aws

ユーザーガイド

Application Auto Scaling

Application Auto Scaling: ユーザーガイド

Table of Contents

Application Auto Scaling とは

Application Auto Scaling は、Amazon EC2 Auto Scaling 以外の個々の AWS サービス用にスケーラブルなリソースを自動的にスケーリングするソリューションを必要とする開発者やシステム管理者向けのウェブサービスです。Application Auto Scaling では、次のリソースの自動スケーリングを設定できます。

- AppStream 2.0 フリート
- Aurora レプリカ
- Amazon Comprehend ドキュメントの分類とエンティティ認識のエンドポイント
- DynamoDB テーブルとグローバルセカンダリインデックス
- Amazon ECS サービス
- ElastiCache レプリケーショングループ (Redis OSS および Valkey) と Memcached クラスター
- Amazon EMR クラスター
- Amazon Keyspaces (Apache Cassandra 用) テーブル
- Lambda 関数のプロビジョニングされた同時実行数
- Amazon Managed Streaming for Apache Kafka (MSK) ブローカーストレージ
- Amazon Neptune クラスター
- SageMaker AI エンドポイントバリアント
- SageMaker AI 推論コンポーネント
- SageMaker AI Serverless プロビジョニングされた同時実行数
- スポットフリートリクエスト
- Amazon WorkSpaces のプール
- 独自のアプリケーションまたはサービスにより提供されるカスタムリソース。詳細については、GitHub リポジトリを参照してください。

上記の AWS サービスのリージョン別可用性を確認するには、「リージョンテーブル」を参照してください。

Auto Scaling グループを使用した Amazon EC2 インスタンスフリートのスケーリングについては、Amazon EC2 Auto Scaling ユーザーガイドを参照してください。

Application Auto Scaling の特徴

Application Auto Scaling では、ユーザー定義の条件に従ってスケーラブルリソースを自動的にスケールすることができます。

- ターゲット追跡スケーリング – 特定の CloudWatch メトリクスのターゲット値に基づいてリソースをスケールします。
- ステップスケーリング – 超過アラームのサイズによって異なる一連のスケーリング調整値に基づいてリソースをスケーリングします。
- スケジュールに基づくスケーリング – 1 回のみ、または定期的なスケジュールでリソースをスケールします。
- 予測スケーリング – 履歴データに基づいて予想される負荷に合わせてリソースをプロアクティブにスケーリングします。

Application Auto Scaling と連携

スケーリングするリソースに応じて、次のインターフェイスを使用してスケーリングを設定できます。

- AWS Management Console – スケーリングを設定する際に使用するウェブインターフェイスを提供します。 AWS アカウントにサインアップし、 にサインインします AWS Management Console。次に、概要に一覧表示されているリソースの 1 つのサービスコンソールを開きます。たとえば、Lambda 関数をスケールするには、 を開きます AWS Lambda console。使用するリソース AWS リージョン と同じ でコンソールを開いてください。

  > **Note**
  >
  > リソースにはコンソールアクセスを利用できないものもあります。詳細については、「AWS のサービス Application Auto Scaling で使用できる」を参照してください。

- AWS Command Line Interface （ AWS CLI ） – Windows、macOS AWS のサービス、および Linux でサポートされているさまざまな セットのコマンドを提供します。開始するには、「AWS Command Line Interface」を参照してください。詳細については、AWS CLI コマンドリファレンスの「application-autoscaling」を参照してください。
- AWS Tools for Windows PowerShell – PowerShell 環境でスクリプトを作成するユーザー向けに、さまざまな AWS 製品用のコマンドを提供します。使用を開始する方法については『AWS

Tools for PowerShell ユーザーガイド』を参照してください。詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」を参照してください。

- AWS SDKs – 言語固有の API オペレーションを提供し、署名の計算、リクエストの再試行の処理、エラーの処理など、接続の詳細の多くを処理します。詳細については、「構築するツール AWS」を参照してください。
- HTTPS API – HTTPS リクエストを使用して呼び出す低レベルの API アクションを提供します。詳細については、Application Auto Scaling API リファレンス を参照してください。
- AWS CloudFormation – CloudFormation テンプレートを使用したスケーリングプランの設定をサポートします。詳細については、「AWS CloudFormationを使用して Application Auto Scaling リソースを設定する」を参照してください。

プログラムで に接続するには AWS のサービス、 エンドポイントを使用します。Application Auto Scaling への呼び出しのエンドポイントの詳細については、「Top Secret Regions UserUser GuideEndpoints in Secret Region」の「Application Auto Scaling endpoints and quotas AWS 全般のリファレンスin the Endpoints and ARNs for Amazon Web Services in China」を参照してください。

Application Auto Scaling の概念

このトピックでは、Application Auto Scaling について学習し、使用を開始するために役立つ主な概念について説明します。

スケーラブルターゲット

スケールするリソースを指定するために作成するエンティティです。各スケーラブルターゲットは、サービス名前空間、リソース ID、およびスケーラブルディメンションによって一意に識別されます。これは、基盤となるサービスの容量ディメンションを表します。例えば、Amazon ECS サービスはそのタスク数のオートスケーリングをサポートし、DynamoDB テーブルはテーブルとそのグローバルセカンダリインデックスの読み込みキャパシティーと書き込みキャパシティーのオートスケーリングをサポートし、Aurora クラスターはそのレプリカ数のスケーリングをサポートします。

> **Tip**
>
> 各スケーラブルターゲットには、最小容量と最大容量もあります。スケーリングポリシーが、最小容量から最大容量までの範囲を超える、または下回ることはありません。Application Auto Scaling が認識しない、この範囲外の帯域外変更を基盤となるリソースに直接行うことができます。ただし、スケーリングポリシー、または

`RegisterScalableTarget` API が呼び出されるときは常に、Application Auto Scaling が現在の容量を取得して、それを最小容量および最大容量と比較します。それが最小容量から最大容量までの範囲内に当てはまらない場合、設定された最小容量と最大容量に適合するように容量が更新されます。

スケールイン

Application Auto Scaling がスケーラブルターゲットの容量を自動的に減少させると、スケーラブルターゲットがスケールインします。スケーリングポリシーが設定されている場合、スケーラブルなターゲットを最小キャパシティよりも小さくスケールインすることはできません。

スケールアウト

Application Auto Scaling がスケーラブルターゲットの容量を自動的に増加させると、スケーラブルターゲットがスケールアウトします。スケーリングポリシーが設定されている場合、スケーラブルなターゲットを最大キャパシティよりも大きくスケールアウトすることはできません。

スケーリングポリシー

スケーリングポリシーは、Application Auto Scaling に対して、特定の CloudWatch メトリクスを追跡するように指示します。その後、メトリクスが特定のしきい値よりも高い、または低いときに実行するスケーリングアクションを決定します。例えば、クラスター全体の CPU 使用率が上昇し始めた場合はスケールアウトし、再び低下した場合はスケールインすることができます。

オートスケーリングに使用されるメトリクスはターゲットサービスによって発行されますが、独自のメトリクスを CloudWatch に発行して、それをスケーリングポリシーで使用することもできます。

スケーリングアクティビティ間のクールダウン期間は、別のスケーリングアクティビティが開始される前にリソースを安定させます。Application Auto Scaling は、クールダウン期間中も引き続きメトリクスを評価します。クールダウン期間が終了すると、スケーリングポリシーが、必要に応じて別のスケーリングアクティビティを開始します。クールダウン期間の実施中、現行のメトリクス値に基づいてより大きなスケールアウトが必要になった場合は、スケーリングポリシーが直ちにスケールアウトします。

スケジュールされたアクション

スケジュールされたアクションは、特定の日付けと時刻にリソースを自動的にスケールします。これらは、スケーラブルターゲットの最小容量と最大容量を変更することによって機能するため、最小容量を高く、または最大容量を低く設定することで、スケジュールに従ってスケールイ

ンおよびスケールアウトするために使用できます。例えば、スケジュールされたアクションを使用して、金曜日の容量を減らし、翌週月曜日の容量を増やすことによって、週末にリソースを消費しないアプリケーションをスケールすることができます。

また、最小値と最大値を経時的に最適化するスケジュールされたアクションを使用して、マーケティングキャンペーンや季節的な変動など、通常よりも多いトラフィックが予想される状況に適応することも可能です。そうすることにより、使用量の増加に合わせてスケールアウトする必要があるときにはパフォーマンスを向上させ、使用するリソースが少ないときにはコストを削減することができます。

詳細情報

AWS のサービス Application Auto Scaling で使用できる – このセクションは、スケール可能なサービスについて紹介し、スケーラブルターゲットを登録することによるオートスケーリングのセットアップに役立ちます。また、ターゲットサービス内のリソースにアクセスするために Application Auto Scaling が作成する、各 IAM サービスリンクロールについても説明します。

Application Auto Scaling のターゲット追跡スケーリングポリシー – Application Auto Scaling の主な機能の 1 つは、ターゲット追跡スケーリングポリシーです。設定されたメトリクスと目標値に基づいて使用量を一定のレベルに保つために、ターゲット追跡ポリシーが望ましい容量を自動的に調整する方法について学びます。例えば、スポットフリートの平均 CPU 使用率を 50% に維持するようにターゲット追跡を設定できます。これが設定されると、Application Auto Scaling は、すべてのサーバー全体で集約された CPU 使用率を 50% に維持するために、必要に応じて EC2 インスタンスを起動または終了します。

AWS のサービス Application Auto Scaling で使用できる

Application Auto Scaling は他の AWS サービスと統合されるため、アプリケーションの需要に合わせてスケーリング機能を追加できます。自動スケーリングは、ほとんどすべての場合にデフォルトで無効になっているサービスのオプション機能です。

次の表に、Auto Scaling を設定するためのサポートされているメソッドに関する情報を含め、Application Auto Scaling で使用できる AWS サービスを示します。Application Auto Scaling は、カスタムリソースで使用することも可能です。

- コンソールアクセス – AWS の互換性があるサービスのコンソールでスケーリングポリシーを設定することによってターゲットサービスを設定し、自動スケーリングを開始できます。
- CLI アクセス – AWS CLIを使用して AWS の互換性があるサービスを設定し、自動スケーリングを開始できます。
- SDK アクセス – AWS SDKs を使用して自動スケーリングを開始するように互換性のある AWS サービスを設定できます。
- CloudFormation アクセス – AWS CloudFormation スタックテンプレートを使用して自動スケーリングを開始するように互換性のある AWS サービスを設定できます。詳細については、「AWS CloudFormationを使用して Application Auto Scaling リソースを設定する」を参照してください。

AWS サービス	コンソールアクセス[1]	CLI アクセス	SDK アクセス	CloudFormation アクセス
AppStream 2.0	はい	はい	はい	はい
Aurora	はい	はい	はい	はい

AWS サービス	コンソールアクセス[1]	CLI アクセス	SDK アクセス	CloudFormation アクセス
Amazon Comprehend	いいえ	はい	はい	はい
Amazon DynamoDB	はい	はい	はい	はい
Amazon ECS	はい	はい	はい	はい
Amazon ElastiCache	はい	はい	はい	はい
Amazon EMR	はい	はい	はい	はい
Amazon Keyspaces	はい	はい	はい	はい
Lambda	いいえ	はい	はい	はい

AWS サービス	コンソールアクセス[1]	CLI アクセス	SDK アクセス	CloudFormation アクセス
Amazon MSK	はい	はい	はい	はい
Amazon Neptune	いいえ	はい	はい	はい
SageMaker AI	はい	はい	はい	はい
スポットフリート	はい	はい	はい	はい
WorkSpaces	はい	はい	はい	はい
カスタムリソース	いいえ	はい	はい	はい

1 スケーリングポリシーを設定するためのコンソールアクセス。ほとんどのサービスは、コンソールからのスケジュールされたスケーリングの設定をサポートしていません。現在、スケジュー

ルされたスケーリングに対するコンソールアクセスを提供しているのは Amazon AppStream 2.0、ElastiCache とスポットフリートのみです。

Amazon AppStream 2.0 と Application Auto Scaling

AppStream 2.0 フリートは、ターゲット追跡スケーリングポリシー、ステップスケーリングポリシー、およびスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、AppStream 2.0 の Application Auto Scaling との統合に役立ててください。

AppStream 2.0 用に作成されたサービスリンクロール

AppStream 2.0 リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `appstream.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての AppStream 2.0 フリートの Application Auto Scaling への登録

Application Auto Scaling では、AppStream 2.0 フリートのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

AppStream 2.0 コンソールを使用してオートスケーリングを設定すると、AppStream 2.0 がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  AppStream 2.0 フリートの登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 1 個のフリートインスタンス、および最大容量を 5 個のフリートインスタンスとして、`sample-fleet` という名前のフリートの希望容量を登録します。

  ```
  aws application-autoscaling register-scalable-target \
     --service-namespace appstream \
     --scalable-dimension appstream:fleet:DesiredCapacity \
     --resource-id fleet/sample-fleet \
     --min-capacity 1 \
     --max-capacity 5
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
  target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、`ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity`、および `MaxCapacity` をパラメータとして指定します。

関連リソース

詳細については、Auto ScalingAmazon AppStream 2.0」を参照してください。 Amazon AppStream 2.0

Amazon Aurora と Application Auto Scaling

Aurora DB クラスターは、ターゲット追跡スケーリングポリシー、ステップスケーリングポリシー、およびスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、Aurora の Application Auto Scaling との統合に役立ててください。

Aurora 用に作成されたサービスリンクロール

Aurora リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウントすると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- `AWSServiceRoleForApplicationAutoScaling_RDSCluster`

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `rds.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての Aurora DB クラスターの Application Auto Scaling への登録

Application Auto Scaling では、Aurora DB クラスターのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

Aurora コンソールを使用してオートスケーリングを設定すると、Aurora がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  Aurora クラスターの登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 1 個の Aurora レプリカ、および最大容量を 8 個の Aurora レプリカとして、`my-db-cluster` という名前のクラスター内の Aurora レプリカの数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace rds \
  --scalable-dimension rds:cluster:ReadReplicaCount \
  --resource-id cluster:my-db-cluster \
  --min-capacity 1 \
  --max-capacity 8
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、「Aurora 用 Amazon RDS ユーザーガイド」の「Aurora レプリカを使用した Amazon Aurora Auto Scaling」を参照してください。

Amazon Comprehend と Application Auto Scaling

Amazon Comprehend のドキュメント分類とエンティティ認識器の各エンドポイントは、ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、Amazon Comprehend の Application Auto Scaling との統合に役立ててください。

Amazon Comprehend 用に作成されたサービスリンクロール

Amazon Comprehend リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許

可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- comprehend.application-autoscaling.amazonaws.com

スケーラブルターゲットとしての Amazon Comprehend リソースの Application Auto Scaling への登録

Application Auto Scaling では、Amazon Comprehend のドキュメント分類とエンティティ認識器の各エンドポイントのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定するには、次のオプションを使用できます。

- AWS CLI:

  ドキュメント分類エンドポイントに対して 登録登録-スケーラブル-ターゲット コマンドを呼び出します。以下の例は、最小容量を 1 個の推論単位、および最大容量を 3 個の推論単位とし、ドキュメント分類器エンドポイントの ARN を使用してそのエンドポイントのモデルによって使用される推論単位の希望数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace comprehend \
  --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits
 \
  --resource-id arn:aws:comprehend:us-west-2:123456789012:document-classifier-
endpoint/EXAMPLE \
  --min-capacity 1 \
```

```
  --max-capacity 3
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

エンティティレコグナイザーエンドポイントの register-scalable-target コマンドを呼び出します。以下の例は、最小容量を 1 個の推論単位、および最大容量を 3 個の推論単位とし、エンティティ認識器エンドポイントの ARN を使用してそのエンドポイントのモデルによって使用される推論単位の希望数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace comprehend \
  --scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
  --resource-id arn:aws:comprehend:us-west-2:123456789012:entity-recognizer-endpoint/EXAMPLE \
  --min-capacity 1 \
  --max-capacity 3
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、Amazon Comprehend デベロッパーガイド」の「エンドポイントを使用した自動スケーリング」を参照してください。

Amazon DynamoDB と Application Auto Scaling

DynamoDB のテーブルとグローバルセカンダリインデックスは、ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、DynamoDB の Application Auto Scaling との統合に役立ててください。

DynamoDB 用に作成されたサービスリンクロール

DynamoDB リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- `AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `dynamodb.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての DynamoDB リソースの Application Auto Scaling への登録

Application Auto Scaling では、DynamoDB のテーブルとグローバルセカンダリインデックスのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

DynamoDB コンソールを使用して自動スケーリングを設定すると、DynamoDB がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  テーブルの書き込み容量に対して register-scalable-target コマンドを呼び出します。次の例では、というテーブルのプロビジョニングされた書き込みキャパシティを登録します。最小キャパシティは 5 書き込みキャパシティユニットmy-table、最大キャパシティは 10 書き込みキャパシティユニットです。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:WriteCapacityUnits \
    --resource-id table/my-table \
    --min-capacity 5 \
    --max-capacity 10
  ```

  成功すると、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
  target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

  テーブルの読み取り容量に対して register-scalable-target コマンドを呼び出します。次の の例では、 というテーブルのプロビジョニングされた読み取りキャパシティを登録します。最小キャパシティは 5 読み込みキャパシティユニットmy-table、最大キャパシティは 10 読み込みユニットです。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --min-capacity 5 \
    --max-capacity 10
  ```

  成功すると、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
  target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

グローバルセカンダリインデックスの書き込み容量に対して register-scalable-target コマンドを呼び出します。次の の例では、 というグローバルセカンダリインデックスのプロビジョニングされた書き込み容量を登録します。最小容量は 5 書き込み容量ユニットmy-table-index、最大容量は 10 書き込み容量ユニットです。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:index:WriteCapacityUnits \
  --resource-id table/my-table/index/my-table-index \
  --min-capacity 5 \
  --max-capacity 10
```

成功すると、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

グローバルセカンダリインデックスの読み取り容量に対して register-scalable-target コマンドを呼び出します。次の の例では、 というグローバルセカンダリインデックスのプロビジョニングされた読み込みキャパシティを登録します。最小容量は 5 読み込みキャパシティユニットmy-table-index、最大容量は 10 読み込みキャパシティユニットです。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:index:ReadCapacityUnits \
  --resource-id table/my-table/index/my-table-index \
  --min-capacity 5 \
  --max-capacity 10
```

成功すると、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

Application Auto Scaling の使用を開始したばかりの場合は、以下のドキュメントで DynamoDB リソースのスケーリングに関する有用な詳細情報を確認できます。

- Amazon DynamoDB デベロッパーガイドの「DynamoDB Auto Scaling によるスループット容量の管理」
- Amazon DynamoDB デベロッパーガイドのテーブルの自動スケーリング設定を評価する
- AWS ブログの AWS CloudFormation を使用して DynamoDB テーブルとインデックスの自動スケーリングを設定する方法

Amazon ECS と Application Auto Scaling

ターゲット追跡スケーリングポリシー、予測スケーリングポリシー、ステップスケーリングポリシー、スケジュールされたスケーリングを使用して ECS サービスをスケーリングできます。

以下の情報を使用して、Amazon ECS の Application Auto Scaling との統合に役立ててください。

Amazon ECS 用に作成されたサービスリンクロール

Amazon ECS リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_ECSService

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `ecs.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての ECS サービスの Application Auto Scaling への登録

Application Auto Scaling では、Amazon ECS サービスのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

Amazon ECS コンソールを使用して自動スケーリングを設定すると、Amazon ECS がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  Amazon ECS サービス用の登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小タスク数を 1 個のタスク、最大タスク数を 10 個のタスクとして、`default` クラスターで実行される `sample-app-service` と呼ばれるサービスのスケーラブルターゲットを登録します。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/sample-app-service \
    --min-capacity 1 \
    --max-capacity 10
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

- AWS SDK:

RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

Application Auto Scaling の使用を開始したばかりの場合は、以下のドキュメントで Amazon ECS リソースのスケーリングに関する有用な詳細情報を確認できます。

- Amazon Elastic Container Service デベロッパーガイドの「サービスの自動スケーリング」
- 「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS サービスの自動スケーリングの最適化」

> **Note**
>
> Amazon ECS デプロイの進行中にスケールアウトプロセスを一時停止する手順については、次のドキュメントを参照してください。
> Amazon Elastic Container Service デベロッパーガイドの「サービスの自動スケーリング」

ElastiCache とアプリケーションの Auto Scaling

ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用して、Amazon ElastiCache レプリケーショングループ (Redis OSS と Valkey) と Memcached 独自設計型クラスターを水平方向にスケーリングできます。

ElastiCache を Application Auto Scaling と統合するには、次の情報を使用します。

ElastiCache 用に作成されたサービスリンクロール

スケーラブルターゲットとして ElastiCache リソースを Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `elasticache.application-autoscaling.amazonaws.com`

Application Auto Scaling でのスケーラブルターゲットとしての ElastiCache リソースの登録

Application Auto Scaling では、ElastiCache レプリケーショングループ、クラスター、またはノードのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要です。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

ElastiCache コンソールを使用して自動スケーリングを設定すると、ElastiCache がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  ElastiCache レプリケーショングループに対して登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 1、最大容量を 5 として、`mycluster1` という名前のレプリケーショングループの希望ノードグループ数を登録します。

```
aws application-autoscaling register-scalable-target \
   --service-namespace elasticache \
   --scalable-dimension elasticache:replication-group:NodeGroups \
   --resource-id replication-group/mycluster1 \
   --min-capacity 1 \
   --max-capacity 5
```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
```

```
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

次の例では、 という名前のレプリケーショングループのノードグループあたりの必要なレプリカ数を登録します。最小容量は 1mycluster2、最大容量は 5 です。

```
aws application-autoscaling register-scalable-target \
   --service-namespace elasticache \
   --scalable-dimension elasticache:replication-group:Replicas \
   --resource-id replication-group/mycluster2 \
   --min-capacity 1 \
   --max-capacity 5
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/234abcd56ab78cd901ef1234567890ab1234"
}
```

次の の例では、 というクラスターに必要なノード数を登録します。最小容量は 20mynode1、最大容量は 50 です。

```
aws application-autoscaling register-scalable-target \
   --service-namespace elasticache \
   --scalable-dimension elasticache:cache-cluster:Nodes \
   --resource-id cache-cluster/mynode1 \
   --min-capacity 20 \
   --max-capacity 50
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/01234abcd56ab78cd901ef1234567890ab12"
}
```

- AWS SDK:

RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、Amazon ElastiCache ユーザーガイド」のAuto Scaling Valkey および Redis OSS クラスター」および「Memcached のクラスターのスケーリング」を参照してください。

Amazon Keyspaces (Apache Cassandra 向け) と Application Auto Scaling

Amazon Keyspaces のテーブルは、ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、Amazon Keyspaces の Application Auto Scaling との統合に役立ててください。

Amazon Keyspaces 用に作成されたサービスリンクロール

Amazon Keyspaces リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_CassandraTable

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- cassandra.application-autoscaling.amazonaws.com

スケーラブルターゲットとしての Amazon Keyspaces テーブルの Application Auto Scaling への登録

Application Auto Scaling では、Amazon Keyspaces テーブルのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

Amazon Keyspaces コンソールを使用して自動スケーリングを設定すると、Amazon Keyspaces がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  Amazon Keyspaces テーブルに対して登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 5 個の書き込みキャパシティーユニット、最大容量を 10 個の書き込みキャパシティーユニットとして、mytable と呼ばれるテーブルのプロビジョニングされた書き込みキャパシティーを登録します。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace cassandra \
    --scalable-dimension cassandra:table:WriteCapacityUnits \
    --resource-id keyspace/mykeyspace/table/mytable \
    --min-capacity 5 \
    --max-capacity 10
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

  以下の例は、最小容量を 5 個の読み取りキャパシティーユニット、最大容量を 10 個の読み取りキャパシティーユニットとして、mytable と呼ばれるテーブルのプロビジョニングされた読み取りキャパシティーを登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace cassandra \
  --scalable-dimension cassandra:table:ReadCapacityUnits \
  --resource-id keyspace/mykeyspace/table/mytable \
  --min-capacity 5 \
  --max-capacity 10
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、「Amazon Keyspaces デベロッパーガイド」の「Amazon Keyspaces 自動スケーリングによるスループットキャパシティの自動管理」を参照してください。

AWS Lambda および Application Auto Scaling

ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用して、 AWS Lambda プロビジョニングされた同時実行をスケーリングできます。

以下の情報を使用して、Lambda の Application Auto Scaling との統合に役立ててください。

Lambda 用に作成されたサービスリンクロール

スケーラブルターゲットとして Lambda リソースを Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- lambda.application-autoscaling.amazonaws.com

スケーラブルターゲットとしての Lambda 関数の Application Auto Scaling への登録

Application Auto Scaling では、Lambda 関数のスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定するには、次のオプションを使用できます。

- AWS CLI:

  Lambda 関数に対して登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 0、最大容量を 100 として、my-function と呼ばれる関数の BLUE というエイリアスに対するプロビジョニングされた同時実行数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace lambda \
  --scalable-dimension lambda:function:ProvisionedConcurrency \
  --resource-id function:my-function:BLUE \
  --min-capacity 0 \
  --max-capacity 100
```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
```

```
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

Application Auto Scaling の使用を開始したばかりの場合は、以下のドキュメントで Lambda 関数のスケーリングに関する有用な詳細情報を確認できます。

- AWS Lambda デベロッパーガイドのプロビジョニングされた同時実行の設定
- AWS ブログの「Lambda プロビジョニングされた同時実行の定期的なピーク使用のスケジューリング」

Amazon Managed Streaming for Apache Kafka (MSK) と Application Auto Scaling

Amazon MSK クラスターストレージは、ターゲット追跡スケーリングポリシーを使用してスケールアウトできます。ターゲット追跡ポリシーによるスケールインが無効になっています。

以下の情報を使用して、Amazon MSK の Application Auto Scaling との統合に役立ててください。

Amazon MSK 用に作成されたサービスリンクロール

Amazon MSK リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_KafkaCluster

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `kafka.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての Amazon MSK クラスターストレージの Application Auto Scaling への登録

Application Auto Scaling では、Amazon MSK クラスターのブローカーごとのストレージボリュームサイズに対するスケーリングポリシーを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

Amazon MSK コンソールを使用して自動スケーリングを設定すると、Amazon MSK がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  Amazon MSK クラスターに対して登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 100 GiB、最大容量を 800 GiB として、Amazon MSK クラスターのブローカーあたりのストレージボリュームサイズを登録します。

```
aws application-autoscaling register-scalable-target \
   --service-namespace kafka \
   --scalable-dimension kafka:broker-storage:VolumeSize \
   --resource-id arn:aws:kafka:us-east-1:123456789012:cluster/demo-
cluster-1/6357e0b2-0e6a-4b86-a0b4-70df934c2e31-5 \
   --min-capacity 100 \
   --max-capacity 800
```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

> **Note**
>
> Amazon MSK クラスターがスケーラブルターゲットである場合は、スケールインが無効化されており、有効にすることはできません。

関連リソース

詳細については、「Amazon Managed Streaming for Apache Kafka デベロッパーガイド」の「Amazon MSK クラスターの自動スケーリング」を参照してください。

Amazon Neptune と Application Auto Scaling

Neptune 関数は、ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、Neptune の Application Auto Scaling との統合に役立ててください。

Neptune 用に作成されたサービスリンクロール

スケーラブルターゲットとして Neptune リソースを Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_NeptuneCluster

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `neptune.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての Neptune クラスターの Application Auto Scaling への登録

Application Auto Scaling では、Neptune クラスターのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定するには、次のオプションを使用できます。

- AWS CLI:

  Neptune クラスターに対して登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 1 個、および最大容量を 8 個のフリートインスタンスとして、`mycluster` という名前のクラスターの希望容量を登録します。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace neptune \
    --scalable-dimension neptune:cluster:ReadReplicaCount \
    --resource-id cluster:mycluster \
    --min-capacity 1 \
    --max-capacity 8
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、「Amazon Neptune DB クラスターのレプリカ数の自動スケーリング」を参照してください。

Amazon SageMaker AI と Application Auto Scaling

ターゲット追跡スケーリングポリシー、ステップスケーリングポリシー、スケジュールされたスケーリングを使用して、SageMaker AI エンドポイントバリアント、サーバーレスエンドポイントのプロビジョニングされた同時実行数、推論コンポーネントをスケーリングできます。

SageMaker AI を Application Auto Scaling と統合するには、次の情報を使用します。

SageMaker AI 用に作成されたサービスにリンクされたロール

SageMaker AI リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- `AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `sagemaker.application-autoscaling.amazonaws.com`

SageMaker AI エンドポイントバリアントをスケーラブルターゲットとして Application Auto Scaling に登録する

Application Auto Scaling には、SageMaker AI モデル (バリアント) のスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要です。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

SageMaker AI コンソールを使用して自動スケーリングを設定すると、SageMaker AI は自動的にスケーラブルターゲットを登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  製品バリアントに対して register-scalable-target コマンドを呼び出します。以下の例は、最小容量を 1 個のインスタンス、最大容量を 8 個のインスタンスとして、my-endpoint エンドポイントで実行される my-variant と呼ばれる製品バリアントに対するインスタンスの希望数を登録します。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace sagemaker \
    --scalable-dimension sagemaker:variant:DesiredInstanceCount \
    --resource-id endpoint/my-endpoint/variant/my-variant \
    --min-capacity 1 \
    --max-capacity 8
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

- AWS SDK:

RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

Application Auto Scaling によってサーバーレスエンドポイントのプロビジョニングされた同時実行数をスケーラブルターゲットとして登録

Application Auto Scaling では、サーバーレスエンドポイントのプロビジョニングされた同時実行数のスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットも必要です。

SageMaker AI コンソールを使用して自動スケーリングを設定すると、SageMaker AI は自動的にスケーラブルターゲットを登録します。

それ以外の場合は、次のいずれかの方法を使用して、スケーラブルターゲットを登録します。

- AWS CLI:

  製品バリアントに対して register-scalable-target コマンドを呼び出します。以下の例は、最小容量を 1、最大容量を 10 として、my-endpoint エンドポイントで実行される my-variant と呼ばれる製品バリアントに対するプロビジョニングされた同時実行数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace sagemaker \
  --scalable-dimension sagemaker:variant:DesiredProvisionedConcurrency \
  --resource-id endpoint/my-endpoint/variant/my-variant \
  --min-capacity 1 \
  --max-capacity 10
```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

スケーラブルターゲットとしての推論コンポーネントの Application Auto Scaling への登録

Application Auto Scaling では、推論コンポーネントのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。

- AWS CLI:

  推論コンポーネントに対して register-scalable-target コマンドを呼び出します。以下の例は、最小容量を 0 個のコピー、最大容量を 3 個のコピーとして、my-inference-component という名前の推論コンポーネントの希望コピー数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace sagemaker \
  --scalable-dimension sagemaker:inference-component:DesiredCopyCount \
  --resource-id inference-component/my-inference-component \
  --min-capacity 0 \
  --max-capacity 3
```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

Application Auto Scaling の使用を開始したばかりの場合は、Amazon SageMaker AI デベロッパーガイドで Amazon SageMakerリソースのスケーリングに関するその他の有用な情報を確認できます。

- Amazon SageMaker AI モデルを自動的にスケーリングする
- サーバーレスエンドポイントのプロビジョニングされた同時実行の自動スケール
- マルチモデルエンドポイントデプロイの自動スケーリングポリシーの設定
- 非同期エンドポイントを自動スケーリングする

> **Note**
>
> 2023 年、SageMaker AI はリアルタイム推論エンドポイント上に構築された新しい推論機能を導入しました。エンドポイントのインスタンスタイプと初期インスタンス数を定義するエンドポイント設定を使用して SageMaker AI エンドポイントを作成します。次に、推論コンポーネントを作成します。これは、モデルをエンドポイントにデプロイするために使用できる SageMaker AI ホスティングオブジェクトです。推論コンポーネントのスケーリングの詳細については、 AWS ブログのAmazon SageMaker AI が基盤モデルのデプロイコストとレイテンシーを削減し、Amazon SageMaker AI の最新機能を使用してモデルデプロイコストを平均 50% 削減するのに役立つ新しい推論機能を追加」を参照してください。 Amazon SageMaker

Amazon EC2 スポットフリートと Application Auto Scaling

スポットフリートは、ターゲット追跡スケーリングポリシー、ステップスケーリングポリシー、およびスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、スポットフリートの Application Auto Scaling との統合に役立ててください。

スポットフリート用に作成されたサービスリンクロール

Application Auto Scaling でスケーラブルターゲットとしてスポットフリートリソースを登録する AWS アカウント と、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可し

ます。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- `AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest`

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `ec2.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしてのスポットフリートの Application Auto Scaling への登録

Application Auto Scaling では、スポットフリートのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

スポットフリートコンソールを使用して自動スケーリングを設定すると、スポットフリートがユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  スポットフリートに対して登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小容量を 2 個のインスタンス、および最大容量を 10 個のインスタンスとし、スポットフリートのリクエスト ID を使用してそのターゲット容量を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --min-capacity 2 \
```

```
  --max-capacity 10
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、Amazon EC2 ユーザーガイド」の「スポットフリートの自動スケーリングを理解する」を参照してください。

Amazon WorkSpaces と Application Auto Scaling

WorkSpaces のプールは、ターゲット追跡スケーリングポリシー、ステップスケーリングポリシー、およびスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、WorkSpaces と Application Auto Scaling との統合に役立ててください。

WorkSpaces 用に作成された、サービスにリンクされたロール

Application Auto Scaling は、WorkSpaces リソースをスケーラブルターゲットとして Application Auto Scaling に登録 AWS アカウント すると、AWSServiceRoleForApplicationAutoScaling_WorkSpacesPoolに という名前のサービスリンクロールを自動的に作成します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

この、サービスにリンクされたロールは、マネージドポリシー AWSApplicationAutoscalingWorkSpacesPoolPolicy を使用します。このポリシーは、ユーザーに代わって Amazon WorkSpaces を呼び出すためのアクセス許可を Application Auto

Scaling に付与します。詳細については、「AWS マネージドポリシーリファレンス」の「AWSApplicationAutoscalingWorkSpacesPoolPolicy」を参照してください。

サービスリンクロールが使用するサービスプリンシパル

サービスにリンクされたロールはその引き受け時に、以下のサービスプリンシパルを信頼します。

- `workspaces.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしての WorkSpaces プールの Application Auto Scaling への登録

Application Auto Scaling では、WorkSpaces のスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

WorkSpaces コンソールを使用して自動スケーリングを設定すると、WorkSpaces がユーザーに代わってスケーラブルターゲットを自動的に登録します。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定する場合は、次のオプションを使用できます。

- AWS CLI:

  WorkSpaces のプールに対して register-scalable-target コマンドを呼び出します。次の例では、リクエスト ID を使用して WorkSpaces プールのターゲット容量を登録します。最小容量は 2 つの仮想デスクトップ、最大容量は 10 の仮想デスクトップです。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace workspaces \
    --resource-id workspacespool/wspool-abcdef012 \
    --scalable-dimension workspaces:workspacespool:DesiredUserSessions \
    --min-capacity 2 \
    --max-capacity 10
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

詳細については、「Amazon WorkSpaces 管理ガイド」の「WorkSpaces Pools の Auto Scaling」を参照してください。 Amazon WorkSpaces

カスタムリソースと Application Auto Scaling

カスタムリソースは、ターゲット追跡スケーリングポリシー、ステップスケーリングポリシー、およびスケジュールされたスケーリングを使用してスケールできます。

以下の情報を使用して、カスタムリソースの Application Auto Scaling との統合に役立ててください。

カスタムリソース用に作成されたサービスリンクロール

Application Auto Scaling にスケーラブルターゲットとしてカスタムリソースを登録する AWS アカウント と、次のサービスにリンクされたロールが に自動的に作成されます。このロールは、アカウント内でサポートされている操作を実行することを Application Auto Scaling に許可します。詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

- AWSServiceRoleForApplicationAutoScaling_CustomResource

サービスリンクロールが使用するサービスプリンシパル

前のセクションで説明したサービスリンクロールを引き受けることができるのは、ロールに定義された信頼関係によって認可されるサービスプリンシパルのみです。Application Auto Scaling が使用するサービスリンクロールは、以下のサービスプリンシパルに対するアクセス権を付与します。

- `custom-resource.application-autoscaling.amazonaws.com`

スケーラブルターゲットとしてのカスタムリソースの Application Auto Scaling への登録

Application Auto Scaling では、カスタムリソースのスケーリングポリシーまたはスケジュールされたアクションを作成する前に、スケーラブルターゲットが必要になります。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。スケーラブルターゲットは、リソース ID、スケーラブルディメンション、および名前空間の組み合わせによって一意に識別されます。

CLI またはいずれかの AWS SDKs AWS を使用して自動スケーリングを設定するには、次のオプションを使用できます。

- AWS CLI:

  カスタムリソース用の登録-スケーラブル-ターゲットコマンドを呼び出します。以下の例は、最小希望数を 1 個のキャパシティーユニット、最大希望数を 10 個のキャパシティーユニットとして、カスタムリソースをスケーラブルターゲットとして登録します。`custom-resource-id.txt` ファイルにはリソース ID を識別する文字列が含まれており、これは Amazon API Gateway エンドポイント経由でのカスタムリソースへのパスを表します。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace custom-resource \
    --scalable-dimension custom-resource:ResourceType:Property \
    --resource-id file://~/custom-resource-id.txt \
    --min-capacity 1 \
    --max-capacity 10
  ```

  `custom-resource-id.txt` の内容:

  ```
  https://example.execute-api.us-west-2.amazonaws.com/prod/
  scalableTargetDimensions/1-23456789
  ```

  成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
  target/1234abcd56ab78cd901ef1234567890ab123"
  ```

```
}
```

- AWS SDK:

  RegisterScalableTargetオペレーションを呼び出し、ResourceId、ScalableDimension、ServiceNamespace、MinCapacity、および MaxCapacity をパラメータとして指定します。

関連リソース

Application Auto Scaling の使用を開始したばかりの場合は、以下のドキュメントでカスタムリソースのスケーリングに関する有用な詳細情報を確認できます。

GitHub リポジトリ

AWS CloudFormationを使用して Application Auto Scaling リソースを設定する

Application Auto Scaling は と統合されています。これは AWS CloudFormation、 AWS リソースとインフラストラクチャの作成と管理に費やす時間を短縮できるように、リソースのモデル化とセットアップに役立つサービスです。必要なすべての AWS リソースを記述するテンプレートを作成し、それらのリソースを AWS CloudFormation プロビジョニングして設定します。

を使用すると AWS CloudFormation、テンプレートを再利用して Application Auto Scaling リソースを一貫して繰り返しセットアップできます。リソースを 1 回記述し、同じリソースを複数の AWS アカウント およびリージョンで何度もプロビジョニングします。

Application Auto Scaling と AWS CloudFormation テンプレート

Application Auto Scaling と関連サービスのリソースをプロビジョニングして設定するには、AWS CloudFormation テンプレートを理解しておく必要があります。テンプレートは、JSON または YAML 形式のテキストファイルです。これらのテンプレートは、 AWS CloudFormation スタックでプロビジョニングするリソースを記述します。JSON または YAML に慣れていない場合は、 AWS CloudFormation デザイナー を使用して AWS CloudFormation テンプレートの使用を開始できます。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation Designer とは」を参照してください。

Application Auto Scaling リソースのスタックテンプレートを作成するときは、以下を指定する必要があります。

- ターゲットサービスの名前空間 (**appstream** など)。サービス名前空間を入手するには、「AWS::ApplicationAutoScaling::ScalableTarget」リファレンスを参照してください。
- ターゲットリソースに関連付けられているスケーラブルディメンション (**appstream:fleet:DesiredCapacity** など)。スケーラブルディメンションを入手するには、「AWS::ApplicationAutoScaling::ScalableTarget」リファレンスを参照してください。
- ターゲットリソースのリソース ID (**fleet/sample-fleet** など)。特定のリソース ID の構文と例については、「AWS::ApplicationAutoScaling::ScalableTarget」リファレンスを参照してください。
- ターゲットリソース用のサービスリンクロール (**arn:aws:iam::012345678910:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/**

AWSServiceRoleForApplicationAutoScaling_AppStreamFleet など)。ロール ARN を入手するには、「サービスリンクロールの ARN リファレンス」の表を参照してください。

Application Auto Scaling リソースの詳細については、AWS CloudFormation ユーザーガイドの Application Auto Scaling リファレンスを参照してください。

サンプルテンプレートスニペット

テンプレートに含めるサンプルスニペットは、 AWS CloudFormation ユーザーガイドの以下のセクション AWS CloudFormation で確認できます。

- スケーリングポリシーとスケジュールされたアクションの例については、「Configure Application Auto Scaling resources with AWS CloudFormation」を参照してください。
- スケーリングポリシーのその他の例については、「AWS::ApplicationAutoScaling::ScalingPolicy」を参照してください。

の詳細 AWS CloudFormation

詳細については AWS CloudFormation、次のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- AWS CloudFormation API リファレンス
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

Application Auto Scaling のスケジュールされたスケーリング

スケジュールされたスケーリングでは、特定の時間に容量を増減するスケジュールアクションを作成することで、予測可能な負荷の変化に基づいてアプリケーションの自動スケーリングを設定できます。これにより、予測可能な負荷の変化に合わせてアプリケーションを事前対応的にスケーリングできます。

例えば、負荷が週の半ばに増加し、週の終わりに近づくと減少する、週ごとの定期的なトラフィックパターンが発生しているとしましょう。Application Auto Scaling では、次のパターンに合わせてスケーリングのスケジュールを設定できます。

- 水曜日の朝、前もって設定したスケール可能なターゲットの最小容量を増やすというスケジュールされた 1 つのアクションが容量を増やします。
- 金曜日の夜、前もって設定したスケール可能なターゲットの最大容量を減らすという別のスケジュールされたアクションが容量を減らします。

これらのスケジュールされたスケーリングアクションにより、コストとパフォーマンスを最適化できます。アプリケーションには、週半ばのトラフィックのピークを処理するのに十分な容量がありますが、それ以外の時間帯に不要な容量を過剰にプロビジョニングすることはありません。

スケジュールされたスケーリングとスケーリングポリシーを併用して、スケーリングに事前対応型アプローチと即応型アプローチの両方のメリットを得ることができます。スケジュールされたスケーリングアクションの実行後、スケーリングポリシーは容量をさらにスケールするかどうかの判断を引き続き行うことができます。これは、アプリケーションの負荷を処理するために十分な容量を確保する上で役立ちます。アプリケーションは需要に合わせてスケールしますが、現行の容量は、スケジュールされたアクションによって設定された最小容量と最大容量内に収まる必要があります。

内容

- を使用して Application Auto Scaling のスケジュールされたアクションを削除する AWS CLI

Application Auto Scaling のスケジュールされたスケーリングの仕組み

このトピックでは、スケジュールされたスケーリングがどのように機能するかを説明し、効果的に使用するために理解すべき重要な考慮事項を紹介します。

内容

仕組み

スケジュールされたスケーリングを使用するには、スケジュールされたアクションを作成します。これは、特定の時間にスケーリングアクティビティを実行するよう Application Auto Scaling に指示します。スケジュールされたアクションを作成するときは、スケーラブルターゲット、スケーリングアクティビティを実行するタイミング、最小容量、および最大容量を指定します。スケジュールされたアクションは、1 度だけスケールする、または定期的なスケジュールに従ってスケールするものを作成できます。

指定された時間がくると、Application Auto Scaling は、現行の容量を指定された最小容量および最大容量と比較することによって、新しい容量値に基づいたスケーリングを実行します。

- 現行の容量が指定された最小容量を下回る場合、Application Auto Scaling は指定された最小容量までスケールアウト (容量を増加) します。
- 現行の容量が指定された最大容量を上回る場合、Application Auto Scaling は指定された最大容量までスケールイン (容量を低減) します。

考慮事項

スケジュールされたアクションを作成する場合、次の点に注意してください。

- スケジュールされたアクションにより、指定された日時に、MinCapacity と MaxCapacity がスケジュールされたアクションで指定した容量に設定されます。リクエストには、オプションで、これらのサイズの 1 つだけを含めることができます。例えば、最小容量のみを指定してスケジュールされたアクションを作成できます。ただし、場合によっては、新しい最小容量が最大容量を上回らない、または新しい最大容量が最小容量を下回らないように、両方のサイズを含める必要があります。
- デフォルトでは、設定した定期的なスケジュールは協定世界時 (UTC) です。ローカルタイムゾーンまたはネットワークの他の部分のタイムゾーンに対応するタイムゾーンに変更できます。夏時間を実施するタイムゾーンを指定すると、夏時間 (DST) に合わせて、アクションが自動的に調整されます。詳細については、「Application Auto Scaling を使用して定期的なスケーリングアクションをスケジュールする」を参照してください。
- スケーラブルターゲットに対してスケジュールされたスケーリングをオフにできます。これにより、スケジュールされたアクションを削除せずにアクティブになるのを防ぐことができます。スケジュールされたスケーリングを再度使用する場合は、スケジュールされたスケーリングを再開できます。詳細については、「Application Auto Scaling のスケーリングの一時停止と再開」を参照してください。
- スケジュールされたアクションが実行される順序は、同一のスケーラブルターゲットに対して保証されますが、複数のスケーラブルターゲットにまたがってスケジュールされたアクションに対しては保証されません。
- スケジュールされたアクションが正常に完了するには、指定されたリソースがターゲットサービスでスケーラブルな状態になっている必要があります。その状態になっていない場合、リクエストは失敗し、エラーメッセージ (`Resource Id [ActualResourceId] is not scalable. Reason: The status of all DB instances must be 'available' or 'incompatible-parameters'` など) が返されます。
- Application Auto Scaling とターゲットサービスには分散的な性質があるため、スケジュールされたアクションがトリガーされてからターゲットサービスがスケーリングアクションを引き受けるまでの遅延が数秒におよぶ可能性があります。スケジュールされたアクションは指定された順序で実行されるため、開始時刻が近いスケジュールされたアクションの実行にはより長い時間がかかる場合があります。

スケジュールされたアクションの作成、管理、および削除用によく使用されるコマンド

スケジュールされたスケーリングの操作用によく使用されるコマンドには以下が含まれます。

- register-scalable-target は、 AWS またはカスタムリソースをスケーラブルターゲット (Application Auto Scaling がスケーリングできるリソース) として登録し、スケーリングを停止および再開します。
- put-scheduled-action 既存のスケーラブルターゲットに対するスケジュールされたアクションを追加または変更します。
- describe-scaling-activities は、 AWS リージョンでのスケーリングアクティビティに関する情報を返します。
- describe-scheduled-actions は、 AWS リージョンでスケジュールされたアクションに関する情報を返します。
- delete-scheduled-action スケジュールされたアクションを削除します。

関連リソース

スケジュールされたスケーリングを使用する詳細な例については、 AWS コンピューティングブログのブログ記事「定期的なピーク使用のための AWS Lambda プロビジョニングされた同時実行のスケジュール」を参照してください。

Auto Scaling グループのスケジュールされたアクションの作成についての詳細は、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling のスケジュールされたスケーリング」を参照してください。

制限

以下は、スケジュールされたスケーリングの使用時における制限事項です。

- スケジュールされたアクションの名前は、スケーラブルターゲットごとに一意である必要があります。
- Application Auto Scaling は、スケジュール式で秒レベルの精度を提供しません。Cron 式を使用した場合の最も細かい粒度は 1 分です。
- スケーラブルターゲットを Amazon MSK クラスターにすることはできません。Amazon MSK はスケジュールされたスケーリングをサポートしません。
- スケーラブルリソースに対するスケジュールされたアクションを表示、追加、更新、削除するためのコンソールアクセスは、使用するリソースによって異なります。詳細については、「AWS のサービス Application Auto Scaling で使用できる」を参照してください。

を使用して Application Auto Scaling のスケジュールされたアクションを作成する AWS CLI

次の例は、 AWS CLI put-scheduled-action コマンドを使用してスケジュールされたアクションを作成する方法を示しています。新しい容量を指定するときは、最小容量、最大容量、またはその両方を指定できます。

これらの例では、Application Auto Scaling と統合するサービスのいくつかにスケーラブルターゲットを使用しています。別のスケーラブルターゲットを使用するには、`--service-namespace` でその名前空間、`--scalable-dimension` でそのスケーラブルディメンション、`--resource-id` でそのリソース ID を指定します。

を使用する場合 AWS CLI、コマンドはプロファイル用に AWS リージョン 設定された で実行されることに注意してください。別のリージョンでコマンドを実行する場合は、プロファイルのデフォルトのリージョンを変更するか、コマンドに `--region` パラメータを使用します。

例

- 1 回だけ実行される、スケジュールされたアクションを作成する
- 定期的な間隔で実行されるスケジュールされたアクションを作成する
- 定期的なスケジュールで実行されるスケジュールされたアクションを作成する
- タイムゾーンを指定する 1 回限りのスケジュールされたアクションを作成する
- タイムゾーンを指定する定期的なスケジュールされたアクションを作成する

1 回だけ実行される、スケジュールされたアクションを作成する

指定した日時にスケーラブルターゲットを 1 度だけ自動的にスケーリングするには、`--schedule "at(yyyy-mm-ddThh:mm:ss)"` オプションを使用します。

Example 例: 1 回限りのスケールアウト

以下は、特定の日時に容量をスケールアウトするためのスケジュールされたアクションを作成する例です。

`--schedule` に指定された日時 (2021 年 3 月 31 日の午後 10:00 (UTC)) の時点で、`MinCapacity` に指定された値が現行の容量を超えている場合、Application Auto Scaling が `MinCapacity` にスケールアウトします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource \
  --scalable-dimension custom-resource:ResourceType:Property \
  --resource-id file://~/custom-resource-id.txt \
  --scheduled-action-name scale-out \
  --schedule "at(2021-03-31T22:00:00)" \
  --scalable-target-action MinCapacity=3
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource ^
  --scalable-dimension custom-resource:ResourceType:Property ^
  --resource-id file://~/custom-resource-id.txt ^
  --scheduled-action-name scale-out ^
  --schedule "at(2021-03-31T22:00:00)" ^
  --scalable-target-action MinCapacity=3
```

このスケジュールされたアクションの実行時に、最大容量が最小容量に指定された値を下回る場合は、新しい最小容量だけではなく、新しい最小容量と最大容量を指定する必要があります。

Example 例: 1 回限りのスケールイン

以下は、特定の日時に容量をスケールインするためのスケジュールされたアクションを作成する例です。

`--schedule` に指定された日時 (2021 年 3 月 31 日の午後 10:30 (UTC)) の時点で、`MaxCapacity` に指定された値が現行の容量を下回る場合、Application Auto Scaling が `MaxCapacity` にスケールインします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource \
  --scalable-dimension custom-resource:ResourceType:Property \
  --resource-id file://~/custom-resource-id.txt \
  --scheduled-action-name scale-in \
  --schedule "at(2021-03-31T22:30:00)" \
  --scalable-target-action MinCapacity=0,MaxCapacity=0
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource ^
```

```
  --scalable-dimension custom-resource:ResourceType:Property ^
  --resource-id file://~/custom-resource-id.txt ^
  --scheduled-action-name scale-in ^
  --schedule "at(2021-03-31T22:30:00)" ^
  --scalable-target-action MinCapacity=0,MaxCapacity=0
```

定期的な間隔で実行されるスケジュールされたアクションを作成する

定期的な間隔でスケーリングをスケジュールするには、`--schedule "rate(value unit)"` オプションを使用します。値は正の整数である必要があります。単位は、`minute`、`minutes`、`hour`、`hours`、`day`、または `days` にすることができます。詳細については、Amazon EventBridge ユーザーガイドのrate 式を参照してください。

以下は、rate 式を使用するスケジュールされたアクションの例です。

指定されたスケジュール (2021 年 1 月 30 日の午後 12:00 (UTC) から 5 時間ごとに実行され、2021 年 1 月 31 日の午後 10:00 (UTC) に終了) で、`MinCapacity` に指定された値が現行の容量を超えている場合、Application Auto Scaling が `MinCapacity` にスケールアウトします。`MaxCapacity` に指定された値が現行の容量を下回る場合は、Application Auto Scaling が `MaxCapacity` にスケールインします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service \
  --scheduled-action-name my-recurring-action \
  --schedule "rate(5 hours)" \
  --start-time 2021-01-30T12:00:00 \
  --end-time 2021-01-31T22:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace ecs ^
  --scalable-dimension ecs:service:DesiredCount ^
  --resource-id service/my-cluster/my-service ^
  --scheduled-action-name my-recurring-action ^
  --schedule "rate(5 hours)" ^
  --start-time 2021-01-30T12:00:00 ^
  --end-time 2021-01-31T22:00:00 ^
```

```
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

定期的なスケジュールで実行されるスケジュールされたアクションを作成する

定期的なスケーリングをスケジュールするには、--schedule "cron(*fields*)" オプションを使用します。詳細については、「Application Auto Scaling を使用して定期的なスケーリングアクションをスケジュールする」を参照してください。

以下は、Cron 式を使用するスケジュールされたアクションの例です。

指定されたスケジュール (毎日午前 9:00 (UTC)) で、MinCapacity に指定された値が現行の容量を超えている場合、Application Auto Scaling が MinCapacity にスケールアウトします。MaxCapacity に指定された値が現行の容量を下回る場合は、Application Auto Scaling が MaxCapacity にスケールインします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action --service-namespace appstream \
  --scalable-dimension appstream:fleet:DesiredCapacity \
  --resource-id fleet/sample-fleet \
  --scheduled-action-name my-recurring-action \
  --schedule "cron(0 9 * * ? *)" \
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace appstream ^
  --scalable-dimension appstream:fleet:DesiredCapacity ^
  --resource-id fleet/sample-fleet ^
  --scheduled-action-name my-recurring-action ^
  --schedule "cron(0 9 * * ? *)" ^
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

タイムゾーンを指定する 1 回限りのスケジュールされたアクションを作成する

スケジュールされたアクションは、デフォルトで UTC タイムゾーンに設定されます。別のタイムゾーンを指定するには、--timezone オプションを含めて、タイムゾーンの正規名 (America/New_York など) を指定します。詳細については、https://www.joda.org/joda-time/timezones.html を

参照してください。このページには、put-scheduled-action を呼び出すときにサポートされる IANA タイムゾーンに関する情報が記載されています。

以下は、特定の日時に容量をスケールするためのスケジュールされたアクションの作成時に --timezone オプションを使用する例です。

--schedule に指定された日時 (2021 年 1 月 31 日の午後 5:00 (ローカルタイム)) の時点で、MinCapacity に指定された値が現行の容量を超えている場合、Application Auto Scaling が MinCapacity にスケールアウトします。MaxCapacity に指定された値が現行の容量を下回る場合は、Application Auto Scaling が MaxCapacity にスケールインします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action --service-namespace comprehend \
  --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits \
  --resource-id arn:aws:comprehend:us-west-2:123456789012:document-classifier-endpoint/
EXAMPLE \
  --scheduled-action-name  my-one-time-action \
  --schedule "at(2021-01-31T17:00:00)" --timezone "America/New_York" \
  --scalable-target-action MinCapacity=1,MaxCapacity=3
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace comprehend ^
  --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits ^
  --resource-id arn:aws:comprehend:us-west-2:123456789012:document-classifier-endpoint/
EXAMPLE ^
  --scheduled-action-name  my-one-time-action ^
  --schedule "at(2021-01-31T17:00:00)" --timezone "America/New_York" ^
  --scalable-target-action MinCapacity=1,MaxCapacity=3
```

タイムゾーンを指定する定期的なスケジュールされたアクションを作成する

以下は、キャパシティーを拡張するための定期的なスケジュール済みアクションを作成できる --timezone オプションの使用例です。詳細については、「Application Auto Scaling を使用して定期的なスケーリングアクションをスケジュールする」を参照してください。

指定されたスケジュール (毎週月曜日から金曜日までの午後 6:00 (ローカルタイム)) で、MinCapacity に指定された値が現行の容量を超えている場合、Application Auto Scaling が

MinCapacity にスケールアウトします。MaxCapacity に指定された値が現行の容量を下回る場合は、Application Auto Scaling が MaxCapacity にスケールインします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action --service-namespace lambda \
  --scalable-dimension lambda:function:ProvisionedConcurrency \
  --resource-id function:my-function:BLUE \
  --scheduled-action-name my-recurring-action \
  --schedule "cron(0 18 ? * MON-FRI *)" --timezone "Etc/GMT+9" \
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace lambda ^
  --scalable-dimension lambda:function:ProvisionedConcurrency ^
  --resource-id function:my-function:BLUE ^
  --scheduled-action-name my-recurring-action ^
  --schedule "cron(0 18 ? * MON-FRI *)" --timezone "Etc/GMT+9" ^
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

を使用して Application Auto Scaling のスケジュールされたスケーリングを記述する AWS CLI

これらの AWS CLI コマンド例では、Application Auto Scaling と統合する サービスのリソースを使用して、スケーリングアクティビティとスケジュールされたアクションについて説明します。別のスケーラブルターゲットについては、--service-namespace でその名前空間、--scalable-dimension でそのスケーラブルディメンション、--resource-id でそのリソース ID を指定します。

を使用する場合 AWS CLI、コマンドはプロファイル用に AWS リージョン 設定された で実行されることに注意してください。別のリージョンでコマンドを実行する場合は、プロファイルのデフォルトのリージョンを変更するか、コマンドに --region パラメータを使用します。

例

- サービスのスケーリングアクティビティを説明する
- サービスのスケジュールされたアクションを説明する
- スケーラブルターゲットに対するスケジュールされたアクションを記述する

サービスのスケーリングアクティビティを説明する

指定されたサービス名前空間にあるすべてのスケーラブルターゲットに対するスケーリングアクティビティを表示するには、describe-scaling-activities コマンドを使用します。

以下の例は、dynamodb サービス名前空間に関連付けられているスケーリングアクティビティを取得します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-activities --service-namespace dynamodb
```

Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace dynamodb
```

Output

成功すると、コマンドは以下のような出力を返します。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 10.",
            "ResourceId": "table/my-table",
            "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
            "StartTime": 1561574415.086,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574449.51,
            "Cause": "maximum capacity was set to 10",
            "StatusMessage": "Successfully set write capacity units to 10. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 5 and max capacity to 10",
            "ResourceId": "table/my-table",
            "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
            "StartTime": 1561574414.644,
            "ServiceNamespace": "dynamodb",
```

```
            "Cause": "scheduled action name my-second-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 5 and max capacity to
 10",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/my-table",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574140.255,
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/my-table",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max capacity to
 20",
            "StatusCode": "Successful"
        }
    ]
}
```

このコマンドを変更して、スケーラブルターゲットのうち 1 つのターゲットのみに関するスケーリングアクティビティを取得するようにするには、`--resource-id` オプションを追加します。

サービスのスケジュールされたアクションを説明する

指定されたサービス名前空間にあるすべてのスケーラブルターゲットに対するスケジュールされたアクションを記述するには、describe-scheduled-actions コマンドを使用します。

以下の例は、`ec2` サービス名前空間に関連付けられているスケジュールされたアクションを取得します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2
```

Output

成功すると、コマンドは以下のような出力を返します。

```
{
    "ScheduledActions": [
        {
            "ScheduledActionName": "my-one-time-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:493a6261-fbb9-432d-855d-3c302c14bdb9:resource/ec2/
spot-fleet-request/sfr-107dc873-0802-4402-a901-37294EXAMPLE:scheduledActionName/my-one-
time-action",
            "ServiceNamespace": "ec2",
            "Schedule": "at(2021-01-31T17:00:00)",
            "Timezone": "America/New_York",
            "ResourceId": "spot-fleet-request/sfr-107dc873-0802-4402-
a901-37294EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "ScalableTargetAction": {
                "MaxCapacity": 1
            },
            "CreationTime": 1607454792.331
        },
        {
            "ScheduledActionName": "my-recurring-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:493a6261-fbb9-432d-855d-3c302c14bdb9:resource/ec2/
spot-fleet-request/sfr-107dc873-0802-4402-a901-37294EXAMPLE:scheduledActionName/my-
recurring-action",
            "ServiceNamespace": "ec2",
            "Schedule": "rate(5 minutes)",
            "ResourceId": "spot-fleet-request/sfr-107dc873-0802-4402-
a901-37294EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "StartTime": 1604059200.0,
```

```
            "EndTime": 1612130400.0,
            "ScalableTargetAction": {
                "MinCapacity": 3,
                "MaxCapacity": 10
            },
            "CreationTime": 1607454949.719
        },
        {
            "ScheduledActionName": "my-one-time-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:4bce34c7-bb81-4ecf-b776-5c726efb1567:resource/ec2/
spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE:scheduledActionName/my-one-
time-action",
            "ServiceNamespace": "ec2",
            "Schedule": "at(2020-12-08T9:36:00)",
            "Timezone": "America/New_York",
            "ResourceId": "spot-fleet-request/sfr-40edeb7b-9ae7-44be-
bef2-5c4c8EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "ScalableTargetAction": {
                "MinCapacity": 1,
                "MaxCapacity": 3
            },
            "CreationTime": 1607456031.391
        }
    ]
}
```

スケーラブルターゲットに対するスケジュールされたアクションを記述する

指定されたスケーラブルターゲットに対するスケジュールされたアクションの情報を取得するには、describe-scheduled-actions コマンドを使用してスケジュールされたアクションを記述するときに `--resource-id` オプションを追加します。

以下の例にあるように、`--scheduled-action-names` オプションを含めて、スケジュールされたアクションの名前をその値として指定すると、コマンドは名前が一致するスケジュールされたアクションのみを返します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2 \
```

```
  --resource-id spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE \
  --scheduled-action-names my-one-time-action
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2 ^
  --resource-id spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE ^
  --scheduled-action-names my-one-time-action
```

Output

成功すると、コマンドは以下のような出力を返します。--scheduled-action-names に複数の値が指定されている場合、名前が一致するスケジュールされたアクションのすべてが出力に含まれます。

```
{
    "ScheduledActions": [
        {
            "ScheduledActionName": "my-one-time-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:4bce34c7-bb81-4ecf-b776-5c726efb1567:resource/ec2/
spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE:scheduledActionName/my-one-
time-action",
            "ServiceNamespace": "ec2",
            "Schedule": "at(2020-12-08T9:36:00)",
            "Timezone": "America/New_York",
            "ResourceId": "spot-fleet-request/sfr-40edeb7b-9ae7-44be-
bef2-5c4c8EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "ScalableTargetAction": {
                "MinCapacity": 1,
                "MaxCapacity": 3
            },
            "CreationTime": 1607456031.391
        }
    ]
}
```

Application Auto Scaling を使用して定期的なスケーリングアクションをスケジュールする

> **Important**
>
> Amazon EC2 Auto Scaling の cron 式の詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「繰り返しのスケジュール」トピックを参照してください。Amazon EC2 Auto Scaling では、Application Auto Scaling が使用するカスタム cron 構文の代わりに、従来の cron 構文を使用します。

cron 式を使用して、定期的なスケジュールで実行されるスケジュールされたアクションを作成できます。

定期的なスケジュールを作成する場合は、cron 式とタイムゾーンを指定して、スケジュールされたアクションがいつ繰り返されるのかを記述します。サポートされているタイムゾーン値は、Joda-Time でサポートされている IANA タイムゾーンの正規名です (`Etc/GMT+9`、`Pacific/Tahiti` など)。必要に応じて、開始時刻、終了時刻、またはその両方の日付と時刻を指定できます。を使用してスケジュールされたアクションを作成するコマンドの例については、 AWS CLI 「」を参照してくださいタイムゾーンを指定する定期的なスケジュールされたアクションを作成する。

サポートされている cron 式の形式は、スペースで区切られた [Minutes] [Hours] [Day_of_Month] [Month] [Day_of_Week] [Year] の 6 つのフィールドで構成されます。例えば、cron 式 `30 6 ? * MON *` は毎週月曜日の午前 6:30 に繰り返すようスケジュールされたアクションを設定します。アスタリスクは、フィールドのすべての値を照合するワイルドカードとして使用されます。

Application Auto Scaling のスケジュールされたアクションの cron 構文の詳細については、「Amazon EventBridge ユーザーガイド」の「cron 式のリファレンス」を参照してください。

定期的なスケジュールを作成するときは、開始時刻と終了時刻を慎重に選択します。以下に留意してください。

- 開始時刻を指定すると、Application Auto Scaling はこの時刻にアクションを実行し、その後は指定された反復周期に基づいてアクションを実行します。
- 終了時刻を指定すると、その時刻以降はアクションが反復されなくなります。Application Auto Scaling は以前の値を記録せず、終了時刻後に以前の値に戻ることはありません。
- AWS CLI または SDKs を使用してスケジュールされたアクションを作成または更新する場合、AWS 開始時刻と終了時刻は UTC で設定する必要があります。

例

Application Auto Scaling のスケーラブルターゲットに対して定期的なスケジュールを作成する場合は、次の表を参照してください。次は、Application Auto Scaling を使用して、スケジュールされたアクションを作成または更新するための正しい構文の例です。

分	時間	日	月	曜日	年	意味
0	10	*	*	?	*	毎日午前 10:00 (UTC) に実行
15	12	*	*	?	*	毎日午後 12:15 (UTC) に実行
0	18	?	*	MON-FRI	*	毎週月曜日から金曜日まで午後 6:00 (UTC) に実行
0	8	1	*	?	*	毎月 1 日の午前 8:00 (UTC) に実行
0/15	*	*	*	?	*	15 分ごとに実行
0/10	*	?	*	MON-FRI	*	月曜日から金曜日まで 10 分ごとに実行
0/5	8-17	?	*	MON-FRI	*	毎週月曜日から金曜日

分	時間	日	月	曜日	年	意味
						まで午前 8:00 から午後 5:55 (UTC) の間に 5 分ごとに実行

Exception

7 つのフィールドを含む文字列値を使用して cron 式を作成することもできます。この場合、最初の 3 つのフィールドを使用して、スケジュールされたアクションを実行する時間を秒単位で指定できます。完全な cron 式には、スペースで区切られた [Seconds] [Minutes] [Hours] [Day_of_Month] [Month] [Day_of_Week] [Year] のフィールドが含まれます。ただし、この方法は、スケジュールされたアクションが指定した秒に正確に実行されることを保証するものではありません。また、一部のサービスコンソールでは、cron 式の 2 番目のフィールドがサポートされていない場合があります。

スケーラブルターゲットに対するスケジュールされたスケーリングをオフにする

スケジュールされたスケーリングは、スケジュールされたアクションを削除せずに一時的に無効化することができます。詳細については、「Application Auto Scaling のスケーリングの一時停止と再開」を参照してください。

スケジュールされたスケーリングを停止する

以下の例にあるように、`--suspended-state` オプションがある register-scalable-target コマンドを使用し、`ScheduledScalingSuspended` 属性の値として `true` を指定することによって、スケーラブルターゲットでスケジュールされたスケーリングを一時停止します。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target --service-namespace rds \
  --scalable-dimension rds:cluster:ReadReplicaCount --resource-id cluster:my-db-cluster
 \
  --suspended-state '{"ScheduledScalingSuspended": true}'
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace rds ^
  --scalable-dimension rds:cluster:ReadReplicaCount --resource-id cluster:my-db-cluster
 ^
  --suspended-state "{\"ScheduledScalingSuspended\": true}"
```

Output

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。以下は出力例です。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

スケーリングスケジュールを再開する

スケジュールされたスケーリングを再開するには、ScheduledScalingSuspended の値として false を指定して、register-scalable-target コマンドを再度実行します。

を使用して Application Auto Scaling のスケジュールされたアクションを削除する AWS CLI

スケジュールされたアクションを使い終わったら削除することができます。

スケジュールされたアクションを削除する

delete-scheduled-action コマンドを使用します。このコマンドが正常に完了した場合は、出力が返されません。

Linux、macOS、または Unix

```
aws application-autoscaling delete-scheduled-action \
  --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE \
  --scheduled-action-name my-recurring-action
```

Windows

```
aws application-autoscaling delete-scheduled-action ^
  --service-namespace ec2 ^
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity ^
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE ^
  --scheduled-action-name my-recurring-action
```

スケーラブルなターゲットを登録解除する

スケーラブルターゲットも完了した場合は、登録解除できます。次の deregister-scalable-target コマンドを使用します。まだ削除されていないスケジュールされたアクションやスケーリングポリシーがある場合は、このコマンドによって削除されます。このコマンドが正常に完了した場合は、出力が返されません。

Linux、macOS、または Unix

```
aws application-autoscaling deregister-scalable-target \
  --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE
```

Windows

```
aws application-autoscaling deregister-scalable-target ^
  --service-namespace ec2 ^
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity ^
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE
```

Application Auto Scaling のターゲット追跡スケーリングポリシー

ターゲット追跡スケーリングポリシーは、ターゲットメトリクス値に基づいてアプリケーションを自動的にスケールします。これにより、手動で操作しなくても、アプリケーションは最適なパフォーマンスとコスト効率を維持できます。

ターゲット追跡を使用することで、アプリケーションの理想的な平均使用率またはスループットレベルを表すメトリクスとターゲット値を選択します。Application Auto Scaling は、メトリクスとターゲットから逸脱したときにスケーリングイベントをトリガーする CloudWatch アラームを作成および管理します。これは、サーモスタットがターゲット温度を維持する仕組みと似ています。

例えば、現在スポットフリートで実行されているアプリケーションがあり、アプリケーションの負荷が変化してもフリートの CPU 使用率を約 50% に維持する必要があるとします。これにより、過剰な数のアイドルリソースを維持することなくトラフィックのスパイクを処理するための追加のキャパシティが得られます。

このニーズを満たすには、50% の平均 CPU 使用率をターゲットとする、ターゲット追跡スケーリングポリシーを作成します。次に、CPU が 50% を超えると、Application Auto Scaling がスケールアウト (容量を増やし) して負荷の増加に対応します。CPU が 50% を下回るとスケールイン (容量が減少) し、使用率が低い期間のコストを最適化します。

ターゲット追跡ポリシーにより、CloudWatch アラームとスケーリング調整を手動で定義する必要がなくなります。Application Auto Scaling は、設定したターゲットに基づいてこれを自動的に処理します。

事前定義されたメトリクスまたはカスタムメトリクスのいずれかを使用して、ターゲット追跡スケーリングポリシーをベースにできます。

- 事前定義メトリクス — Application Auto Scaling によって提供されるメトリクス (ターゲットごとの平均 CPU 使用率や平均リクエスト数など)。
- カスタムメトリクス — メトリクスを組み合わせる、既存のメトリクスを活用する、または CloudWatch に公開された独自のカスタムメトリクスを使用するために、メトリクス計算を使用できます。

スケーラブルなターゲット容量の変化に反比例して変化するメトリクスを選択してください。つまり、容量を 2 倍に増やすと、メトリクスが 50% 減少するという仕組みです。これにより、メトリクスデータが比例スケーリングイベントを正確にトリガーできます。

内容

Application Auto Scaling のターゲット追跡スケーリングの仕組み

このトピックでは、ターゲット追跡スケーリングの仕組みと、ターゲット追跡スケーリングポリシーの主要な要素について説明します。

内容

仕組み

ターゲット追跡スケーリングを使用するには、ターゲット追跡スケーリングポリシーを作成し、以下を指定します。

- メトリクス — 平均 CPU 使用率やターゲットごとの平均リクエスト数など、追跡する CloudWatch メトリクス。

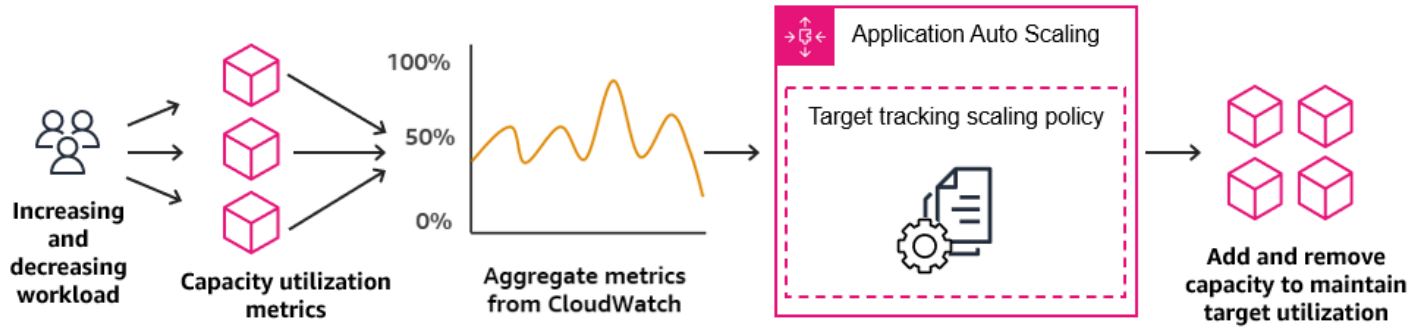
- ターゲット値 — メトリクスのターゲット値 (CPU 使用率 50%、ターゲット 1 分あたり 1000 リクエストなど)。

Application Auto Scaling は、スケーリングポリシーを呼び出す CloudWatch アラームを作成および管理し、メトリクスとターゲット値に基づいてスケーリング調整値を計算します。これは、指定されたターゲット値、またはそれに近い値にメトリクスを維持するため、必要に応じて容量を追加または削除します。

メトリクスが目標値を上回る場合、Application Auto Scaling は容量を追加してメトリクス値とターゲット値の差を減らすことでスケールアウトします。メトリクス値がターゲット値を下回ると、Application Auto Scaling は容量を削除してスケールインします。

スケーリングアクティビティは、容量の急激な変動を防ぐため、クールダウン期間を設けて実行されます。オプションでスケーリングポリシーのクールダウン期間を設定できます。

次の図は、設定完了時におけるターゲット追跡スケーリングポリシーの動作の概要を示しています。



- ターゲット追跡スケーリングポリシーは、使用率が低下したときの容量の削除よりも、使用率が増加したときの容量の追加の方が強力である点に注意してください。例えば、ポリシーの指定されたメトリクスがターゲット値に到達した場合、ポリシーはアプリケーションの負荷がすでに高くなっていると見なします。したがって、できるだけ早くメトリクス値に比例した容量を追加することで対応します。メトリクスが大きいほど、より多くの容量が追加されます。

メトリクスがターゲット値を下回ると、ポリシーは使用率が最終的には再び増加することを期待します。この場合、ポリシーが容量を削除することによってスケーリングの速度を落とすのは、使用率がターゲット値を下回るしきい値未満になり (通常は 10% を超えて低い値の場合)、そのレベルが使用率が減速したとみなされるに十分である場合のみになります。この保守的な動作の意図は、アプリケーションの需要が以前ほど高いレベルでなくなった場合にのみ、容量が削除されるようにすることです。

メトリクスを選択する

事前定義されたメトリクスまたはカスタムメトリクスのいずれかを使用して、ターゲット追跡スケーリングポリシーを作成できます。

事前定義済みメトリクスタイプでターゲット追跡スケーリングポリシーを作成する場合、ターゲット追跡スケーリングポリシーの事前定義メトリクス の事前定義済みメトリクスのリストからメトリクスを選択します。

メトリクスを選択するときは、以下の点に注意してください。

- カスタムメトリクスにはターゲット追跡に使用できないものもあります。メトリクスは、有効な使用率メトリクスで、スケーラブルなターゲットの使用頻度を示す必要があります。メトリクス値は、スケーラブルなターゲットを比例的にスケールするためにメトリクスデータを使用できるようにするため、スケーラブルなターゲットの容量に対して比例的に増減する必要があります。
- `ALBRequestCountPerTarget` メトリクスを使用するには、`ResourceLabel` パラメータを指定して、メトリクスに関連付けられているターゲットグループを識別する必要があります。
- メトリクスが CloudWatch に実数 0 の値を出力する場合 (`ALBRequestCountPerTarget` など)、Application Auto Scaling は、長期間アプリケーションへのトラフィックがない場合に 0 にスケールインできます。スケーラブルターゲットにリクエストがルーティングされないときにターゲットを 0 にスケールインするには、スケーラブルターゲットの最小容量が 0 に設定されている必要があります。
- スケーリングポリシーで使用する新しいメトリクスを公開する代わりに、メトリクス数式を使用して既存のメトリクスを組み合わせることができます。詳細については、「Metric Math を使用して、Application Auto Scaling のターゲット追跡スケーリングポリシーを作成する」を参照してください。
- 使用しているサービスがサービスのコンソールでカスタムメトリクスの指定をサポートするかどうかを確認するには、そのサービスのドキュメントを参照してください。
- 使用率の変化に迅速に対応できるよう、1 分間隔で利用できるメトリクスを使用することをお勧めします。ターゲット追跡では、すべての事前定義済みメトリクスとカスタムメトリクスについて、1 分単位で集計されたメトリクスが評価されますが、基盤となるメトリクスではデータの発行頻度が低くなる可能性があります。たとえば、Amazon EC2 メトリクスはすべてデフォルトで 5 分間隔で送信されますが、1 分に設定できます (詳細モニタリングと呼ばれます)。この選択は個々のサービス次第です。ほとんどの場合、可能な限り短い間隔を使用しようとします。

ターゲット値の定義

ターゲット追跡スケーリングポリシーを作成するときは、ターゲット値を指定する必要があります。ターゲット値は、アプリケーションの最適な平均使用率またはスループットを表します。優れたコスト効率でリソースを使用するには、予期しないトラフィックの増加に対して適切なバッファを使用し、ターゲット値をできる限り高く設定します。アプリケーションが通常のトラフィックフローに対して最適にスケールアウトされる場合、実際のメトリクス値は、ターゲット値以下である必要があります。

スケーリングポリシーが Application Load Balancer のターゲットごとのリクエスト数、ネットワーク I/O、またはその他のカウントメトリクスなどのスループットに基づいている場合、ターゲット値は、1 分間における、単一のエンティティ (Application Load Balancer のターゲットグループの単一ターゲットなど) からの最適な平均スループットを表します。

クールダウン期間を定義する

必要に応じて、ターゲット追跡スケーリングポリシーでクールダウン期間を定義できます。

クールダウン期間は、前回のスケーリングアクティビティが有効になるまでスケーリングポリシーが待機する時間を指定します。

クールダウン期間には次の 2 種類があります。

- スケールアウトクールダウン期間では、スケールアウトが継続的に (ただし過剰になることなく) 行われます。スケーリングポリシーを使用して Application Auto Scaling が正常にスケールアウトすると、クールダウン時間の計算が開始されます。スケーリングポリシーは、より大きなスケールアウトがトリガーされるか、クールダウン期間が終了しない限り、必要な容量を再度増加させません。このスケールアウトクールダウン期間が有効な間は、スケールアウトアクティビティを開始することで追加された容量は、次のスケールアウトアクティビティに予定される容量の一部として繰り入れられます。
- スケールインクールダウン期間では、スケールインを控え目に行ってアプリケーションの可用性を保護することを目的としているため、スケールインアクティビティはスケールインクールダウン期間が終了するまでブロックされます。ただし、スケールインクールダウン期間中に別のアラームがスケールアウトアクティビティをトリガーした場合、Application Auto Scaling scale によってターゲットが即座にスケールアウトされます。この場合、スケールインクールダウン期間は停止し、完了しません。

各クールダウン期間は秒単位で測定され、スケーリングポリシー関連のスケーリングアクティビティにのみ適用されます。クールダウン期間中、スケジュールされたアクションがスケジュールされた時間に開始されると、クールダウン期間の期限が切れるのを待たずにスケーリングアクティビティを即座にトリガーできます。

デフォルト値で開始し、値を後で微調整できます。例えば、ターゲット追跡スケーリングポリシーが短期間に発生する変更に対して積極的になりすぎないように、場合によってはクールダウン期間を延長する必要があります。

デフォルト値

Application Auto Scaling では、ElastiCache のデフォルト値は 600 で、次のスケーラブルターゲットのデフォルト値は 300 です。

- AppStream 2.0 フリート
- Aurora DB クラスター
- ECS サービス
- Neptune クラスター
- SageMaker AI エンドポイントバリアント
- SageMaker AI 推論コンポーネント
- SageMaker AI Serverless でプロビジョニングされた同時実行数
- Spot Fleets
- WorkSpaces のプール
- カスタムリソース

他のすべてのスケーラブルターゲットのデフォルト値は 0 または null です。

- Amazon Comprehend ドキュメントの分類とエンティティ認識のエンドポイント
- DynamoDB テーブルとグローバルセカンダリインデックス
- Amazon Keyspaces テーブル
- Lambda プロビジョニング済み同時実行
- Amazon MSK ブローカーストレージ

Application Auto Scaling がクールダウン期間を評価するとき、null 値はゼロ値と同じように扱われます。

null 値を含む任意のデフォルト値を更新して、独自のクールダウン期間を設定できます。

考慮事項

ターゲット追跡スケーリングポリシーを使用する場合は、次の考慮事項が適用されます。

- ターゲット追跡スケーリングポリシーで使用される CloudWatch アラームを作成、編集、削除しないでください。Application Auto Scaling は、ターゲット追跡スケーリングポリシーに関連付けられている CloudWatch アラームを作成および管理し、不要になるとそれらを削除します。
- メトリクスにデータポイントがない場合、CloudWatch アラームの状態は `INSUFFICIENT_DATA` に変化します。これが発生すると、Application Auto Scaling は、新しいデータポイントが見つかるまでスケーラブルなターゲットをスケールできません。詳細については、Amazon CloudWatch ユーザーガイドの「CloudWatch アラームが欠落データを処理する方法の設定」を参照してください。
- メトリクスが設計上まばらに報告される場合は、メトリクス数式が役立つことがあります。例えば、最新の値を使用するには、`FILL(m1,REPEAT)` という関数を使用します (`m1` がメトリクスです)。
- ターゲット値と実際のメトリクスデータポイント間にギャップが発生する場合があります。これは、Application Auto Scaling が追加または削除する容量を判断するときに、その数を切り上げまたは切り捨てることによって、常に控えめに動作するためです。これにより、不十分な容量を追加したり、必要以上に容量を削除することを防ぎます。ただし、小容量のスケーラブルなターゲットの場合、実際のメトリクスデータポイントがターゲット値からかなり離れているように見えることがあります。

  容量が大きいスケーラブルなターゲットでは、容量を追加または削除することにより、ターゲット値と実際のメトリクスデータポイントの間のギャップが少なくなります。
- ターゲットの追跡スケーリングポリシーでは、指定されたメトリクスがターゲット値を超えている場合、スケールアウトする必要があると見なされます。指定されたメトリクスがターゲット値を下回っている場合、ターゲットの追跡スケーリングポリシーを使用してスケールアウトすることはできません。

複数のスケーリングポリシー

それぞれが異なるメトリクスを使用していれば、スケーラブルなターゲットに対して複数のターゲットの追跡スケーリングポリシーを設定できます。Application Auto Scaling の目的は常に可用性を優先することであるため、その動作は、スケールアウトまたはスケールインに対するターゲット追跡ポリシーの準備が整っているかどうかに応じて異なります。ターゲット追跡ポリシーのいずれかでス

ケールアウトする準備ができると、スケーラブルなターゲットがスケールアウトされますが、すべてのターゲット追跡ポリシー (スケールイン部分が有効) でスケールインする準備ができている場合のみスケールインされます。

複数のスケーリングポリシーが、スケーラブルターゲットに対してスケールアウトまたはスケールインする指示を同時に出す場合、Application Auto Scaling はスケールインとスケールアウトのどちらについても、最大の容量を提供するポリシーに基づいてスケールします。これにより、複数のシナリオに対応する柔軟性が高まり、ワークロードを処理するのに十分な容量が常に確保されます。

ターゲット追跡スケーリングポリシーのスケールイン部分を無効にして、スケールアウトで使用する方法とは別の方法をスケールインで使用できます。例えば、スケールアウトにはターゲットの追跡スケーリングポリシーを使用しながら、スケールインにはステップスケーリングポリシーを使用できます。

ただし、ターゲット追跡スケーリングポリシーをステップスケーリングポリシーとともに使用する場合、ポリシー間の競合によって望ましくない動作が生じる可能性があるため、注意することをお勧めします。例えば、ターゲット追跡ポリシーがスケールインする準備が整う前に、ステップスケーリングポリシーがスケールインアクティビティを開始した場合、スケールインアクティビティはブロックされません。スケールインアクティビティが完了した後で、ターゲット追跡ポリシーにより、スケーラブルなターゲットに再びスケールアウトするよう指示できます。

周期的な性質のワークロードの場合、スケジュールされたスケーリングを使用してスケジュールに従って容量の変更を自動化することもできます。スケジュールされたアクションごとに、新しい最小容量値と新しい最大容量値を定義できます。これらの値は、スケーリングポリシーの境界を形成します。スケジュールされたスケーリングとターゲットトラッキングスケーリングの組み合わせにより、容量がすぐに必要になったときに、使用率レベルの急激な増加による影響を軽減できます。

スケーリングポリシーの作成、管理、および削除用によく使用されるコマンド

スケーリングポリシーの操作用によく使用されるコマンドには以下が含まれます。

- register-scalable-target は、 AWS またはカスタムリソースをスケーラブルターゲット (Application Auto Scaling がスケーリングできるリソース) として登録し、スケーリングを停止して再開します。
- put-scaling-policy 既存のスケーラブルターゲットのスケーリングポリシーを追加または変更します。
- describe-scaling-activities は、 AWS リージョンのスケーリングアクティビティに関する情報を返します。

- describe-scaling-policies AWS リージョン内のスケーリングポリシーに関する情報を返します。
- delete-scaling-policy スケーリングポリシーを削除します。

関連リソース

Auto Scaling グループのターゲット追跡スケーリングポリシーの作成の詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling のターゲットトラッキングスケーリングポリシー」を参照してください。

制限

以下は、ターゲット追跡スケーリングポリシーの使用時における制限事項です。

- スケーラブルターゲットを Amazon EMR クラスターにすることはできません。Amazon EMR はターゲット追跡スケーリングポリシーをサポートしません。
- Amazon MSK クラスターがスケーラブルターゲットである場合は、スケールインが無効化されており、有効にすることはできません。
- `RegisterScalableTarget` または `PutScalingPolicy` API オペレーションを使用して AWS Auto Scaling スケーリングプランを更新することはできません。
- スケーラブルリソースに対するターゲット追跡スケーリングポリシーを表示、追加、更新、削除するためのコンソールアクセスは、使用するリソースによって異なります。詳細については、「AWS のサービス Application Auto Scaling で使用できる」を参照してください。

を使用して Application Auto Scaling のターゲット追跡スケーリングポリシーを作成する AWS CLI

この例では、 AWS CLI コマンドを使用して Amazon EC2 スポットフリートのターゲットラックポリシーを作成します。別のスケーラブルターゲットについては、`--service-namespace` でその名前空間、`--scalable-dimension` でそのスケーラブルディメンション、`--resource-id` でそのリソース ID を指定します。

を使用する場合 AWS CLI、コマンドはプロファイル用に AWS リージョン 設定された で実行されることに注意してください。別のリージョンでコマンドを実行する場合は、プロファイルのデフォルトのリージョンを変更するか、コマンドに `--region` パラメータを使用します。

タスク

ステップ 1: スケーラブルなターゲットを登録する

まだ登録していない場合は、スケーラブルターゲットを登録します。register-scalable-target コマンドを使用して、ターゲットサービス内の特定のリソースをスケーラブルターゲットとして登録します。以下の例は、スポットフリートリクエストを Application Auto Scaling に登録します。Application Auto Scaling は、スポットフリート内のインスタンスの数を最小 2 インスタンス、および最大 10 インスタンスにスケールできます。各##############を独自の情報に置き換えます。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --min-capacity 2 --max-capacity 10
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace ec2 ^
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity ^
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE ^
  --min-capacity 2 --max-capacity 10
```

Output

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。以下は出力例です。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

ステップ 2: ターゲット追跡スケーリングポリシーを作成する

ターゲット追跡スケーリングポリシーを作成するには、次の例を使用して開始できます。

ターゲット追跡スケーリングポリシーを作成する

1. 以下の cat コマンドを使用して、スケーリングポリシーのターゲット値と事前に定義されたメトリクスの仕様を、ホームディレクトリにある config.json という名前の JSON ファイルに保存します。平均 CPU 使用率を 50 パーセントに維持するターゲット追跡設定の例を次に示します。

```
$ cat ~/config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
    }
}
```

詳細については、Application Auto Scaling API リファレンスの「PredefinedMetricSpecification」を参照してください。

または、CloudWatch でカスタマイズされたメトリクス仕様を作成し、各パラメータの値を追加することによって、スケーリング用のカスタムメトリクスを使用することもできます。指定されたメトリクスの平均使用率を 100 パーセントに維持するターゲット追跡設定の例を以下に示します。

```
$ cat ~/config.json
{
   "TargetValue": 100.0,
   "CustomizedMetricSpecification":{
      "MetricName": "MyUtilizationMetric",
      "Namespace": "MyNamespace",
      "Dimensions": [
         {
            "Name": "MyOptionalMetricDimensionName",
            "Value": "MyOptionalMetricDimensionValue"
         }
      ],
      "Statistic": "Average",
      "Unit": "Percent"
   }
}
```

詳細については、Application Auto Scaling API リファレンスの「CustomizedMetricSpecification」を参照してください。

2. 作成した config.json ファイルと共に以下の put-scaling-policy コマンドを使用して、cpu50-target-tracking-scaling-policy という名前のスケーリングポリシーを作成します。

Linux、macOS、または Unix

```
aws application-autoscaling put-scaling-policy --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --policy-name cpu50-target-tracking-scaling-policy --policy-type
 TargetTrackingScaling \
  --target-tracking-scaling-policy-configuration file://config.json
```

Windows

```
aws application-autoscaling put-scaling-policy --service-namespace ec2 ^
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity ^
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE ^
  --policy-name cpu50-target-tracking-scaling-policy --policy-type
 TargetTrackingScaling ^
  --target-tracking-scaling-policy-configuration file://config.json
```

Output

成功した場合、このコマンドはユーザーに代わって作成された 2 つの CloudWatch アラームの ARN と名前を返します。以下は出力例です。

```
{
    "PolicyARN": "arn:aws:autoscaling:region:account-
id:scalingPolicy:policy-id:resource/ec2/spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE:policyName/cpu50-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-
b46e-434a-a60f-3b36d653feca",
            "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca"
```

```
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-
d19b-4a63-a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
        }
    ]
}
```

ステップ 3: ターゲット追跡スケーリングポリシーを記述する

以下の describe-scaling-policies コマンドを使用して、指定したサービス名前空間に対するすべてのスケーリングポリシーを記述することができます。

```
aws application-autoscaling describe-scaling-policies --service-namespace ec2
```

結果をフィルタリングし、--query パラメータを使用してターゲット追跡スケーリングポリシーのみに制限することができます。query 用の構文の詳細については、AWS Command Line Interface ユーザーガイドの「AWS CLIからのコマンド出力の制御」を参照してください。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-policies --service-namespace ec2 \
  --query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

Windows

```
aws application-autoscaling describe-scaling-policies --service-namespace ec2 ^
  --query "ScalingPolicies[?PolicyType==`TargetTrackingScaling`]"
```

Output

以下は出力例です。

```
[
    {
```

```
        "PolicyARN": "PolicyARN",
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
            },
            "TargetValue": 50.0
        },
        "PolicyName": "cpu50-target-tracking-scaling-policy",
        "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
        "ServiceNamespace": "ec2",
        "PolicyType": "TargetTrackingScaling",
        "ResourceId": "spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
        "Alarms": [
            {
                "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-
b46e-434a-a60f-3b36d653feca",
                "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca"
            },
            {
                "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-
d19b-4a63-a812-6c67aaf2910d",
                "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
            }
        ],
        "CreationTime": 1515021724.807
    }
]
```

を使用して Application Auto Scaling のターゲット追跡スケーリングポリシーを削除する AWS CLI

ターゲット追跡スケーリングポリシーが不要になったら、delete-scaling-policy コマンドを使用してポリシーを削除することができます。

次のコマンドは、指定したスポットフリートリクエストに対して指定したターゲット追跡スケーリングポリシーを削除します。これは、Application Auto Scaling がユーザーに代わって作成した CloudWatch アラームも削除します。

Linux、macOS、または Unix

```
aws application-autoscaling delete-scaling-policy --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --policy-name cpu50-target-tracking-scaling-policy
```

Windows

```
aws application-autoscaling delete-scaling-policy --service-namespace ec2 ^
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity ^
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE ^
  --policy-name cpu50-target-tracking-scaling-policy
```

Metric Math を使用して、Application Auto Scaling のターゲット追跡スケーリングポリシーを作成する

メトリクス数学の使用により、複数の CloudWatch メトリクスをクエリし、数表現を使用して、メトリクスに基づく新しい時系列を作成できます。作成された時系列を CloudWatch コンソール内で視覚化でき、ダッシュボードに追加できます。Metric Math の詳細については、「Amazon CloudWatch ユーザーガイド」の「Metric Math を使用する」を参照してください。

Metric Math の数式には、次の考慮事項が適用されます。

- 利用可能な CloudWatch メトリクスをクエリできます。各メトリクスは、メトリクス名、名前空間、0 以上のディメンションの一意の組み合わせです。
- 任意の算術演算子 (+ - * / ^)、統計関数 (AVG や SUM など)、または CloudWatch がサポートするその他の関数を使用できます。
- 数式の関係式では、メトリクスと他の数式の結果の両方を使用できます。
- メトリクスの指定で使用される数式はすべて、最終的に単一の時系列を返す必要があります。
- CloudWatch コンソールまたは CloudWatch GetMetricData API を使用して、Metric Math の数式が有効であることを確認できます。

トピック

- 例: タスクごとの Amazon SQS キューバックログ
- 制限

例: タスクごとの Amazon SQS キューバックログ

タスクごとの Amazon SQS キューバックログを計算するには、キューからの取得に使用できるメッセージの概数を取得し、その数を、サービスで実行されている Amazon ECS タスクの数で割ります。詳細については、 AWS コンピューティングブログの「カスタムメトリクスを使用した Amazon Elastic Container Service (ECS) Auto Scaling」を参照してください。

この数式のロジックは次のとおりです。

```
sum of (number of messages in the queue)/(number of tasks that are
currently in the RUNNING state)
```

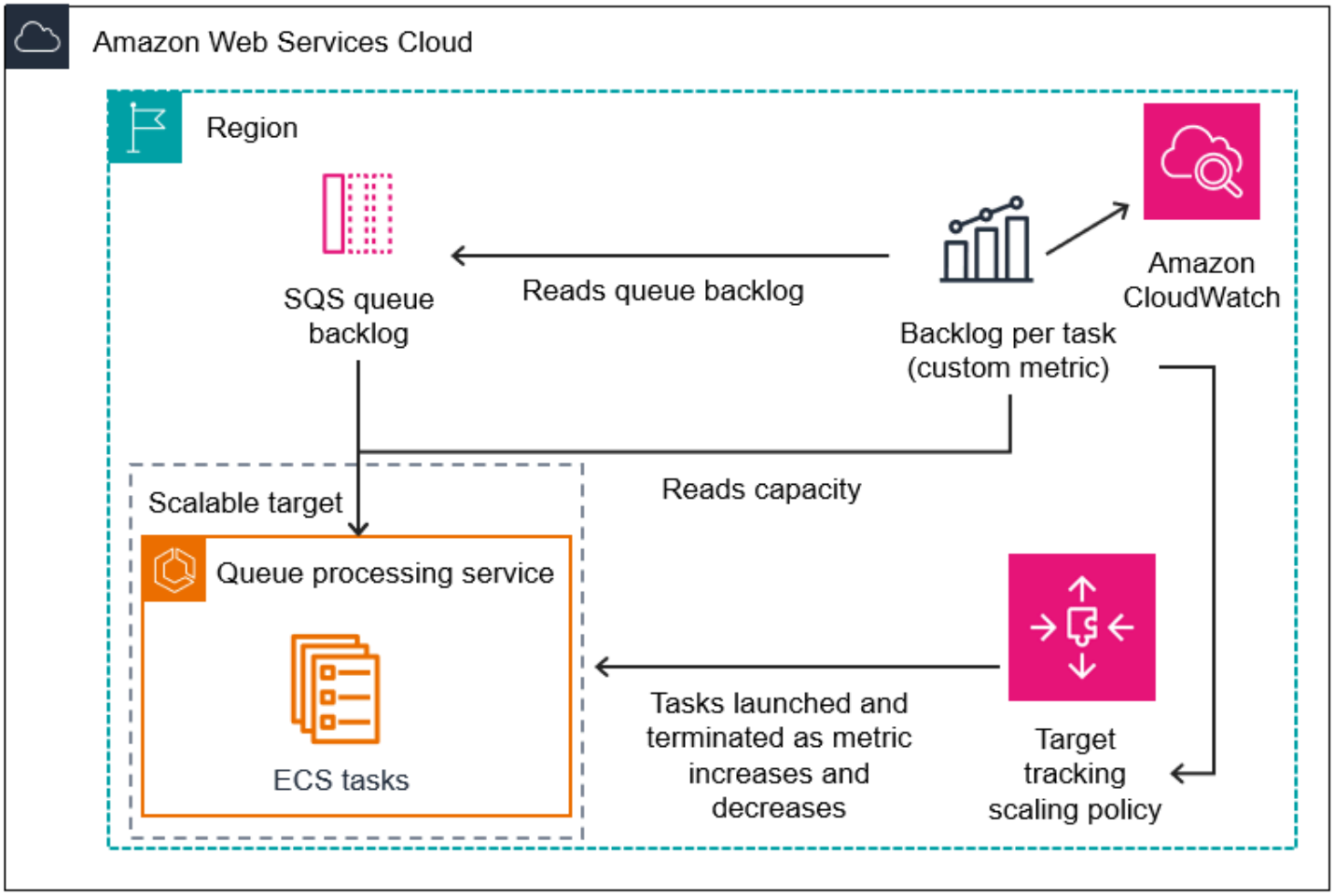
この場合、CloudWatch メトリクス情報は次のようになります。

ID	CloudWatch メトリクス	統計	間隔
m1	ApproximateNumberOfMessagesVisible	合計	1 分
m2	RunningTaskCount	平均	1 分

メトリクス数学 ID と表現は次のとおりです。

ID	表現
e1	`(m1)/(m2)`

このメトリクスのアーキテクチャを以下に図で示します。



この Metric Math を使用してターゲット追跡スケーリングポリシーを作成するには (AWS CLI)

1. Metric Math の数式を、カスタマイズされたメトリクス仕様の一部として、`config.json` という名前の JSON ファイルに保存します。

   次の例を参考にして開始してください。各`###############`を独自の情報に置き換えます。

```
{
    "CustomizedMetricSpecification": {
        "Metrics": [
            {
                "Label": "Get the queue size (the number of messages waiting to be
 processed)",
                "Id": "m1",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "ApproximateNumberOfMessagesVisible",
                        "Namespace": "AWS/SQS",
```

```
                        "Dimensions": [
                            {
                                "Name": "QueueName",
                                "Value": "my-queue"
                            }
                        ]
                    },
                    "Stat": "Sum"
                },
                "ReturnData": false
            },
            {
                "Label": "Get the ECS running task count (the number of currently
running tasks)",
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "RunningTaskCount",
                        "Namespace": "ECS/ContainerInsights",
                        "Dimensions": [
                            {
                                "Name": "ClusterName",
                                "Value": "my-cluster"
                            },
                            {
                                "Name": "ServiceName",
                                "Value": "my-service"
                            }
                        ]
                    },
                    "Stat": "Average"
                },
                "ReturnData": false
            },
            {
                "Label": "Calculate the backlog per instance",
                "Id": "e1",
                "Expression": "m1 / m2",
                "ReturnData": true
            }
        ]
    },
    "TargetValue": 100
```

```
}
```

詳細については、Application Auto Scaling API Reference の「TargetTrackingScalingPolicyConfiguration」を参照してください。

> **Note**
>
> 以下は、CloudWatch メトリクスのメトリクス名、名前空間、ディメンション、および統計を見つけるために役立つ追加のリソースです。
>
> - AWS サービスで利用可能なメトリクスの詳細については、「Amazon AWS CloudWatch ユーザーガイド」の「CloudWatch メトリクスを発行するサービス」を参照してください。 Amazon CloudWatch
> - を使用して CloudWatch メトリクスの正確なメトリクス名、名前空間、ディメンション (該当する場合) を取得するには AWS CLI、「list-metrics」を参照してください。

2. このポリシーを作成するには、以下の例にあるように、JSON ファイルを入力として使用して put-scaling-policy コマンドを実行します。

```
aws application-autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-scaling-policy \
  --service-namespace ecs --scalable-dimension ecs:service:DesiredCount --resource-id service/my-cluster/my-service \
  --policy-type TargetTrackingScaling --target-tracking-scaling-policy-configuration file://config.json
```

成功した場合、このコマンドは、ユーザーに代わって作成したポリシーの Amazon リソースネーム (ARN) および 2 つの CloudWatch アラームの ARN を返します。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:
 8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/my-cluster/my-
service:policyName/sqs-backlog-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/my-cluster/my-service-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
            "AlarmName": "TargetTracking-service/my-cluster/my-service-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
```

```
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/my-cluster/my-service-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
            "AlarmName": "TargetTracking-service/my-cluster/my-service-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
        }
    ]
}
```

> **Note**
>
> このコマンドがエラーをスローする場合は、 を AWS CLI ローカルで最新バージョンに更新していることを確認してください。

制限

- 最大リクエストサイズは 50 KB です。これは、ポリシー定義でメトリクス計算を使用した場合の PutScalingPolicy API リクエストの合計ペイロードサイズです。この制限を超えると、Application Auto Scaling はリクエストを拒否します。
- ターゲット追跡スケーリングポリシーでメトリクス計算を使用する場合、次のサービスはサポートされません。
  - Amazon Keyspaces (Apache Cassandra 向け)
  - DynamoDB
  - Amazon EMR
  - Amazon MSK
  - Amazon Neptune

Application Auto Scaling のステップスケーリングポリシー

ステップスケーリングポリシーは、CloudWatch アラームに基づいて、あらかじめ定義された単位でアプリケーションの容量をスケーリングします。アラームのしきい値を超えると、スケールアウト (容量の増加) とスケールイン (キャパシティの減少) を処理するスケーリングポリシーを個別に定義できます。

ステップスケーリングポリシーを使用して、スケーリングプロセスを呼び出す CloudWatch アラームを作成および管理します。アラームに違反すると、Application Auto Scaling はそのアラームに関連付けられたスケーリングポリシーを開始します。

ステップスケーリングポリシーは、ステップ調整と呼ばれる一連の調整を使用して容量をスケーリングします。調整値の規模は、アラーム違反の大きさに応じて異なります。

- 違反が最初のしきい値を超えると、Application Auto Scaling は最初のステップ調整を適用します。
- 違反が 2 番目のしきい値を超えると、Application Auto Scaling は 2 番目のステップ調整を適用するというように続きます。

これにより、スケーリングポリシーは、アラームメトリクスのマイナーな変更とメジャーな変更の両方に適切に対応できます。

ポリシーは、スケーリングアクティビティの進行中も、引き続き別のアラームに対応します。つまり、Application Auto Scaling はアラーム違反が発生するたびに、それらをすべて評価します。複数のアラーム違反が連続して発生することによるオーバースケーリングを防ぐため、クールダウン期間が設けられています。

ターゲットトラッキングと同様に、ステップスケーリングはトラフィックの変化に応じてアプリケーションの容量を自動スケーリングするのに役立ちます。ただし、安定したスケーリングのニーズに対応するには、ターゲット追跡ポリシーの方が実装と管理が容易な傾向があります。

サポートされているスケーラブルターゲット

ステップスケーリングポリシーは、以下のスケーラブルなターゲットで使用できます。

- AppStream 2.0 フリート
- Aurora DB クラスター
- ECS サービス

- EMR クラスター
- SageMaker AI エンドポイントバリアント
- SageMaker AI 推論コンポーネント
- SageMaker AI Serverless プロビジョニングされた同時実行数
- Spot Fleets
- カスタムリソース

内容

Application Auto Scaling のステップスケーリングの仕組み

このトピックでは、ステップスケーリングの仕組みについて説明し、ステップスケーリングポリシーの主要な要素を紹介します。

内容

仕組み

ステップスケーリングを使用するには、スケーラブルなターゲット用に CloudWatch アラームを作成します。アラーム違反を判断するメトリクス、しきい値、評価期間の数を定義します。また、アラー

ムのしきい値を超えた場合の容量のスケーリング方法を定義するステップスケーリングポリシーを作成し、それをスケーラブルな目標と関連付けることもできます。

ポリシーにステップ調整値を追加します。アラームの違反規模に基づいて、さまざまなステップ調整値を定義できます。以下に例を示します。

- アラームメトリクスが 60% に達したら、10 キャパシティーユニットずつスケールアウトする
- アラームメトリクスが 75% に達したら、30 キャパシティーユニットずつスケールアウトする
- アラームメトリクスが 85% に達したら、40 キャパシティーユニットずつスケールアウトする

指定した評価期間にアラームのしきい値を超えると、Application Auto Scaling はポリシーで定義されたステップ調整を適用します。アラームの状態が OK に戻るまで、さらなるアラーム違反が発生した場合に備えて、調整を続けることができます。

スケーリングアクティビティは、容量の急激な変動を防ぐため、クールダウン期間を設けて実行されます。オプションでスケーリングポリシーのクールダウン期間を設定できます。

ステップ調整値

ステップスケーリングポリシーを作成するときは、アラーム超過のサイズに基づいてターゲット容量を動的にスケーリングする 1 つ以上のステップ調整値を指定します。各ステップ調整値は、次のように指定します。

- メトリクス値の下限
- メトリクス値の上限
- スケーリング調整タイプに基づいてスケールする量

CloudWatch は、CloudWatch アラームに関連付けられたメトリクスの統計に基づいて、メトリクスデータポイントを集計します。アラームに違反すると、適切なスケーリングポリシーが呼び出されます。Application Auto Scaling は、raw メトリクスデータではなく、CloudWatch からの最新のメトリクスデータポイントに指定された集計タイプを適用します。ステップ調整によって定義された上限と下限に対して、この集約メトリクス値を比較することにより、実行するステップ調整が決定されます。

違反しきい値に比例して上限と下限を指定します。例えば、メトリクスが 50% を超えたときの CloudWatch アラームとスケールアウトポリシーを作成したとします。次に、メトリクスが 50% を下回ったときの 2 つ目のアラームとスケールインポリシーを作成しました。ポリシーごとに `PercentChangeInCapacity` の調整タイプを設定して、一連の段階的調整を行いました。

例: スケールアウトポリシーのステップ調整値

下限	上限	調整
0	10	0
10	20	10
20	null	30

例: スケールインポリシーのステップ調整値

下限	上限	調整
-10	0	0
-20	-10	-10
null	-20	-30

これにより、次のスケーリング設定が作成されます。

```
Metric value

-infinity          30%    40%          60%     70%             infinity
-------------------------------------------------------------------------
          -30%      | -10% | Unchanged  | +10%  |       +30%
-------------------------------------------------------------------------
```

次に、例えば、容量が 10 のスケーラブルなターゲットでこのスケーリング設定を使用するとします。以下の点は、スケーラブルターゲットの容量に関連してスケーリング設定の動作をまとめたものです。

- 集合メトリクス値が 40 より大きく 60 未満である間は、元の容量が維持されます。
- メトリクス値が 60 に到達すると、Application Auto Scaling はスケーラブルターゲットの容量に 1 を足して 11 にします。これはスケールアウトポリシーの 2 番目のステップ調整値に基づきます (10 の 10% を追加)。新しい容量が追加されると、Application Auto Scaling は現行の容量を 11 に増やします。この容量の増加後にメトリクス値が 70 に上昇すると、Application Auto Scaling は

ターゲット容量に 3 を足して 14 にします。これはスケールアウトポリシーの 3 番目のステップ調整値に基づきます (11 の 30% である 3.3 を、3 に切り捨てて追加)。

- メトリクス値が 40 になると、Application Auto Scaling はスケールインポリシーの 2 番目のステップ調整値 (14 の 10%、つまり 1.4 を四捨五入した 1 を削除) に基づき、スケーラブルターゲットの容量から 1 を引いて 13 にします。この容量の減少後にメトリクス値がさらに 30 まで減った場合、Application Auto Scaling はスケールインポリシーの 3 番目のステップ調整 (13 の 30%、つまり 3.9 を四捨五入した 3 を削除) に基づき、ターゲット容量から 3 を引いて 10 にします。

スケーリングポリシーのステップ調整を指定するときは、次の点に注意してください。

- ステップ調整値の範囲に重複や間隔があってはなりません。
- 1 つのステップ調整値のみ、下限を null (負の無限大) にすることができます。下限が負のステップ調整値がある場合は、下限が null のステップ調整値が必要です。
- 1 つのステップ調整値のみ、上限を null (正の無限大) にすることができます。上限が正のステップ調整値がある場合は、上限が null のステップ調整値が必要です。
- 同じステップ調整値で上限と下限を null にすることはできません。
- メトリクス値が超過しきい値を上回っている場合、下限にその値を含み、上限には含みません。メトリクス値が超過しきい値を下回っている場合、下限にその値を含まず、上限に含みます。

スケーリング調整タイプ

選択したスケーリング調整タイプに基づいて、最適なスケーリングアクションを実行するスケーリングポリシーを定義できます。調整タイプは、スケーラブルターゲットの現在の容量に対する割合、または絶対数で指定できます。

Application Auto Scaling は、ステップスケーリングポリシーに対して以下の調整タイプをサポートします。

- ChangeInCapacity – スケーラブルターゲットの現行容量を、指定された値に基づいて増減させます。正の値はキャパシティーを増やし、負の値はキャパシティーを減らします。例えば、現行容量が 3 で調整値が 5 の場合、Application Auto Scaling は容量に 5 を追加して合計を 8 にします。
- ExactCapacity – スケーラブルターゲットの現行容量を、指定された値に変更します。この調整タイプには負の値以外を指定します。例えば、現行容量が 3 で調整値が 5 の場合、Application Auto Scaling は容量を 5 に変更します。
- PercentChangeInCapacity – スケーラブルターゲットの現行容量を、指定された割合 (%) に基づいて増減させます。正の値はキャパシティーを増やし、負の値はキャパシティーを減らします。例え

ば、現行容量が 10 で調整値が 10 パーセントの場合、Application Auto Scaling は容量に 1 を追加して合計を 11 にします。

調整後の値が整数ではない場合、Application Auto Scaling はその値を以下のように四捨五入します。

- 1 より大きい値は小数点以下が切り捨てられます。例えば、12.7 は 12 に丸められます。
- 0 と 1 の間の値は 1 に丸められます。例えば、.67 は 1 に丸められます。
- 0 と -1 の間の値は -1 に丸められます。例えば、-.58 は -1 に丸められます。
- -1 未満の値は小数点以下が切り捨てられます。例えば、-6.67 は -6 に丸められます。

PercentChangeInCapacity では、MinAdjustmentMagnitude パラメータを使用してスケーリングする最小の数量を指定できます。例えば、25% 追加するポリシーを作成して、最小数量を 2 に指定するとします。スケーラブルなターゲットの容量が 4 の時にスケーリングポリシーを実行すると、4 の 25% は 1 です。しかし、最小増分が 2 に指定されていることから、Application Auto Scaling は 2 を追加します。

クールダウン期間

必要に応じて、ステップスケーリングポリシーでクールダウン期間を定義できます。

クールダウン期間は、前回のスケーリングアクティビティが有効になるまでスケーリングポリシーが待機する時間を指定します。

ステップスケーリング設定のクールダウン期間の使用を計画する方法は次の 2 つです。

- スケールアウトポリシーのクールダウン期間では、スケールアウトが継続的に (ただし過剰になることなく) 行われます。スケーリングポリシーを使用して Application Auto Scaling が正常にスケールアウトすると、クールダウン時間の計算が開始されます。スケーリングポリシーは、より大きなスケールアウトがトリガーされるか、クールダウン期間が終了しない限り、必要な容量を再度増加させません。このスケールアウトクールダウン期間が有効な間は、スケールアウトアクティビティを開始することで追加された容量は、次のスケールアウトアクティビティに予定される容量の一部として繰り入れられます。
- スケールインポリシーのクールダウン期間では、スケールインを控え目に行ってアプリケーションの可用性を保護することを目的としているため、スケールインアクティビティはスケールインクールダウン期間が終了するまでブロックされます。ただし、スケールインクールダウン期間中に別のアラームがスケールアウトアクティビティをトリガーした場合、Application Auto Scaling scale に

よってターゲットが即座にスケールアウトされます。この場合、スケールインクールダウン期間は停止し、完了しません。

例えば、トラフィックピークが発生すると、アラームがトリガーされ、Application Auto Scaling は、増加したロードを処理できるように容量を自動的に追加します。スケールアウトポリシーのクールダウン期間を設定した場合、アラームがポリシーをトリガーして容量を 2 増やすと、スケーリングアクティビティは正常に完了し、スケールアウトクールダウン期間が始まります。クールダウン期間中にアラームが再度トリガーし、さらに進んだステップ調整を行う場合 (3 の増加)、以前の 2 の増加は現在の容量の一部とみなされます。したがって、容量に追加されるのは 1 だけです。これにより、必要以上に容量を追加しなくても、クールダウンの期限が切れるのを待つよりも速くスケーリングできます。

クールダウン期間は秒単位で測定され、スケーリングポリシー関連のスケーリングアクティビティにのみ適用されます。クールダウン期間中、スケジュールされたアクションがスケジュールされた時間に開始されると、クールダウン期間の期限が切れるのを待たずにスケーリングアクティビティを即座にトリガーできます。

値を指定しない場合、デフォルト値は 300 です。

スケーリングポリシーの作成、管理、および削除用によく使用されるコマンド

スケーリングポリシーの操作用によく使用されるコマンドには以下が含まれます。

- register-scalable-target は、 AWS またはカスタムリソースをスケーラブルターゲット (Application Auto Scaling がスケーリングできるリソース) として登録し、スケーリングを停止および再開します。
- put-scaling-policy 既存のスケーラブルターゲットのスケーリングポリシーを追加または変更します。
- describe-scaling-activities AWS リージョン内でのスケーリングアクティビティに関する情報を返します。
- describe-scaling-policies AWS リージョン内のスケーリングポリシーに関する情報を返します。
- delete-scaling-policy スケーリングポリシーを削除します。

考慮事項

ステップスケーリングポリシーを使用する場合は、次の考慮事項が適用されます。

- ステップスケーリングを使用できるほど正確にアプリケーションのステップ調整を予測できるかどうかを検討してください。スケーリングメトリクスがスケーラブルターゲットの容量に比例して増減する場合は、代わりにターゲット追跡スケーリングポリシーを使用することをお勧めします。より高度な設定には、追加ポリシーとしてステップスケーリングを使用するオプションがあります。例えば、必要に応じて、使用率が一定のレベルに達したときにより積極的なレスポンスを設定できます。
- フラッピングを防ぐために、スケールアウトとスケールインのしきい値の間には適切なマージンを選択してください。フラッピングは、スケールインとスケールアウトの無限ループです。つまり、スケーリングアクションが実行されると、メトリクス値が変化して、逆方向に別のスケーリングアクションが開始されます。

関連リソース

Auto Scaling グループのステップスケーリングポリシーの作成の詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling のステップおよび簡易スケーリングポリシー」を参照してください。

コンソールアクセス

スケーラブルリソースに対するステップスケーリングポリシーを表示、追加、更新、削除するためのコンソールアクセスは、使用するリソースによって異なります。詳細については、「AWS のサービス Application Auto Scaling で使用できる」を参照してください。

を使用して Application Auto Scaling のステップスケーリングポリシーを作成する AWS CLI

この例では、 AWS CLI コマンドを使用して Amazon ECS サービスのステップスケーリングポリシーを作成します。別のスケーラブルターゲットについては、`--service-namespace` でその名前空間、`--scalable-dimension` でそのスケーラブルディメンション、`--resource-id` でそのリソース ID を指定します。

を使用する場合は AWS CLI、 プロファイル用に AWS リージョン 設定された でコマンドが実行されることに注意してください。別のリージョンでコマンドを実行する場合は、プロファイルのデフォルトのリージョンを変更するか、コマンドに `--region` パラメータを使用します。

タスク

- ステップ 1: スケーラブルなターゲットを登録する

- ステップ 2: ステップスケーリングポリシーを作成する
- ステップ 3: スケーリングポリシーを呼び出すアラームを作成する

ステップ 1: スケーラブルなターゲットを登録する

まだ登録していない場合は、スケーラブルターゲットを登録します。register-scalable-target コマンドを使用して、ターゲットサービス内の特定のリソースをスケーラブルターゲットとして登録します。以下の例は、Amazon ECS サービスを Application Auto Scaling に登録します。Application Auto Scaling は、タスクの数を最小 2 タスク、および最大 10 タスクにスケールできます。各*##########*を独自の情報に置き換えます。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service \
  --min-capacity 2 --max-capacity 10
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace ecs ^
  --scalable-dimension ecs:service:DesiredCount ^
  --resource-id service/my-cluster/my-service ^
  --min-capacity 2 --max-capacity 10
```

Output

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。以下は出力例です。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

ステップ 2: ステップスケーリングポリシーを作成する

スケーラブルターゲットのステップスケーリングポリシーを作成するには、次の例を使用して開始できます。

Scale out

スケールアウト (容量増加) 用のステップスケーリングポリシーを作成する

1. 次の cat コマンドを使用して、ステップスケーリングポリシー設定をホームディレクトリの config.json という名前の JSON ファイルに保存します。以下は、調整タイプが PercentChangeInCapacity のサンプル設定で、以下のステップ調整値に基づいてスケーラブルターゲットの容量を増加させます (CloudWatch アラームしきい値を 70 とした場合)。

   - メトリクスの値が 70 パーセント以上、85 パーセント未満の場合は容量を 10 パーセント増やします。
   - メトリクスの値が 85 パーセント以上、95 パーセント未満の場合は容量を 20 パーセント増やします。
   - メトリクスの値が 95 パーセント以上の場合は容量を 30 パーセント増やします。

```
$ cat ~/config.json
{
  "AdjustmentType": "PercentChangeInCapacity",
  "MetricAggregationType": "Average",
  "Cooldown": 60,
  "MinAdjustmentMagnitude": 1,
  "StepAdjustments": [
    {
      "MetricIntervalLowerBound": 0.0,
      "MetricIntervalUpperBound": 15.0,
      "ScalingAdjustment": 10
    },
    {
      "MetricIntervalLowerBound": 15.0,
      "MetricIntervalUpperBound": 25.0,
      "ScalingAdjustment": 20
    },
    {
      "MetricIntervalLowerBound": 25.0,
      "ScalingAdjustment": 30
    }
  ]
}
```

詳細については、Application Auto Scaling API リファレンスの StepScalingPolicyConfiguration を参照してください。

2. 作成した config.json ファイルと共に以下の put-scaling-policy コマンドを使用して、my-step-scaling-policy という名前のスケーリングポリシーを作成します。

Linux、macOS、または Unix

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service \
  --policy-name my-step-scaling-policy --policy-type StepScaling \
  --step-scaling-policy-configuration file://config.json
```

Windows

```
aws application-autoscaling put-scaling-policy --service-namespace ecs ^
  --scalable-dimension ecs:service:DesiredCount ^
  --resource-id service/my-cluster/my-service ^
  --policy-name my-step-scaling-policy --policy-type StepScaling ^
  --step-scaling-policy-configuration file://config.json
```

Output

出力には、ポリシーの一意の名前となる ARN が含まれます。ポリシーの CloudWatch アラームを作成する場合に必要です。以下は出力例です。

```
{
    "PolicyARN":
 "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:resource/ecs/service/my-cluster/my-service:policyName/my-step-scaling-policy"
}
```

Scale in

スケールイン (容量減少) 用のステップスケーリングポリシーを作成する

1. 次の cat コマンドを使用して、ステップスケーリングポリシー設定をホームディレクトリの config.json という名前の JSON ファイルに保存します。以下は、調整タイプが ChangeInCapacity のサンプルステップ設定で、以下のステップ調整値に基づいてスケーラブルターゲットの容量を減少させます (CloudWatch アラームしきい値を 50 とした場合)。

   - メトリクスの値が 50% 以下、40% 超の場合は容量を 1 減らします。
   - メトリクスの値が 40% 以下、30% 超の場合は容量を 2 減らします。
   - メトリクスの値が 30% 以下の場合は容量を 3 減らします。

```
$ cat ~/config.json
{
  "AdjustmentType": "ChangeInCapacity",
  "MetricAggregationType": "Average",
  "Cooldown": 60,
  "StepAdjustments": [
    {
      "MetricIntervalUpperBound": 0.0,
      "MetricIntervalLowerBound": -10.0,
      "ScalingAdjustment": -1
    },
    {
      "MetricIntervalUpperBound": -10.0,
      "MetricIntervalLowerBound": -20.0,
      "ScalingAdjustment": -2
    },
    {
      "MetricIntervalUpperBound": -20.0,
      "ScalingAdjustment": -3
    }
  ]
}
```

   詳細については、Application Auto Scaling API リファレンスの StepScalingPolicyConfiguration を参照してください。

2. 作成した config.json ファイルと共に以下の put-scaling-policy コマンドを使用して、my-step-scaling-policy という名前のスケーリングポリシーを作成します。

Linux、macOS、または Unix

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service \
  --policy-name my-step-scaling-policy --policy-type StepScaling \
  --step-scaling-policy-configuration file://config.json
```

Windows

```
aws application-autoscaling put-scaling-policy --service-namespace ecs ^
  --scalable-dimension ecs:service:DesiredCount ^
  --resource-id service/my-cluster/my-service ^
  --policy-name my-step-scaling-policy --policy-type StepScaling ^
  --step-scaling-policy-configuration file://config.json
```

Output

出力には、ポリシーの一意の名前となる ARN が含まれます。この ARN は、ポリシーの CloudWatch アラームを作成する場合に必要です。以下は出力例です。

```
{
    "PolicyARN":
 "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:resource/ecs/service/my-cluster/my-service:policyName/my-step-scaling-policy"
}
```

ステップ 3: スケーリングポリシーを呼び出すアラームを作成する

最後に、以下の CloudWatch put-metric-alarm コマンドを使用して、ステップスケーリングポリシーで使用するアラームを作成します。この例では、CPU の平均利用率に基づくアラームもあります。アラームは、少なくとも 2 つの連続する 60 秒の評価期間に 70 パーセントのしきい値に達した場合に、ALARM 状態となるよう設定されます。別の CloudWatch メトリクスを指定する、または独自のカスタムメトリクスを使用するには、`--metric-name` でその名前を指定し、`--namespace` でその名前空間を指定します。

Linux、macOS、または Unix

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-ECS:service/my-
cluster/my-service \
  --metric-name CPUUtilization --namespace AWS/ECS --statistic Average \
  --period 60 --evaluation-periods 2 --threshold 70 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions Name=ClusterName,Value=default Name=ServiceName,Value=sample-app-service
 \
  --alarm-actions PolicyARN
```

Windows

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-ECS:service/my-
cluster/my-service ^
  --metric-name CPUUtilization --namespace AWS/ECS --statistic Average ^
  --period 60 --evaluation-periods 2 --threshold 70 ^
  --comparison-operator GreaterThanOrEqualToThreshold ^
  --dimensions Name=ClusterName,Value=default Name=ServiceName,Value=sample-app-service
 ^
  --alarm-actions PolicyARN
```

を使用して Application Auto Scaling のステップスケーリングポリシーを記述する AWS CLI

以下の describe-scaling-policies コマンドを使用して、サービス名前空間に対するすべてのスケーリングポリシーを記述することができます。次の例では、すべての Amazon ECS サービスのすべてのスケーリングポリシーについて説明します。特定の Amazon ECS サービス用にリストするには、`--resource-id` オプションのみを追加します。

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs
```

`--query` パラメータを使用して、結果をステップスケーリングポリシーのみにフィルタリングすることができます。query 用の構文の詳細については、AWS Command Line Interface ユーザーガイドの「AWS CLIからのコマンド出力の制御」を参照してください。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs \
  --query 'ScalingPolicies[?PolicyType==`StepScaling`]'
```

Windows

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs ^
  --query "ScalingPolicies[?PolicyType==`StepScaling`]"
```

Output

以下は出力例です。

```
[
    {
        "PolicyARN": "PolicyARN",
        "StepScalingPolicyConfiguration": {
            "MetricAggregationType": "Average",
            "Cooldown": 60,
            "StepAdjustments": [
                {
                    "MetricIntervalLowerBound": 0.0,
                    "MetricIntervalUpperBound": 15.0,
                    "ScalingAdjustment": 1
                },
                {
                    "MetricIntervalLowerBound": 15.0,
                    "MetricIntervalUpperBound": 25.0,
                    "ScalingAdjustment": 2
                },
                {
                    "MetricIntervalLowerBound": 25.0,
                    "ScalingAdjustment": 3
                }
            ],
            "AdjustmentType": "ChangeInCapacity"
        },
        "PolicyType": "StepScaling",
        "ResourceId": "service/my-cluster/my-service",
        "ServiceNamespace": "ecs",
        "Alarms": [
            {
                "AlarmName": "Step-Scaling-AlarmHigh-ECS:service/my-cluster/my-
service",
                "AlarmARN": "arn:aws:cloudwatch:region:012345678910:alarm:Step-Scaling-
AlarmHigh-ECS:service/my-cluster/my-service"
            }
```

```
        ],
        "PolicyName": "my-step-scaling-policy",
        "ScalableDimension": "ecs:service:DesiredCount",
        "CreationTime": 1515024099.901
    }
]
```

を使用して Application Auto Scaling のステップスケーリングポリシーを削除する AWS CLI

不要になったステップのスケーリングポリシーは削除できます。スケーリングポリシーと CloudWatch アラームの両方を削除するには、以下のタスクを完了します。

スケーリングポリシーを削除する

delete-scaling-policy コマンドを使用します。

Linux、macOS、または Unix

```
aws application-autoscaling delete-scaling-policy --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service \
  --policy-name my-step-scaling-policy
```

Windows

```
aws application-autoscaling delete-scaling-policy --service-namespace ecs ^
  --scalable-dimension ecs:service:DesiredCount ^
  --resource-id service/my-cluster/my-service ^
  --policy-name my-step-scaling-policy
```

CloudWatch アラームを削除する

delete-alarms コマンドを使用します。1 つ以上のアラームを一度に削除することができます。例えば、次のコマンドを使用して Step-Scaling-AlarmHigh-ECS:service/my-cluster/my-service アラームおよび Step-Scaling-AlarmLow-ECS:service/my-cluster/my-service アラームを削除します。

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-ECS:service/my-
cluster/my-service Step-Scaling-AlarmLow-ECS:service/my-cluster/my-service
```

Application Auto Scaling の予測スケーリング

予測スケーリングは、アプリケーションをプロアクティブにスケーリングします。予測スケーリングは、過去の負荷データを分析して、トラフィックフローの日次または週次のパターンを検出します。この情報を使用して、予想される負荷に合わせてアプリケーションの容量を事前に増やすために、将来の容量のニーズを予測します。

予測スケーリングは、次のような状況に適しています。

- 通常の営業時間にはリソースの使用率が高く、夜間や週末はリソースの使用率が低いといったサイクルがあるトラフィック
- バッチ処理、テスト、定期的なデータ分析など、繰り返し発生するon-and-offワークロードパターン。
- 初期化に時間がかかり、スケールアウトイベント中のアプリケーションのパフォーマンスにレイテンシーが顕著な影響を与えるアプリケーション

内容

Application Auto Scaling 予測スケーリングの仕組み

予測スケーリングを使用するには、モニタリングおよび分析する CloudWatch メトリクスを指定する予測スケーリングポリシーを作成します。事前定義されたメトリクスまたはカスタムメトリクスを使用できます。予測スケーリングが将来の値の予測を開始するには、このメトリクスに 24 時間以上のデータが必要です。

ポリシーを作成すると、予測スケーリングは最長過去 14 日間のメトリクスデータの分析を開始し、パターンを特定します。この分析を使用して、今後 48 時間のキャパシティ必要量の時間ごとの予測を生成します。予測は、最新の CloudWatch データを使用して 6 時間ごとに更新されます。新しいデータを取得すると、予測スケーリングは将来の予測の正確性を継続的に向上させることができます。

まず、予測のみモードで予測スケーリングを有効にできます。このモードでは、キャパシティ予測を生成しますが、実際にはそれらの予測に基づいてキャパシティをスケーリングしません。これにより、予測の正確性と適合性を評価できます。

予測データを確認し、そのデータに基づいてスケーリングを開始することを決定したら、スケーリングポリシーを予測とスケーリングのモードに切り替えます。このモードでは、次のようになります。

- 予測で負荷の増加が予想される場合、予測スケーリングは容量を増やします。
- 予測で負荷の減少が予想される場合、予測スケーリングは容量を削除するためにスケールインしません。これにより、予測だけでなく、需要が実際に減少した場合にのみスケールインできます。不要になった容量を削除するには、リアルタイムメトリクスデータに応答するため、ターゲット追跡ポリシーまたは Step Scaling ポリシーを作成する必要があります。

デフォルトでは、予測スケーリングは、その時間の予測に基づいて、各時間の開始時にスケーラブルなターゲットをスケーリングします。オプションで、PutScalingPolicyAPI オペレーションで SchedulingBufferTimeプロパティを使用して、より早い開始時間を指定できます。これにより、予測された需要よりも前に予測されたキャパシティを起動できるため、新しいキャパシティはトラフィックを処理する準備が整います。

最大キャパシティの制限

デフォルトでは、スケーリングポリシーが設定されている場合、最大キャパシティを超えてキャパシティを増やすことはできません。

または、予測容量がスケーラブルターゲットの最大容量に近づいた場合、または超えた場合に、スケーラブルターゲットの最大容量を自動的に増やすことができます。この動作を有効にするには、PutScalingPolicy API オペレーションの MaxCapacityBreachBehavior および MaxCapacityBuffer プロパティ、または AWS Management Consoleの [最大キャパシティーの動作] 設定を使用します。

> **Warning**
>
> 最大キャパシティを自動的に増やす場合は注意してください。最大キャパシティは、自動的に元の最大キャパシティまで減少しません。

スケーリングポリシーの作成、管理、および削除用によく使用されるコマンド

予測スケーリングポリシーを操作するために一般的に使用されるコマンドは次のとおりです。

- `register-scalable-target` AWS またはカスタムリソースをスケーラブルターゲットとして登録し、スケーリングを停止し、スケーリングを再開します。
- `put-scaling-policy` 予測スケーリングポリシーを作成します。
- `get-predictive-scaling-forecast` 予測スケーリングポリシーの予測データを取得するには、 を使用します。
- `describe-scaling-activities` は、 でのスケーリングアクティビティに関する情報を返します AWS リージョン。
- `describe-scaling-policies` は、 のスケーリングポリシーに関する情報を返します AWS リージョン。
- `delete-scaling-policy` スケーリングポリシーを削除します。

カスタムメトリクス

カスタムメトリクスを使用して、アプリケーションに必要な容量を予測できます。カスタムメトリクスは、事前定義されたメトリクスだけではアプリケーションの負荷をキャプチャできない場合に便利です。

考慮事項

予測スケーリングを使用する場合は、以下の考慮事項が適用されます。

- 予測スケーリングがアプリケーションに適しているかどうかを確認します。アプリケーションは、曜日または時刻に固有の定期的な負荷パターンを示す場合、予測スケーリングに適しています。予測スケーリングがアプリケーションをアクティブにスケーリングする前に、予測を評価します。
- 予測スケーリングでは、予測を開始するには 24 時間以上の履歴データが必要です。ただし、履歴データが 2 週間あれば予測がより効果的です。
- アプリケーションのすべての負荷を正確に表し、スケーリングが最も重要なアプリケーションの側面である負荷メトリクスを選択します。

Application Auto Scaling の予測スケーリングポリシーを作成する

次のポリシー例では AWS CLI 、 を使用して Amazon ECS サービスの予測スケーリングポリシーを設定します。各##############を独自の情報に置き換えます。

指定できる CloudWatch メトリクスの詳細については、「Amazon EC2 Auto Scaling API リファレンス」の「PredictiveScalingMetricSpecification」を参照してください。

以下は、メモリ設定が事前定義されたポリシーの例です。

```
cat policy.json
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ECSServiceMemoryUtilization"
            }
        }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
}
```

以下の例は、設定ファイルを指定し put-scaling-policy コマンドを実行して、ポリシーを作成する方法を示しています。

```
aws aas put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json
```

成功した場合、このコマンドはポリシーの ARN を返します。

```
{
```

```
"PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
"Alarms": []
}
```

予定されたアクションを使用して予測値を上書きする

予測計算では考慮できない将来のアプリケーション要件に関する追加情報がある場合があります。例えば、予測の計算では、今後のマーケティングイベントに必要なキャパシティーが過小評価される可能性があります。スケジュールされたアクションを使用して、将来の期間中の予測を一時的に上書きできます。スケジュールされたアクションは、繰り返し実行することも、1 回限りの需要変動がある特定の日時に実行することもできます。

例えば、予測されるキャパシティーを超える最小キャパシティーでスケジュールされたアクションを作成できます。実行時に、Application Auto Scaling はスケーラブルターゲットの最小容量を更新します。予測スケーリングはキャパシティーを最適化するので、予測値を超える最小キャパシティーでスケジュールされたアクションが適用されます。これにより、キャパシティーが想定より少なくなるのを防ぎます。予測の上書きを停止するには、2 番目のスケジュールされたアクションを使用して、最小キャパシティーを元の設定に戻します。

次の手順では、将来の期間中の予測を上書きするステップを示します。

トピック

- ステップ 1: (オプション) 時系列データを分析する
- ステップ 2: 2 つのスケジュールされたアクションを作成する

> **Important**
>
> このトピックでは、予測を上書きして、予測よりも大きなキャパシティにスケールしようとしていることを前提としています。予測スケーリングポリシーの干渉なしに一時的にキャパシティを減らす必要がある場合は、代わりに予測のみモードを使用します。予測のみモードでは、予測スケーリングは予測を生成し続けますが、自動的にキャパシティを増やすことはありません。その後、リソース使用率をモニタリングし、必要に応じてグループのサイズを手動で減らすことができます。

ステップ 1: (オプション) 時系列データを分析する

まず、予測時系列データを分析します。これはオプションのステップですが、予測の詳細を理解したい場合に役立ちます。

1. 予測を取得する

   予測が作成されたら、予測の特定の期間をクエリできます。このクエリの目的は、特定の期間の時系列データの完全なビューを取得することです。

   クエリには、将来の予測データを最大 2 日間含めることができます。予測スケーリングをしばらく使用している場合は、過去の予測データにアクセスすることもできます。ただし、開始時刻と終了時刻の間の最大期間は 30 日間です。

   予測を取得するには、get-predictive-scaling-forecast コマンドを使用します。次の の例では、Amazon ECS サービスの予測スケーリング予測を取得します。

   ```
   aws application-autoscaling get-predictive-scaling-forecast --service-namespace ecs
    \
       --scalable-dimension ecs:service:DesiredCount \
       --resource-id 1234567890abcdef0
       --policy-name predictive-scaling-policy \
       --start-time "2021-05-19T17:00:00Z" \
       --end-time "2021-05-19T23:00:00Z"
   ```

   レスポンスには、 LoadForecastと の 2 つの予測CapacityForecastが含まれます。LoadForecastは、時間単位の負荷予測を示します。 は、指定された を維持しながら、予測負荷を処理するために必要な容量の予測値を時間単位でCapacityForecast表示しますTargetValue。

2. ターゲット期間を特定する

   1 回限りの需要変動が発生する時間または時間範囲を特定します。予測に表示される日付と時刻は UTC であることに注意してください。

ステップ 2: 2 つのスケジュールされたアクションを作成する

次に、アプリケーションの負荷が予測を上回る特定の期間に、2 つのスケジュールされたアクションを作成します。例えば、マーケティングイベントで一時的にトラフィックがサイトに流入する場合

は、1 回限りのアクションをスケジュールして、開始時に最小キャパシティーを更新できます。次に、イベント終了時に最小キャパシティーを元の設定に戻す別のアクションをスケジュールします。

1 回限りのイベントに対して 2 つのスケジュールされたアクションを作成するには (AWS CLI)

スケジュールされたアクションを作成するには、put-scheduled-action コマンドを使用します。

次の例では、Amazon EC2 Auto Scaling のスケジュールを定義し、5 月 19 日の午後 5 時に 8 時間、3 つのインスタンスの最小容量を維持します。以下のコマンドは、このシナリオを実装する方法を示しています。

最初の put-scheduled-update-group-action コマンドは、2021 年 5 月 19 日の午後 5 時 (UTC) に指定された Auto Scaling グループの最小キャパシティーを更新するように Amazon EC2 Auto Scaling に指示します。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-start \
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-capacity 3
```

2 番目のコマンドは、2021 年 5 月 20 日の午前 1 時 (UTC) にグループの最小キャパシティーを 1 に設定するように Amazon EC2 Auto Scaling に指示します。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end \
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-capacity 1
```

これらのスケジュールされたアクションを Auto Scaling グループに追加すると、Amazon EC2 Auto Scaling は次の処理を実行します。

- 2021 年 5 月 19 日の午後 5 時 (UTC) に、最初にスケジュールされたアクションが実行されます。グループのインスタンスが 3 未満である場合、グループは 3 インスタンスにスケールアウトされます。この時刻以降の 8 時間の間、予測キャパシティーが実際のキャパシティーよりも大きい場合、または動的スケーリングポリシーが有効な場合、Amazon EC2 Auto Scaling は引き続きスケールアウトできます。
- 2021 年 5 月 20 日の午前 1 時 (UTC) に、2 番目のスケジュールされたアクションが実行されます。これにより、イベントの終了時に最小キャパシティーが元の設定に戻ります。

繰り返し起こるスケジュールに基づくスケーリング

毎週同じ期間の予測を上書きするには、2 つのスケジュールされたアクションを作成し、cron 式を使用して日時のロジックを指定します。

この cron 式のフォーマットは、スペースで区切られた 5 つのフィールド ([分] [時間] [日] [月] [曜日]) で構成されます。フィールドには、特殊文字を含む任意の許容される値を含めることができます。

例えば、次の cron 式は、毎週火曜日の午前 6:30 にアクションを実行します。アスタリスクは、フィールドのすべての値を照合するワイルドカードとして使用されます。

```
30 6 * * 2
```

カスタムメトリクスを使用した高度な予測スケーリングポリシー

予測スケーリングポリシーでは、事前定義されたメトリクスまたはカスタムメトリクスを使用できます。カスタムメトリクスは、事前定義されたメトリクスがアプリケーションの負荷を十分に説明していない場合に役立ちます。

カスタムメトリクスを使用して予測スケーリングポリシーを作成するときは、 が提供する他の CloudWatch メトリクスを指定するか AWS、自分で定義して公開するメトリクスを指定できます。メトリクス数式を使用して、既存のメトリクスを集計し、自動的に追跡 AWS されない新しい時系列に変換することもできます。新しい合計や平均の計算など、データの値を組み合わせることを、集計すると言います。結果のデータは集計と言います。

以下のセクションには、ポリシー用の JSON 構造を構築する方法のベストプラクティスと例が記載されています。

トピック

ベストプラクティス

次のベストプラクティスは、カスタムメトリクスをより効果的に使用するのに役立ちます。

- 負荷メトリクスの仕様では、最も有用なメトリクスは、アプリケーションの負荷を表すメトリクスです。
- スケーリングメトリクスは、キャパシティーに反比例する必要があります。つまり、スケーラブルターゲットが増加すると、スケーリングメトリクスはほぼ同じ割合で減少します。予測スケーリングが期待どおりに動作するようにするには、負荷メトリクスとスケーリングメトリクスに強い相関がある必要もあります。
- ターゲット使用率は、スケーリングメトリクスのタイプと一致する必要があります。CPU 使用率を使用するポリシー設定の場合、これはパーセンテージのターゲットです。リクエスト数やメッセージ数など、スループットを使用するポリシー設定の場合、これは任意の 1 分間のインスタンスあたりのリクエスト数やメッセージ数のターゲットです。
- これらの推奨事項に従わない場合、予測される将来の時系列の値は、多くの場合、誤りになります。データが正しいことを確認するには、予測値を表示できます。または、予測スケーリングポリシーを作成した後、GetPredictiveScalingForecast API を呼び出して返された `LoadForecast` および `CapacityForecast` オブジェクトを検査します。
- 予測スケーリングがキャパシティーのアクティブスケーリングを開始する前に予測を評価できるように、予測のみモードで予測スケーリングを設定することを強くお勧めします。

前提条件

予測スケーリングポリシーにカスタムメトリクスを追加するには、`cloudwatch:GetMetricData` 許可が必要です。

AWS が提供するメトリクスの代わりに独自のメトリクスを指定するには、まずメトリクスを CloudWatch に公開する必要があります。詳細については、「Amazon CloudWatch ユーザーガイド」の「カスタムメトリクスの発行」を参照してください。

独自のメトリクスを発行するときは、少なくとも 5 分間隔の頻度でデータポイントを発行するようにしてください。Application Auto Scaling は、必要な期間の長さに基づいて CloudWatch からデータポイントを取得します。例えば、負荷メトリクスの指定では、時間単位のメトリクスを使用してアプリケーションの負荷を測定します。CloudWatch は、発行されたメトリクスデータを使用して、各 1 時間の期間内にタイムスタンプがあるすべてのデータポイントを集計することにより、各 1 時間の期間に対して単一のデータ値を提供します。

カスタムメトリクス用の JSON の構築

次のセクションでは、CloudWatch for Amazon EC2 Auto Scaling からデータをクエリするように予測スケーリングを設定する方法の例を示します。このオプションの設定には 2 つの異なる手法あ

り、予測スケーリングポリシーの JSON を構築するために使用する形式は、選択される手法の影響を受けます。メトリクス計算を使用する場合は、実行されるメトリクス計算に基づいて JSON の形式がさらに多様化します。

1. が提供する他の CloudWatch メトリクス AWS または CloudWatch に発行するメトリクスから直接データを取得するポリシーを作成するには、「」を参照してくださいカスタムロードメトリクスとスケーリングメトリクスを使用する予測スケーリングポリシーの例 (AWS CLI)。
2. 複数の CloudWatch メトリクスをクエリし、数式を使用してこれらのメトリクスに基づく新しい時系列を作成できるポリシーを作成するには、「Metric Math 式を使用する」を参照してください。

カスタムロードメトリクスとスケーリングメトリクスを使用する予測スケーリングポリシーの例 (AWS CLI)

を使用してカスタムロードとスケーリングメトリクスを使用して予測スケーリングポリシーを作成するには AWS CLI、 という名前の JSON ファイルに `--predictive-scaling-configuration` の引数を保存しますconfig.json。

カスタムメトリクスの追加は、以下の例にある置き換え可能な値を独自のメトリクスとターゲット使用率に置き換えることによって開始します。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
```

```
            "Stat": "Average"
          }
        }
      ]
    },
    "CustomizedLoadMetricSpecification": {
      "MetricDataQueries": [
        {
          "Id": "load_metric",
          "MetricStat": {
            "Metric": {
              "MetricName": "MyLoadMetric",
              "Namespace": "MyNameSpace",
              "Dimensions": [
                {
                  "Name": "MyOptionalMetricDimensionName",
                  "Value": "MyOptionalMetricDimensionValue"
                }
              ]
            },
            "Stat": "Sum"
          }
        }
      ]
    }
  }
]
}
```

詳細については、「Amazon EC2 Auto Scaling API Reference」 (Amazon EC2 Auto Scaling API リファレンス) の「MetricDataQuery」を参照してください。

> **Note**
>
> 以下は、CloudWatch メトリクスのメトリクス名、名前空間、ディメンション、および統計を見つけるために役立つ追加のリソースです。
>
> - AWS サービスの利用可能なメトリクスの詳細については、「Amazon AWS CloudWatch ユーザーガイド」の「CloudWatch メトリクスを発行するサービス」を参照してください。 Amazon CloudWatch

- を使用して CloudWatch メトリクスの正確なメトリクス名、名前空間、ディメンション (該当する場合) を取得するには AWS CLI、「list-metrics」を参照してください。

このポリシーを作成するには、以下の例にあるように、JSON ファイルを入力として使用して put-scaling-policy コマンドを実行します。

```
aws autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

成功した場合、このコマンドはポリシーの Amazon リソースネーム (ARN) を返します。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
  "Alarms": []
}
```

Metric Math 式を使用する

以下のセクションには、ポリシーでメトリクス計算を使用する方法を説明する予測スケーリングポリシーの情報と例が記載されています。

トピック

- Metric Math について
- メトリクス数式を使用してメトリクスを組み合わせた Amazon EC2 Auto Scaling の予測スケーリングポリシーの例 (AWS CLI)
- ブルー/グリーンデプロイシナリオで使用する予測スケーリングのポリシーの例 (AWS CLI)

Metric Math について

既存のメトリクスデータの集計だけを行いたい場合は、CloudWatch Metric Math により、別のメトリクスを CloudWatch に発行する手間とコストを節約できます。 AWS が提供する任意のメトリクスを使用できます。また、アプリケーションの一部として定義したメトリクスを使用することもできます。

詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch メトリクス数学の使用」を参照してください。

予測スケーリングポリシーで Metric Math の数式を使用する場合は、次の点を考慮してください。

- Metric Math 演算では、メトリクスのメトリクス名、名前空間、ディメンションのキーと値のペアの一意の組み合わせのデータポイントを使用します。
- 任意の算術演算子 (+ - * / ^)、統計関数 (AVG や SUM など)、または CloudWatch がサポートするその他の関数を使用できます。
- 数式の関係式では、メトリクスと他の数式の結果の両方を使用できます。
- Metric Math の数式は、さまざまな集計で構成できます。ただし、最終的な集計結果として、`Average` をスケーリングメトリクスに使用し、`Sum` を負荷メトリクスに使用するのがベストプラクティスです。
- メトリクスの指定で使用される数式はすべて、最終的に単一の時系列を返す必要があります。

Metric Math を使用するには、次の操作を実行します。

- 1 つまたは複数の CloudWatch メトリクスを選択します。次に、数式を作成します。詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch メトリクス数学の使用」を参照してください。
- CloudWatch コンソールまたは CloudWatch GetMetricData API を使用して、Metric Math の数式が有効であることを確認します。

メトリクス数式を使用してメトリクスを組み合わせた Amazon EC2 Auto Scaling の予測スケーリングポリシーの例 (AWS CLI)

場合によっては、メトリクスを直接指定するのではなく、まず何らかの方法でそのデータを処理する必要がある場合があります。例えば、Amazon SQS キューから作業を取り出すアプリケーションがあり、キュー内の項目数を予測スケーリングの基準として使用したいとします。キューにあるメッセージの数だけでは、必要なインスタンスの数は定義されません。インスタンスごとのバックログを計算するために使用できるメトリクスを作成するには、さらに多くの作業が必要です。

このシナリオの予測スケーリングポリシー例を次に示します。Amazon SQS `ApproximateNumberOfMessagesVisible` メトリクス (キューから取得可能なメッセージの数) に基づくスケーリングメトリクスおよび負荷メトリクスを指定します。Amazon EC2 Auto Scaling `GroupInServiceInstances` メトリクスと、スケーリングメトリクスのインスタンスごとのバックログを計算するための数式も使用します。

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
```

```
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                  {
                    "Name": "AutoScalingGroupName",
                    "Value": "my-asg"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
```

```
          {
            "Label": "Calculate the backlog per instance",
            "Id": "scaling_metric",
            "Expression": "queue_size / running_capacity",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ],
              },
              "Stat": "Sum"
            },
            "ReturnData": true
          }
        ]
      }
    }
  ]
}
```

この例では、ポリシーの ARN が返されます。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}
```

ブルー/グリーンデプロイシナリオで使用する予測スケーリングのポリシーの例 (AWS CLI)

検索式には、複数の Auto Scaling グループからメトリクスをクエリし、それらに対して数式を実行できる高度なオプションが用意されています。これは、Blue/Green デプロイで特に有用です。

> **Note**
>
> Blue/Green デプロイ とは、同一の Auto Scaling グループを 2 つ別々に作成するデプロイ方法です。本番トラフィックを受信するグループは 1 つだけです。ユーザートラフィックは、最初は以前の (「青」の) Auto Scaling グループに送信され、新しいグループ (「緑」) はアプリケーションまたはサービスの新しいバージョンのテストと評価に使用されます。新しいデプロイがテストされ、合格すると、ユーザートラフィックは緑の Auto Scaling グループに送信されるようになります。デプロイが成功したら、青のグループを削除できます。

Blue/Green デプロイの一部として新しい Auto Scaling グループが作成されると、メトリクスの指定を変更する必要なく、各グループのメトリクス履歴を自動的に予測スケーリングポリシーに含めることができます。詳細については、 AWS コンピューティングブログの「ブルー/グリーンデプロイでの EC2 Auto Scaling 予測スケーリングポリシーの使用」を参照してください。

次のポリシー例で、これをどのように実行できるかを示します。この例では、ポリシーで、Amazon EC2 が出力する `CPUUtilization` メトリクスを使用します。Amazon EC2 Auto Scaling `GroupInServiceInstances` メトリクスとインスタンスごとのスケーリングメトリクスの値を計算するための数式を使用します。また、キャパシティーメトリック仕様を指定して、`GroupInServiceInstances` メトリクスを取得します。

検索式は、指定した検索条件に基づいて、複数の Auto Scaling グループ内のインスタンスの `CPUUtilization` を検索します。後で同じ検索条件に一致する新しい Auto Scaling グループを作成すると、新しい Auto Scaling グループのインスタンスの `CPUUtilization` が、自動的に含まれます。

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-
policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
      "CustomizedScalingMetricSpecification": {
```

```
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 3600))"
          }
        ]
      },
      "CustomizedCapacityMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))"
          }
        ]
      }
    }
  ]
}
```

この例では、ポリシーの ARN が返されます。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-blue-green-predictive-
scaling-policy",
  "Alarms": []
}
```

予測スケーリングポリシーでのカスタムメトリクスに関する考慮事項

カスタムメトリクスの使用中に問題が発生した場合は、次の操作を実行することをお勧めします。

- エラーメッセージが表示された場合は、メッセージを読み、可能な場合は報告されている問題を解決します。
- 式を事前に検証しなかった場合、put-scaling-policy コマンドはスケーリングポリシーの作成時に式を検証します。ただし、このコマンドでは、検出されたエラーの正確な原因を特定できない可能性があります。問題を解決するには、get-metric-data コマンドへのリクエストからの応答で受け取ったエラーをトラブルシューティングします。CloudWatch コンソールから式をトラブルシューティングすることもできます。
- `MetricDataQueries` で SUM() のような数学関数を使用せずに、独自の SEARCH() 関数を指定する場合、`ReturnData` に `false` を指定する必要があります。これは、検索式が複数の時系列を返す可能性がある一方、数式に基づくメトリクス指定は 1 つの時系列しか返すことができないためです。
- 検索式に含まれるすべてのメトリクスは、同じ解像度である必要があります。

制限

以下の制限が適用されます。

- 1 つのメトリクス指定で最大 10 個のメトリクスのデータポイントをクエリできます。
- この制限に関しては、1 つの式は 1 つのメトリクスとしてカウントされます。

チュートリアル: 大量のワークロードを処理するために自動スケーリングを設定する

このチュートリアルでは、アプリケーションのワークロードが通常よりも多くなる時間枠に基づいてスケールアウトおよびスケールインする方法を学びます。これは、定期的に、または季節に応じて訪問者の数が急増する可能性があるアプリケーションが存在する場合に役立ちます。

追加の負荷を処理するには、ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを併用できます。スケジュールされたスケーリングは、ユーザー指定のスケジュールに基づいて、MinCapacity および MaxCapacity への変更をユーザーに代って自動的に開始します。ターゲット追跡スケーリングポリシーがリソースでアクティブになっていると、新しい最小容量と最大容量の範囲内で、現行のリソース使用率に基づいて動的にスケールすることができます。

このチュートリアルを完了すると、以下を行う方法を理解できます。

- スケジュールされたスケーリングを使用して、高負荷状態になる前にそれらに対応するための容量を追加し、容量が必要なくなったときに削除する。
- ターゲット追跡スケーリングポリシーを使用して、現行のリソース使用率に基づいてアプリケーションをスケールする。

内容

前提条件

このチュートリアルでは、以下を実行済みであることを前提としています。

- を作成しました AWS アカウント。

- をインストールして設定しました AWS CLI。
- スケーラブルターゲットとしてリソースを Application Auto Scaling に登録または登録解除するために必要な許可を取得済みである。また、スケーリングポリシーとスケジュールされたアクションを作成するために必要なすべての許可も取得済みである。詳細については、「Application Auto Scaling の Identity and Access Management」を参照してください。
- 非実稼働環境にこのチュートリアルで使用するためのサポート対象リソースが作成済みである。まだ作成していない場合は、新しく作成してください。Application Auto Scaling と連携する AWS サービスとリソースの詳細については、AWS のサービス Application Auto Scaling で使用できるセクションを参照してください。

> **Note**
>
> このチュートリアルの実行中、 リソースの最小容量と最大容量の値を 0 に設定して、現在の容量を 0 にリセットする 2 つのステップがあります。Application Auto Scaling で使用しているリソースによっては、これらの手順で現在の容量を 0 にリセットできない場合があります。問題に対処しやすくするために、出力のメッセージは、最小容量が指定された値より小さくできないことを示し、 AWS リソースが受け入れることができる最小容量値を提供します。

ステップ 1: スケーラブルターゲットを登録する

スケーラブルターゲットとしてリソースを Application Auto Scaling に登録することから始めます。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。

Application Auto Scaling にスケーラブルなターゲットを登録する

- 以下の register-scalable-target コマンドを使用して、新しいスケーラブルターゲットを登録します。`--min-capacity` および `--max-capacity` の値を 0 に設定して、現行容量を 0 にリセットします。

  `--service-namespace` のサンプルテキストを、Application Auto Scaling で使用している AWS サービスの名前空間、`--scalable-dimension` を登録しているリソースに関連付けられているスケーラブルディメンション、`--resource-id` をリソースの識別子に置き換えます。これらの値は、使用されるリソースとリソース ID の構築方法によって異なります。詳細については、 AWS のサービス Application Auto Scaling で使用できる セクションのトピックを参照し

てください。これらのトピックには、スケーラブルなターゲットを Application Auto Scaling に登録する方法を示すコマンド例が含まれています。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --min-capacity 0 --max-capacity 0
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace namespace
 --scalable-dimension dimension --resource-id identifier --min-capacity 0 --max-
capacity 0
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-
id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

ステップ 2: 要件に従ってスケジュールされたアクションをセットアップする

put-scheduled-action コマンドを使用して、ビジネスニーズを満たすように設定されたスケジュールされたアクションを作成することができます。このチュートリアルでは、容量を 0 に減らすことによって、就業時間外におけるリソースの消費を停止する設定に焦点を当てます。

午前中にスケールアウトするスケジュールされたアクションを作成する

1. スケーラブルターゲットをスケールアウトするには、以下の put-scheduled-action コマンドを使用します。Cron 式を使用して、UTC での定期的なスケジュールが設定された `--schedule` パラメータを含めます。

Application Auto Scaling は、指定されたスケジュール (毎日午前 9:00 (UTC)) に従って、MinCapacity および MaxCapacity の値を希望範囲の 1～5 キャパシティーユニットに更新します。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --scheduled-action-name my-first-scheduled-action \
  --schedule "cron(0 9 * * ? *)" \
  --scalable-target-action MinCapacity=1,MaxCapacity=5
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --scheduled-action-name my-
first-scheduled-action --schedule "cron(0 9 * * ? *)" --scalable-target-action
 MinCapacity=1,MaxCapacity=5
```

このコマンドが正常に完了した場合は、出力が返されません。

2. スケジュールされたアクションが存在することを確認するには、以下の describe-scheduled-actions コマンドを使用します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scheduled-actions \
  --service-namespace namespace \
  --query 'ScheduledActions[?ResourceId==`identifier`]'
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-
namespace namespace --query "ScheduledActions[?ResourceId==`identifier`]"
```

以下は出力例です。

```
[
```

```
    {
        "ScheduledActionName": "my-first-scheduled-action",
        "ScheduledActionARN": "arn",
        "Schedule": "cron(0 9 * * ? *)",
        "ScalableTargetAction": {
            "MinCapacity": 1,
            "MaxCapacity": 5
        },
        ...
    }
]
```

夜間にスケールインするスケジュールされたアクションを作成する

1. 上記の手順を繰り返して、Application Auto Scaling が 1 日の終わりにスケールインするために使用する、別のスケジュールされたアクションを作成します。

   Application Auto Scaling は、以下の put-scheduled-action コマンドの指示通りに、指定されたスケジュール (毎日午後 8:00 (UTC)) に従ってターゲットの `MinCapacity` および `MaxCapacity` を 0 に更新します。

   Linux、macOS、または Unix

   ```
   aws application-autoscaling put-scheduled-action \
     --service-namespace namespace \
     --scalable-dimension dimension \
     --resource-id identifier \
     --scheduled-action-name my-second-scheduled-action \
     --schedule "cron(0 20 * * ? *)" \
     --scalable-target-action MinCapacity=0,MaxCapacity=0
   ```

   Windows

   ```
   aws application-autoscaling put-scheduled-action --service-namespace namespace --
   scalable-dimension dimension --resource-id identifier --scheduled-action-name my-
   second-scheduled-action --schedule "cron(0 20 * * ? *)" --scalable-target-action
    MinCapacity=0,MaxCapacity=0
   ```

2. スケジュールされたアクションが存在することを確認するには、以下の describe-scheduled-actions コマンドを使用します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scheduled-actions \
  --service-namespace namespace \
  --query 'ScheduledActions[?ResourceId==`identifier`]'
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-
namespace namespace --query "ScheduledActions[?ResourceId==`identifier`]"
```

以下は出力例です。

```
[
    {
        "ScheduledActionName": "my-first-scheduled-action",
        "ScheduledActionARN": "arn",
        "Schedule": "cron(0 9 * * ? *)",
        "ScalableTargetAction": {
            "MinCapacity": 1,
            "MaxCapacity": 5
        },
        ...
    },
    {
        "ScheduledActionName": "my-second-scheduled-action",
        "ScheduledActionARN": "arn",
        "Schedule": "cron(0 20 * * ? *)",
        "ScalableTargetAction": {
            "MinCapacity": 0,
            "MaxCapacity": 0
        },
        ...
    }
]
```

ステップ 3: ターゲット追跡スケーリングポリシーを追加する

基本的なスケジュールが設定されたところで、現行のリソース使用率に基づいてスケールするためのターゲット追跡スケーリングポリシーを追加します。

ターゲット追跡では、Application Auto Scaling がポリシーのターゲット値を指定されたメトリクスの現行値と比較します。それらが一定期間同等でなかった場合は、Application Auto Scaling が容量を追加または削除して、安定したパフォーマンスを維持します。アプリケーションに対する負荷とメトリクス値の増加に伴い、Application Auto Scaling は、`MaxCapacity` を超過することなく、可能な限り早急に容量を追加します。負荷が最小限であることを理由に Application Auto Scaling が容量を削除するときは、`MinCapacity` を下回らないように削除します。使用量に基づいて容量を調整することで、料金の支払いがアプリケーションに必要な容量分のみになります。

アプリケーションに負荷がないことが原因でメトリクスに十分なデータがない場合、Application Auto Scaling は容量の追加または削除を行いません。言い換えると、Application Auto Scaling は、十分な情報が利用できない状況では可用性を優先します。

スケーリングポリシーは複数追加できますが、競合するステップスケーリングポリシーは追加しないようにしてください。これらは望ましくない動作の原因となる可能性があります。例えば、ターゲット追跡ポリシーがスケールインする準備が整う前に、ステップスケーリングポリシーがスケールインアクティビティを開始した場合、スケールインアクティビティはブロックされません。ターゲット追跡ポリシーは、スケールインアクティビティの完了後、再度スケールアウトするように Application Auto Scaling に指示できます。

ターゲット追跡スケーリングポリシーを作成する

1. 以下の put-scaling-p コマンドを使用して、ポリシーを作成します。

   ターゲット追跡のために最も頻繁に使用されるメトリクスは事前定義されており、これらは CloudWatch から完全なメトリクス仕様を提供しなくても使用できます。利用可能な事前定義されたメトリクスの詳細については、「Application Auto Scaling のターゲット追跡スケーリングポリシー」を参照してください。

   このコマンドを実行する前に、事前定義されたメトリクスがターゲット値を期待していることを確認してください。例えば、CPU 使用率が 50% に達したときにスケールアウトするには、50.0 のターゲット値を指定します。または、使用量が 70% に達したときに Lambda のプロビジョニングされた同時実行数をスケールアウトするには、0.7 のターゲット値を指定します。特定のリソースのターゲット値に関する情報は、ターゲット追跡の設定方法について、サービス提供のド

キュメントを参照してください。詳細については、「AWS のサービス Application Auto Scaling で使用できる」を参照してください。

Linux、macOS、または Unix

```
aws application-autoscaling put-scaling-policy \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
  --target-tracking-scaling-policy-configuration '{ "TargetValue": 50.0,
 "PredefinedMetricSpecification": { "PredefinedMetricType": "predefinedmetric" }}'
```

Windows

```
aws application-autoscaling put-scaling-policy --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --policy-name my-scaling-
policy --policy-type TargetTrackingScaling --target-tracking-scaling-policy-
configuration "{ \"TargetValue\": 50.0, \"PredefinedMetricSpecification\":
 { \"PredefinedMetricType\": \"predefinedmetric\" }}"
```

正常に完了した場合、このコマンドは、ユーザーに代わって作成された 2 つの CloudWatch アラームの ARN と名前を返します。

2. スケジュールされたアクションが存在することを確認するには、以下の describe-scaling-policies コマンドを使用します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-policies --service-namespace namespace
 \
  --query 'ScalingPolicies[?ResourceId==`identifier`]'
```

Windows

```
aws application-autoscaling describe-scaling-policies --service-namespace namespace
 --query "ScalingPolicies[?ResourceId==`identifier`]"
```

以下は出力例です。

```
[
    {
        "PolicyARN": "arn",
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "predefinedmetric"
            },
            "TargetValue": 50.0
        },
        "PolicyName": "my-scaling-policy",
        "PolicyType": "TargetTrackingScaling",
        "Alarms": [],
        ...
    }
]
```

ステップ 4: 次のステップ

スケーリングアクティビティが発生すると、スケーラブルターゲットのスケーリングアクティビティの出力にそのレコードが表示されます。次に例を示します。

```
Successfully set desired count to 1. Change successfully fulfilled by ecs.
```

Application Auto Scaling を使用してスケーリングアクティビティを監視するには、describe-scaling-activities コマンドを使用できます。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-activities
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier
```

Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace namespace
 --scalable-dimension dimension --resource-id identifier
```

ステップ 5：クリーンアップ

アカウントでアクティブにスケーリングしている最中に作成されたリソースに対する料金が発生しないようにするために、関連付けられたスケーリング設定を以下のようにクリーンアップすることができます。

スケーリング設定を削除しても、基盤となる AWS リソースは削除されません。また、リソースが元の容量に戻されることもありません。リソースの削除、またはその容量の調整は、そのリソースを作成したサービスのコンソールを使用して行うことができます。

スケジュールされたアクションを削除する

以下の delete-scheduled-action コマンドは、指定されているスケールされたアクションを削除します。作成したスケジュールされたアクションを保持したい場合は、このステップをスキップできます。

Linux、macOS、または Unix

```
aws application-autoscaling delete-scheduled-action \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --scheduled-action-name my-second-scheduled-action
```

Windows

```
aws application-autoscaling delete-scheduled-action --service-namespace namespace
 --scalable-dimension dimension --resource-id identifier --scheduled-action-name my-
second-scheduled-action
```

スケーリングポリシーを削除する

以下の delete-scheduled-action コマンドは、指定されたターゲット追跡スケーリングポリシーを削除します。作成したスケーリングポリシーを保持したい場合は、このステップをスキップできます。

Linux、macOS、または Unix

```
aws application-autoscaling delete-scaling-policy \
  --service-namespace namespace \
  --scalable-dimension dimension \
```

```
  --resource-id identifier \
  --policy-name my-scaling-policy
```

Windows

```
aws application-autoscaling delete-scaling-policy --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --policy-name my-scaling-policy
```

スケーラブルなターゲットを登録解除する

以下の deregister-scalable-target コマンドを使用して、スケーラブルターゲットの登録を解除します。自分で作成したスケーリングポリシーや、まだ削除されていないスケジュールされたアクションがある場合は、このコマンドによって削除されます。後で使用できるように、登録されたスケーラブルなターゲットを保持する場合は、このステップをスキップできます。

Linux、macOS、または Unix

```
aws application-autoscaling deregister-scalable-target \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier
```

Windows

```
aws application-autoscaling deregister-scalable-target --service-namespace namespace --
scalable-dimension dimension --resource-id identifier
```

Application Auto Scaling のスケーリングの一時停止と再開

このトピックでは、アプリケーションでスケーラブルターゲットのスケーリングアクティビティの 1 つ、または複数を一時停止し、その後再開する方法について説明します。一時停止/再開機能は、スケーリングポリシーとスケジュールされたアクションによってトリガーされたスケーリングアクティビティを一時的に停止するために使用されます。これは、例えば、変更を行っている間や設定の問題を調査しているときに、自動スケーリングに干渉されないようにする場合などに便利です。スケーリングポリシーとスケジュールされたアクションは保持し、準備が整ったら、スケーリングアクティビティを再開できます。

以下のサンプル CLI コマンドでは、config.json ファイル で JSON 形式のパラメータを渡します。これらのパラメータは、引用符を使用して JSON データ構造を囲むことによって、コマンドラインで渡すこともできます。詳細については、AWS Command Line Interface ユーザーガイドの「AWS CLI での文字列への引用符の使用」を参照してください。

内容

> **Note**
>
> Amazon ECS デプロイの進行中にスケールアウトプロセスを一時停止する手順については、次のドキュメントを参照してください。
> Amazon Elastic Container Service デベロッパーガイドの「サービスの自動スケーリング」

スケーリングアクティビティ

Application Auto Scaling では、以下のスケーリングアクティビティを一時停止状態にすることができます。

- スケーリングポリシーによってトリガーされるすべてのスケールインアクティビティ。
- スケーリングポリシーによってトリガーされるすべてのスケールアウトアクティビティ。
- スケジュールされたアクションに関係するすべてのスケーリングアクティビティ。

以下の説明では、個々のスケーリングアクティビティが停止されると何が起こるかについて説明しています。それぞれ個別に停止および再開できます。スケーリングアクティビティを停止する理由によっては、複数のスケーリングアクティビティをまとめて停止する必要がある場合があります。

`DynamicScalingInSuspended`

- Application Auto Scaling は、ターゲット追跡スケーリングポリシーまたはステップスケーリングポリシーがトリガーされたときに容量を削除しません。これは、スケーリングポリシー、またはそれらに関連する CloudWatch アラームを削除することなく、スケーリングポリシーに関連付けられたスケールインアクティビティを一時的に無効化することを可能にします。スケールインを再開するときは、Application Auto Scaling が違反状態のアラームしきい値があるポリシーを評価します。

`DynamicScalingOutSuspended`

- Application Auto Scaling は、ターゲット追跡スケーリングポリシーまたはステップスケーリングポリシーがトリガーされたときに容量を追加しません。これは、スケーリングポリシー、またはそれらに関連する CloudWatch アラームを削除することなく、スケーリングポリシーに関連付けられたスケールアウトアクティビティを一時的に無効化することを可能にします。スケールアウトを再開するときは、Application Auto Scaling が違反状態のアラームしきい値があるポリシーを評価します。

`ScheduledScalingSuspended`

- Application Auto Scaling は、一時停止期間中に実行がスケジュールされているスケーリングアクションを開始しません。スケジュールされたスケーリングを再開するとき、Application Auto Scaling は、実行時刻がまだ過ぎていないスケジュールされたアクションのみを評価します。

スケーリングアクティビティの一時停止と再開

Application Auto Scaling のスケーラブルターゲットに対するスケーリングアクティビティは、個別に、またはすべてを一時停止して再開することができます。

> **Note**
>
> 簡略化のため、これらの例では、DynamoDB テーブルのスケーリングを一時停止して再開する方法を例示しています。別のスケーラブルターゲットを指定するには、`--service-`

namespace でその名前空間、--scalable-dimension でそのスケーラブルディメンション、--resource-id でそのリソース ID を指定します。各サービスの詳細情報および例については、AWS のサービス Application Auto Scaling で使用できる のトピックを参照してください。

スケーリングアクティビティを停止するには

コマンドラインウィンドウを開き、--suspended-state オプションがある register-scalable-target コマンドを以下のように使用します。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table \
  --suspended-state file://config.json
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table --
suspended-state file://config.json
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

スケーリングポリシーによってトリガーされたスケールインアクティビティのみを停止するには、config.json で次のように指定します。

```
{
    "DynamicScalingInSuspended":true
}
```

スケーリングポリシーによってトリガーされたスケールアウトアクティビティのみを停止するには、config.json で次のように指定します。

```
{
    "DynamicScalingOutSuspended":true
}
```

スケジュールされたアクションに関連するスケーリングアクティビティのみを停止するには、config.json で以下を指定します。

```
{
    "ScheduledScalingSuspended":true
}
```

すべてのスケーリングアクティビティを停止するには

`--suspended-state` オプションがある register-scalable-target コマンドを以下のように使用します。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table \
  --suspended-state file://config.json
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table --
suspended-state file://config.json
```

この例では、ファイル config.json に以下の JSON 形式パラメータが含まれていると仮定しています。

```
{
    "DynamicScalingInSuspended":true,
    "DynamicScalingOutSuspended":true,
    "ScheduledScalingSuspended":true
}
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
```

```
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

一時停止されたスケーリングアクティビティを表示する

describe-scalable-targets コマンドを使用して、スケーラブルターゲットに対するスケーリングアクティビティのどれが一時停止状態になっているかを判断します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scalable-targets --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table
```

Windows

```
aws application-autoscaling describe-scalable-targets --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table
```

以下は出力例です。

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "dynamodb",
            "ScalableDimension": "dynamodb:table:ReadCapacityUnits",
            "ResourceId": "table/my-table",
            "MinCapacity": 1,
            "MaxCapacity": 20,
            "SuspendedState": {
                "DynamicScalingOutSuspended": true,
                "DynamicScalingInSuspended": true,
                "ScheduledScalingSuspended": true
            },
            "CreationTime": 1558125758.957,
            "RoleARN": "arn:aws:iam::123456789012:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable"
        }
    ]
}
```

スケーリングアクティビティを再開する

スケーリングアクティビティを再開する準備が整ったら、register-scalable-target コマンドを使用してそれらを再開できます。

次のコマンド例では、指定されたスケーラブルなターゲットのすべてのスケーリングアクティビティを再開します。

Linux、macOS、または Unix

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table \
  --suspended-state file://config.json
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table --
suspended-state file://config.json
```

この例では、ファイル config.json に以下の JSON 形式パラメータが含まれていると仮定しています。

```
{
    "DynamicScalingInSuspended":false,
    "DynamicScalingOutSuspended":false,
    "ScheduledScalingSuspended":false
}
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Application Auto Scaling のスケーリングアクティビティ

Application Auto Scaling は、スケーリングポリシーの CloudWatch メトリクスを監視し、しきい値を超えるとスケーリングアクティビティを開始します。また、スケーラブルターゲットの最大サイズまたは最小サイズを手動で、またはスケジュールに従って変更すると、スケーリングアクティビティが開始されます。

スケーリングアクティビティが発生すると、Application Auto Scaling は次のいずれかを実行します。

- スケーラブルターゲットの容量を増やします (スケールアウトと呼ばれます)
- スケーラブルターゲットの容量を減らします (スケールインと呼ばれます)

過去 6 週間のスケーリングアクティビティを調べることができます。

スケーラブルターゲットによるスケーリングアクティビティの検索

特定のスケーラブルターゲットに対するスケーリングアクティビティを表示するには次の describe-scaling-activities コマンドを使用します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-activities --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount --resource-id service/my-cluster/my-service
```

Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace ecs --
scalable-dimension ecs:service:DesiredCount --resource-id service/my-cluster/my-service
```

以下はレスポンスの例で、StatusCode にはアクティビティの現在のステータスが、StatusMessage にはスケーリングアクティビティのステータスに関する情報が含まれています。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
```

```
            "Description": "Setting desired count to 1.",
            "ResourceId": "service/my-cluster/my-service",
            "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
            "StartTime": 1462575838.171,
            "ServiceNamespace": "ecs",
            "EndTime": 1462575872.111,
            "Cause": "monitor alarm web-app-cpu-lt-25 in state ALARM triggered policy
 web-app-cpu-lt-25",
            "StatusMessage": "Successfully set desired count to 1. Change successfully
 fulfilled by ecs.",
            "StatusCode": "Successful"
        }
    ]
}
```

レスポンスのフィールドの説明については、「Application Auto Scaling API リファレンス」の「ScalingActivity」を参照してください。

次のステータスコードは、スケーリングアクティビティにつながるスケーリングイベントが完了した状態になったことを示します。

- `Successful` - スケーリングは正常に完了しました
- `Overridden` - 新しいスケーリングイベントにより、希望する容量が更新されました
- `Unfulfilled` - スケーリングがタイムアウトしたか、ターゲットサービスがリクエストを実行できません
- `Failed` - 例外が発生してスケーリングが失敗しました

> **Note**
>
> スケーリングアクティビティは、`Pending` または `InProgress` のステータスである場合もあります。すべてのスケーリングアクティビティには、ターゲットサービスが応答する前に `Pending` ステータスがあります。ターゲットが応答すると、スケーリングアクティビティのステータスは `InProgress` に変わります。

スケーリングされていないアクティビティを含む

デフォルトでは、Application Auto Scaling がスケーリングしないかどうかを決定する時間は、スケーリングアクティビティに反映されません。

例えば、Amazon ECS サービスが特定のメトリクスの最大しきい値を超えているが、タスク数が既に許可されている最大タスク数に達しているとします。この場合、Application Auto Scaling は希望する数のタスクをスケールアウトしません。

スケーリングされていないアクティビティ (スケーリングされたアクティビティでない) をレスポンスに含めるには、describe-scaling-activities コマンドに `--include-not-scaled-activities` オプションを追加します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scaling-activities --include-not-scaled-activities
 \
  --service-namespace ecs --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service
```

Windows

```
aws application-autoscaling describe-scaling-activities --include-not-scaled-activities
 --service-namespace ecs --scalable-dimension ecs:service:DesiredCount --resource-
id service/my-cluster/my-service
```

> **Note**
>
> このコマンドでエラーが発生した場合は、 を AWS CLI ローカルで最新バージョンに更新していることを確認してください。

レスポンスにスケーリングされていないアクティビティが含まれていることを確認するために、失敗したスケーリングアクティビティのすべてではないにしても、一部の NotScaledReasons 要素が出力に表示されます。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Attempting to scale due to alarm triggered",
            "ResourceId": "service/my-cluster/my-service",
            "ActivityId": "4d759079-a31f-4d0c-8468-504c56e2eecf",
            "StartTime": 1664928867.915,
            "ServiceNamespace": "ecs",
```

```
            "Cause": "monitor alarm web-app-cpu-gt-75 in state ALARM triggered policy
 web-app-cpu-gt-75",
            "StatusCode": "Failed",
            "NotScaledReasons": [
                {
                    "Code": "AlreadyAtMaxCapacity",
                    "MaxCapacity": 4
                }
            ]
        }
    ]
}
```

レスポンスのフィールドの説明については、「Application Auto Scaling API リファレンス」の「ScalingActivity」を参照してください。

スケーリングされていないアクティビティが返された場合、Code に記載されている理由コードによっては、CurrentCapacity、MaxCapacity、MinCapacity などの属性がレスポンスに含まれる場合があります。

大量の重複エントリを防ぐために、スケーリングされていない最初のアクティビティのみがスケーリングアクティビティ履歴に記録されます。スケーリングされていない後続のアクティビティは、スケーリングされていない変更についての理由がない限り、新しいエントリを生成しません。

理由コード

以下は、スケーリングされていないアクティビティの理由コードです。

理由コード	定義			
AutoScalingAnticipatedFlapping	自動スケーリングアルゴリズムは、フラッピングの原因となるため、スケーリングアクションを実行しないことを決定しました。フラッ			

理由コード	定義			
	ピングは、スケールインとスケールアウトの無限ループです。つまり、スケーリングアクションが実行されると、メトリクス値が変化して、逆方向に別のスケーリングアクションが開始されます。			
TargetServicePutResourceAsNotScalable	ターゲットサービスが一時的にリソースをスケーリング不可能な状態にしました。Application Auto Scaling は、スケーリングポリシーで指定された自動スケーリング条件が満たされると、再度スケーリングを試みます。			

理由コード	定義			
AlreadyAtMaxCapacity	スケーリングは、指定した最大容量によってブロックされます。Application Auto Scaling をスケールアウトするには、最大容量を増やす必要があります。			
AlreadyAtMinCapacity	スケーリングは、指定した最小容量によってブロックされます。Application Auto Scaling をスケールインさせるには、最小容量を減らす必要があります。			
AlreadyAtDesiredCapacity	Auto Scaling アルゴリズムは、修正後の容量を現在の容量と等しくなるように計算しました。			

Application Auto Scaling をモニタリングする

モニタリングは、Application Auto Scaling やその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、 AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。 は、Application Auto Scaling を監視したり、問題が発生したときに報告したり、必要に応じて自動アクションを実行したりするためのモニタリングツール AWS を提供します。

AWS リソースの管理には、次の機能を使用できます。

AWS CloudTrail

を使用すると AWS CloudTrail、 によって、または に代わって Application Auto Scaling API に対して行われた呼び出しを追跡できます AWS アカウント。CloudTrail は、その情報をログファイルの形で指定した Amazon S3 バケットに格納します。Application Auto Scaling を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出し日時を特定できます。詳細については、「を使用した Application Auto Scaling API コールのログ記録 AWS CloudTrail」を参照してください。

> **Note**
>
> ワークロードに関するデータのログ記録と収集に役立つ他の AWS サービスについては、「 規範ガイダンス」の「アプリケーション所有者のログ記録とモニタリングガイド」を参照してください。 AWS

Amazon CloudWatch

Amazon CloudWatch を使用すると、ログを分析し、リアルタイムで AWS リソースとホストアプリケーションのメトリクスをモニタリングできます。メトリクスを収集および追跡し、カスタマイズされたダッシュボードを作成し、指定されたメトリックが指定したしきい値に達したときに通知またはアクションを実行するアラームを設定できます。例えば、CloudWatch にリソースの使用状況を追跡させて、使用率が非常に高い場合やメトリクスのアラームが `INSUFFICIENT_DATA` 状態になったときに通知するように設定できます。詳細については、「CloudWatch を使用してスケーラブルリソースの使用状況を監視する」を参照してください。

CloudWatch は、Application Auto Scaling の AWS API 使用状況メトリクスも追跡します。これらのメトリクスを使用して、API 呼び出し量が定義したしきい値を超えたときに警告するアラーム

を設定できます。詳細については、Amazon CloudWatch ユーザーガイドの「AWS CloudWatch 使用状況メトリクスの使用」を参照してください。

Amazon EventBridge

Amazon EventBridge は、アプリケーションをさまざまなイベントソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、独自のアプリケーション、Software-as-a-Service (SaaS) アプリケーション、および AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを Lambda などのターゲットにルーティングします。これにより、サービスで発生したイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。詳細については、「Amazon EventBridge を使用して Application Auto Scaling イベントをモニタリングする」を参照してください。

AWS Health Dashboard

AWS Health Dashboard (PHD) は情報を表示し、 AWS リソースの正常性の変化によって呼び出される通知も提供します。情報は 2 つの方法で表示されます。ダッシュボードには、最近のイベントおよび予定されているイベントがカテゴリ別に分類されて表示されます。詳細なイベントログには、過去 90 日間のすべてのイベントが表示されます。詳細については、「Getting started with AWS Health Dashboard」を参照してください。

CloudWatch を使用してスケーラブルリソースの使用状況を監視する

Amazon CloudWatch を使用すると、スケーラブルなリソース全体にわたってアプリケーションをほぼ継続的に可視化できます。CloudWatch は、 AWS リソースのモニタリングサービスです。CloudWatch を使用して、メトリクスの収集と追跡、アラーム設定、 AWS リソースの変更に対する自動的な対応ができます。ダッシュボードを作成して、特定のメトリクスや必要なメトリクスのセットをモニタリングすることもできます。

Application Auto Scaling と統合するサービスとやり取りするときは、サービスが以下の表にあるメトリクスを CloudWatch に送信します。CloudWatch では、メトリクスがまずサービス名前空間ごとにグループ化され、次に各名前空間内のさまざまなディメンションの組み合わせでグループ化されます。これらのメトリクスは、リソース使用量をモニタリングし、アプリケーションの容量を計画するのに役立ちます。アプリケーションのワークロードが一定ではない場合は、自動スケーリングの使用を検討する必要があることを示しています。これらのメトリクスの詳細な説明については、対象となるメトリクスのドキュメントを参照してください。

内容

リソースの使用状況をモニタリングするための CloudWatch メトリクス

次の表は、リソース使用量のモニタリングをサポートするために使用できる CloudWatch メトリクスを示しています。このリストは、すべてを網羅しているわけではありませんが、適切な開始点です。CloudWatch コンソールにこれらのメトリクスが表示されない場合は、リソースのセットアップが完了していることを確認してください。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。

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
AppStream 2.0			
フリート	AWS/AppStream	名前: AvailableCapacity ディメンション: フリート	AppStream 2.0 のメトリクス
フリート	AWS/AppStream	名前: CapacityUtilization ディメンション: フリート	AppStream 2.0 のメトリクス
Aurora			

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
レプリカ	AWS/RDS	名前: CPUUtilization ディメンション: DBClusterIdentifier、ロール (閲覧者)	Amazon Aurora でのクラスターレベルのメトリクス
レプリカ	AWS/RDS	名前: DatabaseConnections ディメンション: DBClusterIdentifier、ロール (閲覧者)	Amazon Aurora でのクラスターレベルのメトリクス
Amazon Comprehend			
ドキュメント分類のエンドポイント	AWS/Comprehend	名前: InferenceUtilization ディメンション: EndpointArn	Amazon Comprehend エンドポイントのメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
エンティティ認識機能のエンドポイント	AWS/Comprehend	名前: InferenceUtilization ディメンション: EndpointArn	Amazon Comprehend エンドポイントのメトリクス
DynamoDB			
テーブルとグローバルセカンダリインデックス	AWS/DynamoDB	名前: ProvisionedReadCapacityUnits ディメンション: TableName、GlobalSecondaryIndexName	DynamoDB のメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
テーブルとグローバルセカンダリインデックス	AWS/DynamoDB	名前: ProvisionedWriteCapacityUnits ディメンション: TableName、GlobalSecondaryIndexName	DynamoDB のメトリクス
テーブルとグローバルセカンダリインデックス	AWS/DynamoDB	名前: ConsumedReadCapacityUnits ディメンション: TableName、GlobalSecondaryIndexName	DynamoDB のメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
テーブルとグローバルセカンダリインデックス	AWS/DynamoDB	名前: ConsumedWriteCapacityUnits ディメンション: TableName、GlobalSecondaryIndexName	DynamoDB のメトリクス
Amazon ECS			
サービス	AWS/ECS	名前: CPUUtilization ディメンション: ClusterName、ServiceName	Amazon ECS のメトリクス
サービス	AWS/ECS	名前: MemoryUtilization ディメンション: ClusterName、ServiceName	Amazon ECS のメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
サービス	AWS/ApplicationELB	名前: RequestCountPerTarget ディメンション: TargetGroup	Application Load Balancer のメトリクス
ElastiCache			
クラスター (レプリケーショングループ)	AWS/ElastiCache	名前: DatabaseMemoryUsageCountedForEvictPercentage ディメンション: ReplicationGroupId	ElastiCache Valkey および Redis OSS メトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
クラスター (レプリケーショングループ)	AWS/ElastiCache	名前: DatabaseCapacityUsageCountedForEvictPercentage ディメンション: ReplicationGroupId	ElastiCache Valkey および Redis OSS メトリクス
クラスター (レプリケーショングループ)	AWS/ElastiCache	名前: EngineCPUUtilization ディメンション: ReplicationGroupId、ロール (プライマリ)	ElastiCache Valkey および Redis OSS メトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
クラスター (レプリケーショングループ)	AWS/ElastiCache	名前: EngineCPUUtilization ディメンション: ReplicationGroupId、ロール (レプリカ)	ElastiCache Valkey および Redis OSS メトリクス
クラスター (キャッシュ)	AWS/ElastiCache	名前: EngineCPUUtilization ディメンション: CacheClusterId、Node	ElastiCache Memcached メトリクス
クラスター (キャッシュ)	AWS/ElastiCache	名前: DatabaseCapacityMemoryUsagePercentage ディメンション: CacheClusterId	ElastiCache Memcached メトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
Amazon EMR			
クラスター	AWS/ElasticMapReduce	名前: YARNMemoryAvailablePercentage ディメンション: ClusterId	Amazon EMR のメトリクス
Amazon Keyspaces			
テーブル	AWS/Cassandra	名前: ProvisionedReadCapacityUnits ディメンション: Keyspace、TableName	Amazon Keyspaces のメトリクス
テーブル	AWS/Cassandra	名前: ProvisionedWriteCapacityUnits ディメンション: Keyspace、TableName	Amazon Keyspaces のメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
テーブル	AWS/Cassandra	名前: ConsumedReadCapacityUnits ディメンション: Keyspace、TableName	Amazon Keyspaces のメトリクス
テーブル	AWS/Cassandra	名前: ConsumedWriteCapacityUnits ディメンション: Keyspace、TableName	Amazon Keyspaces のメトリクス
Lambda			
プロビジョニングされた同時実行	AWS/Lambda	名前: ProvisionedConcurrencyUtilization ディメンション: FunctionName、Resource	Lambda 関数のメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
Amazon MSK			
ブローカーストレージ	AWS/Kafka	名前: KafkaDataLogsDiskUsed ディメンション: クラスター名	Amazon MSK のメトリクス
ブローカーストレージ	AWS/Kafka	名前: KafkaDataLogsDiskUsed ディメンション: クラスター名、ブローカー ID	Amazon MSK のメトリクス
Neptune			

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
クラスター	AWS/Neptune	名前: CPUUtilization ディメンション: DBClusterIdentifier、ロール (閲覧者)	Neptune メトリクス
SageMaker AI			
エンドポイントバリアント	AWS/SageMaker	名前: InvocationsPerInstance ディメンション: EndpointName、VariantName	呼び出しメトリクス
推論コンポーネント	AWS/SageMaker	名前: InvocationsPerCopy ディメンション: InferenceComponentName	呼び出しメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
サーバーレスエンドポイントのプロビジョニングされた同時実行数	AWS/SageMaker	名前: ServerlessProvisionedConcurrencyUtilization ディメンション: EndpointName、VariantName	サーバーレスエンドポイントのメトリクス
スポットフリート (Amazon EC2)			
Spot Fleets	AWS/EC2Spot	名前: CPUUtilization ディメンション: FleetRequestId	スポットフリートのメトリクス
Spot Fleets	AWS/EC2Spot	名前: NetworkIn ディメンション: FleetRequestId	スポットフリートのメトリクス

スケーラブルなリソース	名前空間	[CloudWatch メトリクス]	ドキュメントへのリンク
Spot Fleets	AWS/EC2Spot	名前: NetworkOut ディメンション: FleetRequestId	スポットフリートのメトリクス
Spot Fleets	AWS/ApplicationELB	名前: RequestCountPerTarget ディメンション: TargetGroup	Application Load Balancer のメトリクス

ターゲット追跡スケーリングポリシーの事前定義メトリクス

次の表は、Application Auto Scaling API リファレンスの事前定義済みメトリクスタイプおよびそれに対応する CloudWatch メトリクス名を示しています。事前定義済みメトリクスはそれぞれ、基になっている CloudWatch メトリクスの値の集計を表します。結果は、1 分間の平均リソース使用量で、特に明記されていない限りパーセント表記です。事前定義済みメトリクスは、ターゲット追跡スケーリングポリシー設定のコンテキスト内でのみ使用されます。

これらのメトリクスの詳細については、リソースの使用状況をモニタリングするための CloudWatch メトリクス の表から入手できる、サービスのドキュメントを参照してください。

事前定義済みメトリクスタイプ	CloudWatch メトリクス名
AppStream 2.0	

事前定義済みメトリクスタイプ	CloudWatch メトリクス名
AppStreamAverageCapacityUtilization	CapacityUtilization
Aurora	
RDSReaderAverageCPUUtilization	CPUUtilization
RDSReaderAverageDatabaseConnections	DatabaseConnections[1]
Amazon Comprehend	
ComprehendInferenceUtilization	InferenceUtilization
DynamoDB	
DynamoDBReadCapacityUtilization	ProvisionedReadCapacityUnits、ConsumedReadCapacityUnits[2]
DynamoDBWriteCapacityUtilization	ProvisionedWriteCapacityUnits、ConsumedWriteCapacityUnits[2]
Amazon ECS	
ECSServiceAverageCPUUtilization	CPUUtilization
ECSServiceAverageMemoryUtilization	MemoryUtilization
ALBRequestCountPerTarget	RequestCountPerTarget[1]
ElastiCache	
ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage	DatabaseMemoryUsageCountedForEvictPercentage
ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage	DatabaseCapacityUsageCountedForEvictPercentage

事前定義済みメトリクスタイプ	CloudWatch メトリクス名
`ElastiCachePrimaryEngineCPUUtilization`	EngineCPUUtilization
`ElastiCacheReplicaEngineCPUUtilization`	EngineCPUUtilization
`ElastiCacheEngineCPUUtilization`	EngineCPUUtilization
`ElastiCacheDatabaseMemoryUsagePercentage`	DatabaseMemoryUsagePercentage
Amazon Keyspaces	
`CassandraReadCapacityUtilization`	ProvisionedReadCapacityUnits、ConsumedReadCapacityUnits[2]
`CassandraWriteCapacityUtilization`	ProvisionedWriteCapacityUnits、ConsumedWriteCapacityUnits[2]
Lambda	
`LambdaProvisionedConcurrencyUtilization`	ProvisionedConcurrencyUtilization
Amazon MSK	
`KafkaBrokerStorageUtilization`	KafkaDataLogsDiskUsed
Neptune	
`NeptuneReaderAverageCPUUtilization`	CPUUtilization
SageMaker AI	
`SageMakerVariantInvocationsPerInstance`	InvocationsPerInstance[1]

事前定義済みメトリクスタイプ	CloudWatch メトリクス名
SageMakerInferenceComponentInvocationsPerCopy	InvocationsPerCopy[1]
SageMakerVariantProvisionedConcurrencyUtilization	ServerlessProvisionedConcurrencyUtilization
SageMakerInferenceComponentConcurrentRequestsPerCopyHighResolution	ConcurrentRequestsPerCopy
SageMakerVariantConcurrentRequestsPerModelHighResolution	ConcurrentRequestsPerModel
スポットフリート	
EC2SpotFleetRequestAverageCPUUtilization	CPUUtilization[3]
EC2SpotFleetRequestAverageNetworkIn[3]	NetworkIn[1] [3]
EC2SpotFleetRequestAverageNetworkOut[3]	NetworkOut[1] [3]
ALBRequestCountPerTarget	RequestCountPerTarget[1]

[1] メトリクスは割合ではなくカウントです。

[2] DynamoDB と Amazon Keyspaces の場合、事前定義済みメトリクスは 2 つの CloudWatch メトリクスを集計したもので、プロビジョニングされたスループット消費量に基づくスケーリングをサポートします。

[3] 最高のスケーリングパフォーマンスを得るには、Amazon EC2 の詳細モニタリングを使用する必要があります。

を使用した Application Auto Scaling API コールのログ記録 AWS CloudTrail

Application Auto Scaling は AWS CloudTrail と統合されています。このサービスは、ユーザー、ロール、または AWS の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、Application Auto Scaling への API コールをイベントとしてキャプチャします。キャプチャされたコールには、 AWS Management Console からのコールと、Application Auto Scaling API 操作へのコードコールが含まれます。CloudTrail によって収集された情報を使用して、 Application Auto Scaling に対して行われた要求、要求が行われた IP アドレス、要求を行った人、いつ行われたか、および追加の詳細を判別できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか。
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail は、アカウント AWS アカウント を作成すると でアクティブになり、CloudTrail イベント履歴に自動的にアクセスできます。CloudTrail の [イベント履歴] では、 AWS リージョンで過去 90 日間に記録された 管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント 過去 90 日間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。

CloudTrail 証跡

> 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。を使用して作成された証跡はすべてマルチリージョン AWS Management Console です。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント AWS リージョン 内のすべての でアクティビティをキャプチャするため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail での Application Auto Scaling 管理イベント

管理イベントは、 のリソースで実行される管理オペレーションに関する情報を提供します AWS アカウント。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。CloudTrail は、デフォルトで管理イベントをログ記録します。

Application Auto Scaling は、すべての Application Auto Scaling コントロールプレーンオペレーションを管理イベントとしてログに記録します。Application Auto Scaling が CloudTrail にログ記録する Application Auto Scaling コントロールプレーンオペレーションのリストについては、「Application Auto Scaling API リファレンス」を参照してください。

Application Auto Scaling イベントの例

各イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。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、イベントは特定の順序で表示されません。

次の例は、`DescribeScalableTargets` オペレーションを示す CloudTrail イベントを示しています。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-08-21T17:05:42Z"
            }
        }
```

```
    },
    "eventTime": "2018-08-16T23:20:32Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "DescribeScalableTargets",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "72.21.196.68",
    "userAgent": "EC2 Spot Console",
    "requestParameters": {
        "serviceNamespace": "ec2",
        "scalableDimension": "ec2:spot-fleet-request:TargetCapacity",
        "resourceIds": [
            "spot-fleet-request/sfr-05ceaf79-3ba2-405d-e87b-612857f1357a"
        ]
    },
    "responseElements": null,
    "additionalEventData": {
        "service": "application-autoscaling"
    },
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

CloudTrail レコードの内容については、「AWS CloudTrail ユーザーガイド」の「CloudTrail record contents」を参照してください。

CloudWatch での Application Auto Scaling RemoveAction 呼び出し

Application Auto Scaling が CloudWatch にアラームから自動スケーリングアクションを削除するように指示すると、Application Auto Scaling が CloudWatch `RemoveAction` API を呼び出すことが AWS CloudTrail ログに表示される場合があります。これは、スケーラブルターゲットの登録を解除したり、スケーリングポリシーを削除したり、アラームが存在しないスケーリングポリシーを呼び出したりした場合に発生する可能性があります。

Amazon EventBridge を使用して Application Auto Scaling イベントをモニタリングする

Amazon EventBridge (旧称: CloudWatch Events) は、Application Auto Scaling に固有のイベントをモニタリングし、他の AWS のサービスを使用するターゲットアクションを開始するのに役立ちます。からのイベント AWS のサービス は、ほぼリアルタイムで EventBridge に配信されます。

EventBridge を使用すると、受信イベントを照合し、処理のためにターゲットにルーティングするルールを作成できます。

詳細については、Amazon EventBridge ユーザーガイドの「Getting started with Amazon EventBridge」を参照してください。

Application Auto Scaling イベント

次の例は、Application Auto Scaling のイベントを示しています。イベントは、ベストエフォートベースで生成されます。

現在、Application Auto Scaling で使用できるのは、最大スケーリングに特化したイベントと CloudTrail 経由の API 呼び出しのみです。

イベントタイプ

- 状態変化のイベント: 最大までスケーリング
- CloudTrail を介した API コールのイベント

状態変化のイベント: 最大までスケーリング

次のイベント例は、Application Auto Scaling がスケーラブルなターゲットのキャパシティを最大サイズ制限まで引き上げた (スケールアウトした) ことを示しています。需要が再び増加した場合であっても、Application Auto Scaling は、ターゲットが既に最大サイズにスケールされているため、ターゲットをより大きなサイズにスケーリングできません。

`detail` オブジェクトでは、`resourceId`、`serviceNamespace`、および `scalableDimension` 属性の値がスケーラブルなターゲットを識別します。`newDesiredCapacity` および `oldDesiredCapacity` 属性の値は、スケールアウトイベント後の新しいキャパシティと、スケールアウトイベント前の元のキャパシティを参照します。`maxCapacity` は、スケーラブルなターゲットの最大サイズ制限です。

```
{
  "version": "0",
  "id": "11112222-3333-4444-5555-666677778888",
  "detail-type": "Application Auto Scaling Scaling Activity State Change",
  "source": "aws.application-autoscaling",
  "account": "123456789012",
  "time": "2019-06-12T10:23:40Z",
  "region": "us-west-2",
  "resources": [],
```

```
  "detail": {
    "startTime": "2022-06-12T10:20:43Z",
    "endTime": "2022-06-12T10:23:40Z",
    "newDesiredCapacity": 8,
    "oldDesiredCapacity": 5,
    "minCapacity": 2,
    "maxCapacity": 8,
    "resourceId": "table/my-table",
    "scalableDimension": "dynamodb:table:WriteCapacityUnits",
    "serviceNamespace": "dynamodb",
    "statusCode": "Successful",
    "scaledToMax": true,
    "direction": "scale-out"
}
```

すべてのスケーラブルなターゲットについて、すべての `scaledToMax` 状態の変更イベントをキャプチャするルールを作成するには、次のサンプルイベントパターンを使用します。

```
{
  "source": [
    "aws.application-autoscaling"
  ],
  "detail-type": [
    "Application Auto Scaling Scaling Activity State Change"
  ],
  "detail": {
    "scaledToMax": [
      true
    ]
  }
}
```

CloudTrail を介した API コールのイベント

証跡は、 AWS CloudTrail がイベントをログファイルとして Amazon S3 バケットに配信するために使用する設定です。CloudTrail ログファイルにはログエントリがあります。1 つのイベントが 1 つのログエントリを表し、リクエストされたアクション、アクションの日時、リクエストパラメータに関する情報が含まれます。CloudTrail の使用を開始する方法については、「AWS CloudTrail ユーザーガイド」の「証跡の作成」を参照してください。

CloudTrail 経由で送信されたイベントの `detail-type` の値は `AWS API Call via CloudTrail` です。

次のイベント例は、コンソールユーザーが Application Auto Scaling の RegisterScalableTarget アクションを呼び出したことを示す CloudTrail ログファイルエントリを表しています。


```
{
  "version": "0",
  "id": "99998888-7777-6666-5555-444433332222",
  "detail-type": "AWS API Call via CloudTrail",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2022-07-13T16:50:15Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "eventVersion": "1.08",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "123456789012",
      "arn": "arn:aws:iam::123456789012:user/Bob",
      "accountId": "123456789012",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "sessionContext": {
        "sessionIssuer": {
          "type": "Role",
          "principalId": "123456789012",
          "arn": "arn:aws:iam::123456789012:role/Admin",
          "accountId": "123456789012",
          "userName": "Admin"
        },
        "webIdFederationData": {},
        "attributes": {
          "creationDate": "2022-07-13T15:17:08Z",
          "mfaAuthenticated": "false"
        }
      }
    },
    "eventTime": "2022-07-13T16:50:15Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "RegisterScalableTarget",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "EC2 Spot Console",
    "requestParameters": {
      "resourceId": "spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
```

```
      "serviceNamespace": "ec2",
      "scalableDimension": "ec2:spot-fleet-request:TargetCapacity",
      "minCapacity": 2,
      "maxCapacity": 10
    },
    "responseElements": null,
    "additionalEventData": {
      "service": "application-autoscaling"
    },
    "requestID": "e9caf887-8d88-11e5-a331-3332aa445952",
    "eventID": "49d14f36-6450-44a5-a501-b0fdcdfaeb98",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
  }
}
```

すべてのスケーラブルなターゲットに対するすべての DeleteScalingPolicy および DeregisterScalableTarget API コールに基づいてルールを作成するには、次のサンプルイベントパターンを使用します。

```
{
  "source": [
    "aws.autoscaling"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "autoscaling.amazonaws.com"
    ],
    "eventName": [
       "DeleteScalingPolicy",
       "DeregisterScalableTarget"
    ],
    "additionalEventData": {
      "service": [
        "application-autoscaling"
      ]
```

```
    }
  }
}
```

CloudTrail の使用の詳細については、を使用した Application Auto Scaling API コールのログ記録 AWS CloudTrailを参照してください。

AWS SDK でのこのサービスの使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	AWS Tools for PowerShell コード例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

SDK を使用した Application Auto Scaling のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Application Auto Scaling を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- SDK を使用した Application Auto Scaling の基本的な例 AWS SDKs
  - SDK を使用した Application Auto Scaling のアクション AWS SDKs
    - AWS SDK または CLI DeleteScalingPolicyで を使用する
    - CLI で DeleteScheduledAction を使用する
    - CLI で DeregisterScalableTarget を使用する
    - CLI で DescribeScalableTargets を使用する
    - CLI で DescribeScalingActivities を使用する
    - AWS SDK または CLI DescribeScalingPoliciesで を使用する
    - CLI で DescribeScheduledActions を使用する
    - CLI で PutScalingPolicy を使用する
    - CLI で PutScheduledAction を使用する
    - AWS SDK または CLI RegisterScalableTargetで を使用する

SDK を使用した Application Auto Scaling の基本的な例 AWS SDKs

次のコード例は、 AWS SDK で Application Auto Scaling を使用する方法を示しています。

例

- SDK を使用した Application Auto Scaling のアクション AWS SDKs
  - AWS SDK または CLI DeleteScalingPolicyで を使用する
  - CLI で DeleteScheduledAction を使用する
  - CLI で DeregisterScalableTarget を使用する
  - CLI で DescribeScalableTargets を使用する
  - CLI で DescribeScalingActivities を使用する
  - AWS SDK または CLI DescribeScalingPoliciesで を使用する
  - CLI で DescribeScheduledActions を使用する
  - CLI で PutScalingPolicy を使用する
  - CLI で PutScheduledAction を使用する
  - AWS SDK または CLI RegisterScalableTargetで を使用する

SDK を使用した Application Auto Scaling のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の Application Auto Scaling アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、Application Auto Scaling API リファレンス を参照してください。

例

- AWS SDK または CLI DeleteScalingPolicyで を使用する
- CLI で DeleteScheduledAction を使用する
- CLI で DeregisterScalableTarget を使用する
- CLI で DescribeScalableTargets を使用する
- CLI で DescribeScalingActivities を使用する
- AWS SDK または CLI DescribeScalingPoliciesで を使用する
- CLI で DescribeScheduledActions を使用する
- CLI で PutScalingPolicy を使用する
- CLI で PutScheduledAction を使用する
- AWS SDK または CLI RegisterScalableTargetで を使用する

AWS SDK または CLI **DeleteScalingPolicy**で を使用する

次のサンプルコードは、DeleteScalingPolicy を使用する方法を説明しています。

CLI

AWS CLI

スケーリングポリシーを削除する

この例では、デフォルトのクラスターで実行されている Amazon ECS サービスウェブアプリのスケーリングポリシーを削除します。

コマンド:

```
aws application-autoscaling delete-scaling-policy --policy-name web-app-cpu-lt-25
 --scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-
app --service-namespace ecs
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteScalingPolicy」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReq
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
```

```
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();
```

```
        DescribeScalingPoliciesResponse response =
 appAutoScalingClient.describeScalingPolicies(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    public static void deregisterScalableTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        try {
            DeregisterScalableTargetRequest targetRequest =
 DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「DeleteScalingPolicy」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、Application Auto Scaling スケーラブルターゲットの指定されたスケーリングポリシーを削除します。

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDeleteScalingPolicy」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、Application Auto Scaling スケーラブルターゲットの指定されたスケーリングポリシーを削除します。

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDeleteScalingPolicy」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeleteScheduledAction** を使用する

次のサンプルコードは、DeleteScheduledAction を使用する方法を説明しています。

CLI

AWS CLI

スケジュールされたアクションを削除するには

次の delete-scheduled-action の例では、指定された Amazon AppStream 2.0 フリートから指定されたスケジュール済みアクションを削除します。

```
aws application-autoscaling delete-scheduled-action \
    --service-namespace appstream \
    --scalable-dimension appstream:fleet:DesiredCapacity \
    --resource-id fleet/sample-fleet \
    --scheduled-action-name my-recurring-action
```

このコマンドでは何も出力されません。

詳細については、「Application Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteScheduledAction」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、Application Auto Scaling スケーラブルターゲットの指定されたスケジュールされたアクションを削除します。

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDeleteScheduledAction」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、Application Auto Scaling スケーラブルターゲットの指定されたスケジュールされたアクションを削除します。

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDeleteScheduledAction」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DeregisterScalableTarget** を使用する

次のサンプルコードは、DeregisterScalableTarget を使用する方法を説明しています。

CLI

AWS CLI

スケーラブルターゲットを登録解除する

この例では、デフォルトのクラスターで実行されているウェブアプリと呼ばれる Amazon ECS サービスのスケーラブルターゲットを登録解除します。

コマンド:

```
aws application-autoscaling deregister-scalable-target --service-namespace ecs --
scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-app
```

この例では、カスタムリソースのスケーラブルターゲットを登録解除します。custom-resource-id.txt ファイルにはリソース ID を識別する文字列が含まれており、これはカスタムリソースの場合、Amazon API Gateway エンドポイント経由でのカスタムリソースへのパスを表します。

コマンド:

```
aws application-autoscaling deregister-scalable-target --
service-namespace custom-resource --scalable-dimension custom-
resource:ResourceType:Property --resource-id file://~/custom-resource-id.txt
```

custom-resource-id.txt ファイルの内容:

```
https://example.execute-api.us-west-2.amazonaws.com/prod/
scalableTargetDimensions/1-23456789
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeregisterScalableTarget」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、Application Auto Scaling のスケーラブルターゲットを登録解除します。スケーラブルターゲットの登録を解除すると、それに関連付けられているスケーリングポリシーが削除されます。

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDeregisterScalableTarget」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、Application Auto Scaling のスケーラブルターゲットを登録解除します。スケーラブルターゲットの登録を解除すると、それに関連付けられているスケーリングポリシーが削除されます。

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDeregisterScalableTarget」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeScalableTargets** を使用する

次のサンプルコードは、DescribeScalableTargets を使用する方法を説明しています。

CLI

AWS CLI

スケーラブルターゲットを記述する

次の describe-scalable-targets 例では、ecs サービス名前空間のスケーラブルターゲットについて説明します。

```
aws application-autoscaling describe-scalable-targets \
    --service-namespace ecs
```

出力:

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "ecs",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "MinCapacity": 1,
            "MaxCapacity": 10,
            "RoleARN": "arn:aws:iam::123456789012:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService",
            "CreationTime": 1462558906.199,
            "SuspendedState": {
                "DynamicScalingOutSuspended": false,
                "ScheduledScalingSuspended": false,
                "DynamicScalingInSuspended": false
            },
            "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
        }
    ]
}
```

詳細については、「Application Auto Scaling ユーザーガイド」の「Application Auto Scaling と共に使用可能なAWS サービス」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeScalableTargets」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: この例では、指定された名前空間内の Application Autoscaling Scalable ターゲットに関する情報を提供します。

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

出力:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDescribeScalableTargets」を参照してください。

Tools for PowerShell V5

例 1: この例では、指定された名前空間内の Application Autoscaling Scalable ターゲットに関する情報を提供します。

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

出力:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
```

```
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDescribeScalableTargets」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeScalingActivities** を使用する

次のサンプルコードは、DescribeScalingActivities を使用する方法を説明しています。

CLI

AWS CLI

例 1: 指定された Amazon ECS サービスのスケーリングアクティビティを記述する方法

次の describe-scaling-activities の例では、default クラスターで実行されている「web-app」という Amazon ECS サービスのスケーリングアクティビティについて記述します。出力には、スケーリングポリシーによって開始されたスケーリングアクティビティが表示されます。

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace ecs \
    --resource-id service/default/web-app
```

出力:

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Setting desired count to 1.",
            "ResourceId": "service/default/web-app",
            "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
            "StartTime": 1462575838.171,
            "ServiceNamespace": "ecs",
            "EndTime": 1462575872.111,
```

```
            "Cause": "monitor alarm web-app-cpu-lt-25 in state ALARM triggered
 policy web-app-cpu-lt-25",
            "StatusMessage": "Successfully set desired count to 1. Change
 successfully fulfilled by ecs.",
            "StatusCode": "Successful"
        }
    ]
}
```

詳細については、「Application Auto Scaling ユーザーガイド」の「Application Auto Scaling のスケーリングアクティビティ」を参照してください。

例 2: 指定された DynamoDB テーブルのスケーリングアクティビティを記述する方法

次の `describe-scaling-activities` の例では、`TestTable` という DynamoDB テーブルのスケーリングアクティビティについて記述します。出力には、2 つの異なるスケジュールされたアクションによって開始されたスケーリングアクティビティが表示されます。

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace dynamodb \
    --resource-id table/TestTable
```

出力:

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 10.",
            "ResourceId": "table/my-table",
            "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
            "StartTime": 1561574415.086,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574449.51,
            "Cause": "maximum capacity was set to 10",
            "StatusMessage": "Successfully set write capacity units to 10. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 5 and max capacity to 10",
```

```
            "ResourceId": "table/my-table",
            "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
            "StartTime": 1561574414.644,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-second-scheduled-action was
 triggered",
            "StatusMessage": "Successfully set min capacity to 5 and max capacity
 to 10",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/my-table",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574140.255,
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/my-table",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was
 triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max
 capacity to 20",
            "StatusCode": "Successful"
        }
    ]
}
```

詳細については、「Application Auto Scaling ユーザーガイド」の「Application Auto Scaling のスケーリングアクティビティ」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeScalingActivities」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定された名前空間における過去 6 週間のスケーリングアクティビティに関する詳細情報を提供します。

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

出力:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDescribeScalingActivities」を参照してください。

Tools for PowerShell V5

例 1: 指定された名前空間における過去 6 週間のスケーリングアクティビティに関する詳細情報を提供します。

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

出力:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
```

```
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDescribeScalingActivities」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeScalingPolicies**で を使用する

次のサンプルコードは、DescribeScalingPolicies を使用する方法を説明しています。

CLI

AWS CLI

スケーリングポリシーを記述する方法

この例では、ECS サービス名前空間のスケーリングポリシーについて記述します。

コマンド:

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs
```

出力:

```
{
    "ScalingPolicies": [
        {
            "PolicyName": "web-app-cpu-gt-75",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462561899.23,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 60,
```

```
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": 200,
                        "MetricIntervalLowerBound": 0.0
                    }
                ],
                "AdjustmentType": "PercentChangeInCapacity"
            },
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-gt-75",
            "PolicyType": "StepScaling",
            "Alarms": [
                {
                    "AlarmName": "web-app-cpu-gt-75",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-gt-75"
                }
            ],
            "ServiceNamespace": "ecs"
        },
        {
            "PolicyName": "web-app-cpu-lt-25",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462562575.099,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 1,
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": -50,
                        "MetricIntervalUpperBound": 0.0
                    }
                ],
                "AdjustmentType": "PercentChangeInCapacity"
            },
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-lt-25",
            "PolicyType": "StepScaling",
            "Alarms": [
                {
                    "AlarmName": "web-app-cpu-lt-25",
```

```
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-lt-25"
                }
            ],
            "ServiceNamespace": "ecs"
        }
    ]
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeScalingPolicies」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、指定されたサービス名前空間の Application Auto Scaling スケーリングポリシーを記述します。

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

出力:

```
Alarms                                    : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-out
PolicyName                                : default-scale-out
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                    : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
```

```
PolicyARN                                   : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                              policyName/default-scale-in
PolicyName                                  : default-scale-in
PolicyType                                  : StepScaling
ResourceId                                  : fleet/LabFleet
ScalableDimension                           : appstream:fleet:DesiredCapacity
ServiceNamespace                            : appstream
StepScalingPolicyConfiguration              :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDescribeScalingPolicies」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、指定されたサービス名前空間の Application Auto Scaling スケーリングポリシーを記述します。

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

出力:

```
Alarms                                      : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                                : 9/3/2019 2:48:15 AM
PolicyARN                                   : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                              policyName/default-scale-out
PolicyName                                  : default-scale-out
PolicyType                                  : StepScaling
ResourceId                                  : fleet/LabFleet
ScalableDimension                           : appstream:fleet:DesiredCapacity
ServiceNamespace                            : appstream
StepScalingPolicyConfiguration              :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                      : {Appstream2-LabFleet-default-scale-in-
Alarm}
```

```
CreationTime                                : 9/3/2019 2:48:15 AM
PolicyARN                                   : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                              policyName/default-scale-in
PolicyName                                  : default-scale-in
PolicyType                                  : StepScaling
ResourceId                                  : fleet/LabFleet
ScalableDimension                           : appstream:fleet:DesiredCapacity
ServiceNamespace                            : appstream
StepScalingPolicyConfiguration              :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDescribeScalingPolicies」を参照してください。

Rust

SDK for Rust

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    println!("Auto Scaling Policies:");
    for policy in response.scaling_policies() {
        println!("{:?}\n", policy);
    }
    println!("Next token: {:?}", response.next_token());

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「DescribeScalingPolicies」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **DescribeScheduledActions** を使用する

次のサンプルコードは、DescribeScheduledActions を使用する方法を説明しています。

CLI

AWS CLI

スケジュールされたアクションの説明

次の describe-scheduled-actions の例では、指定されたサービス名前空間でスケジュールが設定されたアクションの詳細を表示します。

```
aws application-autoscaling describe-scheduled-actions \
    --service-namespace dynamodb
```

出力:

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:35:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571888.361,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-first-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 15,
```

```
                "MaxCapacity": 20
            },
            "ScheduledActionName": "my-first-scheduled-action",
            "ServiceNamespace": "dynamodb"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:40:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571946.021,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-second-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 5,
                "MaxCapacity": 10
            },
            "ScheduledActionName": "my-second-scheduled-action",
            "ServiceNamespace": "dynamodb"
        }
    ]
}
```

詳細については、「Application Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeScheduledActions」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットには、Auto Scaling グループにスケジュールされた、実行されていないか終了時刻に達していないアクションが一覧表示されます。

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

出力:

```
CreationTime         : 12/22/2019 9:25:52 AM
```

```
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のDescribeScheduledActions」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットには、Auto Scaling グループにスケジュールされた、実行されていないか終了時刻に達していないアクションが一覧表示されます。

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

出力:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のDescribeScheduledActions」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **PutScalingPolicy** を使用する

次のサンプルコードは、PutScalingPolicy を使用する方法を説明しています。

CLI

AWS CLI

例 1: 事前定義されたメトリクス指定を使用してターゲット追跡スケーリングポリシーを適用するには

以下の put-scaling-policy の例では、デフォルトクラスター内の web-app と呼ばれる Amazon ECS サービスに、事前に定義されたメトリック仕様でターゲット追跡スケーリングポリシーを適用します。このポリシーでは、サービスの平均 CPU 使用率を 75% に保ち、スケールアウトとスケールインのクールダウン期間は 60 秒です。出力には、自動的に作成された 2 つの CloudWatch アラームの ARN と名前が含まれます。

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cpu75-target-tracking-scaling-policy --policy-
type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

この例では、現在のディレクトリに次の内容の config.json ファイルがあることを前提としています。

```
{
     "TargetValue": 75.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ECSServiceAverageCPUUtilization"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

出力:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/cpu75-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
        }
    ]
}
```

例 2: カスタマイズされたメトリクス仕様を使用してターゲット追跡スケーリングポリシーを適用するには

以下の put-scaling-policy の例では、デフォルトクラスター内の web-app と呼ばれる Amazon ECS サービスに、カスタマイズされたメトリック仕様でターゲット追跡スケーリングポリシーを適用します。このポリシーでは、サービスの平均使用率を 75% に保ち、スケールアウトとスケールインのクールダウン期間は 60 秒です。出力には、自動的に作成された 2 つの CloudWatch アラームの ARN と名前が含まれます。

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cms75-target-tracking-scaling-policy
--policy-type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

この例では、現在のディレクトリに次の内容の config.json ファイルがあることを前提としています。

```
{
    "TargetValue":75.0,
    "CustomizedMetricSpecification":{
        "MetricName":"MyUtilizationMetric",
        "Namespace":"MyNamespace",
        "Dimensions": [
            {
                "Name":"MyOptionalMetricDimensionName",
                "Value":"MyOptionalMetricDimensionValue"
            }
        ],
        "Statistic":"Average",
        "Unit":"Percent"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

出力:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:
 8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/default/web-
app:policyName/cms75-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
        }
    ]
}
```

例 3: スケールアウトにのみターゲット追跡スケーリングポリシーを適用するには

次の例では、ターゲット追跡スケーリングポリシーを、デフォルトのクラスターで web-app という Amazon ECS サービスに適用します。このポリシーは、Application Load Balancer の RequestCountPerTarget メトリクスがしきい値を超えたときに ECS サービスをスケールアウトするために使用されます。出力には、自動的に作成された CloudWatch アラームの ARN と名前が含まれます。

```
aws application-autoscaling put-scaling-policy \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --policy-name alb-scale-out-target-tracking-scaling-policy \
    --policy-type TargetTrackingScaling \
    --target-tracking-scaling-policy-configuration file://config.json
```

config.json の内容:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ALBRequestCountPerTarget",
         "ResourceLabel": "app/EC2Co-EcsEl-1TKLTMITMM0EO/f37c06a68c1748aa/
targetgroup/EC2Co-Defau-LDNM7Q3ZH1ZN/6d4ea56ca2d6a18d"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60,
    "DisableScaleIn": true
}
```

出力:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/alb-scale-out-target-tracking-scaling-
policy",
    "Alarms": [
        {
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
```

```
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        }
    ]
}
```

詳細については、「AWS Application Auto Scaling ユーザーガイド」の「Application Auto Scaling のターゲット追跡スケーリングポリシー」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutScalingPolicy」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、Application Auto Scaling のスケーラブルターゲットのポリシーを作成または更新します。各スケーラブルターゲットは、サービス名前空間、リソース ID、スケーラブルなディメンションによって識別されます。

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

出力:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のPutScalingPolicy」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、Application Auto Scaling のスケーラブルターゲットのポリシーを作成または更新します。各スケーラブルターゲットは、サービス名前空間、リソース ID、スケーラブルなディメンションによって識別されます。

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

出力:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のPutScalingPolicy」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **PutScheduledAction** を使用する

次のサンプルコードは、PutScheduledAction を使用する方法を説明しています。

CLI

AWS CLI

スケジュールされたアクションを DynamoDB テーブルに追加する方法

この例では、TestTable という名前の DynamoDB テーブルにスケジュールされたアクションを追加して、定期的なスケジュールに合わせてスケールアウトします。特定のスケ

ジュール (毎日午後 12:15 (UTC)) で、現在の容量が MinCapacity に指定された値を下回る場合、Application Auto Scaling は MinCapacity で指定された値にスケールアウトします。

コマンド:

```
aws application-autoscaling put-scheduled-action --service-
namespace dynamodb --scheduled-action-name my-recurring-action --
schedule "cron(15 12 * * ? *)" --resource-id table/TestTable --
scalable-dimension dynamodb:table:WriteCapacityUnits --scalable-target-
action MinCapacity=6
```

詳細については、「Application Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「PutScheduledAction」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、Application Auto Scaling のスケーラブルターゲットのスケジュールされたアクションを作成または更新します。各スケーラブルターゲットは、サービス名前空間、リソース ID、スケーラブルなディメンションによって識別されます。

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のPutScheduledAction」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、Application Auto Scaling のスケーラブルターゲットのスケジュールされたアクションを作成または更新します。各スケーラブルターゲットは、サービス名前空間、リソース ID、スケーラブルなディメンションによって識別されます。

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
```

```
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のPutScheduledAction」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RegisterScalableTarget**で を使用する

次のサンプルコードは、RegisterScalableTarget を使用する方法を説明しています。

CLI

AWS CLI

例 1: ECS サービスをスケーラブルターゲットとして登録する方法

以下の register-scalable-target の例は、Amazon ECS サービスを Application Auto Scaling に登録します。また、スケーラブルターゲットにキー名「environment」と値「production」を含むタグを追加します。

```
aws application-autoscaling register-scalable-target \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --min-capacity 1 --max-capacity 10 \
    --tags environment=production
```

出力:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

他の AWS サービスやカスタムリソースの例については、AWS 「Application Auto Scaling ユーザーガイド」の「Application Auto Scaling で使用できる サービスのトピック」を参照してください。 Auto Scaling

例 2: スケーラブルターゲットのスケーリングアクティビティを一時停止する方法

次の register-scalable-target の例では、既存のスケーラブルターゲットのスケーリングアクティビティを一時停止します。

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=true,DynamicScalingOutSuspended=true,ScheduledScalingSusp
```

出力:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

詳細については、「Application Auto Scaling ユーザーガイド」の「Application Auto Scaling のスケーリングの一時停止と再開」を参照してください。

例 3: スケーラブルターゲットのスケーリングアクティビティを再開する方法

次の register-scalable-target の例では、既存のスケーラブルターゲットのスケーリングアクティビティを再開します。

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=false,DynamicScalingOutSuspended=false,ScheduledScalingSu
```

出力:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

詳細については、「Application Auto Scaling ユーザーガイド」の「Application Auto Scaling のスケーリングの一時停止と再開」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「RegisterScalableTarget」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecificati
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetReque
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
```

```
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicy
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <roleARN> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
 permissions.
               policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table,
 which is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
```

```
            .region(Region.US_EAST_1)
            .build();

       registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
       verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
       configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
   }

   public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
       try {
           RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
               .serviceNamespace(ns)
               .scalableDimension(tableWCUs)
               .resourceId(tableId)
               .roleARN(roleARN)
               .minCapacity(5)
               .maxCapacity(10)
               .build();

           appAutoScalingClient.registerScalableTarget(targetRequest);
           System.out.println("You have registered " + tableId);

       } catch (ApplicationAutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
   }

   // Verify that the target was created.
   public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
       DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
           .scalableDimension(tableWCUs)
           .serviceNamespace(ns)
           .resourceIds(tableId)
           .build();
```

```
        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
    public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
        // Check if the policy exists before creating a new one.
        DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
            .serviceNamespace(ns)
            .resourceId(tableId)
            .scalableDimension(tableWCUs)
            .build());

        if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
            // If policies exist, consider updating an existing policy instead of
creating a new one.
            System.out.println("Policy already exists. Consider updating it
instead.");
            List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
            for (ScalingPolicy pol : polList) {
                System.out.println("Policy name:" +pol.policyName());
            }
        } else {
            // If no policies exist, proceed with creating a new policy.
            PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
                .build();

            TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
                .predefinedMetricSpecification(specification)
                .targetValue(50.0)
                .scaleInCooldown(60)
                .scaleOutCooldown(60)
                .build();
```

```
            PutScalingPolicyRequest putScalingPolicyRequest =
 PutScalingPolicyRequest.builder()
                .targetTrackingScalingPolicyConfiguration(policyConfiguration)
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling
 policy for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " +
 e.awsErrorDetails().errorMessage());
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「RegisterScalableTarget」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: このコマンドレットは、スケーラブルなターゲットを登録または更新します。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V4)」のRegisterScalableTarget」を参照してください。

Tools for PowerShell V5

例 1: このコマンドレットは、スケーラブルなターゲットを登録または更新します。スケーラブルターゲットとは、Application Auto Scaling がスケールアウトおよびスケールインできるリソースです。

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- API の詳細については、AWS Tools for PowerShell 「 コマンドレットリファレンス (V5)」のRegisterScalableTarget」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Application Auto Scaling のタグ付けサポート

AWS CLI または SDK を使用して、Application Auto Scaling のスケーラブルターゲットにタグを付けることができます。スケーラブルターゲットは、Application Auto Scaling がスケーリングできる AWS またはカスタムリソースを表すエンティティです。

各タグは、Application Auto Scaling API を使用してユーザー定義のキーと値で構成されるラベルです。タグは、組織のニーズに応じて、特定のスケーラブルターゲットへのきめ細かいアクセスを構成するのに役立ちます。詳細については、「ABAC と Application Auto Scaling」を参照してください。

新しいスケーラブルターゲットを登録するときにタグを追加したり、既存のスケーラブルターゲットにタグを追加したりできます。

タタグを管理するために一般的に使用されるコマンドには、以下があります。

- register-scalable-target は、新しいスケーラブルターゲットを登録するときにタグ付けします。
- tag-resource は、既存のスケーラブルターゲットにタグを追加します。
- list-tags-for-resource は、スケーラブルターゲットでタグを返します。
- untag-resource は、タグを削除します。

タグ付けの例

`--tags` オプションがある、以下の register-scalable-target コマンドを使用します。この例では、2 つのタグでスケーラブルターゲットにタグを付けます。**`production`** のタグ値で名前が **`environment`** であるタグキーと、**`true`** のタグ値で名前が **`iscontainerbased`** であるタグキーです。

`--min-capacity` と のサンプル値`--max-capacity`、および のサンプルテキスト`--service-namespace`を、Application Auto Scaling で使用している AWS サービスの名前空間`--scalable-dimension`、登録するリソースに関連付けられたスケーラブルなディメンション、および リソースの識別子`--resource-id`に置き換えます。各サービスの詳細情報および例については、AWS のサービス Application Auto Scaling で使用できる のトピックを参照してください。

```
aws application-autoscaling register-scalable-target \
  --service-namespace namespace \
  --scalable-dimension dimension \
```

```
  --resource-id identifier \
  --min-capacity 1 --max-capacity 10 \
  --tags environment=production,iscontainerbased=true
```

成功した場合、このコマンドはスケーラブルターゲットの ARN を返します。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

> **Note**
>
> このコマンドがエラーをスローする場合は、 を AWS CLI ローカルで最新バージョンに更新していることを確認してください。

セキュリティ用のタグ

タグを使用して、リクエスター (IAM ユーザーやロールなど) が特定のアクションを実行するアクセス許可を持っていることを確認します。以下の条件キーを 1 つ以上使用して、IAM ポリシーの条件要素にタグ情報を指定します。

- 特定のタグを持つスケーラブルターゲットに対してユーザーアクションを許可 (または拒否) するには、aws:ResourceTag/*tag-key*: *tag-value* を使用します。
- リクエストに特定のタグが含める (または含めない) ことを要求するには、aws:RequestTag/*tag-key*: *tag-value* を使用します。
- リクエストに特定のタグキーを含める (または含めない) ことを要求するには、aws:TagKeys [*tag-key*, ...] を使用します。

例えば、次の IAM ポリシーでは、DeregisterScalableTarget、DeleteScalingPolicy、DeleteScheduledAction アクションを使用するアクセス許可を付与します。ただし、処理対象のスケーラブルターゲットにタグ **environment=production** がある場合は、そのアクションも拒否します。

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

```
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeleteScheduledAction"
            ],
            "Resource": "*"
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeleteScheduledAction"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

タグへのアクセスを制御する

タグを使用してリクエスター (IAM ユーザーまたはロールなど) が スケーラブルターゲットのタグを追加、変更、削除するアクセス許可を持っていることを確認します。

例えば、IAM ポリシーを作成して、スケーラブルターゲットから **temporary** キーでタグのみ削除することを許可できます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "application-autoscaling:UntagResource",
```

```
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary"] }
            }
        }
    ]
}
```

Application Auto Scaling のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証します。Application Auto Scaling に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスAWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Application Auto Scaling の使用時に責任共有モデルがどのように適用されるかを理解するために役立ちます。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上の目的を達成するために Application Auto Scaling を設定する方法について説明します。また、Application Auto Scaling リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

内容

Application Auto Scaling でのデータ保護

責任 AWS 共有モデル、Application Auto Scaling でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Application Auto Scaling AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

Application Auto Scaling の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインインを許可) し、誰に Application Auto Scaling リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

完全な IAM ドキュメントについては、「IAM ユーザーガイド」を参照してください。

アクセスコントロール

リクエストを認証するための有効な認証情報があっても、許可がなければ Application Auto Scaling リソースを作成、またはそれらにアクセスすることはできません。例えば、スケーリングポリシーの作成、スケジュールされたスケーリングの設定などのアクセス権限が必要です。

以下のセクションでは、Application Auto Scaling API アクションを実行できるユーザーを制御することで、IAM 管理者が IAM を使用して AWS リソースを保護する方法について詳しく説明します。

内容

Application Auto Scaling で IAM が機能する仕組み

> **Note**
>
> 2017 年 12 月、Application Auto Scaling の更新が行われ、Application Auto Scaling 統合サービスのために複数のサービスリンクロールが有効化されました。ユーザーがスケーリングを設定できるようにするには、特定の IAM 許可、および Application Auto Scaling サービスリンクロール (または Amazon EMR オートスケーリング用のサービスロール) が必要です。

IAM を使用して Application Auto Scaling へのアクセスを管理する前に、Application Auto Scaling で使用できる IAM 機能を理解しておく必要があります。

Application Auto Scaling で使用できる IAM 機能

IAM 機能	アプリケーションの自動スケーリングのサポート
アイデンティティベースポリシー	はい
ポリシーアクション	はい
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
リソースベースのポリシー	いいえ
ACL	いいえ
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	はい
サービスロール	はい
サービスリンクロール	はい

Application Auto Scaling およびその他の がほとんどの IAM 機能と AWS のサービス 連携する方法の概要については、AWS のサービス 「IAM ユーザーガイド」の「IAM と連携する」を参照してください。

Application Auto Scaling のアイデンティティベースポリシー

ID ベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Application Auto Scaling のアイデンティティベースポリシー例

Application Auto Scaling のアイデンティティベースポリシーの例については、「Application Auto Scaling のアイデンティティベースポリシー例」を参照してください。

アクション

ポリシーアクションのサポート:あり

IAM ポリシーステートメントで、IAM をサポートするすべてのサービスからの任意の API アクションを指定できます。Application Auto Scaling の場合、API アクション `application-autoscaling:` の名前に次のプレフィックスを使用します。例えば、`application-autoscaling:RegisterScalableTarget`、`application-autoscaling:PutScalingPolicy`、および `application-autoscaling:DeregisterScalableTarget` のようになります。

1 つのステートメントで複数のアクションを指定するには、次の例のようにカンマで区切ります。

```
"Action": [
      "application-autoscaling:DescribeScalingPolicies",
      "application-autoscaling:DescribeScalingActivities"
```

ワイルドカード (*) を使用して複数のアクションを指定することができます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "application-autoscaling:Describe*"
```

Application Auto Scaling アクションのリストについては、「サービス認可リファレンス」のAWS「Application Auto Scaling で定義されるアクション」を参照してください。

リソース

ポリシーリソースのサポート: あり

IAM ポリシーステートメントで、Resource 要素は、ステートメントがカバーするオブジェクトを指定します。Application Auto Scaling の場合、Amazon リソースネーム (ARN) を使用して指定したスケーラブルターゲットに、各 IAM ポリシーステートメントが適用されます。

スケーラブルターゲットの ARN リソース形式:

```
arn:aws:application-autoscaling:region:account-id:scalable-target/unique-identifier
```

例えば、以下の要領で ARN を使用して、ステートメント内で特定のスケーラブルターゲットを指定することができます。ユニーク ID (1234abcd56ab78cd901ef1234567890ab123) は、Application Auto Scaling によってスケーラブルターゲットに割り当てられる値です。

```
"Resource": "arn:aws:application-autoscaling:us-east-1:123456789012:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
```

次のように、一意の識別子をワイルドカード (*) に置き換えることで、特定のアカウントに属するすべてのインスタンスを指定できます。

```
"Resource": "arn:aws:application-autoscaling:us-east-1:123456789012:scalable-target/*"
```

すべてのリソースを指定する場合、または特定の API アクションが ARN をサポートしていない場合は、以下のように、Resource エレメント内でワイルドカード (*) を使用します。

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

詳細については、「サービス認可リファレンス」のAWS 「Application Auto Scaling で定義されるリソースタイプ」を参照してください。

条件キー

サービス固有のポリシー条件キーのサポート: あり

Application Auto Scaling リソースへのアクセスを制御する IAM ポリシーで条件を指定できます。ポリシーステートメントは、条件が true の場合にのみ有効です。

Application Auto Scaling は、Application Auto Scaling API アクションを実行できるユーザーを決定するためにアイデンティティベースのポリシーで使用できる次のサービス定義条件キーをサポートしています。

- application-autoscaling:scalable-dimension

- `application-autoscaling:service-namespace`

条件キーを使用できる Application Auto Scaling API アクションについては、「サービス認可リファレンス」のAWS 「Application Auto Scaling で定義されるアクション」を参照してください。Application Auto Scaling 条件キーの使用の詳細については、AWS 「Application Auto Scaling の条件キー」を参照してください。

すべてのサービスで使用できるグローバル条件キーを確認するには、IAM User Guide の「AWS global condition context keys」を参照してください。

リソースベースのポリシー

リソースベースのポリシーのサポート: なし

Amazon Simple Storage Service などの他の AWS サービスは、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。例えば、ポリシーを S3 バケットにアタッチして、そのバケットに対するアクセス許可を管理できます。

Application Auto Scaling は、リソースベースポリシーをサポートしません。

アクセスコントロールリスト (ACL)

ACL のサポート: なし

Application Auto Scaling は、アクセスコントロールリスト (ACL) をサポートしません。

ABAC と Application Auto Scaling

ABAC (ポリシー内のタグ) のサポート: 一部

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

ABAC はタグをサポートするリソースでは可能ですが、すべてのリソースがタグをサポートしているわけではありません。スケジュールされたアクションとスケーリングポリシーはタグをサポートしていませんが、スケーラブルターゲットはタグをサポートしています。詳細については、「Application Auto Scaling のタグ付けサポート」を参照してください。

ABAC の詳細については、IAM ユーザーガイドのABAC とは?を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、IAM ユーザーガイドの属性に基づくアクセスコントロール (ABAC) を使用するを参照してください。

Application Auto Scaling での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報と AWS のサービス 連携する などの詳細については、AWS のサービス IAM ユーザーガイドの「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用してアクセスすることができます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

サービス役割

サービスロールのサポート: あり

Amazon EMR クラスターがオートスケーリングを使用する場合、この機能は、Application Auto Scaling がユーザーに代わってサービスロールを引き受けることを許可します。サービスリンクロールと同様に、サービスロールは、サービスがユーザーに代わって他のサービスのリソースにアクセスし、アクションを完了することを許可します。サービス役割はIAM アカウントに表示され、アカウントによって所有されます。つまり、IAM 管理者はこの役割の権限を変更できます。ただし、それにより、サービスの機能が損なわれる場合があります。

Application Auto Scaling は、Amazon EMR に対してのみサービスロールをサポートします。EMR サービスロールのドキュメントについては、Amazon EMR 管理ガイドの「Using automatic scaling with a custom policy for instance groups」を参照してください。

> **Note**
>
> サービスにリンクされたロールの導入により、いくつかのレガシーサービスロールは不要になりました。例えば、Amazon ECS やスポットフリートなどです。

サービスにリンクされた役割

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンク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。

Application Auto Scaling 用のサービスリンクロールの詳細については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

AWS Application Auto Scaling の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、多くの一般的なユースケースにアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS 管理ポリシーを更新する可能性が高くなります。

詳細については「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS マネージドポリシー: AppStream 2.0 と CloudWatch

ポリシー名: AWSApplicationAutoscalingAppStreamFleetPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_AppStreamFleet という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Amazon AppStream と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `appstream:DescribeFleets`
- アクション: `appstream:UpdateFleet`
- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Aurora と CloudWatch

ポリシー名: AWSApplicationAutoscalingRDSClusterPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_RDSCluster という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Aurora と CloudWatch を呼び出して、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `rds:AddTagsToResource`
- アクション: `rds:CreateDBInstance`
- アクション: `rds:DeleteDBInstance`
- アクション: `rds:DescribeDBClusters`
- アクション: `rds:DescribeDBInstance`

- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Amazon Comprehend と CloudWatch

ポリシー名: AWSApplicationAutoscalingComprehendEndpointPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Amazon Comprehend と CloudWatch を呼び出して、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `comprehend:UpdateEndpoint`
- アクション: `comprehend:DescribeEndpoint`
- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: DynamoDB と CloudWatch

ポリシー名: AWSApplicationAutoscalingDynamoDBTablePolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_DynamoDBTable という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は DynamoDB と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `dynamodb:DescribeTable`
- アクション: `dynamodb:UpdateTable`

- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Amazon ECS と CloudWatch

ポリシー名: AWSApplicationAutoscalingECSServicePolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_ECSService という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Amazon ECS と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `ecs:DescribeServices`
- アクション: `ecs:UpdateService`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:GetMetricData`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: ElastiCache と CloudWatch

ポリシー名: AWSApplicationAutoscalingElastiCacheRGPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は ElastiCache と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。このサービスにリンクされたロールは、ElastiCache Memcached、Redis OSS、および Valkey に使用できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling が指定されたリソースで以下のアクションを完了することを許可します。

- アクション: すべてのリソースでの elasticache:DescribeReplicationGroups
- アクション: すべてのリソースでの elasticache:ModifyReplicationGroupShardConfiguration
- アクション: すべてのリソースでの elasticache:IncreaseReplicaCount
- アクション: すべてのリソースでの elasticache:DecreaseReplicaCount
- アクション: すべてのリソースでの elasticache:DescribeCacheClusters
- アクション: すべてのリソースでの elasticache:DescribeCacheParameters
- アクション: すべてのリソースでの elasticache:ModifyCacheCluster
- アクション: リソース arn:aws:cloudwatch:*:*:alarm:* での cloudwatch:DescribeAlarms
- アクション: リソース arn:aws:cloudwatch:*:*:alarm:TargetTracking* での cloudwatch:PutMetricAlarm
- アクション: リソース arn:aws:cloudwatch:*:*:alarm:TargetTracking* での cloudwatch:DeleteAlarms

AWS マネージドポリシー: Amazon Keyspaces と CloudWatch

ポリシー名: AWSApplicationAutoscalingCassandraTablePolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_CassandraTable という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Amazon Keyspaces と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling が指定されたリソースで以下のアクションを完了することを許可します。

- アクション: 次のリソースでの cassandra:Select:
  - arn:*:cassandra:*:*:/keyspace/system/table/*
  - arn:*:cassandra:*:*:/keyspace/system_schema/table/*
  - arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*
- アクション: すべてのリソースでの cassandra:Alter
- アクション: すべてのリソースでの cloudwatch:DescribeAlarms
- アクション: すべてのリソースでの cloudwatch:PutMetricAlarm

- アクション: すべてのリソースでの `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Lambda と CloudWatch

ポリシー名: AWSApplicationAutoscalingLambdaConcurrencyPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Lambda と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `lambda:PutProvisionedConcurrencyConfig`
- アクション: `lambda:GetProvisionedConcurrencyConfig`
- アクション: `lambda:DeleteProvisionedConcurrencyConfig`
- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Amazon MSK と CloudWatch

ポリシー名: AWSApplicationAutoscalingKafkaClusterPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_KafkaCluster という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Amazon MSK と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `kafka:DescribeCluster`
- アクション: `kafka:DescribeClusterOperation`
- アクション: `kafka:UpdateBrokerStorage`

- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Neptune と CloudWatch

ポリシー名: AWSApplicationAutoscalingNeptuneClusterPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_NeptuneCluster という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Neptune と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling が指定されたリソースで以下のアクションを完了することを許可します。

- アクション: すべてのリソースでの `rds:ListTagsForResource`
- アクション: すべてのリソースでの `rds:DescribeDBInstances`
- アクション: すべてのリソースでの `rds:DescribeDBClusters`
- アクション: すべてのリソースでの `rds:DescribeDBClusterParameters`
- アクション: すべてのリソースでの `cloudwatch:DescribeAlarms`
- アクション: Amazon Neptune データベースエンジン (`"Condition":{"StringEquals":{"rds:DatabaseEngine":"neptune"}`) のプレフィックス Autoscaled 閲覧者が付いたリソースの `rds:AddTagsToResource`
- アクション: Amazon Neptune データベースエンジン (`"Condition":{"StringEquals":{"rds:DatabaseEngine":"neptune"}`) のすべての DB クラスター (`"Resource":"arn:*:rds:*:*:db:autoscaled-reader*", "arn:aws:rds:*:*:cluster:*"`) のプレフィックス Autoscaled 閲覧者が付いたリソースの `rds:CreateDBInstance`
- アクション: リソース `arn:aws:rds:*:*:db:autoscaled-reader*` での `rds:DeleteDBInstance`
- アクション: リソース `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` での `cloudwatch:PutMetricAlarm`
- アクション: リソース `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` での `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: SageMaker AI と CloudWatch

ポリシー名: AWSApplicationAutoscalingSageMakerEndpointPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint という名前のサービスにリンクされたロールにアタッチされ、Application Auto Scaling が SageMaker AI と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できるようにします。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling が指定されたリソースで以下のアクションを完了することを許可します。

- アクション: すべてのリソースでの `sagemaker:DescribeEndpoint`
- アクション: すべてのリソースでの `sagemaker:DescribeEndpointConfig`
- アクション: すべてのリソースでの `sagemaker:DescribeInferenceComponent`
- アクション: すべてのリソースでの `sagemaker:UpdateEndpointWeightsAndCapacities`
- アクション: すべてのリソースでの `sagemaker:UpdateInferenceComponentRuntimeConfig`
- アクション: すべてのリソースでの `cloudwatch:DescribeAlarms`
- アクション: すべてのリソースでの `cloudwatch:GetMetricData`
- アクション: リソース `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` での `cloudwatch:PutMetricAlarm`
- アクション: リソース `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` での `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: EC2 スポットフリートと CloudWatch

ポリシー名: AWSApplicationAutoscalingEC2SpotFleetRequestPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Amazon EC2 と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `ec2:DescribeSpotFleetRequests`
- アクション: `ec2:ModifySpotFleetRequest`
- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: WorkSpaces と CloudWatch

ポリシー名: AWSApplicationAutoscalingWorkSpacesPoolPolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_WorkSpacesPool という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は WorkSpaces と CloudWatch を呼び出し、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling が指定されたリソースで以下のアクションを完了することを許可します。

- アクション: SLR と同じアカウントのすべてのリソースでの `workspaces:DescribeWorkspacesPools`
- アクション: SLR と同じアカウントのすべてのリソースでの `workspaces:UpdateWorkspacesPool`
- アクション: SLR と同じアカウントのすべてのアラームでの `cloudwatch:DescribeAlarms`
- アクション: SLR と同じアカウントのすべてのアラームでの `cloudwatch:PutMetricAlarm`。アラーム名は TargetTracking で始まります。
- アクション: SLR と同じアカウントのすべてのアラームでの `cloudwatch:DeleteAlarms`。アラーム名は TargetTracking で始まります。

AWS マネージドポリシー: カスタムリソースと CloudWatch

ポリシー名: AWSApplicationAutoScalingCustomResourcePolicy

このポリシーは、AWSServiceRoleForApplicationAutoScaling_CustomResource という、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は、API Gateway と CloudWatch を通じて利用できるカスタムリソースを呼び出して、ユーザーに代わってスケーリングを実行できます。

アクセス許可の詳細

許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」:「*」) で以下のアクションを完了することを許可します。

- アクション: `execute-api:Invoke`
- アクション: `cloudwatch:DescribeAlarms`
- アクション: `cloudwatch:PutMetricAlarm`
- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシーへの Application Auto Scaling の更新

このサービスがこれらの変更の追跡を開始してからの Application Auto Scaling の AWS マネージドポリシーの更新に関する詳細を表示します。このページへの変更に関する自動アラートを受け取るには、Application Auto Scaling のドキュメント履歴ページで RSS フィードにサブスクライブしてください。

変更	説明	日付
AWSApplicationAutoscalingElastiCacheRGPolicy – 既存のポリシーを更新する	Memcached 自動スケーリングをサポートするために ElastiCache `ModifyCacheCluster` API アクションを呼び出すアクセス許可を追加しました。	2025 年 4 月 10 日
AWSApplicationAutoscalingECSServicePolicy – 既存のポリシーを更新する	予測スケーリングをサポートするために CloudWatch `GetMetricData` API アクションを呼び出すアクセス許可を追加しました。	2024 年 11 月 21 日
AWSApplicationAutoscalingWorkSpacesPoolPolicy - 新しいポリシー	Amazon WorkSpaces のマネージドポリシーが追加されました。このポリシーは、サービスにリンクされたロールにアタッチされます。こ	2024 年 6 月 24 日

変更	説明	日付
	れにより Application Auto Scaling は WorkSpaces と CloudWatch を呼び出して、ユーザーに代わってスケーリングを実行できます。	
AWSApplicationAutoscalingSageMakerEndpointPolicy - 既存ポリシーへの更新	今後の統合のために SageMaker AI リソースの自動スケーリングの互換性をサポートするために、SageMaker AI `DescribeInferenceComponent` および `UpdateInferenceComponentRuntimeConfig` API アクションを呼び出すアクセス許可を追加しました。また、このポリシーは、CloudWatch `PutMetricAlarm` と `DeleteAlarms` API アクションをターゲット追跡スケーリングポリシーで使用される CloudWatch アラームに制限するようになりました。	2023 年 11 月 13 日
AWSApplicationAutoscalingNeptuneClusterPolicy - 新しいポリシー	Neptune のマネージドポリシーが追加されました。このポリシーは、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は Neptune と CloudWatch を呼び出して、ユーザーに代わってスケーリングを実行できます。	2021 年 10 月 6 日

変更	説明	日付
AWSApplicationAutoscalingRDSClusterPolicy - 新しいポリシー	ElastiCache のマネージドポリシーが追加されました。このポリシーは、サービスにリンクされたロールにアタッチされます。これにより Application Auto Scaling は ElastiCache と CloudWatch を呼び出して、ユーザーに代わってスケーリングを実行できます。	2021 年 8 月 19 日
Application Auto Scaling が変更の追跡を開始	Application Auto Scaling は AWS 、管理ポリシーの変更の追跡を開始しました。	2021 年 8 月 19 日

Application Auto Scaling 用のサービスリンクロール

Application Auto Scaling は、ユーザーに代わって他の サービスを呼び出すために必要なアクセス許可に、サービスにリンクされたロールを使用します。 AWS サービスにリンクされたロールは、AWS サービスに直接リンクされた一意のタイプの AWS Identity and Access Management (IAM) ロールです。サービスにリンクされたロールは、リンクされたサービスのみが AWS サービスにリンクされたロールを引き受けることができるため、サービスにアクセス許可を委任する安全な方法を提供します。

Application Auto Scaling と統合されるサービスについては、Application Auto Scaling がユーザーのためにサービスリンクロールを作成します。サービスリンクロールはサービスごとに 1 つあります。サービスリンクロールはそれぞれ、指定されたサービスプリンシパルを信頼してそのロールを継承します。詳細については、「サービスリンクロールの ARN リファレンス」を参照してください。

Application Auto Scaling は、各サービスリンクロールに必要な許可のすべてを含めます。これらのマネージド許可は、Application Auto Scaling によって作成および管理され、各リソースタイプに対して許可されるアクションを定義します。各ロールが付与する許可の詳細については、「AWS Application Auto Scaling の マネージドポリシー」を参照してください。

内容

- サービスリンクロールの作成に必要な許可

- サービスリンクロールを作成する (自動)
- サービスリンクロールを作成する (手動)
- サービスリンクロールを編集する
- サービスリンクロールを削除する
- Application Auto Scaling サービスリンクロールがサポートされるリージョン
- サービスリンクロールの ARN リファレンス

サービスリンクロールの作成に必要な許可

Application Auto Scaling では、 のユーザーが特定のサービスを初めて AWS アカウント 呼び出すときに、サービスにリンクされたロールを作成RegisterScalableTargetするためのアクセス許可が必要です。Application Auto Scaling は、アカウントにターゲットサービス用のサービスリンクロールが既に存在しない場合、そのロールを作成します。サービスリンクロールは Application Auto Scaling に許可を付与して、ユーザーに代わってターゲットサービスを呼び出すことができるようにします。

この自動ロール作成が正常に行われるには、ユーザーが iam:CreateServiceLinkedRole アクションに対する許可を持っている必要があります。

```
"Action": "iam:CreateServiceLinkedRole"
```

以下は、スポットフリート用のサービスリンクロールを作成するアクセス許可を付与するアイデンティティベースのポリシーです。以下にあるように、サービスリンクロールは ARN としてポリシーの Resource フィールドに指定し、サービスリンクロールのサービスプリンシパルは条件として指定できます。各サービスの ARN については、「サービスリンクロールの ARN リファレンス」を参照してください。

JSON

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

```
            "Resource": "arn:aws:iam::*:role/aws-
service-role/ec2.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName":"ec2.application-
autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> `iam:AWSServiceName` IAM 条件キーは、ロールがアタッチされるサービスプリンシパルを指定するもので、このポリシー例では *ec2.application-autoscaling*.amazonaws.com として記述されています。サービスプリンシパルを推測しようとしないでください。サービスのサービスプリンシパルを確認するには、「AWS のサービス Application Auto Scaling で使用できる」を参照してください。

サービスリンクロールを作成する (自動)

サービスリンクロールを手動で作成する必要はありません。Application Auto Scaling は、ユーザーが `RegisterScalableTarget` を呼び出す時に、適切なサービスリンクロールを作成します。例えば、Amazon ECS サービスのオートスケーリングをセットアップする場合は、Application Auto Scaling が `AWSServiceRoleForApplicationAutoScaling_ECSService` ロールを作成します。

サービスリンクロールを作成する (手動)

サービスにリンクされたロールを作成するには、IAM コンソール AWS CLI、または IAM API を使用できます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの作成」を参照してください。

サービスリンクロールの作成 (AWS CLI)

次の create-service-linked-role コマンドを使用して、Application Auto Scaling サービスにリンクされたロールを作成します。リクエストでは、サービス名の「prefix」を指定します。

サービス名のプレフィックスを確認するには、「AWS のサービス Application Auto Scaling で使用できる」セクションで、各サービス用のサービスリンクロールのサービスプリンシパルに関する情報を参照してください。サービス名とサービスプリンシパルは同じプレフィックスを共有します。たとえば、 AWS Lambda サービスにリンクされたロールを作成するには、 を使用しますlambda.application-autoscaling.amazonaws.com。

```
aws iam create-service-linked-role --aws-service-name prefix.application-
autoscaling.amazonaws.com
```

サービスリンクロールを編集する

Application Auto Scaling によって作成されたサービスリンクロールで編集できるのは、それらの説明のみです。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

サービスリンクロールを削除する

サポートされているサービスで Application Auto Scaling を使用しなくなった場合は、対応するサービスリンクロールを削除することをお勧めします。

サービスリンクロールは、関連する AWS リソースを削除した後でしか削除できません。これは、リソースに対する Application Auto Scaling 許可を誤って取り消すことがないようにします。詳細については、スケーラブルリソースのドキュメントを参照してください。例えば、Amazon ECS サービスを削除するには、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS サービスの削除」を参照してください。

IAM を使用して、サービスにリンクされたロールを削除できます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

サービスリンクロールの削除後に `RegisterScalableTarget` を呼び出すと、Application Auto Scaling がそのロールを再度作成します。

Application Auto Scaling サービスリンクロールがサポートされるリージョン

Application Auto Scaling は、サービスが利用可能なすべての AWS リージョンでサービスにリンクされたロールの使用をサポートしています。

サービスリンクロールの ARN リファレンス

次の表に、Application Auto Scaling で動作する各 のサービスにリンク AWS のサービス されたロールの Amazon リソースネーム (ARN) を示します。

サービス	ARN
AppStream 2.0	arn:aws:iam:: *012345678910* :role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
Aurora	arn:aws:iam:: *012345678910* :role/aws-service-role/rds.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster
Comprehend	arn:aws:iam:: *012345678910* :role/aws-service-role/comprehend.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint
DynamoDB	arn:aws:iam:: *012345678910* :role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable
ECS	arn:aws:iam:: *012345678910* :role/aws-service-role/ecs.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService
ElastiCache	arn:aws:iam:: *012345678910* :role/aws-service-role/elasticache.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG
Keyspaces	arn:aws:iam:: *012345678910* :role/aws-service-role/cassandra.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CassandraTable
Lambda	arn:aws:iam:: *012345678910* :role/aws-service-role/lambda.application-autoscaling.amazonaws.com/AWSS

サービス	ARN
	erviceRoleForApplicationAutoScaling_LambdaCon currency
MSK	arn:aws:iam:: *012345678910* :role/aws-service-role/ kafka.application-autoscaling.amazonaws.com/AWSSe rviceRoleForApplicationAutoScaling_KafkaCluster
Neptune	arn:aws:iam:: *012345678910* :role/aws-service-role/ neptune.application-autoscaling.amazonaws.com/ AWSServiceRoleForApplicationAutoScaling_NeptuneC luster
SageMaker AI	arn:aws:iam:: *012345678910* :role/aws-service-role/ sagemaker.application-autoscaling.amazonaws.com/ AWSServiceRoleForApplicationAutoScaling_SageMa kerEndpoint
Spot Fleets	arn:aws:iam:: *012345678910* :role/aws-service-role/ ec2.application-autoscaling.amazonaws.com/AWSServ iceRoleForApplicationAutoScaling_EC2SpotFleet Request
WorkSpaces	arn:aws:iam:: *012345678910* :role/aws-service-role/ workspaces.application-autoscaling.amazonaws.com/ AWSServiceRoleForApplicationAutoScaling_WorkS pacesPool
カスタムリソース	arn:aws:iam:: *012345678910* :role/aws-service-role/cust om-resource.application-autoscaling.amazonaws.com/ AWSServiceRoleForApplicationAutoScaling_CustomRes ource

> **Note**
>
> 指定されたサービスにリンクされたロールがまだ存在しない場合でも、 AWS CloudFormation スタックテンプレートの AWS::ApplicationAutoScaling::ScalableTarget

リソースの RoleARN プロパティにサービスにリンクされたロールの ARN を指定できます。Application Auto Scaling が、そのロールを自動的に作成します。

Application Auto Scaling のアイデンティティベースポリシー例

デフォルトでは、 のまったく新しいユーザー AWS アカウント には、何もするアクセス許可がありません。IAM 管理者は、IAM アイデンティティ (ユーザーやロールなど) に Application Auto Scaling API アクションを実行するアクセス許可を与える IAM ポリシーを作成して割り当てる必要があります。

以下のサンプル JSON ポリシードキュメントを使用して IAM ポリシーを作成する方法については、IAM ユーザーガイドの「[JSON] タブでのポリシーの作成」を参照してください。

内容

- Application Auto Scaling API アクションに必要な許可
- ターゲットサービスと CloudWatch での API アクションに必要な許可
- で作業するためのアクセス許可 AWS Management Console

Application Auto Scaling API アクションに必要な許可

以下のポリシーは、Application Auto Scaling API の呼び出し時に、一般的なユースケースに対して許可を付与します。アイデンティティベースのポリシーを作成するときは、このセクションを参照してください。各ポリシーは、Application Auto Scaling API アクションのすべて、または一部に対するアクセス許可を付与します。また、エンドユーザーがターゲットサービスと CloudWatch に対するアクセス許可があることを確認する必要もあります (詳細については、次のセクションを参照してください)。

以下のアイデンティティベースのポリシーは、すべての Application Auto Scaling API アクションへのアクセス許可を付与します。

JSON

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

```
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:*"
            ],
            "Resource": "*"
        }
    ]
}
```

以下のアイデンティティベースのポリシーは、スケジュールされたアクションではなく、スケーリングポリシーを設定するために必要なすべての Application Auto Scaling API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:RegisterScalableTarget",
              "application-autoscaling:DescribeScalableTargets",
              "application-autoscaling:DeregisterScalableTarget",
              "application-autoscaling:PutScalingPolicy",
              "application-autoscaling:DescribeScalingPolicies",
              "application-autoscaling:DescribeScalingActivities",
              "application-autoscaling:DeleteScalingPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

以下のアイデンティティベースのポリシーは、スケーリングポリシーではなく、スケジュールされたアクションを設定するために必要なすべての Application Auto Scaling API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:RegisterScalableTarget",
              "application-autoscaling:DescribeScalableTargets",
              "application-autoscaling:DeregisterScalableTarget",
              "application-autoscaling:PutScheduledAction",
              "application-autoscaling:DescribeScheduledActions",
              "application-autoscaling:DescribeScalingActivities",
              "application-autoscaling:DeleteScheduledAction"
            ],
            "Resource": "*"
        }
    ]
}
```

ターゲットサービスと CloudWatch での API アクションに必要な許可

ターゲットサービスで Application Auto Scaling を正常に設定して使用するには、Amazon CloudWatch、およびスケーリングを設定する各ターゲットサービスに対するアクセス許可をエンドユーザーに付与する必要があります。以下のポリシーを使用して、ターゲットサービスと CloudWatch での作業に必要な最小限のアクセス許可を付与します。

内容

- AppStream 2.0 フリート
- Aurora レプリカ
- Amazon Comprehend ドキュメントの分類とエンティティ認識のエンドポイント
- DynamoDB テーブルとグローバルセカンダリインデックス
- ECS サービス
- ElastiCache レプリケーショングループ
- Amazon EMR クラスター
- Amazon Keyspaces テーブル

- Lambda 関数
- Amazon Managed Streaming for Apache Kafka (MSK) ブローカーストレージ
- Neptune クラスター
- SageMaker AI エンドポイント
- スポットフリート (Amazon EC2)
- カスタムリソース

AppStream 2.0 フリート

以下のアイデンティティベースのポリシーは、必要とされるすべての AppStream 2.0 および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "appstream:DescribeFleets",
              "appstream:UpdateFleet",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Aurora レプリカ

以下のアイデンティティベースのポリシーは、必要とされるすべての Aurora および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "rds:AddTagsToResource",
              "rds:CreateDBInstance",
              "rds:DeleteDBInstance",
              "rds:DescribeDBClusters",
              "rds:DescribeDBInstances",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Comprehend ドキュメントの分類とエンティティ認識のエンドポイント

以下のアイデンティティベースのポリシーは、必要とされるすべての Amazon Comprehend および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "comprehend:UpdateEndpoint",
              "comprehend:DescribeEndpoint",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
```

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

DynamoDB テーブルとグローバルセカンダリインデックス

以下のアイデンティティベースのポリシーは、必要とされるすべての DynamoDB および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "dynamodb:DescribeTable",
              "dynamodb:UpdateTable",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

ECS サービス

以下のアイデンティティベースのポリシーは、必要とされるすべての ECS および CloudWatch API アクションへのアクセス許可を付与します。

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
              "ecs:DescribeServices",
              "ecs:UpdateService",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

ElastiCache レプリケーショングループ

以下のアイデンティティベースのポリシーは、必要とされるすべての ElastiCache および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "elasticache:ModifyReplicationGroupShardConfiguration",
              "elasticache:IncreaseReplicaCount",
              "elasticache:DecreaseReplicaCount",
              "elasticache:DescribeReplicationGroups",
              "elasticache:DescribeCacheClusters",
              "elasticache:DescribeCacheParameters",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EMR クラスター

以下のアイデンティティベースのポリシーは、必要とされるすべての Amazon EMR および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "elasticmapreduce:ModifyInstanceGroups",
              "elasticmapreduce:ListInstanceGroups",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Keyspaces テーブル

以下のアイデンティティベースのポリシーは、必要とされるすべての Amazon Keyspaces および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "cassandra:Select",
              "cassandra:Alter",
              "cloudwatch:DescribeAlarms",
```

```
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Lambda 関数

以下のアイデンティティベースのポリシーは、必要とされるすべての Lambda および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "lambda:PutProvisionedConcurrencyConfig",
              "lambda:GetProvisionedConcurrencyConfig",
              "lambda:DeleteProvisionedConcurrencyConfig",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Managed Streaming for Apache Kafka (MSK) ブローカーストレージ

以下のアイデンティティベースのポリシーは、必要とされるすべての Amazon MSK および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "kafka:DescribeCluster",
              "kafka:DescribeClusterOperation",
              "kafka:UpdateBrokerStorage",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Neptune クラスター

以下のアイデンティティベースのポリシーは、必要とされるすべての Neptune および CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "rds:AddTagsToResource",
              "rds:CreateDBInstance",
              "rds:DescribeDBInstances",
              "rds:DescribeDBClusters",
              "rds:DescribeDBClusterParameters",
              "rds:DeleteDBInstance",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
```

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

SageMaker AI エンドポイント

次のアイデンティティベースのポリシーは、必要なすべての SageMaker AI および CloudWatch API アクションにアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "sagemaker:DescribeEndpoint",
              "sagemaker:DescribeEndpointConfig",
              "sagemaker:DescribeInferenceComponent",
              "sagemaker:UpdateEndpointWeightsAndCapacities",
              "sagemaker:UpdateInferenceComponentRuntimeConfig",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

スポットフリート (Amazon EC2)

以下のアイデンティティベースのポリシーは、必要とされるすべてのスポットフリートおよび CloudWatch API アクションへのアクセス許可を付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "ec2:DescribeSpotFleetRequests",
              "ec2:ModifySpotFleetRequest",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

カスタムリソース

以下のアイデンティティベースのポリシーは、API Gateway API 実行アクションへのアクセス許可を付与します。また、このポリシーは、必要とされるすべての CloudWatch アクションへのアクセス許可も付与します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "execute-api:Invoke",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

で作業するためのアクセス許可 AWS Management Console

Application Auto Scaling にスタンドアロンコンソールはありません。Application Auto Scaling と統合するほとんどのサービスには、それらのコンソールでスケーリングを設定することを目的とした機能があります。

ほとんどの場合、各サービスは、Application Auto Scaling API アクションへのアクセス許可を含むコンソールへのアクセスを定義する AWS マネージド (事前定義) IAM ポリシーを提供します。詳細については、コンソールを使用するサービスのドキュメントを参照してください。

AWS Management Consoleで特定の Application Auto Scaling アクションを表示して使用するためのきめ細かな許可をユーザーに付与する、独自のカスタム IAM ポリシーを作成することもできます。前のセクションのサンプルポリシーを使用できますが、 AWS CLI または SDK で行われたリクエスト用に設計されています。コンソールではこの機能を実行するために追加の API アクションを使用するので、これらのポリシーは正常に動作しない可能性があります。例えば、ステップスケーリングを設定するには、CloudWatch アラームを作成して管理するための追加の許可がユーザーに必要となる場合があります。

> **Tip**
>
> コンソールでタスクを実行するために必要な API アクションを探すには、 AWS CloudTrail などのサービスを使用できます。詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

以下のアイデンティティベースのポリシーは、スポットフリートのスケーリングポリシーを設定するためのアクセス許可を付与します。スポットフリートの IAM アクセス許可に加えて、Amazon EC2 コンソールからフリートスケーリング設定にアクセスするコンソールユーザーには、動的スケーリングをサポートするサービスに対する適切なアクセス許可が必要です。

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ec2:DescribeSpotFleetRequests",
                "ec2:ModifySpotFleetRequest",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-
service-role/ec2.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName":"ec2.application-
autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

このポリシーは、コンソールユーザーが Amazon EC2 コンソールでスケーリングポリシーを表示して変更し、CloudWatch コンソールで CloudWatch アラームを作成して管理することを可能にします。

API アクションを調整して、ユーザーアクセスを制限できます。例えば、`application-autoscaling:*` を `application-autoscaling:Describe*` に置き換えると、ユーザーには読み取り専用アクセスが与えられます。

また、必要に応じて CloudWatch 許可を調整して、CloudWatch 機能へのユーザーアクセスを制限することもできます。詳細については、Amazon CloudWatch ユーザーガイドの「CloudWatch コンソールへのアクセス許可」を参照してください。

Application Auto Scaling へのアクセスのトラブルシューティング

Application Auto Scaling の使用時に `AccessDeniedException` または同様の問題が発生する場合は、このセクションの情報を参考にしてください。

Application Auto Scaling でアクションを実行する権限がありません

AWS API オペレーションを呼び出す`AccessDeniedException`ときに を受け取った場合、使用している AWS Identity and Access Management (IAM) 認証情報に、その呼び出しを行うために必要なアクセス許可がないことを意味します。

以下のサンプルエラーは、`mateojackson` ユーザーがスケーラブルターゲットに関する詳細を表示しようとしているが、`application-autoscaling:DescribeScalableTargets` アクセス許可を持っていないという場合に発生します。

```
An error occurred (AccessDeniedException) when calling the DescribeScalableTargets
 operation: User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: application-autoscaling:DescribeScalableTargets
```

このエラー、または同様のエラーが発生する場合は、管理者に問い合わせてサポートを受ける必要があります。

アカウントの管理者は、Application Auto Scaling がターゲットサービスと CloudWatch のリソースへのアクセスに使用するすべての API アクションに対するアクセス許可があることを確認する必要があります。どのリソースで作業を行っているかに応じて、異なる許可が必要になります。Application Auto Scaling には、ユーザーが所定のリソースに対するスケーリングを初めて設定するときにサービスリンクロールを作成するための許可も必要です。

管理者ですが、IAM ポリシーからエラーが返される、またはポリシーが期待どおりに動作しません

Application Auto Scaling アクションに加えて、IAM ポリシーは、ターゲットサービスと CloudWatch を呼び出すためのアクセス権も付与する必要があります。ユーザーまたはアプリケーションがこれらの追加のアクセス許可を持っていない場合、それらのアクセスが予期せず拒否される可能性があります。アカウントのユーザーとアプリケーション用の IAM ポリシーを記述するには、「Application Auto Scaling のアイデンティティベースポリシー例」の情報を参考にしてください。

検証の実行方法については、「ターゲットリソースでの Application Auto Scaling API コールに対するアクセス許可の検証」を参照してください。

一部の許可問題は、Application Auto Scaling が使用するサービスリンクロールの作成に関する問題に起因する可能性があることに注意してください。これらのサービスリンクロールの作成については、「Application Auto Scaling 用のサービスリンクロール」を参照してください。

ターゲットリソースでの Application Auto Scaling API コールに対するアクセス許可の検証

Application Auto Scaling API アクションに対して承認されたリクエストを行うには、API 呼び出し元がターゲットサービスと CloudWatch の AWS リソースにアクセスするためのアクセス許可を持っている必要があります。Application Auto Scaling は、リクエストを続行する前に、ターゲットサービスと CloudWatch の両方に関連付けられているリクエストに対する許可を検証します。これを行うには、一連のコールを発行してターゲットリソースに対する IAM 許可を検証します。レスポンスが返されると、Application Auto Scaling がそのレスポンスを読み取ります。IAM 許可が所定のアクションが許可しない場合、Application Auto Scaling はリクエストを失敗させ、欠落している許可に関する情報が含まれたエラーをユーザーに返します。これは、ユーザーがデプロイするスケーリング設定が意図したとおりに機能することと、リクエストが失敗した場合に有用なエラーが返されることを確実にします。

以下の情報は、この仕組みの例として、Application Auto Scaling が Aurora と CloudWatch で許可の検証を実行する方法を詳しく説明します。

ユーザーが Aurora DB クラスターに対して `RegisterScalableTarget` API を呼び出すと、Application Auto Scaling は以下のすべてのチェックを実行してユーザーに必要なアクセス許可 (太字) があることを確認します。

- rds:CreateDBInstance: ユーザーにこの許可があるかどうかを判断するため、`CreateDBInstance` API オペレーションにリクエストを送信して、ユーザーが指定した Aurora DB クラスターで無効

なパラメータ (空のインスタンス ID) を使った DB インスタンスの作成を試みます。許可があるユーザーの場合、API は、リクエストを監査した後で InvalidParameterValue エラーコードレスポンスを返します。しかし、許可がないユーザーの場合は、AccessDenied エラーが発生し、欠落している許可がリストされた、ユーザーへの ValidationException エラーを伴って Application Auto Scaling リクエストが失敗します。

- rds:DeleteDBInstance: DeleteDBInstance API オペレーションに空のインスタンス ID を送信します。許可があるユーザーの場合、このリクエストの結果は InvalidParameterValue エラーになります。許可がないユーザーの場合は、結果が AccessDenied になり、ユーザーに検証例外が送信されます (最初の箇条書きで説明されているものと同じ対応)。
- rds:AddTagsToResource: AddTagsToResource API オペレーションには Amazon リソースネーム (ARN) が必要であるため、無効なアカウント ID (12345) とダミーインスタンス ID (non-existing-db) を使用した「ダミー」リソースを指定して ARN (arn:aws:rds:us-east-1:12345:db:non-existing-db) を作成する必要があります。許可があるユーザーの場合、このリクエストの結果は InvalidParameterValue エラーになります。許可がないユーザーの場合は、結果が AccessDenied になり、ユーザーに検証例外が送信されます
- rds:DescribeDBClusters: 自動スケーリングに登録されているリソースのクラスター名を記述します。許可があるユーザーの場合、有効な記述結果が得られます。許可がないユーザーの場合は、結果が AccessDenied になり、ユーザーに検証例外が送信されます
- rds:DescribeDBInstances: スケーラブルターゲットを登録するためにユーザーが提供したクラスター名をフィルタリングする db-cluster-id フィルターを使って、DescribeDBInstances API を呼び出します。許可があるユーザーの場合、DB クラスター内のすべての DB インスタンスを記述することが許可されます。許可がないユーザーの場合は、この呼び出しの結果が AccessDenied になり、ユーザーに検証例外が送信されます
- cloudwatch:PutMetricAlarm: パラメータなしで PutMetricAlarm API を呼び出します。アラーム名が欠落しているため、リクエストの結果は、許可があるユーザーに対する ValidationError になります。許可がないユーザーの場合は、結果が AccessDenied になり、ユーザーに検証例外が送信されます
- cloudwatch:DescribeAlarms: 最大レコード数の値を 1 に設定して DescribeAlarms API を呼び出します。許可があるユーザーの場合、レスポンスに 1 つのアラームに関する情報があることを期待できます。許可がないユーザーの場合は、この呼び出しの結果が AccessDenied になり、ユーザーに検証例外が送信されます
- cloudwatch:DeleteAlarms: 上記の PutMetricAlarm と同じく、DeleteAlarms リクエストにパラメータを指定しません。リクエストにアラーム名がないため、この呼び出しは、許可があるユーザーに対する ValidationError を伴って失敗します。許可がないユーザーの場合は、結果が AccessDenied になり、ユーザーに検証例外が送信されます

これらの検証例外は、そのうちのどれかが発生するたびにログに記録されます。を使用して、検証に失敗した呼び出しを手動で特定する手順を実行できます AWS CloudTrail。詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

> **Note**
>
> CloudTrail を使用して Application Auto Scaling イベントのアラートを受信した場合、これらのアラートには、ユーザーアクセス許可を検証するための Application Auto Scaling に対するコールがデフォルトで含まれています。これらのアラートを除外する場合は、これらの検証チェックのための `application-autoscaling.amazonaws.com` が含まれている `invokedBy` フィールドを使用します。

インターフェイス VPC エンドポイントを使用して Application Auto Scaling にアクセスする

を使用して AWS PrivateLink 、VPC と Application Auto Scaling の間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続を使用せずに、VPC 内にあるかのように Application Auto Scaling にアクセスできます。VPC 内のインスタンスは Application Auto Scaling にアクセスするためのパブリック IP アドレスを必要としません。

このプライベート接続を確立するには、 AWS PrivateLinkを利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、Application Auto Scaling 宛てのトラフィックのエントリポイントとして機能するリクエスタ管理型ネットワークインターフェイスです。

詳細については、「 AWS PrivateLink ガイド」の「Access AWS のサービス through AWS PrivateLink」を参照してください。

内容

- インターフェイス VPC エンドポイントを作成する
- VPCエンドポイントポリシーを作成する

インターフェイス VPC エンドポイントを作成する

Application Auto Scaling 用のエンドポイントは、以下のサービス名を使用して作成します。

```
com.amazonaws.region.application-autoscaling
```

詳細については、「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

その他の設定を変更する必要はありません。Application Auto Scaling は AWS 、サービスエンドポイントまたはプライベートインターフェイス VPC エンドポイントのいずれかを使用して、他のサービスを呼び出します。

VPCエンドポイントポリシーを作成する

Application Auto Scaling API へのアクセスを制御するために、VPC エンドポイントにポリシーをアタッチすることができます。このポリシーでは以下の内容を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

以下の例では、エンドポイントを介してスケーリングポリシーを削除するためのアクセス許可を全員に対して拒否する VPC エンドポイントポリシーを示しています。このポリシー例では、他のすべてのアクションを実行するアクセス許可も全員に付与しています。

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "application-autoscaling:DeleteScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

詳細については、AWS PrivateLink ガイドの「VPC エンドポイントポリシー」を参照してください。

Application Auto Scaling の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。

AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンと分離されたアベイラビリティーゾーンを提供します。

アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Application Auto Scaling のインフラストラクチャセキュリティ

マネージドサービスである Application Auto Scaling は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で Application Auto Scaling にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。またはAWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Application Auto Scaling のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス「コンプライアンスプログラムによる対象範囲内」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「 Compliance ProgramsAssurance」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、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の機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべての AWS のサービス が HIPAA の対象となるわけではありません。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティコントロールを保護および AWS のサービス マッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に把握できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポー

トされているサービスとコントロールの一覧については、Security Hub のコントロールリファレンスを参照してください。

- Amazon GuardDuty – 不審なアクティビティや悪意のあるアクティビティがないか環境をモニタリングすることで AWS アカウント、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Application Auto Scaling のクォータ

AWS アカウント には、それぞれ の制限と呼ばれるデフォルトのクォータがあります AWS のサービス。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、その他のクォータについては引き上げることはできません。

Application Auto Scaling のクォータを表示するには、Service Quotas コンソールを開きます。ナビゲーションペインで、[AWS サービス] 、[Application Auto Scaling] の順に選択します。

クォータの引き上げをリクエストするには、Service Quotas ユーザーガイド の「クォータ引き上げリクエスト」を参照してください。

AWS アカウント には、Application Auto Scaling に関連する次のクォータがあります。

名前	デフォルト	引き上げ可能
リソースタイプごとのスケーラブルターゲット	Amazon DynamoDB: 5,000 | Amazon ECS: 3,000 | Amazon Keyspaces: 1,500 | その他のリソースタイプ: 500	はい
スケーラブルターゲットあたりのスケーリングポリシー (ステップスケーリングポリシーおよびターゲット追跡ポリシー)	50	いいえ
スケーラブルなターゲットあたりのスケジュールされたアクション	200	いいえ
ステップスケーリングポリシーあたりのステップ調整	20	はい

ワークロードをスケールアウトする際は、サービスのクォータを念頭に置いてください。例えば、サービスで許可されるキャパシティーユニットの最大数に達すると、スケールアウトは停止します。需要が低下し、現行の容量が減少すると、Application Auto Scaling が再びスケールアウトできるようになります。この容量制限に再度到達しないようにするために、引き上げをリクエストします。各サービスには、リソースの最大容量に対する独自のデフォルトのクォータがあります。その他の Amazon Web Services のデフォルトクォータについては、「Amazon Web Services 全般のリファレンス」の「サービスのエンドポイントとクォータ」を参照してください。

Application Auto Scaling のドキュメント履歴

以下の表は、2018 年 1 月以降の Application Auto Scaling ドキュメントへの重要な追加項目をまとめたものです。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

変更	説明	日付
ElastiCache Memcached クラスターのサポートを追加	Application Auto Scaling を使用して、Memcached クラスターのノード数を水平方向にスケーリングします。詳細については、ElastiCache と Application Auto Scaling」を参照してください。	2025 年 4 月 10 日
AWS マネージドポリシーの更新	Application Auto Scaling が`AWSApplicationAutoscalingElastiCacheRGPolicy` ポリシーを更新しました。	2025 年 4 月 10 日
ガイドの変更点	Application Auto Scaling ユーザーガイドの新しいトピックは、Application Auto Scaling で予測スケーリングの使用を開始するのに役立ちます。「Application Auto Scaling 予測スケーリング」を参照してください。	2024 年 11 月 21 日
AWS マネージドポリシーの更新	Application Auto Scaling が`AWSApplicationAutoscalingECSServicePolicy` ポリシーを更新しました。	2024 年 11 月 21 日

WorkSpaces プールのサポートを追加する	Application Auto Scaling を使用して WorkSpaces のプールをスケールします。詳細については、「Amazon WorkSpaces とApplication Auto Scaling」を参照してください。トピック Application Auto Scaling が更新される AWS マネージドポリシーが更新され、WorkSpaces との統合に関する新しい管理ポリシーが一覧表示されました。	2024 年 6 月 27 日
ガイドの変更点	クォータに関するドキュメントのリソースタイプエントリあたりのスケーラブルターゲットの最大数が更新されました。「Quotas for Application Auto Scaling」(Application Auto Scaling のクォータ) を参照してください。	2024 年 1 月 16 日
SageMaker AI 推論コンポーネントのサポート	Application Auto Scaling を使用して、推論コンポーネントのコピーの数をスケールします。	2023 年 11 月 29 日
AWS マネージドポリシーの更新	Application Auto Scaling が`AWSApplicationAutoscalingSageMakerEndpointPolicy` ポリシーを更新しました。	2023 年 11 月 13 日

SageMaker AI Serverless プロビジョニングされた同時実行のサポート	Application Auto Scaling を使用して、サーバーレスエンドポイントのプロビジョニングされた同時実行数をスケールします。	2023 年 5 月 9 日
タグを使用してスケーラブルなターゲットを分類する	これで、Application Auto Scaling スケーラブルターゲットにメタデータをタグ形式で割り当てることができます。「Tagging support for Application Auto Scaling」(Application Auto Scaling のタグ付けサポート) を参照してください。	2023 年 3 月 20 日
CloudWatch の Metric Math のサポート	ターゲット追跡スケーリングポリシーの作成時に Metric Math を使用できるようになりました。Metric Math により、複数の CloudWatch メトリクスをクエリし、数式を使用して、これらのメトリクスに基づく新しい時系列を作成できます。Metric Math を使用して、Application Auto Scaling のターゲット追跡スケーリングポリシーを作成するを参照してください。	2023 年 3 月 14 日

スケーリングしない理由	Application Auto Scaling API を使用して、Application Auto Scaling がリソースをスケーリングしない理由を機械的に読み取りできるようになりました。「Scaling activities for Application Auto Scaling」(Application Auto Scaling のスケーリングアクティビティ) を参照してください。	2023 年 1 月 4 日
ガイドの変更点	クォータに関するドキュメントのリソースタイプエントリあたりのスケーラブルターゲットの最大数が更新されました。「Quotas for Application Auto Scaling」(Application Auto Scaling のクォータ) を参照してください。	2022 年 5 月 6 日
Amazon Neptune クラスターのサポートを追加	アプリケーションの Auto Scaling を使用して、Amazon Neptune DB クラスター内のレプリカ数をスケーリングします。詳細については、「Amazon Neptune と Application Auto Scaling」を参照してください。。トピック Application Auto Scaling が更新される AWS マネージドポリシーが更新され、Neptune との統合に関する新しい管理ポリシーが一覧表示されました。	2021 年 10 月 6 日

Application Auto Scaling が AWS マネージドポリシーの変更を報告するようになりました	2021 年 8 月 19 日以降、マネージドポリシーへの変更は、「Application Auto Scaling による AWS マネージドポリシーの更新」のトピックで報告されます。リストされている最初の変更は、ElastiCache (Redis OSS) に必要なアクセス許可の追加です。	2021 年 8 月 19 日
ElastiCache (Redis OSS) レプリケーショングループのサポートを追加	Application Auto Scaling を使用して、ElastiCache (Redis OSS) レプリケーショングループ (クラスター) のノードグループ数と、ノードグループあたりのレプリカ数をスケールします。詳細については、「ElastiCache (Redis OSS) と Application Auto Scaling」を参照してください。	2021 年 8 月 19 日

ガイドの変更点	Application Auto Scaling ユーザーガイドの新しい IAM トピックは、Application Auto Scaling へのアクセスのトラブルシューティングに役立ちます。詳細については、「Application Auto Scaling の Identity and Access Management」を参照してください。また、ターゲットサービスと Amazon CloudWatch でのアクションに対する新しい IAM 許可ポリシーの例も追加されました。詳細については、「 AWS CLI または SDK を操作するためのポリシーの例」を参照してください。	2021 年 2 月 23 日
ローカルタイムゾーンのサポートを追加	ローカルタイムゾーンでスケジュールされたアクションを作成できるようになりました。タイムゾーンが夏時間を実施する場合は、夏時間 (DST) に合わせて自動的に調整されます。詳細については、「スケジュールされたスケーリング」を参照してください。	2021 年 2 月 2 日

ガイドの変更点	Application Auto Scaling ユーザーガイドの新しいチュートリアルは、Application Auto Scaling の使用時に、ターゲット追跡スケーリングポリシーとスケジュールされたスケーリングを使用してアプリケーションの可用性を向上させる方法を理解するために役立ちます。	2020 年 10 月 15 日
Amazon Managed Streaming for Apache Kafka クラスターストレージのサポートを追加	ターゲット追跡スケーリングポリシーを使用して、Amazon MSK クラスターに関連付けられているブローカーストレージの量をスケールアウトします。	2020 年 9 月 30 日
Amazon Comprehend エンティティ認識器エンドポイントのサポートを追加	Application Auto Scaling を使用して、Amazon Comprehend エンティティ認識器エンドポイントにプロビジョニングされた推論単位の数をスケールします。	2020 年 9 月 28 日
Amazon Keyspaces (Apache Cassandra 用) テーブルのサポートを追加	Application Auto Scaling を使用して、Amazon Keyspaces テーブルのプロビジョニングされたスループット (読み込みキャパシティーと書き込みキャパシティー) をスケールします。	2020 年 4 月 23 日

新しい「セキュリティ」章	Application Auto Scaling ユーザーガイドの新しい「セキュリティ」章は、Application Auto Scaling の使用時に責任共有モデルを適用する方法を理解するために役立ちます。この更新の一環として、ユーザーガイドの「認証とアクセスコントロール」章が、新しいより有益な「Application Auto Scaling の Identity and Access Management」セクションに置き換えられました。	2020 年 1 月 16 日
マイナーな更新	さまざまな改善と修正。	2020 年 1 月 15 日
通知機能の追加	Application Auto Scaling は、特定のアクションが発生した AWS Health Dashboard ときにイベントを Amazon EventBridge に送信し、 に通知を送信するようになりました。詳細については、「Application Auto Scaling のモニタリング」を参照してください。	2019 年 12 月 20 日
AWS Lambda 関数のサポートを追加する	Application Auto Scaling を使用して、Lambda 関数のプロビジョニングされた同時実行数をスケールします。	2019 年 12 月 3 日

Amazon Comprehend ドキュメント分類エンドポイントのサポートを追加	Application Auto Scaling を使用して、Amazon Comprehend ドキュメント分類エンドポイントのスループット容量をスケールします。	2019 年 11 月 25 日
ターゲット追跡スケーリングポリシーに AppStream 2.0 サポートを追加	ターゲット追跡スケーリングポリシーを使用して、AppStream 2.0 フリートのサイズをスケールします。	2019 年 11 月 25 日
Amazon VPC エンドポイントのサポート	VPC と Application Auto Scaling の間でプライベート接続を確立できるようになりました。移行の考慮事項と手順については、「Application Auto Scaling とインターフェイス VPC エンドポイント」を参照してください。	2019 年 11 月 22 日
スケーリングの一時停止と再開	スケーリングの中断と再開のサポートが追加されました。詳細については、「Application Auto Scaling のスケーリングの一時停止と再開」を参照してください。	2019 年 8 月 29 日
ガイドの変更点	Application Auto Scaling ドキュメントの「スケジュールされたスケーリング」、「ステップスケーリングポリシー」、および「ターゲット追跡スケーリングポリシー」の各セクションを改善しました。	2019 年 3 月 11 日

カスタムリソースのサポートを追加	Application Auto Scaling を使用して、独自のアプリケーションまたはサービスによって提供されるカスタムリソースをスケールします。詳細については、GitHub リポジトリを参照してください。	2018 年 7 月 9 日
SageMaker AI エンドポイントバリアントのサポートを追加	Application Auto Scaling を使用して、バリアントに対してプロビジョニングされたエンドポイントインスタンスの数をスケールします。	2018 年 2 月 28 日

以下の表は、2018 年 1 月までの Application Auto Scaling ドキュメントへの重要な変更をまとめたものです。

変更	説明	日付
Aurora レプリカのサポートを追加	Application Auto Scaling を使用して、希望数をスケールします。詳細については、Amazon RDS ユーザーガイドの「Aurora レプリカでの Amazon Aurora Auto Scaling の使用」を参照してください。	2017 年 11 月 17 日
スケジュールに基づくスケーリングのサポートを追加	スケジュールに基づくスケーリングを使用して、事前設定された特定の日時または間隔でリソースをスケールします。詳細については、「Application Auto Scaling のスケジュールされたスケーリング」を参照してください。	2017 年 11 月 8 日

変更	説明	日付
ターゲット追跡スケーリングポリシーのサポートを追加	ターゲットの追跡スケーリングポリシーを使用して、いくつかの簡単なステップでアプリケーションの動的スケーリングをセットアップします。詳細については、「Application Auto Scaling のターゲット追跡スケーリングポリシー」を参照してください。	2017 年 7 月 12 日
DynamoDB のテーブルとグローバルセカンダリインデックスのプロビジョニングされた読み込みキャパシティーと書き込みキャパシティーのサポートを追加	Application Auto Scaling を使用して、プロビジョニングされたスループット (読み込みキャパシティーと書き込みキャパシティー) をスケールします。詳細については、Amazon DynamoDB デベロッパーガイドの「DynamoDB Auto Scaling によるスループット容量の管理」を参照してください。	2017 年 6 月 14 日
AppStream 2.0 フリートのサポートを追加	Application Auto Scaling を使用して、フリートのサイズをスケールします。詳細については、Amazon AppStream 2.0 管理ガイドの「AppStream 2.0 向け Fleet Auto Scaling」を参照してください。	2017 年 3 月 23 日

変更	説明	日付
Amazon EMR クラスターのサポートを追加	Application Auto Scaling を使用して、コアノードとタスクノードをスケールします。詳細については、Amazon EMR 管理ガイドの「Using automatic scaling in Amazon Neptune」を参照してください。	2016 年 11 月 18 日
スポットフリートのサポートを追加	Application Auto Scaling を使用して、ターゲット容量をスケールします。詳細については、Amazon EC2 ユーザーガイドの「スポットフリートの自動スケーリング」を参照してください。	2016 年 9 月 1 日
Amazon ECS サービスのサポートを追加	Application Auto Scaling を使用して、希望数をスケールします。詳細については、Amazon Elastic Container Service デベロッパーガイドの「サービスのオートスケーリング」を参照してください。	2016 年 8 月 9 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。