Amazon ECR

ユーザーガイド

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、安全性と信頼性に優れたスケーラブルなマネージド AWS Docker レジストリサービスです。Amazon ECR では、AWS IAM を使用してプライベート Docker リポジトリにリソースベースのアクセス権限が付与されるため、特定のユーザーまたは Amazon EC2 インスタンスからリポジトリとイメージにアクセスできるようになります。開発者は、Docker CLI を使用してイメージをプッシュ、プル、および管理できます。

AWS コンテナサービスチームは、GitHub でパブリックロードマップを維持しています。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、AWS のすべてのお客様が直接フィードバックを提供することができます。詳細については、「AWS コンテナロードマップ」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 レジストリは、AWS アカウントごとに用意されています。レジストリ内にイメージリポジトリを作成して、これらのリポジトリにイメージを保存できます。詳細については、「Amazon ECR レジストリ (p. 13)」を参照してください。

認証トークン

Docker 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 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、「レジストリの認証 (p. 13)」を参照してください。

リポジトリ

Amazon ECR イメージリポジトリには、Docker または Open Container Initiative (OCI) イメージが含まれます。詳細については、「Amazon ECR Repositories (p. 16)」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イメージへのアクセス権を制御できます。詳細については、「Amazon ECR のリポジトリポリシー (p. 19)」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をローカルに使用したり、Amazon ECS タスク定義と Amazon EKS ポッド仕様で使用したりすることができます。詳細については、「Amazon ECR イメージを Amazon ECS で使用する (p. 51)」および「Amazon ECR イメージを Amazon EKS で使用する (p. 52)」を参照してください。

Amazon ECR の使用を開始する方法

Amazon ECR を使用するには、AWS Command Line Interface と Docker をインストールするようにセットアップする必要があります。詳細については、「Amazon ECR でのセットアップ (p. 2)」および「AWS CLI を使用した Amazon ECR の開始方法 (p. 7)」を参照してください。

Amazon ECR でのセットアップ

AWS にサインアップ済みで、Amazon Elastic Container Service (Amazon ECS) または (Amazon Elastic Kubernetes Service) Amazon EKS を使用している場合は、すぐに Amazon ECR を使用し始めることができます。Amazon ECR はこれらのサービスの拡張機能であるため、これら 2 つのサービスのセットアッププロセスはよく似ています。AWS CLI と Amazon ECR をともに使用するには、最新の Amazon ECR 機能をサポートしているバージョンの AWS CLI を使用する必要があります。AWS CLI で Amazon ECR 機能のサポートが表示されない場合は、最新バージョンにアップグレードする必要があります。詳細については、http://aws.amazon.com/cli/ を参照してください。

コンテナイメージを初めて Amazon ECR にプッシュするように設定するには、以下のタスクを実行します。完了済みのステップがあればスキップして、次のステップに進むことができます。

Sign up for AWS

AWS にサインアップすると、Amazon ECR などすべてのサービスに対して AWS アカウントが自動的にサインアップされます。料金が発生するのは、実際に使用したサービスの分のみです。

既に AWS アカウントをお持ちの場合は、次のタスクに進んでください。AWS アカウントをお持ちでない場合は、次に説明する手順にしたがってアカウントを作成してください。

AWS アカウントを作成するには

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを用いて確認コードを入力することが求められます。

次のタスクで AWS アカウント番号が必要となるので、メモしておいてください。

IAM ユーザーを作成する

AWS のサービス (Amazon ECR など) の場合は、サービスにアクセスする際に認証情報を提供する必要があります。このため、サービスのリソースにアクセスする権限があるかどうかがサービスによって判定されます。コンソールを使用するにはパスワードが必要です。AWS アカウントのアクセスキーを作成して、コマンドラインインターフェイスまたは API にアクセスすることができます。ただし、AWS アカウントの認証情報を使って AWS にアクセスすることはお勧めしません。代わりに AWS Identity and Access Management (IAM) を使用することをお勧めします。IAM ユーザーを作成して、管理権限を使ってこのユーザーを IAM グループに追加するか、管理権限を付与します。これで、特殊な URL と IAM ユーザーの認証情報を使って、AWS にアクセスできます。

AWS にサインアップしたけれど、自身の IAM ユーザーをまだ作成していない場合は、IAM コンソールを使用して作成できます。

自分用の管理者ユーザーを作成し、そのユーザーを管理者グループに追加するには (コンソール)

1. ルートユーザー を選択し AWS アカウントの E メールアドレスを入力して、アカウントの所有者としてIAM コンソールにサインインします。次のページでパスワードを入力します。

Note

以下の**Administrator**IAMユーザーの使用に関するベストプラクティスに従い、ルートユーザー認証情報を安全な場所に保管しておくことを強くお勧めします。ルートユーザーとしてサインインして、少数のアカウントおよびサービス管理タスクのみを実行します。

2. ナビゲーションペインで [Users]、[Add user] の順に選択します。
3. [ユーザー名] に「**Administrator**」と入力します。
4. [AWS マネジメントコンソール access (アクセス)] の横にあるチェックボックスをオンにします。[Custom password (カスタムパスワード)] を選択し、その後テキストボックスに新しいパスワードを入力します。
5. (オプション) AWS では、デフォルトで、新しいユーザーに対して初回のサインイン時に新しいパスワードを作成することが必要です。必要に応じて [User must create a new password at next sign-in (ユーザーは次回のサインイン時に新しいパスワードを作成する必要がある)] のチェックボックスをオフにして、新しいユーザーがサインインしてからパスワードをリセットできるようにできます。
6. [Next: Permissions (次へ: アクセス許可)] を選択します。
7. [Set permissions (アクセス許可の設定)] で、[Add user to group (ユーザーをグループに追加)] を選択します。
8. [Create group] を選択します。
9. [グループの作成] ダイアログボックスで、[グループ名] に「**Administrators**」と入力します。
10. [Filter policies (フィルタポリシー)] を選択し、その後 [AWS managed -job function (AWS 管理ジョブの機能] を選択してテーブルのコンテンツをフィルタリングします。
11. ポリシーリストで、[AdministratorAccess] のチェックボックスをオンにします。次に、[Create group] を選択します。

    Note

    `AdministratorAccess` アクセス許可を使用して、AWS Billing and Cost Management コンソールを使用する前に、IAM ユーザーおよびロールの請求へのアクセスをアクティブ化する必要があります。これを行うには、請求コンソールへのアクセスの委任に関するチュートリアルのステップ 1 の手順に従ってください。
12. グループのリストに戻り、新しいグループのチェックボックスをオンにします。必要に応じて [Refresh] を選択し、リスト内のグループを表示します。
13. [次へ: タグ] を選択します。
14. (オプション) タグをキー - 値のペアとしてアタッチして、メタデータをユーザーに追加します。IAM でのタグの使用の詳細については、『IAM ユーザーガイド』の「IAM エンティティのタグ付け」を参照してください。
15. [Next: Review] を選択して、新しいユーザーに追加するグループメンバーシップのリストを表示します。続行する準備ができたら、[Create user] を選択します。

この同じプロセスを繰り返して新しいグループとユーザーを作成し、AWS アカウントのリソースへのアクセス権をユーザーに付与できます。ポリシーを使用して特定の AWS リソースに対するユーザーのアクセス許可を制限する方法については、「アクセス管理」と「ポリシーの例」を参照してください。

新規の IAM ユーザーとしてサインインするには、AWS コンソールからサインアウトし、次の URL を使用します。このとき、your_aws_account_id はハイフンを除いた AWS アカウント番号です (たとえば AWS アカウント番号が `1234-5678-9012` であれば、AWS アカウント ID は `123456789012` となります)。

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

作成した IAM ユーザー名とパスワードを入力します。サインインすると、ナビゲーションバーに「your_user_name @ your_aws_account_id」が表示されます。

サインページの URL に AWS アカウント ID を含めない場合は、アカウントのエイリアスを作成します。IAM ダッシュボードから [カスタマイズ] を選択し、[アカウントエイリアス] (会社名など) を入力します。詳細については、IAM ユーザーガイドの「AWS アカウント ID とその別名」を参照してください。

アカウントエイリアスを作成した後、サインインするには、次の URL を使用します。

```
https://your_account_alias.signin.aws.amazon.com/console/
```

アカウントの IAM ユーザーのサインインリンクを確認するには、IAM コンソールを開いて、ダッシュボードの [IAM ユーザーのサインインリンク] を確認します。

IAM の詳細については、「AWS Identity and Access Management ユーザーガイド」を参照してください。

AWS マネジメントコンソール を使用した Amazon ECR の開始方法

Amazon ECR コンソールでリポジトリを作成することにより、Amazon ECR の使用を開始します。Amazon ECR コンソールの指示に従って、最初のリポジトリの作成を開始します。

開始する前に、必ず「Amazon ECR でのセットアップ (p. 2)」の手順を完了してください。

イメージリポジトリを作成するには

リポジトリは、Docker または Open Container Initiative (OCI) イメージを Amazon ECR に保存する場所です。Amazon ECR からイメージをプッシュまたはプルするたびに、イメージをプッシュまたはプルするリポジトリとレジストリの場所を指定します。

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. [Get Started] を選択します。
3. [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。タグが変更不可能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、「Image tag mutability (p. 45)」を参照してください。
4. [Scan on push (プッシュ時にスキャン)] で、リポジトリのイメージスキャン設定を選択します。プッシュ時にスキャンするように設定されたリポジトリは、イメージがプッシュされるたびにイメージスキャンを開始します。それ以外の場合は、イメージスキャンを手動で開始する必要があります。詳細については、「Image scanning (p. 46)」を参照してください。
5. [リポジトリの作成] を選択します。

Docker イメージの構築、タグ付け、プッシュ

ウィザードのこのセクションでは、Docker CLI を使用して既存のローカルイメージ (Dockerfile から構築したもの、または Docker Hub などの別のレジストリから取得したもの) にタグを付け、そのタグ付きイメージを Amazon ECR レジストリにプッシュします。Docker CLI の使用に関する詳細な手順については、「AWS CLI を使用した Amazon ECR の開始方法 (p. 7)」を参照してください。

1. 作成したレポジトリを選択し、[プッシュコマンドの表示] を選択して、イメージを新しいリポジトリにプッシュする手順を表示します。
2. コンソールからターミナルウィンドウに aws ecr get-login コマンドを貼り付けて、Docker クライアントをレジストリに対して認証するために使用できる docker login コマンドを取得します。

   Note

   get-login コマンドは AWS CLI のバージョン 1.9.15 以降で利用できます。ただし、Docker の最新バージョン (17.06 以降) を使用している場合は、バージョン 1.11.91 以降をお勧めします。AWS CLI のバージョンは、aws --version コマンドで確認できます。Docker のバージョン 17.06 以降を使用している場合は、`get-login` の後に `--no-include-email` オプションを含めます。`Unknown options: --no-include-email` エラーが返された場合は、AWS CLI の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 コマンドラインインターフェイスのインストール」を参照してください。
3. 前のステップで返された docker login コマンドを実行します。このコマンドは、12 時間有効な認証トークンを提供します。

Important

docker login コマンドを実行すると、システムの他のユーザーに対して、プロセスリスト (ps -e) 表示でコマンド文字列が表示されます。docker login コマンドには認証情報が含まれているため、システムの他のユーザーがこのようにして認証情報を表示するリスクがあります。また、その認証情報を使用して、リポジトリへのプッシュアクセスおよびプルアクセスを取得する可能性があります。安全なシステムを使用していない場合は、このリスクを考慮してください。`-p` *`password`* オプションを省略してインタラクティブにログインし、求められたときにパスワードを入力することを検討してください。

4. (オプション) プッシュするイメージの Dockerfile がある場合は、イメージを構築し、新しいレポジトリ用にタグを付けます。コンソールからターミナルウィンドウに docker build コマンドを貼り付けます。Dockerfile と同じディレクトリであることを確認します。
5. コンソールからターミナルウィンドウに docker tag コマンドを貼り付けて、ECR レジストリと新しいレポジトリ用のイメージにタグを付けます。コンソールコマンドでは、前のステップの Dockerfile からイメージが構築されたことを前提とします。Dockerfile からイメージを構築していない場合は、*`repository`*`:latest` の最初のインスタンスを、プッシュするローカルイメージのイメージ ID またはイメージ名と置き換えます。
6. docker push コマンドをターミナルウィンドウに貼り付けて、新しくタグ付けされたイメージを ECR レポジトリにプッシュします。
7. [Close] を選択します。

AWS CLI を使用した Amazon ECR の開始方法

以下では、Docker CLI と AWS CLI を使用して初めてコンテナイメージを Amazon ECR にプッシュするために必要な手順について説明します。

さまざまな AWS SDK、IDE ツールキット、および Windows PowerShell コマンドラインツールを含む、AWS リソースを管理するための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

開始する前に、必ず「Amazon ECR でのセットアップ (p. 2)」の手順を完了してください。

最新の AWS CLI と Docker がまだインストールされておらず、使用する準備ができていない場合は、次の手順を使用してこれら両方のツールをインストールします。

AWS CLI をインストールする

AWS コマンドラインツールを使用して、システムのコマンドラインでコマンドを発行することで、Amazon ECR および他の AWS タスクを実行できます。これは、コンソールを使用するよりも高速でより便利になります。コマンドラインツールは、AWS のタスクを実行するスクリプトの作成にも便利です。

AWS CLI を Amazon ECR とともに使用するには、最新の AWS CLI バージョンをインストールします (Amazon ECR の機能は、バージョン 1.9.15 から AWS CLI で利用可能)。AWS CLI のバージョンは、aws --version コマンドで確認できます。AWS CLI のインストールまたは最新バージョンへのアップグレードについては、AWS Command Line Interface ユーザーガイドの「AWS CLI バージョン 2 のインストール」を参照してください。

Docker のインストール

Docker は、Ubuntu のような最新の Linux ディストリビューションから Mac OSX 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 2 インスタンスを起動し、Docker をインストールして使用し始めることができます。

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 (p. 8)」に進んでください。

Amazon EC2 インスタンスに Docker をインストールするには

1. Amazon Linux 2 AMI でインスタンスを起動します。詳細については、Linux インスタンス用 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、Linux インスタンス用 Amazon EC2 ユーザーガイド の「Linux インスタンスへの接続」を参照してください。
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 amazon-linux-extras 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

このセクションでは、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 EC2 インスタンスでテストしてから、コンテナレジストリ (Amazon ECR や Docker Hub など) にプッシュして、ECS タスク定義で使用できるようにします。

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 ubuntu:18.04

# Install dependencies
RUN apt-get update && \
 apt-get -y install apache2

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

```
# Configure apache
RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && \
 echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
 echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

この Dockerfile は Ubuntu 18.04 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パッケージ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まれます。`EXPOSE` 命令はコンテナ上のポート 80 を公開し、`CMD` 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. docker images を実行して、イメージが正しく作成されたことを確認します。

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

出力:

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

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 server's 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 コンピューターで docker-machine を使用している場合は、docker-machine ip コマンドを使用して Docker のホスト VirtualBox VM の IP アドレスを見つけ、*machine-name* を、使用中の 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

AWS CLI のインストールと設定が完了したら、デフォルト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。AWS CLI には、認証プロセスをシンプルにする get-login-password コマンドが用意されています。

get-login-password を使用して 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に、ユーザー名の `AWS` 値と認証先の 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 コマンドラインインターフェイスのインストール」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-world` 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-world \
    --image-scanning-configuration scanOnPush=true \
    --region us-east-1
```

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 を使ってイメージをプッシュしますが、この作業を正しく完了するために満たす必要がある前提条件がいくつかあります。

- docker の最小バージョン 1.7 がインストールされている。
- Amazon ECR 認証トークンが docker login で設定されている。
- 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、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.us-east-1.amazonaws.com/hello-
world:latest
```

3. イメージをプッシュします。

```
docker push aws_account_id.dkr.ecr.us-east-1.amazonaws.com/hello-world:latest
```

出力:

```
The push refers to a repository [aws_account_id.dkr.ecr.us-east-1.amazonaws.com/hello-
world] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b
 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、他の場所からプルできます。docker CLI を使ってイメージをプルしますが、この作業を正しく完了するために満たす必要がある前提条件がいくつかあります。

- docker の最小バージョン 1.7 がインストールされている。
- Amazon ECR 認証トークンが docker login で設定されている。
- 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権がユーザーにある。

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 からイメージの例をプルするには、次のコマンドを実行します。

```
docker pull aws_account_id.dkr.ecr.us-east-1.amazonaws.com/hello-world:latest
```

出力:

```
latest: Pulling from hello-world
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b
Status: Downloaded newer image for aws_account_id.dkr.ecr.us-east-1.amazonaws.com/hello-
world: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で、イメージが必要なくなったと判断した場合、batch-delete-image コマンドで削除できます。イメージを削除するには、イメージがあるレポジトリ、およびイメージの `imageTag` または `imageDigest` の値を指定する必要があります。次の例では、イメージタグが `hello-world` で、`latest` リポジトリにあるイメージを削除します。

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

出力:

```
{
    "failures": [],
    "imageIds": [
        {
            "imageTag": "latest",
            "imageDigest":
 "sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b"
        }
    ]
}
```

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必要ないと判断した場合は、リポジトリを削除できます。デフォルトでは、イメージを含むリポジトリを削除することはできません。ただし、`--force` フラグを使用すると、この操作を行うことができます。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-world \
      --force
```

Amazon ECR レジストリ

Amazon ECR レジストリによって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がホストされるため、コンテナをアプリケーションに確実にデプロイできます。レジストリを使用して、Docker および Open Container Initiative (OCI) イメージで構成されるイメージリポジトリを管理できます。AWS アカウントごとに、(デフォルト) Amazon ECR レジストリが 1 つずつ提供されます。

レジストリの概念

- デフォルトレジストリの URL は `https://aws_account_id.dkr.ecr.region.amazonaws.com` です。
- デフォルトでは、アカウントにはデフォル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アクセス権があります。ただし、IAM ユーザーには、Amazon ECR API への呼び出しと、リポジトリへのイメージのプッシュまたはプルを行う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コントロールするマネージド型ポリシーがいくつか用意されています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」を参照してください。
- 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、「レジストリの認証 (p. 13)」を参照してください。
- リポジトリは、IAM の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 のリポジトリポリシー (p. 19)」を参照してください。

レジストリの認証

AWS マネジメントコンソール、AWS CLI、または AWS SDK を使用して、レポジトリを作成して管理することができます。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。技術的には 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、ほとんどの場合 Docker CLI (または言語固有の Docker ライブラリ) を使用します。

Docker CLI では、ネイティブ 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、承認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 認証情報ヘルパーの使用

Amazon ECR には Docker 認証情報ヘルパーが用意されており、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。

認証トークンの使用

承認トークンのアクセス許可スコープは、認証トークンの取得に使用される IAM プリンシパルのアクセス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する必要があります。これにより、ユーザー名 `AWS`

とエンコードされたパスワードを含む base64 エンコード認証トークンを取得します。AWS CLI の `get-login-password` コマンドでは、よりシンプルな方法で、認証トークンを取得し、デコードして、認証のために docker login コマンドにパイプできます。

get-login-password 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 を使用して 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に、ユーザー名の `AWS` 値と認証先の 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 コマンドラインインターフェイスのインストール」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
```

get-login を使用して Amazon ECR レジストリに対して Docker を認証するには

AWS CLI バージョン 1.17.10 より前のバージョンを使用している場合は、get-login コマンドを使用して Amazon ECR レジストリに対して認証できます。AWS CLI のバージョンは、aws --version コマンドで確認できます。

1. aws ecr get-login コマンドを実行します。次の例は、リクエストを実行するアカウントに関連付けられたデフォルトレジストリ用です。他のアカウントのレジストリにアクセスするには、`--registry-ids aws_account_id` オプションを使用します。詳細については、AWS CLI Command Reference の「get-login」を参照してください。

   ```
   aws ecr get-login --region region --no-include-email
   ```

   結果の出力は docker login コマンドです。このコマンドを使用して、Amazon ECR レジストリに対して Docker クライアントを認証します。

   ```
   docker login -u AWS -p password https://aws_account_id.dkr.ecr.region.amazonaws.com
   ```

2. Docker CLI をレジストリに対して認証するには、docker login コマンドをコピーしてターミナルに貼り付けます。このコマンドは、指定されたレジストリに対して 12 時間有効な認証トークンを提供します。

   > **Note**
   >
   > Windows PowerShell を使用している場合、このような長い文字列をコピーして貼り付けることはできません。代わりに次のコマンドを使用します。

   ```
   Invoke-Expression -Command (Get-ECRLoginCommand -Region region).Command
   ```

Important

docker login コマンドを実行すると、システムの他のユーザーに対して、プロセスリスト (ps -e) 表示でコマンド文字列が表示されます。docker login コマンドには認証情報が含まれているため、システムの他のユーザーがこのようにして認証情報を表示するリスクがあります。また、その認証情報を使用して、リポジトリへのプッシュアクセスおよびプルアクセスを取得する可能性があります。安全なシステムを使用していない場合は、このリスクを考慮してください。`-p` *password* オプションを省略してインタラクティブにログインし、求められたときにパスワードを入力することを検討してください。

HTTP API 認証の使用

Amazon ECR は Docker Registry HTTP API をサポートしています。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。curl の `-H` オプションを使用して HTTP 認証ヘッダーを追加し、get-authorization-token AWS CLI コマンドで提供される認証トークンを渡します。

Amazon ECR HTTP API を使用して認証するには

1. AWS CLI を使用して認証トークンを取得し、そのトークンを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:

```
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 Repositories

Amazon Elastic Container Registry (Amazon ECR) には、イメージリポジトリの作成、モニタリング、削除と、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権限を設定する API オペレーションが用意されています。Amazon ECR コンソールの [リポジトリ] セクションでは、同じアクションを実行できます。Amazon ECR には、Docker CLI も統合されており、開発環境とリポジトリの間でイメージをプッシュおよびプルできます。

トピック

Repository Concepts

- By default, your account has read and write access to the repositories in your default registry (`aws_account_id.dkr.ecr.region.amazonaws.com`). However, IAM users require permissions to make calls to the Amazon ECR APIs and to push or pull images from your repositories. Amazon ECR provides several managed policies to control user access at varying levels; for more information, see Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64).
- Repositories can be controlled with both IAM user access policies and repository policies. For more information, see Amazon ECR のリポジトリポリシー (p. 19).
- Repository names can support namespaces, which you can use to group similar repositories. For example if there are several teams using the same registry, Team A could use the `team-a` namespace while Team B uses the `team-b` namespace. Each team could have their own image called `web-app`, but because they are each prefaced with the team namespace, the two images can be used simultaneously without interference. Team A's image would be called `team-a/web-app`, while Team B's image would be called `team-b/web-app`.

Creating a Repository

Docker イメージを Amazon ECR にプッシュするには、保存先のリポジトリを作成する必要があります。AWS マネジメントコンソール、または AWS CLI と AWS SDK を使用して Amazon ECR リポジトリを作成できます。

リポジトリを作成するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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、[リポジトリの作成] を選択します。
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。
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。タグが変更不可能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、「Image tag mutability (p. 45)」を参照してください。
7. [Scan on push (プッシュ時にスキャン)] で、リポジトリのイメージスキャン設定を選択します。プッシュ時にスキャンするように設定されたリポジトリは、イメージがプッシュされるたびにイメージスキャンを開始します。それ以外の場合は、イメージスキャンを手動で開始する必要があります。詳細については、「Image scanning (p. 46)」を参照してください。
8. 対象: KMS暗号化、 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、KMS暗号化が有効になっている場合 Amazon ECR は、 AWS エイリアス付き管理顧客マスターキー(CMK) `aws/ecr`, KMS暗号化を有効にして初めてリポジトリを作成すると、アカウントで作成されます。詳細については、「Encryption at rest (p. 70)」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、 顧客の暗号化設定(詳細) お好きなCMKをお選びいただけます。CMK はクラスタと同じ地域に存在する必要があります。選択 作成 AWS KMS キー 移動するには AWS KMS 独自のキーを作成します。
10. [リポジトリの作成] を選択します。
11. (オプション) 作成したレポジトリを選択し、[プッシュコマンドの表示] を選択して、イメージを新しいリポジトリにプッシュするステップを表示します。

    a. コンソールからターミナルウィンドウに aws ecr get-login コマンドを貼り付けて、Docker クライアントをレジストリに対して認証するために使用できる docker login コマンドを取得します。

       Note

       get-login コマンドは AWS CLI のバージョン 1.9.15 以降で利用できます。ただし、Docker の最新バージョン (17.06 以降) を使用している場合は、バージョン 1.11.91 以降をお勧めします。AWS CLI のバージョンは、aws --version コマンドで確認できます。Docker のバージョン 17.06 以降を使用している場合は、`get-login` の後に `--no-include-email` オプションを含めます。`Unknown options: --no-include-email` エラーが返された場合は、AWS CLI の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 コマンドラインインターフェイスのインストール」を参照してください。

    b. 前のステップで返された docker login コマンドを実行します。このコマンドは、12 時間有効な認証トークンを提供します。

       Important

       docker login コマンドを実行すると、システムの他のユーザーに対して、プロセスリスト (ps -e) 表示でコマンド文字列が表示されます。docker login コマンドには認証情報が含まれているため、システムの他のユーザーがこのようにして認証情報を表示するリスクがあります。また、その認証情報を使用して、リポジトリへのプッシュアクセスおよびプルアクセスを取得する可能性があります。安全なシステムを使用していない場合は、このリスクを考慮してください。`-p` *`password`* オプションを省略してインタラクティブにログインし、求められたときにパスワードを入力することを検討してください。

    c. (オプション) プッシュするイメージの Dockerfile がある場合は、イメージを構築し、新しいレポジトリ用にタグを付けます。コンソールからターミナルウィンドウに docker build コマンドを貼り付けます。Dockerfile と同じディレクトリであることを確認します。

    d. コンソールからターミナルウィンドウに docker tag コマンドを貼り付けて、ECR レジストリと新しいレポジトリ用のイメージにタグを付けます。コンソールコマンドでは、前のステップの Dockerfile からイメージが構築されたことを前提とします。Dockerfile からイメージを構築していない場合は、*`repository`*`:latest` の最初のインスタンスを、プッシュするローカルイメージのイメージ ID またはイメージ名と置き換えます。

e. docker push コマンドをターミナルウィンドウに貼り付けて、新しくタグ付けされたイメージを ECR レポジトリにプッシュします。
f. [Close] を選択します。

Viewing Repository Information

リポジトリを作成したら、その情報を AWS マネジメントコンソール で表示できます。

- Which images are stored in a repository
- Whether an image is tagged
- The tags for the image
- The SHA digest for the images
- The size of the images in MiB
- When the image was pushed to the repository

> 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 コマンドの出力には、非圧縮のイメージサイズが表示されるため、AWS マネジメントコンソール に表示されるイメージサイズよりも大きいイメージサイズが返されることがあります。

リポジトリ情報を表示するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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] ページで、表示するリポジトリを選択します。
5. の リポジトリ: ***repository_name*** ページで、ナビゲーションバーを使用して画像に関する情報を表示します。
   - [イメージ] を選択して、リポジトリ内のイメージに関する情報を表示します。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] を選択できます。詳細については、「Deleting an image (p. 32)」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 のリポジトリポリシー (p. 19)」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 Lifecycle Policies (p. 34)」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。

Editing a Repository

既存のリポジトリを編集して、イメージタグの変更可能性およびイメージスキャン設定を変更できます。

リポジトリを編集するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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. [レポジトリ] ページで、削除するレポジトリを選択して [編集] を選択します。
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。タグが変更不可能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、「Image tag mutability (p. 45)」を参照してください。
6. [Scan on push (プッシュ時にスキャン)] で、リポジトリのイメージスキャン設定を選択します。プッシュ時にスキャンするように設定されたリポジトリは、イメージがプッシュされるたびにイメージスキャンを開始します。それ以外の場合は、イメージスキャンを手動で開始する必要があります。詳細については、「Image scanning (p. 46)」を参照してください。
7. [保存] を選択してリポジトリ設定を更新します。

Deleting a Repository

リポジトリを使用し終わったら、削除できます。AWS マネジメントコンソール でリポジトリを削除すると、リポジトリに含まれているすべてのイメージも削除されます。これを元に戻すことはできません。

リポジトリを削除するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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. [レポジトリ] ページで、削除するレポジトリを選択して [削除] を選択します。
5. の 削除 ***repository_name*** 選択したリポジトリが削除されることを確認し、 削除.

   Important

   選択されたリポジトリ内のイメージもすべて削除されます。

Amazon ECR の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、リポジトリの所有者のみ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権限を許可するポリシードキュメントを適用できます。

レポジトリポリシーと IAM ポリシー

Amazon ECR レ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対象として具体的に使用される 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 レポジトリと IAM ポリシーはどちらも、特定の IAM ユーザーあるいはロールがレポジトリで実行できるアクションを決定するために使用されます。ユーザーあるい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ながら、IAM ポリシーからは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に対して認証したり、Amazon ECR レ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できるためには、事前に IAM ポリシーを通じ

て ecr:GetAuthorizationToken API へのアクセス権限を許可されている必要があります。Amazon ECR は、さまざまなレベルに応じて複数の IAM 管理ポリシーを提供しています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」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、指定する IAM ユーザーがレポジトリ内でレポジトリとイメージを説明することを許可する Amazon ECR レポジトリポリシーを示しています。

```
{
  "Version": "2008-10-17",
  "Statement": [{
    "Sid": "ECR Repository Policy",
    "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、「Resources (p. 60)」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ECR Repository Policy",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::account-id:user/username"
    },
    "Action": [
      "ecr:DescribeImages",
      "ecr:DescribeRepositories"
    ],
    "Resource": [
      "arn:aws:ecr:region:account-id:repository/repository-name"
    ]
    }]
}
```

トピック

- リポジトリポリシーステートメントの設定 (p. 20)
- リポジトリポリシーステートメントの削除 (p. 21)
- Amazon ECR リポジトリポリシーの例 (p. 22)

リポジトリポリシーステートメントの設定

次の手順に従って、AWS マネジメントコンソール でレ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 リポジトリポリシーの例 (p. 22)」を参照してください。

Important

Amazon ECR では、ユーザーがレジストリに対して認証したり、Amazon ECR レ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できるためには、事前に IAM ポリシーを通じて `ecr:GetAuthorizationToken` API へのアクセス権限を許可されている必要があります。Amazon ECR は、さまざまなレベルに応じて複数の IAM 管理ポリシーを提供しています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」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. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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. [Repositories] ページで、ポリシーステートメントをオンに設定するリポジトリを選択します。
5. ナビゲーションペインの [アクセス許可]、[編集] を選択します。
6. [アクセス権限の編集] ページの [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 ポリシーの要素: プリンシパル」を参照してください。

   - [全員 (*)] チェックボックスをオンにすることにより、すべての認証済み AWS ユーザーにステートメントを適用できます。
   - [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 (例: `ecs.amazonaws.com`) を指定します。
   - [AWS アカウント ID] で、AWS アカウント番号 (例: `111122223333`) を指定し、特定の AWS アカウント内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。
   - [IAM エンティティ] で、ステートメントを適用する AWS アカウント内のロールまたはユーザーを選択します。

     Note

     AWS マネジメントコンソール 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は、set-repository-policy AWS CLI コマンドを使用して適用できます。
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。

リポジトリポリシーステートメントの削除

既存のリポジトリポリシーステートメントをリポジトリに適用する必要がなくなった場合、削除できます。

リポジトリポリシーステートメントを削除するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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削除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. [Repositories] ページで、ポリシーステートメントを削除するリポジトリを選択します。
5. ナビゲーションペインの [アクセス許可]、[編集] を選択します。
6. [アクセス権限の編集] ページの [削除] を選択します。

Amazon ECR リポジトリポリシーの例

以下の例では、Amazon ECR リポジトリに対してユーザーが所有するアクセス権限を制御するために使用できるポリシー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に対して認証したり、Amazon ECR レ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できるためには、事前に IAM ポリシーを通じて `ecr:GetAuthorizationToken` API へのアクセス権限を許可されている必要があります。Amazon ECR は、さまざまなレベルに応じて複数の IAM 管理ポリシーを提供しています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」を参照してください。

例: アカウント内での IAM ユーザーの許可

次のリポジトリポリシーでは、アカウント内の IAM ユーザーにイメージのプッシュとプルが許可されます。

```
{
    "Version": "2008-10-17",
    "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:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload"
            ]
        }
    ]
}
```

例: 別のアカウントの許可

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。

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPush",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:root"
            },
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload"
            ]
        }
    ]
}
```

次のリポジトリポリシーでは、一部の IAM 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

AWS マネジメントコンソール 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は、set-repository-policy AWS CLI コマンドを使用して適用できます。

```
{
    "Version": "2008-10-17",
    "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:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ]
        },
        {
            "Sid": "AllowAll",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:user/admin-user"
            },
            "Action": [
                "ecr:*"
            ]
        }
    ]
}
```

例: すべての AWS アカウントにイメージのプルを許可する

次のリポジトリポリシーでは、すべての AWS アカウントにイメージのプルが許可されます。

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowPull",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ]
        }
    ]
}
```

例: すべて拒否

次のリポジトリポリシーでは、すべてのユーザーに対してすべてのイメージのプルが拒否されます。

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "DenyPull",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ]
        }
    ]
}
```

例: 特定の IP アドレスへのアクセスの制限

次の例は、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付与します。ただし、リクエストは条件で指定された IP アドレス範囲からのリクエストである必要があります。

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。ただし、`54.240.143.188` を除きます。

`Condition` ブロックでは、`IpAddress` および `NotIpAddress` 条件と、`aws:SourceIp` 条件キーが使用されています。これは AWS 全体を対象とする条件キーです。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` Pv4 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": "Allow",
            "Principal": "*",
            "Action": "ecr:*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "54.240.143.188/32"
                },
                "IpAddress": {
                    "aws:SourceIp": "54.240.143.0/24"
                }
```

```
            }
        }
    ]
}
```

例: サービスにリンクされたロール

次のリポジトリポリシーは、AWS CodeBuild に、そのサービスとの統合に必要な Amazon ECR API アクションへのアクセスを許可します。詳細については、AWS CodeBuild ユーザーガイドの「CodeBuild の Amazon ECR サンプル」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"
            ]
        }
    ]
}
```

Amazon ECR リポジトリのタグ付け

Amazon ECR リポジトリを管理しやすいように、必要に応じて、独自のメタデータをタグ形式で各リポジトリに割り当てることができます。ここでは、タグとその作成方法について説明します。

コンテンツ

タグの基本

タグとは、AWS リソースに付けるラベルです。タグはそれぞれ、1 つのキーとオプションの 1 つの値で構成されており、どちらもお客様側が定義します。

タグを使用すると、AWS リソースを用途、所有者、環境などのさまざまな方法で分類できます。同じ型のリソースが多い場合に役立ちます — 割り当てたタグに基づいて特定のリソースをすばやく識別できます。たとえば、各リポジトリの所有者を追跡しやすくするため、アカウントの Amazon ECR リポジトリに対して一連のタグを定義できます。

ニーズを満たす一連のタグキーを考案されることをお勧めします。一貫性のあるタグキーセットを使用することで、リソースの管理が容易になります。追加したタグに基づいてリソースを検索およびフィルタリングできます。

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。また、タグは自動的にリソースに割り当てられます。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、古い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。

タグは、AWS マネジメントコンソール、AWS CLI、および Amazon ECR API を使用して操作できます。

AWS Identity and Access Management (IAM) を使用している場合は、AWS アカウント内のどのユーザーがタグを作成、編集、削除するアクセス許可を持つかを制御できます。

リソースにタグを付ける

新規または既存の Amazon ECR リポジトリにタグ付けできます。

Amazon ECR コンソールを使用している場合、新規リソースには作成時にタグを適用でき、既存のリソースには、ナビゲーションペインの [Tags (タグ)] オプションを使用していつでもタグを適用できます。

Amazon ECR API、AWS CLI、または AWS SDK を使用している場合、CreateRepository API アクションの `tags` パラメータを使用して新規リポジトリにタグを適用するか、`TagResource` API アクションを使用して既存のリソースにタグを適用することができます。詳細については、「TagResource」を参照してください。

また、リポジトリの作成時にタグを適用できない場合は、リポジトリ作成プロセスがロールバックされます。これにより、リポジトリがタグ付きで作成されるか、まったく作成されないようになるため、タグ付けされていないリポジトリが存在することがなくなります。作成時にリポジトリにタグ付けすることで、リポジトリ作成後にカスタムタグ付けスクリプトを実行する必要がなくなります。

タグの制限

タグには以下のような基本制限があります。

- リポジトリあたりのタグの最大数 – 50
- タグキーは、リポジトリごとにそれぞれ一意である必要があります。また、各タグキーに設定できる値は 1 つのみです。
- キーの最大長 – 128 文字 (Unicode) (UTF-8)
- 値の最大長 – 256 文字 (Unicode) (UTF-8)
- 複数のサービス間およびリソース間でタグ付けスキーマを使用する場合、他のサービスでも許可される文字に制限が適用されることがあるのでご注意ください。一般的に使用が許可される文字は、UTF-8 で表現できる文字、数字、スペース、および + 、 - 、 = 、 . 、 _ 、 : 、 / 、 @。
- タグのキーと値は大文字と小文字が区別されます。
- キーと値のどちらにも `aws:` プレフィックスは使用しないでください。このプレフィックスは AWS で使用するために予約されています。このプレフィックスが含まれるタグのキーや値を編集したり削除することはできません。このプレフィックスを持つタグは、リソースあたりのタグ数の制限時には計算されません。

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。詳細については、「Amazon ECR Usage Reports (p. 84)」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。たとえば、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 and Cost Management 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。

コンソールでのタグの処理

新規または既存のリポジトリに関連付けられたタグは、Amazon ECR コンソールを使用して管理することができます。

Amazon ECR コンソールで特定のリポジトリを選択する場合、ナビゲーションペインの [タグ] を選択してタグを表示することができます。

リポジトリにタグを追加するには

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、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. [Repositories] ページで、表示するリポジトリを選択します。
5. [Repositories: ***repository_name***] ページで、ナビゲーションペインの [タグ] を選択します。
6. [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. [Edit Tags (タグの編集)] ページで、各タグのキーと値を指定してから [Save (保存)] を選択します。

個々のリソースからタグを削除するには

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、使用するリージョンを選択します。
3. [Repositories] ページで、表示するリポジトリを選択します。
4. [Repositories: ***repository_name***] ページで、ナビゲーションペインの [タグ] を選択します。
5. [タグ] ページで、[編集] を選択します。
6. [Edit Tags (タグの編集)] ページで、削除するタグごとに [Remove (削除)] を選択してから [Save (保存)] を選択します。

AWS CLI または API でのタグの操作

リソースのタグの追加、更新、リスト表示、および削除には、次を使用します。対応するドキュメントに例が記載されています。

Amazon ECR リソースへのタグ付けのサポート

タスク	AWS CLI	API アクション
1 つ以上のタグを追加、または上書きします。	tag-resource	TagResource
1 つ以上のタグを削除します。	untag-resource	UntagResource

AWS CLI を使用してタグを管理する方法を以下の例に示します。

例 1: 既存のリポジトリにタグ付けする

次のコマンドでは、既存のリポジトリにタグ付けします。

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

例 2: 既存のリポジトリに複数のタグをタグ付けする

次のコマンドでは、既存のリポジトリにタグ付けします。

```
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: 既存のリポジトリからタグを削除する

次のコマンドでは、既存のリポジトリからタグを削除します。

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

例 4: リポジトリのタグを一覧表示する

次のコマンドでは、既存の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 5: リポジトリを作成してタグを適用する

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
```

Images

Amazon Elastic Container Registry (Amazon ECR) は Docker および Open Container Initiative (OCI) イメージをイメージリポジトリに保存します。Docker CLI 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。

Important

Amazon ECR では、ユーザーがレジストリに対して認証したり、Amazon ECR レ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できるためには、事前に IAM ポリシーを通じて `ecr:GetAuthorizationToken` API へのアクセス権限を許可されている必要があります。Amazon ECR は、さまざまなレベルに応じて複数の IAM 管理ポリシーを提供しています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」を参照してください。

Pushing an image

docker push コマンドを使用して、Docker イメージまたは Open Container Initiative (OCI) イメージを Amazon ECR リポジトリにプッシュできます。

Important

Amazon ECR では、ユーザーがレジストリに対して認証したり、Amazon ECR レ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できるためには、事前に IAM ポリシーを通じて `ecr:GetAuthorizationToken` API へのアクセス権限を許可されている必要があります。Amazon ECR は、さまざまなレベルに応じて複数の IAM 管理ポリシーを提供しています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」を参照してください。

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。マニフェストリストで参照される各イメージは、すでにリポジトリにプッシュされている必要があります。詳細については、「Pushing a multi-architecture image (p. 30)」を参照してください。

Docker または Open Container Initiative (OCI) 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、「レジストリの認証 (p. 13)」を参照してください。
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、「Creating a Repository (p. 16)」を参照してください。
3. プッシュするイメージを識別します。docker images コマンドを実行し、システム上のイメージを一覧表示します。

   ```
   docker images
   ```

   イメージは、*repository:tag* 値、または結果のコマンド出力内のイメージID。
4. 使用する Amazon ECR レジストリ、リポジトリ、オプションのイメージタグ名の組み合わせによってイメージをタグ付けします。レジストリ形式は *aws_account_id*.dkr.ecr.*region*.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 を持つイメージにタグを付けます。*e9ae3c220b23* として *aws_account_id*.dkr.ecr.*region*.amazonaws.com/my-web-app

```
docker tag e9ae3c220b23 aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app
```

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.region.amazonaws.com/my-web-app
```

6. (オプション) 任意の追加のタグをイメージに適用し、「Step 4 (p. 29)」および「Step 5 (p. 30)」を繰り返して、それらのタグを Amazon ECR にプッシュします。Amazon ECR では、イメージあたり最大 100 個のタグを適用できます。

Pushing a multi-architecture image

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をサポートしています。A manifest list 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。通常、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が、これ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。

> Important
>
> この機能を使用するには、Docker CLI で実験機能が有効になっている必要があります。詳細については、「実験機能」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。

次のステップを使用して、Docker マニフェストリストを作成して Amazon ECR リポジトリにプッシュできます。Docker マニフェストで参照するには、イメージがすでにリポジトリにプッシュされている必要があります。イメージのプッシュについては、「Pushing an image (p. 29)」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、「レジストリの認証 (p. 13)」を参照してください。
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-web-app
```

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.region.amazonaws.com/my-web-app aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app:image_one_tag aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app: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.region.amazonaws.com/my-web-app
```

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.region.amazonaws.com/my-web-app
```

Pulling an image

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他の AWS アカウントに関連付けられたレジストリから行うことができます。Amazon ECS タスク定義で Amazon ECR イメージを使用する場合は、「Amazon ECR イメージを Amazon ECS で使用する (p. 51)」を参照してください。

Important

Amazon ECR では、ユーザーがレジストリに対して認証したり、Amazon ECR レ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できるためには、事前に IAM ポリシーを通じて `ecr:GetAuthorizationToken` API へのアクセス権限を許可されている必要があります。Amazon ECR は、さまざまなレベルに応じて複数の IAM 管理ポリシーを提供しています。詳細については、「Amazon Elastic Container Registry Identity-Based Policy Examples (p. 64)」を参照してください。

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、「レジストリの認証 (p. 13)」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、「レジストリの認証 (p. 13)」を参照してください。

Deleting an image

イメージの使用が終わったら、リポジトリから削除できます。イメージは、AWS マネジメントコンソールまたは AWS CLI を使用して削除できます。

> Note
>
> リポジトリを使い終えた場合、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、「Deleting a Repository (p. 19)」を参照してください。

AWS マネジメントコンソール を使用してイメージを削除するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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. [Repositories] 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. の リポジトリ: ***repository_name*** ページで、削除する画像の左側にあるボックスを選択し、 削除.
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。

AWS CLI を使用してイメージを削除するには

1. リポジトリ内のイメージを一覧表示し、イメージタグまたはダイジェストを使用して識別できるようにします。

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

2. (オプション) 削除するイメージのタグを指定して、イメージの不要なタグを削除します。

> Note
>
> イメージの最後のタグを削除すると、イメージは削除されます。

```
aws ecr batch-delete-image --repository-name my-repo --image-ids imageTag=latest
```

3. 削除するイメージのダイジェストを指定することで、イメージを削除します。

> Note
>
> 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=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304c7c2c1a9d6fa3e9de6bf552d
```

Retagging an image

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。

To retag an image (AWS CLI)

AWS CLI を使用してイメージにもう一度タグを付けるには

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環境変数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、 タグ付きのイメージのマニフェストです。*latest*リポジトリ内の、*amazonlinux*は環境変数に書き込まれます。*MANIFEST*.

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

2. put-image コマンドの `--image-tag` 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、イメージは次のようにタグ付けされています。*2017.03*.

   Note

   使用している AWS CLI のバージョンで `--image-tag` オプションが使用できない場合は、最新バージョンにアップグレードします。詳細については、https://docs.aws.amazon.com/cli/latest/userguide/ の「AWS Command Line Interface ユーザーガイドAWS コマンドラインインターフェイスのインストール」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。出力ウィンドウで、イメージにはタグ `latest` および `2017.03` が付いています。

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

   Output

```
{
    "imageDetails": [
        {
            "imageSizeInBytes": 98755613,
            "imageDigest":
 "sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a2685dfe6f247227",
            "imageTags": [
                "latest",
                "2017.03"
            ],
            "registryId": "aws_account_id",
            "repositoryName": "amazonlinux",
            "imagePushedAt": 1499287667.0
        }
    ]
}
```

To retag an image (AWS Tools for Windows PowerShell)

AWS Tools for Windows PowerShell を使用してイメージにもう一度タグを付けるには

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、タグ、*latest*リポジトリ内の、*amazonlinux*は環境変数に書き込まれます。*$Image*.

Note

お持ちでない場合は、 Get-ECRImageBatch システムで cmdlet を使用できます。 を参照してください。 の設定 AWS Tools for Windows PowerShell の AWS Tools for Windows PowerShell ユーザーガイド.

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

2. イメージのマニフェストを *$Manifest* 環境変数

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

3. Write-ECRImage コマンドレットの `-ImageTag` 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、イメージは次のようにタグ付けされています。*2017.09*.

```
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。出力ウィンドウで、イメージにはタグ `latest` および `2017.09` が付いています。

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

Output

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

Amazon ECR Lifecycle Policies

Amazon ECR ライフサイクルポリシーで、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。ライフサイクルポリシーは 1 つまたは複数のルールのセットであり、各ルールでは Amazon ECR へのアクションが定義されています。アクションは、指定された文字列がプレフィックスとして付いているタグを含むイメージに適用されます。これにより、未使用のイメージ、たとえば、経過時間またはカウントに基づく有効期限切れイメージなどのクリーンアップを自動化できます。ライフサイクルポリシーの作成後、影響を受けるイメージは 24 時間以内に有効期限切れになります。

トピック

Lifecycle Policy Template

ライフサイクルポリシーのコンテンツは、リポジトリに関連付けられ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。ライフサイクルポリシーの例については、「Examples of Lifecycle Policies (p. 39)」を参照してください。

```
{
    "rules": [
        {
            "rulePriority": integer,
            "description": "string",
            "selection": {
                "tagStatus": "tagged"|"untagged"|"any",
                "tagPrefixList": list<string>,
                "countType": "imageCountMoreThan"|"sinceImagePushed",
                "countUnit": "string",
                "countNumber": integer
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Note

は、 tagPrefixList パラメータは、 tagStatus はです tagged。 は、 countUnit パラメータは、 countType はです sinceImagePushed。 は、 countNumber パラメータは、 countType は に設定されます。 imageCountMoreThan.

Lifecycle Policy Parameters

ライフサイクルポリシーは、次の部分に分けられます。

トピック

Rule Priority

`rulePriority`

タイプ: 整数

必須: はい

ルールが評価される順序を低いものから高いものへと設定します。優先順位が 1 のライフサイクルポリシールールが最初に処理され、次に優先順位 2 のルールが処理されます。以降も優先順位に従って

処理されます。ライフサイクル ポリシーにルールを追加する場合は、それぞれに `rulePriority`。値はポリシー内のルール間でシーケンシャルであ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。

Description

`description`

タイプ: 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。

Tag Status

`tagStatus`

タイプ: 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。許容されるオプションは次のとおりです。 `tagged`、 `untagged`、または `any`。 指定する場合 `any`すべての画像にルールが適用されます。`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。

Tag Prefix List

`tagPrefixList`

タイプ: list[string]

必須: `tagStatus` が [tagged] に設定されている場合のみ、必須

指定した場合のみ使用 `"tagStatus": "tagged"`。 ライフサイクル ポリシーでアクションを実行するイメージ タグ プレフィックスのコンマ区切り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。

Count Type

`countType`

タイプ: 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。

Count Unit

`countUnit`

タイプ: 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。

Count Number

`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。

Action

`type`

タイプ: 文字列

必須: はい

アクションタイプを指定します。サポートされる値は `expire` です。

Lifecycle Policy Evaluation Rules

ライフサイクルポリシーの評価者は、プレーンテキスト JSON を解析し、指定したリポジトリ内のイメージへ適用します。ライフサイクルのポリシーを作成する際は次のルールに注意が必要です。

- An image is expired by exactly one or zero rules.
- An image that matches the tagging requirements of a rule cannot be expired by a rule with a lower priority.
- Rules can never mark images that are marked by higher priority rules, but can still identify them as if they haven't been expired.
- The set of rules must contain a unique set of tag prefixes.
- Only one rule is allowed to select untagged images.
- Expiration is always ordered by `pushed_at_time`, and always expires older images before newer ones.
- When using the `tagPrefixList`, an image is successfully matched if all of the tags in the `tagPrefixList` value are matched against any of the image's tags.

- With `countType = imageCountMoreThan`, images are sorted from youngest to oldest based on `pushed_at_time` and then all images greater than the specified count are expired.
- With `countType = sinceImagePushed`, all images whose `pushed_at_time` is older than the specified number of days based on `countNumber` are expired.

Creating a Lifecycle Policy Preview

ライフサイクルポリシーのプレビューでは、実行する前にイメージリポジトリのライフサイクルポリシーの影響を確認できます。以下の手順では、ライフサイクルポリシーのプレビューを作成する方法を示します。

コンソールを使用してライフサイクルポリシーのプレビューを作成するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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 [Repositories] を選択し、リポジトリを選択します。
4. の リポジトリ: ***repository_name*** ページのナビゲーションペインで、 ライフサイクル ポリシー.
5. の リポジトリ: ***repository_name***:ライフサイクル ポリシー ページ、選択 テストルールの編集、ルールの作成.
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。
   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. [Image status (イメージステータス)] で、[Tagged (タグ付け)]、[Untagged (タグ付けなし)]、または [Any (すべて)] を選択します。
   d. [イメージのステータス] で `Tagged` を指定した場合は、[タグのプレフィックス] で、ライフサイクルポリシーでアクションを実行するイメージタグのリストをオプションで指定できます。[`Untagged`] を指定した場合は、このフィールドは空にする必要があります。
   e. [一致条件] では、[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] の値を選択します (該当する場合)。
7. [Save] を選択します。
8. 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。
9. 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。
10. [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。
11. 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。

    Note

    ライフサイクルポリシーの作成後、影響を受けるイメージは 24 時間以内に有効期限切れになります。

Creating a Lifecycle Policy

ライフサイクルポリシーで、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できます。以下の手順では、ライフサイクルポリシーを作成する方法を示します。ライフサイクルポリシーの作成後、影響を受けるイメージは 24 時間以内に有効期限切れになります。

AWS CLI を使用してライフサイクルポリシーを作成するには

1. ライフサイクルポリシーを作成するためのリポジトリの ID を取得します。

```
aws ecr describe-repositories
```

2. ライフサイクルポリシーの作成

```
aws ecr put-lifecycle-policy [--registry-id <string>] --repository-name <string> --
lifecycle-policy-text <string>
```

コンソールを使用してライフサイクルポリシーを作成するには

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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 [Repositories] を選択し、リポジトリを選択します。
4. の リポジトリ: ***repository_name*** ページのナビゲーションペインで、 ライフサイクル ポリシー.
5. の リポジトリ: ***repository_name***:ライフサイクル ポリシー ページ、選択 ルールの作成.
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。
   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. [Image status (イメージステータス)] で、[Tagged (タグ付け)]、[Untagged (タグ付けなし)]、または [Any (すべて)] を選択します。
   d. [イメージのステータス] で `Tagged` を指定した場合は、[タグのプレフィックス] で、ライフサイクルポリシーでアクションを実行するイメージタグのリストをオプションで指定できます。[`Untagged`] を指定した場合は、このフィールドは空にする必要があります。
   e. [一致条件] では、[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] の値を選択します (該当する場合)。
7. [Save] を選択します。

Examples of Lifecycle Policies

次に、ライフサイクルポリシーの例を構文で示します。

トピック

Filtering on Image Age

次の例で、タグ付けされていない `14` 日以上経ったイメージを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": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Filtering on Image Count

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"
            }
        }
    ]
}
```

Filtering on Multiple Rules

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。

Example 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",
                "tagPrefixList": ["prod"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["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。

- Rule 1 identifies images tagged with prefix `prod`. It should mark images, starting with the oldest, until there is one or fewer images remaining that match. It marks Image A for expiration.
- Rule 2 identifies images tagged with prefix `beta`. It should mark images, starting with the oldest, until there is one or fewer images remaining that match. It marks both Image A and Image B for expiration. However, Image A has already been seen by Rule 1 and if Image B were expired it would violate Rule 1 and thus is skipped.
- Result: Image A is expired.

Example 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",
                "tagPrefixList": ["beta"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
```

```
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["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。

- Rule 1 identifies images tagged with `beta`. It should mark images, starting with the oldest, until there is one or fewer images remaining that match. It sees all three images and would mark Image A and Image B for expiration.
- Rule 2 identifies images tagged with `prod`. It should mark images, starting with the oldest, until there is one or fewer images remaining that match. It would see no images because all available images were already seen by Rule 1 and thus would mark no additional images.
- Result: Images A and B are expired.

Filtering on Multiple Tags in a Single Rule

次の例で、単一のルールでの複数のタグプレフィックス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。

Example A

単一のルールで複数のタグプレフィックス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プレフィックス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",
                "tagPrefixList": ["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。

- Rule 1 identifies images tagged with `alpha` and `beta`. It sees images C and F. It should mark images that are older than five days, which would be Image C.
- Result: Image C is expired.

Example 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",
                "tagPrefixList": ["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。

- Rule 1 identifies images tagged with `alpha` and `beta`. It sees all images. It should mark images, starting with the oldest, until there is one or fewer images remaining that match. It marks image A and B for expiration.
- Result: Images A and B are expired.

Filtering on All Images

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。

Example 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。

- Rule 1 identifies all images. It sees images A, B, C, and D. It should expire all images other than the newest one. It marks images A, B, and C for expiration.
- Result: Images A, B, and C are expired.

Example 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-1", "beta-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",
                "tagPrefixList": ["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ロジックは次のようになります。

- Rule 1 identifies images tagged with `alpha`. It identifies images A and C. It should keep the newest image and mark the rest for expiration. It marks image A for expiration.
- Rule 2 identifies untagged images. It identifies images B and E. It should mark all images older than one day for expiration. It marks image B for expiration.
- Rule 3 identifies all images. It identifies images A, B, C, D, and E. It should keep the newest image and mark the rest for expiration. However, it can't mark images A, B, C, or E because they were identified by higher priority rules. It marks image D for expiration.
- Result: Images A, B, and D are expired.

Image tag mutability

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を変更不可に設定できます。リポジトリをタグ変更不可に設定すると、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した場合に `ImageTagAlreadyExistsException` エラーが返されます。

イメージタグ変更不可の設定は、AWS マネジメントコンソールと AWS CLI ツールを使用して、新しいリポジトリの作成中または既存のリポジトリでいつでも行うことができます。コンソール出の手順については、「Creating a Repository (p. 16)」および「Editing a Repository (p. 18)」を参照してください。

To create a repository with immutable tags configured

Use one of the following commands to create a new image repository with immutable tags configured.

- 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
```

To update the image tag mutability settings for an existing repository

Use one of the following commands to update the image tag mutability settings for an existing repository.

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

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

- Write-ECRImageTagMutability (AWS Tools for Windows PowerShell)

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

Image scanning

Amazon ECR イメージスキャンは、コンテナイメージのソフトウェアの脆弱性を特定するのに役立ちます。 Amazon ECR は、オープンソースの Clair プロジェクトから Common Vulnerabilities and Exposures (CVE) データベースを使用し、スキャン所見のリストを提供します。デプロイされているコンテナイメージのセキュリティに関する情報については、スキャンの結果を確認できます。Clair の詳細については、以下を参照してください。 クレア GitHubで

Amazon ECR では、可能であれば、アップストリームのディストリビューションソースからの CVE の重大度 が使用されます。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。

Amazon ECR に保存されているコンテナイメージを手動でスキャンすることも、リポジトリにイメージをプッシュするときにイメージをスキャンするようにリポジトリを設定することもできます。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。Amazon ECR は、イメージスキャンの完了時にイベントを Amazon EventBridge (旧称 CloudWatch イベント) に送信します。詳細については、「Amazon ECR Events and EventBridge (p. 84)」を参照してください。

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、「Troubleshooting Image Scanning Issues (p. 105)」を参照してください。

トピック

- Configuring a repository to scan on push (p. 47)
- Manually scanning an image (p. 48)
- Retrieving image scan findings (p. 49)

Configuring a repository to scan on push

イメージスキャン設定は、作成時に新しいリポジトリに対しても、既存のリポジトリに対しても設定できます。[scan on push (プッシュ時にスキャン)] が有効な場合、イメージはリポジトリにプッシュされた後にスキャンされます。リポジトリで [scan on push (プッシュ時にスキャン)] が無効になっている場合は、スキャン結果を得るために各イメージスキャンを手動で開始する必要があります。

トピック

Creating a new repository to scan on push

新しいリポジトリが [scan on push (プッシュ時にスキャン)] するように設定されている場合、リポジトリにプッシュされたすべての新しいイメージがスキャンされます。最後に完了したイメージスキャンの結果を取得できます。詳細については、「Retrieving image scan findings (p. 49)」を参照してください。

AWS マネジメントコンソール の手順については、「Creating a Repository (p. 16)」を参照してください。

To create a repository configured for scan on push (AWS CLI)

イメージの [scan on push (プッシュ時にスキャン)] が設定された新しいリポジトリを作成するには、次のコマンドを使用します。

- create-repository (AWS CLI)

```
aws ecr create-repository --repository-name name --image-scanning-configuration
 scanOnPush=true --region us-east-2
```

To create a repository configured for scan on push (AWS Tools for Windows PowerShell)

イメージの [scan on push (プッシュ時にスキャン)] が設定された新しいリポジトリを作成するには、次のコマンドを使用します。

- New-ECRRepository (AWS Tools for Windows PowerShell)

```
New-ECRRepository -RepositoryName name -ImageScanningConfiguration_ScanOnPush true -
Region us-east-2 -Force
```

Configure an existing repository to scan on push

既存のリポジトリは、イメージをリポジトリにプッシュするときにスキャンするように設定できます。この設定は、今後のイメージプッシュに適用されます。最後に完了したイメージスキャンの結果を取得できます。詳細については、「Retrieving image scan findings (p. 49)」を参照してください。

AWS マネジメントコンソール の手順については、「Editing a Repository (p. 18)」を参照してください。

To edit the settings of an existing repository (AWS CLI)

既存のリポジトリのイメージスキャン設定を編集するには、次のコマンドを使用します。

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

```
aws ecr put-image-scanning-configuration --repository-name name --image-scanning-
configuration scanOnPush=true --region us-east-2
```

Note

To disable image scan on push for a repository, specify `scanOnPush=false`.

To edit the settings of an existing repository (AWS Tools for Windows PowerShell)

既存のリポジトリのイメージスキャン設定を編集するには、次のコマンドを使用します。

- New-ECRRepository (AWS Tools for Windows PowerShell)

```
Write-ECRImageScanningConfiguration -RepositoryName name -
ImageScanningConfiguration_ScanOnPush true -Region us-east-2 -Force
```

Manually scanning an image

[scan on push (プッシュ時にスキャン)] するように設定されていないリポジトリ内のイメージをスキャンする場合は、イメージスキャンを手動で開始できます。イメージは 1 日に 1 回しかスキャンできません。この制限には、最初の [scan on push (プッシュ時にスキャン)] (有効になっている場合) 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、「Troubleshooting Image Scanning Issues (p. 105)」を参照してください。

To start a manual scan of an image (console)

AWS マネジメントコンソール 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. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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. [Repositories (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。

To start a manual scan of an image (AWS CLI)

イメージの手動スキャンを開始するには、次の AWS CLI コマンドを使用します。`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。

- start-image-scan (AWS CLI)

The following example uses an image tag.

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

The following example uses an image digest.

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

To start a manual scan of an image (AWS Tools for Windows PowerShell)

イメージの手動スキャンを開始するには、次の AWS Tools for Windows PowerShell コマンドを使用します。`ImageId_ImageTag` または `ImageId_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 Get-ECRImage CLI コマンドを使用して取得できます。

- Get-ECRImageScanFinding (AWS Tools for Windows PowerShell)

  The following example uses an image tag.

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

  The following example uses an image digest.

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

Retrieving image scan findings

最後に完了したイメージスキャンのスキャン結果を取得できます。結果には、検出されたソフトウェアの脆弱性が、共通の脆弱性およびエクスポージャー (CVE) データベースに基づく重大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、「Troubleshooting Image Scanning Issues (p. 105)」を参照してください。

To retrieve image scan findings (console)

AWS マネジメントコンソール 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。

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) 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. [Repositories (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. [Images (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。

To retrieve image scan findings (AWS CLI)

以下の AWS CLI コマンドを使用して、AWS CLI でイメージスキャンの結果を取得します。`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)

  The following example uses an image tag.

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

The following example uses an image digest.

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

To retrieve image scan findings (AWS Tools for Windows PowerShell)

以下の AWS Tools for Windows PowerShell コマンドを使用して、イメージスキャンの結果を取得します。`ImageId_ImageTag` または `ImageId_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 Get-ECRImage CLI コマンドを使用して取得できます。

- Get-ECRImageScanFinding (AWS Tools for Windows PowerShell)

  The following example uses an image tag.

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

  The following example uses an image digest.

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

Container image manifest formats

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 (used with Docker version 1.9 and older)
- Docker Image Manifest V2 Schema 2 (used with Docker version 1.10 and newer)
- Open Container Initiative (OCI) Specifications (v1.0 and up)

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

- The ability to use multiple tags per image.
- Support for storing Windows container images. For more information, see Pushing Windows Images to Amazon ECR in the Amazon Elastic Container Service Developer Guide.

Amazon ECR image manifest conversion

Amazon ECR との間で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て、クライアントとレジストリで理解される、イメージで使用す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。

画像を Amazon ECR by tag、 Amazon ECR リポジトリに保存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、クライアントによって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。以下の表は、がサポートする使用可能な変換を示しています。 Amazon ECR イメージが引き出されたとき by tag:

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

画像を引っ張って by digest、使用可能な翻訳はありません。クライアントは、 Amazon ECR. 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、同じイメージをリクエストすると、 by tag、 Amazon ECR は、イメージマニフェストをクライアントが理解できる形式に変換します。イメージのプルが成功します。

Amazon ECR イメージを Amazon ECS で使用する

Amazon ECR でホストされているコンテナイメージを Amazon ECS タスク定義で使用できますが、次の前提条件を満たす必要があります。

- Amazon ECS タスクで EC2 起動タイプを使用する場合、コンテナインスタンスでは、バージョン 1.7.0 以降の Amazon ECS コンテナエージェントを使用する必要があります。Amazon ECS に最適化された最新バージョンの AMI では、タスク定義の Amazon ECR イメージがサポートされています。最新の Amazon ECS 最適化 AMI の ID などの詳細については、Amazon Elastic Container Service Developer Guideの「Amazon ECS に最適化された AMI バージョン」を参照してください。
- 使用する Amazon ECS コンテナインスタンス IAM ロール (`ecsInstanceRole`) には、Amazon ECR に対する次の IAM ポリシーアクセス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": "*"
```

```
        }
    ]
}
```

AmazonEC2ContainerServiceforEC2Role 管理ポリシーを使用する場合、コンテナインスタンスの IAM ロールに適切なアクセス許可が付与されます。ロールが Amazon ECR をサポートしていることを確認するには、Amazon Elastic Container Service Developer Guideの「Amazon ECS コンテナインスタンスの IAM ロール」を参照してください。

- Amazon ECS タスク定義で、`registry/repository:tag` イメージに完全な Amazon ECR 名前付けを使用していることを確認してください。たとえば、*aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-web-app*`:`*latest* と指定します。

  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-web-app:latest",
            ...
        }
    ],
    ...
}
```

Amazon ECR イメージを Amazon EKS で使用する

Amazon EKS で Amazon ECR イメージを使用できます。ただし、以下の前提条件を満たす必要があります。

- ワーカーノードで使用する Amazon EKS ワーカーノード IAM ロール (`NodeInstanceRole`) には、Amazon ECR の次の IAM ポリシーアクセス許可が必要です。

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

  Note

  `eksctl` または「Amazon EKS の使用開始」の AWS CloudFormation テンプレートを使用してクラスターとワーカーノードグループを作成した場合、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 IAM ロールに適用されます。

- Amazon ECR からイメージを参照する場合は、イメージに完全な `registry/repository:tag` 名前を使用する必要があります(*aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest* など)。

Amazon Linux container image

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。これは、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用にすでに Amazon Linux AMI を使用している場合、Amazon Linux コンテナイメージで簡単に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 を使ってアプリケーションを AWS クラウドにプッシュできます。詳細については、「Amazon ECR イメージを Amazon ECS で使用する (p. 51)」を参照してください。

Amazon Linux コンテナイメージは Amazon ECR の および Docker Hub で使用できます。Amazon Linux コンテナイメージのサポートは、AWS 開発者フォーラムにアクセスして見つけることができます。

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

1. Amazon Linux コンテナイメージの 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 12 時間有効です。詳細については、「レジストリの認証 (p. 13)」を参照してください。

   Note

   は、 get-login-password コマンドは、 AWS CLI バージョンから開始 `1.17.10`。 詳細については、以下を参照してください。 AWS コマンド ライン インターフェイスのインストール の AWS Command Line Interface ユーザーガイド.

   ```
   aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 137112412989.dkr.ecr.us-east-1.amazonaws.com
   ```

   Output

   ```
   Login succeeded
   ```

   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、https://docs.aws.amazon.com/cli/latest/userguide/installing.html の「AWS Command Line Interface ユーザーガイドAWS コマンドラインインターフェイスのインストール」を参照してください。
2. (オプション) Amazon Linux リポジトリ内のイメージは、aws ecr list-images コマンドでリストすることができます。`latest` タグは、使用可能な最新の Amazon Linux コンテナイメージに常に対応します。

   ```
   aws ecr list-images --region us-east-1 --registry-id 137112412989 --repository-name amazonlinux
   ```
3. 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 137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest
   ```
4. (オプション) コンテナをローカルに実行します。

```
docker run -it 137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest /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
   ```

Security in Amazon Elastic Container Registry

AWS では、クラウドのセキュリティが最優先事項です。AWS のお客様は、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利点を得られます。

セキュリティは、AWS とお客様の間の共有責任です。は、 共有責任モデル では、これをセキュリティとして説明します。 of クラウドとセキュリティ in クラウド:

- Security of the cloud – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon ECR, see AWS Services in Scope by Compliance Program.
- Security in the cloud – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

このドキュメントでは、Amazon ECR を使用する際に責任共有モデルを適用する方法について説明します。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように 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セキュリティ保護に役立つ他の 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:

Identity and Access Management for Amazon Elastic Container Registry

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コントロールするために役立つ AWS のサービスです。IAM 管理者は、Amazon ECR リソースを使用するために認証 (サインイン) および承認 (アクセス許可を持つ) される者を制御します。IAM は、追加料金なしで使用できる AWS のサービスです。

トピック

Audience

AWS Identity and Access Management (IAM) の用途は、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、管理者が必要なアクセス許可と認証情報を用意します。作業を実行するためにさらに多くの Amazon ECR 機能を使用するとき、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者から適切なアクセス許可をリクエストするのに役に立ちます。Amazon ECR の機能にアクセスできない場合は、「Troubleshooting Amazon Elastic Container Registry Identity and Access (p. 68)」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場合は、おそらく Amazon ECR へのフルアクセスがあります。従業員がどの Amazon ECR 機能とリソースアクセスする必要があるかを決定するのは管理者の仕事です。その後で、サービスユーザーのアクセス許可を変更するために、IAM 管理者にリクエストを送信する必要があります。IAM の基本概念については、このページの情報を確認します。お客様の会社で Amazon ECR の IAM を利用する方法の詳細については、「How Amazon Elastic Container Registry Works with IAM (p. 59)」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について確認する場合があります。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 Identity-Based Policy Examples (p. 64)」を参照して ください。

Authenticating With Identities

認証は、アイデンティティ認証情報を使用して AWS にサインインする方法です。AWS マネジメントコンソール を使用するサインインの詳細については、IAM ユーザーガイド の「IAM コンソールとサインインページ」を参照してください。

AWS アカウントのルートユーザー、IAM ユーザーとして、または IAM ロールを引き受けて、認証されている (AWS にサインインしている) 必要があります。会社のシングルサインオン認証を使用することも、Google や Facebook を使用してサインインすることもできます。このような場合、管理者は以前に IAM ロールを使用して ID フェデレーションを設定しました。他の会社の認証情報を使用して AWS にアクセスした場合、ロールを間接的に割り当てられています。

AWS マネジメントコンソール へ直接サインインするには、ルートユーザー E メールまたは IAM ユーザー名とパスワードを使用します。ルートユーザー または IAM を使用して AWS にプログラム的にアクセスできます。AWS では、SDK とコマンドラインツールを提供して、お客様の認証情報を使用して、リクエストに暗号で署名できます。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。これには、インバウンド API リクエストを認証するためのプロトコル、署名バージョン 4 を使用します。リクエストの認証の詳細については、AWS General Referenceの「署名バージョン 4 の署名プロセス」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供を要求される場合もあります。たとえば、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高めることを推奨しています。詳細については、IAM ユーザーガイドの「AWS のデバイスに多要素認証 (MFA) を使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを初めて作成する場合は、このアカウントのすべての AWS サービスとリソースに対して完全なアクセス権限を持つシングルサインインアイデンティティで始めます。このアイデンティティは 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のサインインによりアクセスします。強くお勧めしているのは、日常的なタスクには、それが管理者タスクであっても、ルートユーザーを使用しないことです。代わりに、最初の IAM ユーザーを作成するためだけに ルートユーザー を使用するというベストプラクティスに従います。その後、ルートユーザー認証情報を安全な場所に保管し、それらを使用して少数のアカウントおよびサービス管理タスクのみを実行します。

IAM Users and Groups

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特定のアクセス許可がある AWS アカウント内のアイデンティティです。IAM ユーザーは、ユーザー名とパスワード、アクセスキーのセットなど、長期的な認証情報を持つことができます。アクセスキーを生成する方法の詳細については、IAM ユーザーガイド の「IAM ユーザーのアクセスキーの管理」を参照してください。IAM ユーザーにアクセスキーを生成するとき、必ずキーペアを表示して安全に保存してください。後になって、シークレットアクセスキーを回復することはできません。新しいアクセスキーペアを生成する必要があります。

IAM グループは、IAM ユーザーのコレクション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、一度に複数のユーザーに対してアクセス許可を指定できます。多数の組のユーザーがある場合、グループを使用すると管理が容易になります。たとえば、IAM Admin 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アクセス許可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 人の特定の人または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利用できます。詳細については、IAM ユーザーガイド の「IAM ユーザーの作成が適している場合 (ロールではなく)」を参照してください。

IAM Roles

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、特定のユーザーに関連付けられていません。ロールを切り替えて、AWS マネジメントコンソール 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、AWS CLI または AWS API 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。

- 一時的な IAM ユーザーアクセス許可 – IAM ユーザーは、特定のタスクに対して複数の異なるアクセス許可を一時的に IAM ロールで引き受けることができます。
- フェデレーティッドユーザーアクセス – IAM ユーザーを作成する代わりに、AWS Directory Service、エンタープライズユーザーディレクトリ、またはウェブ ID プロバイダーに既存のアイデンティティを使用できます。このようなユーザーはフェデレーティッドユーザーと呼ばれます。AWS では、ID プロバイダーを通じてアクセスがリクエストされたとき、フェデレーティッドユーザーにロールを割り当てます。フェデレーティッドユーザーの詳細については、IAM ユーザーガイドの「フェデレーティッドユーザーとロール」を参照してください。
- 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信頼済みプリンシパルに許可できます。ロールは、クロスアカウントアクセスを許可する主な方法です。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。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。
- AWS サービスアクセス – サービスロールは、サービスがお客様に代わってお客様のアカウントでアクションを実行するために引き受ける IAM ロールです。一部の AWS のサービス環境を設定するときに、サービスが引き受けるロールを定義する必要があります。このサービスロールには、サービスが必要とする AWS のリソースにサービスがアクセスするために必要なすべてのアクセス権限を含める必要があります。サービスロールはサービスによって異なりますが、多くのサービスロールでは、そのサービスの文書化された要件を満たしている限り、アクセス権限を選択することができます。サービスロールは、お客様のアカウント内のみでアクセスを提供します。他のアカウントのサービスへのアクセス権を付与するためにサービスロールを使用することはできません。IAM 内部からロールを作成、修正、削除できます。たとえば、Amazon Redshift がお客様に代わって Amazon S3 バケットにアクセスし、バケットからデータを Amazon Redshift クラスターにロードすることを許可するロールを作成できます。詳細については、IAM ユーザーガイドのAWS サービスにアクセス権限を委任するロールの作成を参照してください。

- 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 リクエストを作成している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認証情報を取得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「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス権限を付与する」を参照してください。

IAM ロールを使用するべきかどうかについては、IAM ユーザーガイド の「IAM ロール (ユーザーではない) の作成が適している場合」を参照してください。

Managing Access Using Policies

AWS でアクセスをコントロールするには、ポリシーを作成して IAM アイデンティティや AWS リソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、エンティティ (ルートユーザー、IAM ユーザーまたは IAM ロール) によってリクエストが行われると、それらのポリシーを評価します。ポリシーでのアクセス許可により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS 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。

IAM 管理者は、ポリシーを使用して、AWS リソースへのアクセスを許可するユーザーと、これらのリソースで実行できるアクションを指定できます。すべての IAM エンティティ (ユーザーまたはロール) は、アクセス許可のない状態からスタートします。言い換えると、デフォルト設定では、ユーザーは何もできず、自分のパスワードを変更することすらできません。何かを実行するアクセス許可をユーザーに付与するには、管理者がユーザーにアクセス許可ポリシーをアタッチする必要があります。また、管理者は、必要なアクセス許可があるグループにユーザーを追加できます。管理者がグループにアクセス許可を付与すると、そのグループ内のすべてのユーザーにこれらのアクセス許可が付与されます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションのアクセス許可を定義します。たとえば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS マネジメントコンソール、AWS CLI、または AWS API からロールの情報を取得できます。

Identity-Based Policies

アイデンティティベースのポリシーは、IAM ユーザー、ロール、グループなどのアイデンティティに JSON ドキュメントとしてアタッチできるアクセス許可ポリシーです。これらのポリシーは、アイデンティティが実行できるアクション、リソース、および条件を制御します。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のポリシーは、さらにインラインポリシーまたは管理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、AWS アカウント内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。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。管理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「管理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。

Resource-Based Policies

リソースベースのポリシーは、Amazon S3 バケットなどのリソースにアタッチする JSON ポリシードキュメントです。サービス管理者は、これらのポリシーを使用して、特定のプリンシパル (アカウントメンバー、ユーザー、またはロール) がそのリソースに対して実行する条件およびアクションを定義することができます。リソースベースのポリシーはインラインポリシーです。マネージド型のリソースベースのポリシーはありません。

Other Policy Types

AWS では、別のあまり一般的ではないポリシータイプもサポートしています。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アクセス許可を設定できます。

- 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が IAM エンティティ (IAM ユーザーまたはロール) に付与できるアクセス許可の上限を設定する高度な機能です。エンティティの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アクセス許可は、エンティティの ID ベースのポリシーとそのアクセス許可の境界の共通部分です。`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は、アクセス許可の境界では制限されません。これらのポリシーのいずれかを明示的に拒否した場合、その許可は無効になります。アクセス許可の境界の詳細については、IAM ユーザーガイド の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) – SCP は、AWS Organizations で 組織や組織単位 (OU) に最大権限を指定する JSON ポリシーです。AWS Organizations は、お客様のビジネスが所有する複数の AWS アカウントをグループ化し、一元的に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、サービス制御ポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対するアクセス許可を制限します (各 AWS アカウントのルートユーザー など)。Organizations および SCP の詳細については、AWS Organizations ユーザーガイド の「SCP の動作」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ティッドユーザーの一時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。結果として得られるセッションのアクセス許可は、ユーザーまたはロールの ID ベースのポリシーとセッションポリシーの共通部分です。また、リソースベースのポリシーからアクセス許可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、その許可は無効になります。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。

Multiple Policy Types

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アクセス許可を理解するのがさらに複雑になります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、IAM ユーザーガイド の「ポリシーの評価ロジック」を参照してください。

How Amazon Elastic Container Registry Works with IAM

Amazon ECR へのアクセスを管理するために IAM 使用する前に、Amazon ECR で使用できる IAM 機能を理解しておく必要があります。どのように Amazon ECR およびその他 AWS サービスはIAMと連携します。を参照してください。 AWS 連携するサービス IAM の IAM ユーザーガイド.

トピック

Amazon ECR Identity-Based Policies

IAM アイデンティティベースのポリシーでは、許可または拒否されたアクションとリソースを指定でき、さらにアクションが許可または拒否された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ます。JSONポリシーで使用するすべての要素については、以下を参照してください。 IAM JSON ポリシー要素参照 の IAM ユーザーガイド.

Actions

IAM アイデンティティベースのポリシーの `Action` エレメントは、そのポリシーにより許可または拒否される特定のアクションについて説明します。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。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アクセス許可を付与するポリシーで使用されます。

でのポリシーアクション Amazon ECR アクションの前に次のプレフィックスを使用します。`ecr:`。たとえば、 Amazon ECR リポジトリと Amazon ECR `CreateRepository` API 操作では、`ecr:CreateRepository` 行動に取り組めます。ポリシーステートメントには、`Action` または `NotAction` 要素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説明する独自の一連のアクションを定義します。

単一のステートメントに複数のアクションを指定するには、以下のようにコンマで区切ります。

```
"Action": [
      "ecr:action1",
      "ecr:action2"
```

ワイルドカード (*) を使用して複数のアクションを指定できます。たとえば、`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、以下のアクションを含めます。

```
"Action": "ecr:Describe*"
```

次のリストを表示するには: Amazon ECR アクション、を参照 のアクション、リソース、および条件キー Amazon Elastic Container Registry の IAM ユーザーガイド.

Resources

`Resource` エレメントは、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` エレメントを含める必要があります。ARN を使用して、また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ワイルドカード (*) を使用して、リソースを指定します。

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 (ARN) と AWS サービスの名前空間」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/*"
```

単一のステートメントで複数のリソースを指定するには、ARN 間をカンマで区切ります。

```
"Resource": [
```

```
    "resource1",
    "resource2"
```

次のリストを表示するには: Amazon ECR リソースタイプとそのARNについては、を参照してください。リソースの定義者 Amazon Elastic Container Registry の IAM ユーザーガイド。 各リソースのARNを指定できるアクションについては、以下を参照してください。 によって定義されたアクション Amazon Elastic Container Registry.

Condition Keys

`Condition` エレメント (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`エレメントはオプションです。イコールや以下などの条件演算子を使用する条件式を構築して、リクエスト内に値のあるポリシーの条件に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` エレメントを指定する場合、または 1 つの `Condition` エレメントに複数のキーを指定する場合、AWS が論理 `AND` 演算を使用してそれらを評価します。単一の条件キーに複数の値を指定する場合、AWS が論理 `OR` 演算を使用して条件を評価します。ステートメントのアクセス許可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。たとえば、IAM ユーザー名でタグ付けされている場合のみ、リソースにアクセスする IAM ユーザーアクセス許可を付与できます。詳細については、IAM ユーザーガイド の「IAM ポリシーエレメント: 変数およびタグ」を参照してください。

Amazon ECR は独自の一連の条件キーを定義し、一部のグローバル条件キーの使用をサポートしています。すべてを表示するには AWS グローバル条件キー、参照 AWS グローバル条件コンテキスト キー の IAM ユーザーガイド.

すべての Amazon ECR アクションは、`aws:ResourceTag` および `ecr:ResourceTag` 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 (p. 67)」を参照してください。

次のリストを表示するには: Amazon ECR 条件キー、を参照 条件キーの定義者 Amazon Elastic Container Registry の IAM ユーザーガイド。 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、以下を参照してください。 によって定義されたアクション Amazon Elastic Container Registry.

Examples

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 Identity-Based Policy Examples (p. 64)」を参照してください。

Amazon ECR Resource-Based Policies

リソースベースのポリシーは、指定されたプリンシパルが Amazon ECR リソースに対して実行できるアクションおよび実行条件を指定する 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ています。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、全体のアカウント、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、リソースにアクセスするためのアクセス許可をプリンシパルエンティティに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、アイデンティティベースのポリ

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、以下を参照してください。 方法 IAM リソースベースのポリシーとは異なるロール の IAM ユーザーガイド.

は、 Amazon ECR サービスは、 repository policyは、 repository。 このポリシーは、リポジトリに対してどの主要エンティティ(アカウント、ユーザー、ロール、連合ユーザー)がアクションを実行できるかを定義します。

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 のリポジトリポリシー (p. 19)」を参照してください。

Examples

Amazon ECR リソースベースのポリシーの例については、「Amazon ECR リポジトリポリシーの例 (p. 22)」を参照してください。

Authorization Based on Amazon ECR Tags

タグを Amazon ECR リソースにアタッチするか、Amazon ECR へのリクエストでタグを渡すことができます。タグに基づいてアクセスを制御するには、`ecr:ResourceTag/key-name`、`aws:RequestTag/key-name`、`aws:TagKeys` 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 リポジトリのタグ付け (p. 25)」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の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 (p. 67)」を参照してください。

Amazon ECR IAM Roles

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。

Using Temporary Credentials with Amazon ECR

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、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 または GetFederationToken などの AWS STS API オペレーションを呼び出します。

Amazon ECRでは、一時認証情報の使用をサポートしています。

Service-Linked Roles

サービスにリンクされたロールによって、AWS サービスが他のサービスのリソースにアクセスして自動的にアクションを完了できます。サービスにリンクされたロールは、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、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

Amazon ECR ではサービスにリンクされたロールをサポートしていません。

Amazon ECR 管理ポリシー

Amazon ECR では、IAM ユーザーや EC2 インスタンスにアタッチして、Amazon ECR リソースや API オペレーションで異なる制御レベルを使用できる複数のマネージドポリシーを提供しています。これらのポリシーを直接適用することも、独自のポリシーを作成する開始点として使用することもできます。これらのポリシーに記載された各 API オペレーションの詳細については、Amazon Elastic Container Registry API Reference の「アクション」を参照してください。

トピック

- AmazonEC2ContainerRegistryFullAccess (p. 63)

- AmazonEC2ContainerRegistryPowerUser (p. 63)
- AmazonEC2ContainerRegistryReadOnly (p. 63)

AmazonEC2ContainerRegistryFullAccess

この管理ポリシーは、Amazon ECR の使用を管理するための完全な管理者アクセスを IAM ユーザーまたはロールに 提供することを検討しているお客様にとっての開始ポイントです。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 イベントとしてレポートされます。また、Amazon ECR が AWS CloudTrail と統合されているため、お客様のライフサイクルポリシーイベントは Amazon ECR コンソールに直接表示されます。`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。

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

AmazonEC2ContainerRegistryPowerUser

この管理ポリシーでは、パワーユーザーによる Amazon ECR へのアクセスを許可して、リポジトリへの読み書きアクセスを許可しますが、リポジトリの削除、適用されるポリシードキュメントの変更は許可しません。

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

AmazonEC2ContainerRegistryReadOnly

この管理ポリシーでは、リポジトリとリポジトリ内のイメージの一覧表示や、Docker CLI を使用した Amazon ECR からのイメージのプルなど、Amazon ECR への読み取り専用アクセスを許可します。

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

Amazon Elastic Container Registry Identity-Based Policy Examples

デフォルトでは、IAM 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 マネジメントコンソール や AWS CLI、AWS API を使用してタスクを実行することもできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行するアクセス許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらのアクセス権限が必要な IAM ユーザーまたはグループにそのポリシーをアタッチします。

作成方法を学ぶには IAM JSONポリシードキュメントの例を使用したIDベースのポリシーについては、を参照してください。 [JSON]タブでのポリシーの作成 の IAM ユーザーガイド.

トピック

- Policy Best Practices (p. 64)
- Using the Amazon ECR Console (p. 65)
- Allow Users to View Their Own Permissions (p. 65)
- Accessing One Amazon ECR Repository (p. 66)

Policy Best Practices

アイデンティティベースのポリシーは非常に強力です。アカウント内で、Amazon ECR リソースを作成、アクセス、または削除できるか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 アカウントに追加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のポリシーを作成または編集するときは、以下のガイドラインと推奨事項に従います。

- AWS 管理ポリシーの使用を開始する – Amazon ECR の使用をすばやく開始するには、AWS 管理ポリシーを使用して、従業員に必要なアクセス許可を付与します。これらのポリシーはアカウントですでに有効になっており、AWS によって管理および更新されています。詳細については、IAM ユーザーガイドの「AWS 管理ポリシーを使用したアクセス許可の使用開始」を参照してください 。
- 最小権限を付与する – カスタムポリシーを作成するときは、タスクを実行するために必要なアクセス許可のみを付与します。最小限のアクセス権限から開始し、必要に応じて追加のアクセス権限を付与します。この方法は、寛容なアクセス権限で始め、後でそれらを強化しようとするよりも安全です。詳細については、IAM ユーザーガイド の「最小権限を付与する」を参照してください。
- 機密性の高いオペレーションに MFA を有効にする – 追加セキュリティとして、機密性の高リソースまたは API オペレーションにアクセスするために IAM ユーザーに対して、多要素認証 (MFA) の使用を要

求します。詳細については、IAM ユーザーガイドの「AWS のデバイスに 多要素認証 (MFA) を使用」を参照してください。

- 追加セキュリティに対するポリシー条件を使用する – 実行可能な範囲内で、アイデンティティベースのポリシーがリソースにアクセスできる条件を定義します。たとえば、要求が発生しなければならない許容 IP アドレスの範囲を指定するための条件を記述できます。指定された日付または時間範囲内でのみリクエストを許可する条件を書くことも、SSL や MFA の使用を要求することもできます。ポリシー要素の詳細については、IAM ユーザーガイド の「IAM JSON ポリシー要素: 条件」を参照してください。

Using the Amazon ECR Console

Amazon Elastic Container Registry コンソールにアクセスするには、一連の最小限のアクセス許可が必要です。これらのアクセス許可により、AWS アカウントの Amazon ECR リソースの一覧と詳細を表示できます。最小限必要なアクセス許可よりも制限されたアイデンティティベースのポリシーを作成すると、そのポリシーをアタッチしたエンティティ (IAM ユーザーまたはロール) に対しては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、エンティティに `AmazonEC2ContainerRegistryReadOnly`AWS 管理ポリシーをアタッチします。詳細については、以下を参照してください。 ユーザーへの権限の追加 の IAM ユーザーガイド:

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

AWS CLI または AWS API のみを呼び出すユーザーには、コンソールに対する最小限の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。

Allow Users to View Their Own Permissions

この例では、ユーザー ID にアタッチされたインラインおよび管理ポリシーの表示を IAM ユーザーに許可するポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了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": "*"
        }
    ]
}
```

Accessing One Amazon ECR Repository

この例では、 IAM のユーザーが AWS アカウントへのアクセスを Amazon ECR リポジトリ、 `my-repo`。また、ユーザーが画像をプッシュ、プル、リストできるようにする必要があります。

```
{
   "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 アクションでは、リポジトリ作成時にタグを指定することができます。詳細については、「Amazon ECR リポジトリのタグ付け (p. 25)」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承認を実行してユーザーがタグを作成するアクセス権限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 ポリシーで使用できます。次に例を示します。

次のポリシーでは、IAM 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"
                }
            }
        }
    ]
}
```

次のポリシーでは、IAM ユーザーは、`key=environment,value=prod` としてタグ付けされていない限り、すべてのリポジトリにアクセスすることができます。

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

```
        },
        {
            "Effect": "Deny",
            "Action": "ecr:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/environment": "prod"
                }
            }
        }
    ]
}
```

Troubleshooting Amazon Elastic Container Registry Identity and Access

以下の情報は、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

I Am Not Authorized to Perform an Action in Amazon ECR

AWS マネジメントコンソール から、アクションを実行する権限がないと通知された場合、管理者に問い合わせ、サポートを依頼する必要があります。お客様のユーザー名とパスワードを発行したのが、担当の管理者です。

以下の例のエラーは、`mateojackson` IAM ユーザーがコンソールを使用してリポジトリの詳細を表示しようとしているが、`ecr:DescribeRepositories` アクセス許可がない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:DescribeRepositories on resource: my-repo
```

この場合、Mateo は管理者に依頼し、`ecr:DescribeRepositories` アクションを使用して `my-repo` リソースにアクセスできるようにポリシーを更新してもらいます。

I Am Not Authorized to Perform iam: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、管理者に問い合わせ、サポートを依頼する必要があります。お客様のユーザー名とパスワードを発行したのが、担当の管理者です。Amazon ECR にロールを渡すことができるようにポリシーを更新するよう、管理者に依頼します。

一部の AWS 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せずに、既存のロールをサービスに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アクセス許可が必要です。

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、アクションでは、サービスロールによって付与されたア

クセス許可がサービスにある必要があります。メアリーには、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

この場合、メアリーは担当の管理者に `iam:PassRole` アクションを実行できるようにポリシーの更新を依頼します。

I Want to View My Access Keys

IAM ユーザーアクセスキーを作成した後は、いつでもアクセスキー ID を表示できます。ただし、シークレットアクセスキーをもう一度表示することはできません。シークレットアクセスキーを紛失した場合は、新しいキーペアを作成する必要があります。

アクセスキーは、アクセスキー ID (例: `AKIAIOSFODNN7EXAMPLE`) とシークレットアクセスキー (例: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`) の 2 つの部分から構成されます。ユーザー名とパスワードと同様に、リクエストを認証するために、アクセスキー ID とシークレットアクセスキーの両方を使用する必要があります。ユーザー名とパスワードと同様に、アクセスキーをしっかり管理してください。

> Important
>
> 正規ユーザー ID を確認するためであっても、アクセスキーをサードパーティーに提供しないでください。提供すると、第三者がアカウントへの永続的アクセスを取得する場合があります。

アクセスキーペアを作成する場合、アクセスキー ID とシークレットアクセスキーを安全な場所に保存するように求めるプロンプトが表示されます。このシークレットアクセスキーは、作成時にのみ使用できます。シークレットアクセスキーを紛失した場合、新しいアクセスキーを IAM ユーザーに追加する必要があります。最大 2 つのアクセスキーを持つことができます。すでに 2 つある場合は、新しいキーペアを作成する前に、いずれかを削除する必要があります。手順を表示するには、IAM ユーザーガイド の「アクセスキーの管理 」を参照してください。

I'm an Administrator and Want to Allow Others to Access Amazon ECR

Amazon ECR へのアクセスを他のユーザーに許可するには、アクセスを必要とする人またはアプリケーションの IAM エンティティ (ユーザーまたはロール) を作成する必要があります。ユーザーは、このエンティティの認証情報を使用して AWS にアクセスします。次に、Amazon ECR の適切なアクセス許可を付与するポリシーを、そのエンティティにアタッチする必要があります。

すぐに開始するには、IAM ユーザーガイド の「IAM が委任した最初のユーザーおよびグループの作成」を参照してください 。

I Want to Allow People Outside of My AWS Account to Access My Amazon ECR Resources

他のアカウントのユーザーや組織外のユーザーが、リソースへのアクセスに使用できるロールを作成できます。ロールを引き受けるように信頼されたユーザーを指定することができます。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 でこれらの機能がサポートされるかどうかを確認するには、「How Amazon Elastic Container Registry Works with IAM (p. 59)」を参照してください。

- 所有している AWS アカウント間でリソースへのアクセスを付与する方法については、IAM ユーザーガイド の「所有している別の AWS アカウントへのアクセスを IAM ユーザーに許可」を参照してください。
- サードパーティーの AWS アカウントに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「第三者が所有する AWS アカウントへのアクセス権を付与する」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) へのアクセスの許可」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。

Data protection in Amazon ECR

Amazon Elastic Container Registry (Amazon ECR)は、 AWS 共有責任モデルデータ保護に関する規制とガイドラインを含む。 AWS は、 AWS サービス。 AWS は、顧客コンテンツおよび個人データを処理するためのセキュリティ構成制御を含む、このインフラストラクチャでホストされるデータの管理を維持します。 AWS 顧客と APN パートナーは、データ管理者またはデータ処理者として、 AWS クラウド。

データ保護目的の場合、AWS アカウント認証情報を保護して IAM (AWS Identity and Access Management) で個々のユーザーアカウントをセットアップし、そのユーザーに各自の職務を果たすために必要なアクセス許可のみが付与されるようにすることをお勧めします。また、以下の方法でデータを保護することをお勧めします。

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

顧客の口座番号などの機密性の高い識別情報を、 Name フィールド。これは、コンソール、API、AWS CLI、または AWS で Amazon ECR または他の AWS サービスを使用する場合も同様です。Amazon ECR や他のサービスに入力したすべてのデータは、診断ログに取り込まれる可能性があります。外部サーバーへの URL を指定するときは、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 に含めないでください。

データ保護の詳細については、 AWS 共有責任モデルとGDPR のブログ投稿 AWS Security Blog.

トピック

- Encryption at rest (p. 70)

Encryption at rest

Amazon ECR 画像の保存先 Amazon S3 バケツは Amazon ECR 管理します。デフォルトでは、 Amazon ECR サーバー側の暗号化を Amazon S3-AES-256暗号化アルゴリズムを使用して保存データを暗号化する管理された暗号化キー。これは、お客様側でのアクションが不要で、追加料金なしで提供されます。詳細については、以下を参照してください。 Amazon S3-Managed Encryption Keys (SSE-S3) によるサーバー側暗号化を使用したデータの保護 の Amazon Simple Storage Service 開発者ガイド.

暗号化をより細かく制御するには、 Amazon ECR サーバー側の暗号化は、 AWS Key Management Service (AWS KMS)。使用する時 AWS KMS データを暗号化するには、デフォルトの AWS-マネージド

CMKは、 Amazon ECR、または独自のCMK(顧客管理CMK)を指定します。詳細については、以下を参照してください。 AWS Key Management Service (SSE-KMS) に保存された CMK によるサーバー側暗号化を使用したデータの保護 の Amazon Simple Storage Service 開発者ガイド.

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構成があります。各リポジトリで異なる暗号化構成を使用できます。詳細については、「Creating a Repository (p. 16)」を参照してください。

でリポジトリが作成された場合 AWS KMS 暗号化を有効にすると、CMK 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、 AWS KMS CMKに Amazon ECR 付与者本人としてのレポジトリ。

以下では、 Amazon ECR は、 AWS KMS リポジトリを暗号化および復号化するには:

1. リポジトリを作成する場合、 Amazon ECR は、 説明キー への通話 AWS KMS 暗号化構成で指定されたCMKのAmazonリソース名(ARN)を検証および取得します。
2. Amazon ECR 2回送信 付与の作成 へのリクエスト AWS KMS CMKで権限を作成し、 Amazon ECR データキーを使用してデータを暗号化および復号化します。
3. 画像をプッシュする場合、 データキーを生成 要求は、 AWS KMS 画像レイヤーとマニフェストの暗号化に使用する CMK を指定します。
4. AWS KMS 新しいデータキーを生成し、指定されたCMK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. 画像をプルする場合、 復号化 要求は、 AWS KMS暗号化されたデータ キーを指定します。
6. AWS KMS は、暗号化されたデータキーを復号して、Amazon S3 に復号されたデータキーを送信します。
7. 画像レイヤーを取り出す前に、画像レイヤーを復号化するために使用される のデータ キー。
8. リポジトリが削除されると、 Amazon ECR 2回送信 退職付与 へのリクエスト AWS KMS リポジトリに対して作成された権限をリタイアします。

Considerations

使用に際しては、以下の点に留意すること。 AWS KMS 暗号化 Amazon ECR.

- If you create your Amazon ECR repository with KMS encryption and you do not specify a CMK, Amazon ECR uses an AWS-managed CMK with the alias `aws/ecr` by default. This CMK is created in your account the first time that you create a repository with KMS encryption enabled.
- When you use KMS encryption with your own CMK, the key must exist in the same Region as your repository.
- AWS KMS enforces a limit of 500 grants per CMK. As a result, there is a limit of 500 Amazon ECR repositories that can be encrypted per CMK.
- The grants that Amazon ECR creates on your behalf should not be revoked. If you revoke the grant that gives Amazon ECR permission to use the AWS KMS keys in your account, Amazon ECR cannot access this data, encrypt new images pushed to the repository, or decrypt them when they are pulled. When you revoke a grant for Amazon ECR, the change occurs immediately. To revoke access rights, you should delete the repository rather than revoking the grant. When a repository is deleted, Amazon ECR retires the grants on your behalf.
- There is a cost associated with using AWS KMS keys. For more information, see AWS Key Management Service pricing.

Required IAM permissions

作成時または削除時 Amazon ECR サーバー側の暗号化を使用するリポジトリ AWS KMS必要な許可は、使用している特定のカスタマーマスターキー(CMK)によって異なります。

Required IAM permissions when using the AWS managed CMK for Amazon ECR

デフォルトでは、 AWS KMS 暗号化は、 Amazon ECR CMKは指定されていません。 AWS-管理されたCMK Amazon ECR が使用されています。が AWS-管理されたCMK Amazon ECR リポジトリを暗号化するために使用され、リポジトリを作成する権限を持つすべてのプリンシパルは、 AWS KMS 保存場所の暗号化。ただし、 IAM リポジトリを削除するプリンシパルには、 `kms:RetireGrant` 許可を得ます。これにより、 AWS KMS キー(リポジトリ作成時)

次の例は、 IAM ポリシーをインライン ポリシーとしてユーザーに追加し、暗号化が有効になっているリポジトリを削除するために必要な最小限の権限がユーザーに付与されるようにすることができます。は、AWS KMS リポジトリの暗号化に使用するキーは、リソース パラメーターを使用して指定できます。

```
{
    "Version": "2012-10-17",
    "Id": "ecr-kms-permissions",
    "Statement": [
        {
            "Sid": "Allow access to retire the grants associated with the key",
            "Effect": "Allow",
            "Action": [
                "kms:RetireGrant"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Required IAM permissions when using a customer-managed CMK

でリポジトリを作成する場合 AWS KMS 顧客が管理するCMKを使用して暗号化を有効にした場合、CMK キーポリシーと IAM リポジトリを作成するユーザーまたは役割のポリシー。

独自のCMKを作成する場合、デフォルトのキーポリシーを使用できます。 AWS KMS 独自の を作成するか、独自の を指定できます。顧客管理のCMKがアカウント所有者によって管理可能な状態を維持するため、CMKの主要ポリシーでは、 AWS KMS アカウントのrootユーザーのアクション。適用範囲の権限は、キーポリシーに追加できますが、最低でも、CMKを管理するための権限をrootユーザーに付与する必要があります。Amazon ECR から実行されるリクエストにのみ CMK が使用されるようにするには、値 `ecr.<region>.amazonaws.com` で kms:ViaService 条件キーを使用できます。

次のキーポリシーの例は、 AWS CMKへのフルアクセスを所有するアカウント(rootユーザー)。このキーポリシーの例の詳細については、以下を参照してください。 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 の AWS Key Management Service Developer Guide.

```
{
    "Version": "2012-10-17",
    "Id": "ecr-key-policy",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

は、 IAM ユーザー、 IAM 役割、または AWS リポジトリを作成するアカウントには、 `kms:CreateGrant`、 `kms:RetireGrant`、および `kms:DescribeKey` 必要な許可に加えて、 Amazon ECR 許可。

> Note
>
> は、 `kms:RetireGrant` 権限を IAM リポジトリを作成するユーザーまたは役割のポリシー。は、 `kms:CreateGrant` および `kms:DescribeKey` パーミッションは、CMK または IAM リポジトリを作成するユーザーまたは役割のポリシー。詳細については、 AWS KMS 許可作業、参照 AWS KMS API権限: アクションとリソース参照 の AWS Key Management Service Developer Guide.

次の例は、 IAM ポリシーをインライン ポリシーとしてユーザーに追加し、暗号化を有効にしてリポジトリを作成し、リポジトリが終了したら削除するために必要な最小限の権限がユーザーにあることを確認します。は、 AWS KMS リポジトリの暗号化に使用するキーは、リソース パラメーターを使用して指定できます。

```
{
    "Version": "2012-10-17",
    "Id": "ecr-kms-permissions",
    "Statement": [
        {
            "Sid": "Allow access to create and retire the grants associated with the key as
 well as describe the key",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:RetireGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
        }
    ]
}
```

Allow a user to list CMKs in the console when creating a repository

を使用するとき Amazon ECR コンソールを使用してリポジトリを作成すると、ユーザーが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が管理する CMK をリストできるように許可できます。以下 IAM ポリシーの例は、コンソールの使用時にCMKとエイリアスを一覧表示するために必要な権限を示しています。

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

Monitoring Amazon ECR interaction with AWS KMS

以下を使用できます。 AWS CloudTrail リクエストを追跡し、 Amazon ECR 送信先 AWS KMS を代理します。のログ エントリ CloudTrail ログには、より簡単に識別できるように、暗号化キーが含まれています。

Amazon ECR encryption context

A encryption context は、任意の非秘密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組み込むと、AWS KMS は暗号化コンテキストを暗号化されたデータに暗号化してバインドします。データを復号化するには、同じ暗号化コンテキストに渡す必要があります。

その データキーを生成 および 復号化 へのリクエスト AWS KMS、 Amazon ECR 2つの名前を持つ暗号化コンテキストを使用–リポジトリを識別する値ペアと Amazon S3 バケットが使用されています。これを次の例で示します: 名前は変化しませんが、組み合わせた暗号化コンテキスト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、Amazon CloudWatch Logs、AWS CloudTrail 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確認できます。また、ポリシーと許可で認可の条件としても確認できます。

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 – The first name–value pair identifies the bucket. The key is `aws:s3:arn`. The value is the Amazon Resource Name (ARN) of the Amazon S3 bucket.

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

  For example, if the ARN of the bucket is `arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`, the encryption context would include the following pair.

  ```
  "arn:aws:s3:::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
  ```

- aws:ecr:arn – The second name–value pair identifies the Amazon Resource Name (ARN) of the repository. The key is `aws:ecr:arn`. The value is the ARN of the repository.

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

  For example, if the ARN of the repository is `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`, the encryption context would include the following pair.

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

Troubleshooting

_を削除する場合 Amazon ECR リポジトリが正常に削除され、 Amazon ECR リポジトリのCMKに追加された権限を失効できません。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、 AWS KMS 自分自身がリポジトリに付与する権限です。

退職 AWS KMS リポジトリへの手動権限

1. の権限を一覧表示します。 AWS KMS リポジトリに使用されるキー。は、 `key-id` 値は、コンソールから受信したエラーに含まれます。また、 `list-keys` アカウントの特定の地域のAWS管理CMKと顧客管理CMKの両方を一覧表示するコマンドです。

```
aws kms list-grants \
     --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
     --region us-west-2
```

   出力には、 `EncryptionContextSubset` リポジトリの Amazon リソース名 (ARN) を使用します。これは、キーに追加された権限が、リタイアする権限かどうかを判断するのに使用できます。は、 `GrantId` 値は、次のステップで付与を終了するときに使用されます。
2. 各交付金を AWS KMS キーがリポジトリに追加されました。次の値を置換します: *`GrantId`* 前のステップの出力からの付与のID。

```
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

サードパーティーの監査者は、複数の AWS コンプライアンスプログラムの一環として Amazon Elastic Container Registry のセキュリティとコンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「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。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。

サードパーティーの監査レポートをダウンロードするには、AWS Artifact を使用します。詳細については、「AWS Artifact でレポートをダウンロードする」を参照してください。

Amazon ECR サービスを使用する際のお客様のコンプライアンス責任は、データの機密性、企業のコンプライアンス目的、適用法規や規則によって決まります。AWS ではコンプライアンスに役立つ以下のリソースを用意しています。

- セキュリティおよび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を AWS でデプロイするための手順を説明します。
- HIPAA のセキュリティとコンプライアンスに関するホワイトペーパーを作成する – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスのリソース – このワークブックおよびガイドのコレクションは、お客様の業界や場所に適用される場合があります。
- AWS Config 開発者ガイドの「ルールでのリソースの評価」– AWS Config サービスでは、リソース設定が社内のプラクティス、業界のガイドライン、規制にどの程度適合しているかを評価します。
- AWS Security Hub – この AWS サービスでは、AWS 内のセキュリティ状態を包括的に表示しており、セキュリティ業界の標準およびベストプラクティスへの準拠を確認するのに役立ちます。

Amazon Elastic Container Registry のインフラストラクチャセキュリティ

マネージド型サービスとして、Amazon Elastic Container Registry は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティの手順で保護されています。

AWS が公開した 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。クライアントで Transport Layer Security (TLS) 1.0 以降がサポートされている必要があります。TLS 1.2 以降が推奨されています。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、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の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ます。これには送信元 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 からのアクセスを制御することもできます。これにより、実質的に AWS ネットワーク内の特定の VPC から特定の Amazon ECR リソースへのネットワークアクセスのみが分離されます。詳細については、「Amazon ECR interface VPC endpoints (AWS PrivateLink) (p. 76)」を参照してください。

Amazon ECR interface VPC endpoints (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 プライベートアクセスを可能にするテクノロジー、PrivateLink Amazon ECR プライベート IP アドレス経由の API。 AWSPrivateLink は、VPC および Amazon ECR 間のすべてのネットワークトラフィックを Amazon ネットワークに限定します。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。

詳細については、以下を参照してください。 AWS PrivateLinkおよびVPCエンドポイントについては、を参照してください。 VPCエンドポイント の Amazon VPC ユーザーガイド.

Considerations for Amazon ECR VPC endpoints

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。

- To allow your Amazon ECS tasks that use the EC2 launch type to pull private images from Amazon ECR, ensure that you also create the interface VPC endpoints for Amazon ECS. For more information, see Interface VPC Endpoints (AWS PrivateLink) in the Amazon Elastic Container Service Developer Guide.

  Important

  Amazon ECS tasks that use the Fargate launch type don't require the Amazon ECS interface VPC endpoints.
- Amazon ECS tasks using the Fargate launch type and platform version 1.3.0 or earlier only require the com.amazonaws.*region*.ecr.dkr Amazon ECR VPC endpoint and the Amazon S3 gateway endpoint to take advantage of this feature.

- Amazon ECS tasks using the Fargate launch type and platform version 1.4.0 or later require both the com.amazonaws.*region*.ecr.dkr and com.amazonaws.*region*.ecr.api Amazon ECR VPC endpoints as well as the Amazon S3 gateway endpoint to take advantage of this feature.
- Amazon ECS tasks using the Fargate launch type that pull container images from Amazon ECR can restrict access to the specific VPC their tasks use and to the VPC endpoint the service uses by adding condition keys to the task execution IAM role for the task. For more information, see Optional IAM Permissions for Fargate Tasks Pulling Amazon ECR Images over Interface Endpoints in the Amazon Elastic Container Service Developer Guide.
- Amazon ECS tasks using the Fargate launch type that pull container images from Amazon ECR that also use the `awslogs` log driver to send log information to CloudWatch Logs require the CloudWatch Logs VPC endpoint. For more information, see Create the CloudWatch Logs endpoint (p. 80).
- The security group attached to the VPC endpoint must allow incoming connections on port 443 from the private subnet of the VPC.
- VPC endpoints currently don't support cross-Region requests. Ensure that you create your VPC endpoints in the same Region where you plan to issue your API calls to Amazon ECR.
- VPC endpoints only support Amazon provided DNS through Amazon Route 53. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see DHCP Options Sets in the Amazon VPC ユーザーガイド.
- If your containers have existing connections to Amazon S3, their connections might be briefly interrupted when you add the Amazon S3 gateway endpoint. If you want to avoid this interruption, create a new VPC that uses the Amazon S3 gateway endpoint and then migrate your Amazon ECS cluster and its containers into the new VPC.

Considerations for Windows images

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 これらのアーティファクトを含むレイヤーは、 foreign layers。 アーティファクトが Microsoft によって提供されると、Microsoft Azure インフラストラクチャから外部レイヤーが取得されます。このため、コンテナが Azure からこれらの外部レイヤーをプルできるようにするには、VPC エンドポイントの作成以外に追加のステップが必要です。

Windowsイメージを Amazon ECR 使用するには `--allow-nondistributable-artifacts` というフラグが表示されます。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 これらの画像を Amazon ECR Azureへの追加アクセスが不要なVPCエンドポイント経由。

> **Important**
>
> の使用 `--allow-nondistributable-artifacts` フラグは、Windowsコンテナベースイメージライセンスの条件を遵守する義務を排除するものではありません。公開または第三者の再配布のためにWindowsコンテンツを投稿することはできません。自分の環境内での使用が許可されています。

Dockerの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デーモン構成ファイルを変更する必要があります。このファイルは、Dockerのインストールに応じて、通常、 ドッカーエンジン セクションを編集するか、 `C:\ProgramData\docker\config\daemon.json` ファイルを直接保存します。

以下は、必要な構成の例です。値を、イメージをプッシュするリポジトリ URI に置き換えます。

```
{
    "allow-nondistributable-artifacts": [
        "111122223333.dkr.ecr.us-west-2.amazonaws.com"
    ]
}
```

Dockerデーモン構成ファイルを変更した後、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、プッシュが動作したことを確認します。

Note

Windowsイメージのベースレイヤーが大きくなります。レイヤーサイズにより、プッシュ時間が長くなり、ストレージコストが増加します。 Amazon ECR これらの画像用。このような理由から、ビルド時間と継続的なストレージ コストの削減が厳密に必要な場合のみ、このオプションを使用することをお勧めします。たとえば、 `mcr.microsoft.com/windows/servercore` 圧縮した場合、イメージのサイズは約1.7 GiBです。 Amazon ECR.

Create the VPC endpoints for Amazon ECR

のVPCエンドポイントを作成するには Amazon ECR サービス、 インタフェース エンドポイントの作成 手順 Amazon VPC ユーザーガイド.

EC2 起動タイプを使用する Amazon ECS タスクには、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。

Amazon ECS タスク、 Fargate 起動タイプおよびプラットフォームバージョン1.3.0以前では、com.amazonaws。***region***.ecr.dkr Amazon ECR VPCエンドポイントおよび Amazon S3 ゲートウェイエンドポイント。

Amazon ECS タスク、 Fargate 起動タイプとプラットフォームバージョン1.4.0以降には、com.amazonaws。***region***.ecr.dkr および com.amazonaws。***region***.ecr.api(エクリュ.api) Amazon ECR VPCエンドポイントおよび Amazon S3 ゲートウェイエンドポイント。

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。

_を作成するとき com.amazonaws。***region***.ecr.dkr 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、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(エクリュ.api)

Note

指定された *`region`* は、 AWS によってサポートされる地域 Amazon ECRなど `us-east-2` の 米国東部 (オハイオ) リージョン.

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 CreateRepositories などの API アクションは、このエンドポイントに送られます。

が com.amazonaws。***region***.ecr.api(エクリュ.api) エンドポイントが作成されると、プライベート DNSホスト名を有効にするオプションがあります。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 エンドポイントでプライベート DNS ホスト名を有効にする場合は、SDK または AWS CLI を使用する際にエンドポイント URL を指定しなくてもいいように、SDK または AWS CLI を最新バージョンに更新します。

プライベート DNS ホスト名が有効で SDK または AWS CLI の 2019 年 1 月 24 日以前にリリースされた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

の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
```

Create the Amazon S3 gateway endpoint

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 で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。コンテナが Amazon ECR からイメージをダウンロードするときは、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。 以下に示しているのは、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) です。

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

_を使用 ゲートウェイ エンドポイントの作成 手順 Amazon VPC ユーザーガイド 以下を作成します。 Amazon S3 ゲートウェイ エンドポイント Amazon ECR. 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。 フルアクセスポリシーを選択することもできます。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はこのポリシーに優越して適用されるためです。Amazon S3 バケットへのアクセスを 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、「Minimum Amazon S3 Bucket Permissions for Amazon ECR (p. 79)」を参照してください。

Minimum Amazon S3 Bucket Permissions for Amazon ECR

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 で必要な最小限の 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 が必要とする Amazon S3 バケットポリシーの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。Amazon ECR によってサポートされている AWS リージョンのリージョン ID (例: 米国東部 (オハイオ) リージョン の場合は `us-east-2`) を表します。

Example

以下の例では、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/*"]
    }
  ]
}
```

Create the CloudWatch Logs endpoint

Amazon ECS タスク、 Fargate 起動タイプは、VPCを使用しており、インターネット ゲートウェイも使用していません。 `awslogs` ログ情報を送信するログ ドライバ CloudWatch Logs は、com.amazonaws。***region***.logs(ログ) インターフェースVPCエンドポイント CloudWatch Logs. 詳細については、以下を参照してください。 ゲートウェイ エンドポイントの作成 の Amazon CloudWatch Logs User Guide.

Create an endpoint policy for your Amazon ECR VPC endpoints

VPC エンドポイントポリシーは、エンドポイントを作成または変更するときにエンドポイントにアタッチする IAM 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、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がアタッチされます。エンドポイントポリシーは、IAM 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、エンドポイント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。エンドポイントのポリシーは、JSON 形式で記述される必要があります。詳細については、Amazon VPC ユーザーガイドの「VPC エンドポイントによるサービスのアクセスコントロール」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 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"
  ],
  "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"
   ],
   "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、[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、「Create the VPC endpoints for Amazon ECR (p. 78)」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。

Amazon ECR Monitoring

Amazon CloudWatch を使用して Amazon ECR API の使用状況をモニタリングすることで、Amazon ECR から未加工データを収集し、リアルタイムに近い読み取り可能なメトリクスに加工できます。これらの統計情報は 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 分間で CloudWatch に自動的に送信されます。CloudWatch の詳細については、Amazon CloudWatch ユーザーガイドを参照してください。

Amazon ECR には、認証、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うえで重要な要素です。マルチポイント障害が発生した場合は、デバッグしやすくなるように、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。

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

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、お客様の環境における Amazon ECR の正常な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

Visualizing Your Service Quotas and Setting Alarms

CloudWatch コンソールを使用して、サービスクォータを可視化し、サービスクォータと照らして現在の使用状況を確認できます。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/ にある 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。以下 より、入力 **`80`**。 これにより、クォータの 80% を超えるとアラーム状態になるアラームが生成されます。
   c. [次] を選択します。
   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。続いて、[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. [アラームの作成] を選択します。

Amazon ECR Usage Metrics

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の使用状況を把握できます。これらのメトリクスを使用して、CloudWatch グラフやダッシュボードで現在のサービスの使用状況を可視化できます。

Amazon ECR 使用状況メトリクスは、AWS の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 のサービスクォータの詳細については、「Amazon ECR service quotas (p. 95)」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。

以下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 のサービスの名前。Amazon ECR 使用量メトリクスの場合、このディメンションの値は `ECR` です。
`Type`	レポートされるエンティティのタイプ。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`	実行中のリソースのタイプ。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 Usage Reports

AWS は、Amazon ECR リソースのコストおよび使用量を分析できる、Cost Explorer と呼ばれる無料のレポートツールを提供します。

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 (p. 26)」を参照してください。

作成についての詳細は、 AWS コストと使用レポート、を参照 AWS コストと使用レポート の AWS Billing and Cost Management ユーザーガイド.

Amazon ECR Events and EventBridge

Amazon EventBridge を使用すると、AWS のサービスを自動化して、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、ほぼリアルタイムに EventBridge に提供されます。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。

- Adding events to log groups in CloudWatch Logs

- Invoking an AWS Lambda function
- Invoking Amazon EC2 Run Command
- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine
- Notifying an Amazon SNS topic or an AWS SMS queue

詳細については、Amazon EventBridge の「Amazon EventBridge ユーザーガイド の使用開始」を参照してください。

Sample Events from Amazon ECR

以下に、Amazon ECR からのイベントの例を示します。

Event for a Completed Image Push

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、「Pushing an image (p. 29)」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-repo",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "PUSH",
        "image-tag": "latest"
    }
}
```

Event for a Completed Image Scan

各イメージスキャンが完了すると、次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、「Image scanning (p. 46)」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-repo"
    ],
    "detail": {
        "scan-status": "COMPLETE",
        "repository-name": "my-repo",
```

```
        "finding-severity-counts": {
        "CRITICAL": 10,
        "MEDIUM”: 9
      },
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "image-tags": []
    }
}
```

Event for an Image Deletion

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、「Deleting an image (p. 32)」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-repo",
        "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
        "action-type": "DELETE",
        "image-tag": "latest"
    }
}
```

Logging Amazon ECR Actions with AWS CloudTrail

Amazon ECR は AWS CloudTrail と統合されています。このサービスは、Amazon ECR のユーザー、ロール、または AWS サービスによって実行されたアクションを記録するサービスです。CloudTrail は、次の Amazon ECR のアクションをイベントとしてキャプチャします。

- All API calls, including calls from the Amazon ECR console
- All actions taken due to the encryption settings on your repositories
- All actions taken due to lifecycle policy rules, including both successful and unsuccessful actions

証跡を作成する場合は、Amazon ECR のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [Event history (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 に対して実行されたリクエスト、リクエスト元の IP アドレス、リクエストの実行者、実行日時などの詳細を確認できます。

詳細については、AWS CloudTrail User Guideを参照してください。

Amazon ECR Information in CloudTrail

CloudTrail は、アカウント作成時に AWS 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 AWS の他のサービスのイベントと共に CloudTrail イベントとして

[イベント履歴] に記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」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。証跡では、AWS パーティションのイベントがログに記録され、指定した Amazon S3 バケットにログファイルが配信されます。さらに、CloudTrail ログで収集されたデータの分析とそのデータに基づいたアクションのために、その他の AWS のサービスを設定できます。詳細については、「」を参照してください。

- Creating a trail for your AWS account
- AWS Service Integrations With CloudTrail Logs
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon ECR API アクションはすべて CloudTrail によって記録されます。また、これらのアクションは Amazon Elastic Container Registry API Reference で説明されています。一般的なタスクを実行すると、そのタスクに含まれる各 API アクションのセクションが CloudTrail ログファイルに生成されます。たとえば、リポジトリを作成すると、`GetAuthorizationToken`、`CreateRepository`、`SetRepositoryPolicy` のセクションが 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 Log Entry Examples (p. 87)」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、リクエストの生成者に関する情報が含まれます。この ID 情報は以下のことを確認するのに役立ちます。

- Whether the request was made with root or IAM user credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

詳細については、「CloudTrail `userIdentity` 要素」を参照してください。

Understanding Amazon ECR Log File Entries

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 1 つ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、リクエストのパラメータなどに関する情報が含まれます。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 Log Entry Examples

いくつかの一般的な Amazon ECR タスクの CloudTrail ログエントリの例を次に示します。

> 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記録されます。

トピック

Example: Create Repository Action

以下の例では、`CreateRepository` アクションを表す 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"
```

```
}
```

Example: AWS KMS CreateGrant API action when creating an Amazon ECR repository

次の例は、 CloudTrail ログエントリは、 AWS KMS CreateGrantアクションは、 Amazon ECR KMS暗号化を有効にしたリポジトリ。KMS 暗号化を使用して作成された各リポジトリに対して、 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"
}
```

Example: Image Push Action

以下の例は、PutImage アクションを使用するイメージのプッシュの 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、InitiateLayerUpload、UploadLayerPart、および CompleteLayerUpload の参照も CloudTrail ログに表示されます。

```
{
    "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"
}
```


Example: Image Pull Action

以下の例は、`BatchGetImage` アクションを使用するイメージのプルの 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、すでにイメージがローカルにない場合、`GetDownloadUrlForLayer` の参照も CloudTrail ログに表示されます。


```
{
    "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"
}
```

Example: Image Lifecycle Policy Action

次の例は、ライフサイクルポリシールールによってイメージの期限が切れたことを示す 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。

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

Amazon ECR service quotas

Amazon Elastic Container Registry (Amazon ECR) のデフォルトのクォータを以下の表に示します。

サービスクォータ	説明:	デフォルトのクォータ値
登録済みリポジトリ	リージョンごとに作成できるリポジトリの最大数。	10,000*
リポジトリあたりのイメージ	リポジトリあたりのイメージの最大数。	10,000*

以下の表に示しているのは、イメージプッシュおよびイメージプルアクションに関連する Amazon ECR API アクションそれぞれのデフォルトのレートクォータです。

Amazon ECR アクション	API オペレーション	説明:	デフォルトのクォータ値
認証	GetAuthorizationToken リクエストのレート	現在のリージョンで 1 秒あたりに実行できる GetAuthorizationToken API リクエストのレート。	200 USD
イメージプッシュ	BatchCheckLayerAvailability リクエストのレート	現在のリージョンで 1 秒あたりに実行できる BatchCheckLayerAvailability API リクエストのレート。 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。	200 USD
	InitiateLayerUpload リクエストのレート	現在のリージョンで 1 秒あたりに実行できる InitiateLayerUpload API リクエストのレート。 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	10*

Amazon ECR アクション	API オペレーション	説明:	デフォルトのクォータ値
		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。	
	CompleteLayerUpload リクエストのレート	現在のリージョンで 1 秒あたりに実行できる CompleteLayerUpload API リクエストのレート。 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。	10*
	UploadLayerPart リクエストのレート	現在のリージョンで 1 秒あたりに実行できる UploadLayerPart API リクエストのレート。 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ます。各イメージレイヤーパートの最大サイズは 20,971,520 バイト (約 20 MB) です。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。	260 。
	PutImage リクエストのレート	現在のリージョンで 1 秒あたりに実行できる PutImage API リクエストのレート。 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。	10*

Amazon ECR アクション	API オペレーション	説明:	デフォルトのクォータ値
イメージプル	BatchGetImage リクエストのレート	現在のリージョンで 1 秒あたりに実行できる BatchGetImage API リクエストのレート。 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。	1,000。
	GetDownloadUrlForLayer リクエストのレート	現在のリージョンで 1 秒あたりに実行できる GetDownloadUrlForLayer API リクエストのレート。 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。	1,500

以下の表は、変更できない Amazon ECR および Docker イメージの他のクォータを示しています。

Note

以下の表に示されているレイヤーパート情報は、Amazon ECR API アクションを直接呼び出し、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。このアクションはまれにしか発生しません。Docker CLI を使用してイメージをプル、タグ付け、プッシュすることをお勧めします。

サービスクォータ	説明:	クォータ値
レイヤーパート	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。	1,000。
レイヤーの最大サイズ	レイヤーの最大サイズ (MiB)。**	10,000*
レイヤーパートの最小サイズ	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。	5*

サービスクォータ	説明:	クォータ値
レイヤーパートの最大サイズ	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、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。	10*
イメージあたりのタグ	イメージあたりのタグの最大数。	1000*
ライフサイクルポリシーの長さ	ライフサイクルポリシーの最大文字数。	30,720
ライフサイクルポリシーあたりのルール	ライフサイクルポリシーのルールの最大数。	(50)
イメージスキャンのレート	1 日あたり、イメージあたりのイメージスキャンの最大数。	1

** ここに示されている最大レイヤーサイズは、レイヤーパートの最大サイズ (10 MiB) をレイヤーパートの最大数 (1,000) で乗算することにより計算されます。

Managing your Amazon ECR service quotas in the AWS マネジメントコンソール

Amazon ECR 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 のサービスである サービスクォータ と統合されています。詳細については、以下を参照してください。 サービス・クォータとは? の サービスクォータ ユーザーガイド.

サービスクォータ 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 サービスクォータを表示するには (AWS マネジメントコンソール)

1. サービスクォータ コンソール (https://console.aws.amazon.com/servicequotas/) 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 サービス] リストから、[Amazon Elastic Container Registry (Amazon ECR)] を検索して選択します。

   [Service quotas (サービスクォータ)] 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、サービスクォータ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。

クォータの増加を要求するには、以下を参照してください。 割当増額の要求 の サービスクォータ ユーザーガイド.

Creating a CloudWatch alarm to monitor API usage metrics

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 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。サービスクォータコンソールでは、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 Usage Metrics (p. 83)」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 マネジメントコンソール)

1. サービスクォータ コンソール (https://console.aws.amazon.com/servicequotas/) 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 サービス] リストから、[Amazon Elastic Container Registry (Amazon ECR)] を検索して選択します。
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 Troubleshooting

この章では、Amazon Elastic Container Registry (Amazon ECR) の診断情報と、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ステップを示します。

トピック

Enabling Docker Debug Output

Docker 関連の問題をデバッグを開始するには、最初にホストインスタンスで実行している Docker デーモンで Docker デバッグ出力を有効にします。Dockerデバッグの有効化についての詳細は、 Amazon ECR オン Amazon ECS コンテナインスタンス、「コンテナインスタンス」を参照してください。 Dockerデバッグ出力の有効化 の Amazon Elastic Container Service Developer Guide.

Enabling AWS CloudTrail

Amazon ECR によって返されるエラーに関する追加情報は、AWS CloudTrail を有効にすることで検出できます。これは、AWS アカウントの AWS コールを記録するサービスです。CloudTrail は、Amazon S3 バケットにログファイルを配信します。CloudTrail が収集した情報を使用して、成功した AWS サービスリクエスト、そのリクエストの送信者、そのリクエストの送信時間などを確認できます。CloudTrail の詳細 (有効にする方法、ログファイルを検索する方法を含む) については、AWS CloudTrail User Guide を参照してください。CloudTrail と Amazon ECR の使用の詳細については、「Logging Amazon ECR Actions with AWS CloudTrail (p. 86)」を参照してください。

Optimizing Performance for Amazon ECR

次のセクションでは、Amazon ECR の使用時にパフォーマンスを最適化するために使用できる設定と戦略の推奨事項を示します。

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 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 パラメーターを使用して AWS CLI コマンドに渡すか、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できます。また、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。

Troubleshooting Errors with Docker Commands When Using Amazon ECR

トピック

- HTTP 403 Errors or "no basic auth credentials" Error When Pushing to Repository (p. 103)

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。

Error: "Filesystem Verification Failed" or "404: Image Not Found" When Pulling an Image From an Amazon ECR Repository

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はエラー `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。

Error: "Filesystem Layer Verification Failed" When Pulling Images from Amazon ECR

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。

HTTP 403 Errors or "no basic auth credentials" Error When Pushing to Repository

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) 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) 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。

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 のリポジトリポリシー (p. 19)」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 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。このバグを回避するには、 `https://` 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のスキーム aws ecr get-login。 HTTPS スキームを使用しない Docker ログインコマンドの例を以下に示します。

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Troubleshooting Amazon ECR Error Messages

Amazon ECS コンソールまたは AWS CLI を通じてトリガーされた API コールがエラーメッセージで終了することがあります。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。

Error: "Error Response from Daemon: Invalid Registry Endpoint" When Running aws ecr get-login

aws ecr get-login コマンドを実行して Amazon ECR リポジトリのログイン認証情報を取得するときに、次のエラーが表示される場合があります。

```
Error response from daemon: invalid registry endpoint
    https://xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/v0/: unable to ping registry
 endpoint
    https://xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/v0/
v2 ping attempt failed with error: Get https://xxxxxxxxxxxx.dkr.ecr.us-
east-1.amazonaws.com/v2/:
    dial tcp: lookup xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com on 172.20.10.1:53:
    read udp 172.20.10.1:53: i/o timeout
```

このエラーは、Docker Toolbox、Docker for Windows、または Docker for Mac を実行している MacOS X および Windows システムで発生することがあります。これは、他のアプリケーションがローカルゲートウェイ (192.168.0.1) を介してルートを変更するときに、よく発生します。仮想マシンはこのゲートウェイを介して Amazon ECR サービスにアクセスする必要があります。Docker Toolbox の使用中にこのエラーが発生した場合は、Docker Machine 環境またはローカルクライアントのオペレーティングシステムを再起動することで、しばしば解決できます。これによって問題が解決しない場合は、docker-machine ssh コマンドを使用してコンテナインスタンスにログインします。外部ホストで DNS ルックアップを行って、ローカルホストと同じ結果となることを確認します。結果が異なる場合は、Docker Toolbox のドキュメントを参照して、Docker Machine 環境が適切に設定されていることを確認します。

HTTP 429: Too Many Requests or ThrottleException

1 つ以上の Amazon ECR コマンドまたは API コールから、`429: Too Many Requests` エラーまたは `ThrottleException` エラーを受け取ることがあります。Dockerツールを Amazon ECR、Dockerバージョン1.12.0以降では、 `TOOMANYREQUESTS: Rate exceeded`。 1.12.0未満のバージョンのDockerでは、 `Unknown: Rate exceeded`.

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 のさまざまな API オペレーションでは、異なるスロットリングが行われます。

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。

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、アマゾン ウェブ サービス全般のリファレンスの「AWS でのエラーの再試行とエクスポネンシャルバックオフ」を参照してください。

HTTP 403: "User [arn] is not authorized to perform [operation]"

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
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 のアクセス許可の作成と検証の詳細については、「Identity and Access Management for Amazon Elastic Container Registry (p. 55)」を参照してください。

HTTP 404: "Repository Does Not Exist" Error

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 Repositories (p. 16)」を参照してください。

Troubleshooting Image Scanning Issues

以下は、一般的なイメージスキャンの失敗です。このようなエラーを Amazon ECR コンソールで表示するには、イメージの詳細を確認するか、`DescribeImageScanFindings` API を使用して API またはAWS CLI を通して行うことができます。

UnsupportedImageError

皆さんは、 `UnsupportedImageError` オペレーティングシステムを使用して作成された画像のスキャンを試行する際にエラーが発生しました。 Amazon ECR は画像スキャンをサポートしていません。 Amazon ECR のメジャーバージョンのパッケージ脆弱性スキャンをサポート Amazon Linux、Amazon Linux 2、Debian、Ubuntu、CentOS、Oracle Linux、Alpine、およびRHEL Linuxディストリビューション。流通がベンダーからの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い場合があります。 Amazon ECR は、 ドッカーの傷 画像。

`UNDEFINED` 深刻度が返される

重大度レベルが `UNDEFINED`。 これの一般的な原因は次のとおりです。

- The vulnerability was not assigned a priority by the CVE source.
- The vulnerability was assigned a priority that Amazon ECR did not recognize.

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。

Document History

次の表に、Amazon ECR の前回のリリース以後に行われた、文書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
保管時の暗号化	Amazon ECR サーバー側の暗号化を使用してリポジトリの暗号化を構成するためのサポートを追加。 AWS Key Management Service (AWS KMS)。 詳細については、「Encryption at rest (p. 70)」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のサポートが追加されました。 詳細については、「Pushing a multi-architecture image (p. 30)」を参照してください。	2020 年 4 月 28 日
Amazon ECR 使用状況に関するメトリクス	Amazon ECR で、アカウントのリソースの使用状況を可視化する CloudWatch 使用量メトリクスが追加されました。また、CloudWatch と サービスクォータ コンソールの両方から CloudWatch アラームを作成でき、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られるようになりました。 詳細については、「Amazon ECR Usage Metrics (p. 83)」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られるようになりました。 詳細については、「Amazon ECR service quotas (p. 95)」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、「認証トークンの使用 (p. 13)」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およびエクスポージャー (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、「Image scanning (p. 46)」を参照してください。	2019 年 10 月 24 日

変更	説明:	日
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、「Create an endpoint policy for your Amazon ECR VPC endpoints (p. 80)」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、「Image tag mutability (p. 45)」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)	AWS PrivateLink を使用したインターフェイス VPC エンドポイントの設定のサポートが追加されました。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 とをプライベートに接続できます。 詳細については、「Amazon ECR interface VPC endpoints (AWS PrivateLink) (p. 76)」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	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 Amazon ECR に追加されました。 詳細については、「Amazon ECR リポジトリのタグ付け (p. 25)」を参照してください。	2018 年 12 月 18 日
Amazon ECR の名前変更	Amazon Elastic Container Registry の名前が変更されました (旧 Amazon EC2 コンテナレジストリ)。	2017 年 11 月 21 日
ライフサイクルポリシー	Amazon ECR ライフサイクルポリシーで、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 Lifecycle Policies (p. 34)」を参照してください。	2017 年 10 月 11 日
Amazon ECR が Docker Image Manifest 2、Schema 2 をサポート	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、「Container image manifest formats (p. 50)」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、安全性と信頼性に優れたスケーラブルなマネージド AWS Docker レジストリサービスです。	2015 年 12 月 21 日

AWS の用語集

最新の AWS の用語については、『AWS General Reference』の「AWS の用語集」を参照してください。

英語の翻訳が提供されている場合で、内容が矛盾する場合には、英語版がオリジナルとして取り扱われます。翻訳は機械翻訳により提供されています。