新しいレポジトリが間違って作成されるのを防止し、適切なセキュリティアクセスポリシーが明示的に新しいレポジトリに割り当てられます。レポジトリの作成方法の詳細については、「Amazon ECR プライベートリポジトリ」を参照してください。

エラー: Cannot perform an interactive login from a non TTY device (TTY 以外のデバイスから対話型ログインを実行できません)

`Cannot perform an interactive login from a non TTY device` エラーが表示される場合は、以下のトラブルシューティング手順を試してください。

- AWS CLI バージョン 2 を使用していて、システム上にバージョン 1 の競合 AWS CLI バージョンがないことを確認します。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。
- AWS CLI に有効な認証情報が設定されていることを確認します。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。
- AWS CLI コマンドの構文が正しいことを確認します。

ドキュメント履歴

次の表に、Amazon ECR の前回のリリース以後に行われたドキュメントの重要な変更を示します。また、お客様からいただいたフィードバックに対応するために、ドキュメントを頻繁に更新しています。

変更	説明	日付
中国リージョンにクロスリージョンおよびクロスアカウントレプリケーションを追加	Amazon ECR は、レプリケートされるリポジトリをフィルタリングするためのサポートを中国リージョンに追加しました。	2024 年 5 月 15 日
キャッシュルールをプルスルーする GitLab コンテナレジストリを追加	Amazon ECR に、 GitLab コンテナレジストリのプルスルーキャッシュルールの作成のサポートが追加されました。	2024 年 5 月 8 日
Amazon ECR ライフサイクルポリシーが更新され、ワイルドカードの使用がサポートされるようになりました	Amazon ECR は、ライフサイクルポリシールールで `tagPatternList` パラメータを使用することにより、ライフサイクルポリシーでのワイルドカードをサポートできるようになりました。詳細については、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。	2023 年 12 月 18 日
Amazon ECR リポジトリ作成テンプレート	Amazon ECR にリポジトリの作成テンプレートのサポートが追加されました。詳細については、「プルスルーキャッシュアクション中に作成されたリポジトリを制御するテンプレート」を参照してください。	2023 年 11 月 15 日
Amazon ECR プルスルーキャッシュで、認証済みのアップストリームレジストリでサポートされるようになりました	Amazon ECR で、プルスルーキャッシュルールに認証が必要なアップストリームレジストリの使用がサポートされるようになりました。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。	2023 年 11 月 15 日

変更	説明	日付
AWSECRPullThroughCache_ServiceRolePolicy – 既存ポリシーへの更新	Amazon ECR は、新しいアクセス許可を AWSECRPullThroughCache_ServiceRolePolicy ポリシーに追加しました。これらの権限により、Amazon ECR は Secrets Manager シークレットの暗号化されたコンテンツを取得できます。これは、プルスルーキャッシュルールを使用して、認証が必要なアップストリームレジストリからイメージをキャッシュする場合に必要です。	2023 年 11 月 15 日
Amazon ECR イメージ署名	Amazon ECR と に、Notary クライアントを使用したコンテナイメージ署名の作成とプッシュのサポート AWS Signer が追加されました。詳細については、「Amazon ECR プライベートリポジトリに保存されているイメージの署名」を参照してください。	2023 年 6 月 6 日
キャッシュルールのプルスルーに Kubernetes コンテナレジストリが追加	Amazon ECR で、Kubernetes コンテナレジストリのプルスルーキャッシュルールを作成するサポートが追加されました。詳細については、「アップストリームレジストリと Amazon ECR プライベートレジストリを同期する」を参照してください。	2023 年 6 月 1 日
Amazon ECR 拡張スキャン期間のサポート	Amazon Inspector で、拡張スキャンが有効になっている場合にリポジトリが監視される期間を設定できるようになりました。詳細については、「Amazon Inspector でのイメージの拡張スキャン期間の変更」を参照してください。	2022 年 6 月 28 日
Amazon ECR がリポジトリのプルカウントメトリクスを Amazon に送信する CloudWatch	Amazon ECR は、リポジトリのプルカウントメトリクスを Amazon に送信します CloudWatch。詳細については、「Amazon ECR リポジトリメトリクス」を参照してください。	2022 年 1 月 6 日
拡張されたレプリケーションサポート	Amazon ECR が、レプリケートされるリポジトリのフィルタリングに対するサポートを追加しました。詳細については、「Amazon ECR でのプライベートイメージのレプリケーション」を参照してください。	2021 年 9 月 21 日

変更	説明	日付
AWS Amazon ECR の マネージドポリシー	Amazon ECR に AWS マネージドポリシーのドキュメントが追加されました。詳細については、「AWS Amazon Elastic Container Registry の マネージドポリシー」を参照してください。	2021 年 6 月 24 日
クロスリージョンおよびクロスアカウントレプリケーション	Amazon ECR で、プライベートレジストリのレプリケーション設定を構成するためのサポートが追加されました。詳細については、「Amazon ECR のプライベートレジストリ設定」を参照してください。	2020 年 12 月 8 日
OCI アーティファクトサポート	Amazon ECR では、Open Container Initiative (OCI) アーティファクトのプッシュおよびプルがサポートされました。アーティファクトのタイプを示すために `DescribeImages` API レスポンスに新しいパラメータ `artifactMediaType` が追加されました 詳細については、「Helm チャートを Amazon ECR プライベートリポジトリにプッシュする」を参照してください。	2020 年 8 月 24 日
保管中の暗号化	Amazon ECR で、 AWS Key Management Service (AWS KMS) に格納されたカスタマーマネージドキーでサーバー側の暗号化を使用してレポジトリを暗号化するための設定のサポートが追加されました。 詳細については、「保管中の暗号化」を参照してください。	2020 年 7 月 29 日
マルチアーキテクチャイメージ	Amazon ECR で、マルチアーキテクチャイメージに使用される Docker マニフェストリストの作成とプッシュのサポートが追加されました。 詳細については、「マルチアーキテクチャイメージを Amazon ECR プライベートリポジトリにプッシュする」を参照してください。	2020 年 4 月 28 日

変更	説明	日付
Amazon ECR 使用状況メトリクス	Amazon ECR は、アカウントのリソース CloudWatch 使用状況を可視化する使用状況メトリクスを追加しました。また、 CloudWatch と Service Quotas コンソールの両方からアラームを作成して CloudWatch、適用されたサービスクォータに使用量が近づいたときにアラートを受け取ることもできます。 詳細については、「Amazon ECR 使用状況メトリクス」を参照してください。	2020 年 2 月 28 日
Amazon ECR サービスクォータの更新	Amazon ECR サービスクォータが更新されて、API 別のクォータを含めることができるようになりました。 詳細については、「Amazon ECR のサービスクォータ」を参照してください。	2020 年 2 月 19 日
`get-login-password` コマンドの追加	get-login-password のサポートが追加されて、シンプルで安全な方法で認証トークンを取得できるようになりました。 詳細については、「認可トークンを使用する」を参照してください。	2020 年 2 月 4 日
イメージスキャン	コンテナイメージ内のソフトウェアの脆弱性を識別するのに役立つイメージスキャンのサポートが追加されました。Amazon ECR は、オープンソースの CoreOS Clair プロジェクトの共通脆弱性識別子 (CVE) データベースを使用し、スキャン結果のリストを提供します。 詳細については、「Amazon ECR でソフトウェアの脆弱性がないかイメージをスキャンする」を参照してください。	2019 年 10 月 24 日

変更	説明	日付
VPC エンドポイントポリシー	Amazon ECR インターフェイス VPC エンドポイントで IAM ポリシーを設定するためのサポートが追加されました。 詳細については、「Amazon ECR VPC エンドポイントのエンドポイントポリシーを作成する」を参照してください。	2019 年 9 月 26 日
イメージタグの変更可能性	イメージタグが上書きされるのを防止するためにリポジトリの設定で変更不可のサポートが追加されました。 詳細については、「Amazon ECR でイメージタグが上書きされないようにする」を参照してください。	2019 年 7 月 25 日
インターフェイス VPC エンドポイント (AWS PrivateLink)	によるインターフェイス VPC エンドポイントの設定のサポートが追加されました AWS PrivateLink。これにより、インターネット、NAT インスタンス、VPN 接続、または AWS Direct Connectを経由せずに、VPC と Amazon ECR をプライベートに接続できます。 詳細については、「Amazon ECR インターフェイス VPC エンドポイント (AWS PrivateLink) 」を参照してください。	2019 年 1 月 25 日
リソースへのタグ付け	Amazon ECR で、メタデータタグをリポジトリに追加するためのサポートが追加されました。 詳細については、「Amazon ECR でのプライベートリポジトリのタグ付け」を参照してください。	2018 年 12 月 18 日
Amazon ECR の名称変更	Amazon Elastic Container Registry の名称が変更されました (以前の名称は Amazon EC2 Container Registry)。	2017 年 11 月 21 日

変更	説明	日付
ライフサイクルポリシー	Amazon ECR ライフサイクルポリシーを使用して、リポジトリ内のイメージのライフサイクル管理を指定することができます。 詳細については、「Amazon ECR のライフサイクルポリシーを使用してイメージのクリーンアップを自動化する」を参照してください。	2017 年 10 月 11 日
Amazon ECR での Docker Image Manifest 2、Schema 2 のサポート	Amazon ECR が Docker Image Manifest V2 Schema 2 (Docker バージョン 1.10 以降で使用) をサポートするようになりました。 詳細については、「Amazon ECR でのコンテナイメージマニフェスト形式のサポート」を参照してください。	2017 年 1 月 27 日
Amazon ECR の一般提供	Amazon Elastic Container Registry (Amazon ECR) は、安全でスケーラブル、信頼性の高いマネージド AWS Docker レジストリサービスです。	2015 年 12 月 21 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。