



ユーザーガイド

Application Auto Scaling



Application Auto Scaling: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

| | |
|---|----|
| Application Auto Scaling とは | 1 |
| Application Auto Scaling の特徴 | 2 |
| Application Auto Scaling と連携 | 2 |
| 概念 | 3 |
| 詳細情報 | 5 |
| 統合するサービス | 6 |
| Amazon AppStream 2.0 | 9 |
| サービスリンクロール | 9 |
| サービスプリンシパル | 9 |
| スケーラブルターゲットとしての AppStream 2.0 フリーの Application Auto Scaling への登録 | 9 |
| 関連リソース | 10 |
| Amazon Aurora | 10 |
| サービスリンクロール | 11 |
| サービスプリンシパル | 11 |
| 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 | 11 |
| 関連リソース | 12 |
| Amazon Comprehend | 12 |
| サービスリンクロール | 12 |
| サービスプリンシパル | 13 |
| 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 | 13 |
| 関連リソース | 14 |
| Amazon DynamoDB | 15 |
| サービスリンクロール | 15 |
| サービスプリンシパル | 15 |
| 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 | 15 |
| 関連リソース | 18 |
| Amazon ECS | 18 |
| サービスリンクロール | 18 |
| サービスプリンシパル | 19 |
| 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 | 19 |

| | |
|--|----|
| 関連リソース | 20 |
| Amazon ElastiCache | 20 |
| サービスリンクロール | 20 |
| サービスプリンシパル | 21 |
| スケーラブルターゲットとしての ElastiCache Redis レプリケーショングループの Application Auto Scaling への登録 | 21 |
| 関連リソース | 22 |
| Amazon Keyspaces (Apache Cassandra 向け) | 23 |
| サービスリンクロール | 23 |
| サービスプリンシパル | 23 |
| 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 | 23 |
| 関連リソース | 25 |
| AWS Lambda | 25 |
| サービスリンクロール | 25 |
| サービスプリンシパル | 25 |
| 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 | 26 |
| 関連リソース | 27 |
| Amazon Managed Streaming for Apache Kafka (MSK) | 27 |
| サービスリンクロール | 27 |
| サービスプリンシパル | 27 |
| 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 | 28 |
| 関連リソース | 29 |
| Amazon Neptune | 29 |
| サービスリンクロール | 29 |
| サービスプリンシパル | 29 |
| 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 | 30 |
| 関連リソース | 31 |
| Amazon SageMaker | 31 |
| サービスリンクロール | 31 |
| サービスプリンシパル | 31 |
| スケーラブルターゲットとしての SageMaker エンドポイントバリエーションの Application Auto Scaling への登録 | 31 |

| | |
|--|----|
| 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 | 33 |
| 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 | 34 |
| 関連リソース | 34 |
| スポットフリート (Amazon EC2) | 35 |
| サービスリンクロール | 35 |
| サービスプリンシパル | 36 |
| 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 | 36 |
| 関連リソース | 37 |
| カスタムリソース | 37 |
| サービスリンクロール | 37 |
| サービスプリンシパル | 37 |
| 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 | 38 |
| 関連リソース | 39 |
| を使用してスケーリングを設定する AWS CloudFormation | 40 |
| Application Auto Scaling と AWS CloudFormation テンプレート | 40 |
| サンプルテンプレートスニペット | 41 |
| の詳細 AWS CloudFormation | 41 |
| スケジュールされたスケーリング | 42 |
| スケジュールされたスケーリングのしくみ | 43 |
| 仕組み | 43 |
| 考慮事項 | 43 |
| よく使われるコマンド | 44 |
| 関連リソース | 45 |
| 制限事項 | 45 |
| cron 式の使用 | 46 |
| スケジュールされたアクションの例 | 48 |
| 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 | 49 |
| 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 | 50 |
| 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 | 51 |
| 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 | 52 |
| 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 | 53 |
| スケジュールされたスケーリングを管理する | 54 |
| 指定されたサービスのスケーリングアクティビティを表示する | 54 |
| 指定されたサービスに対するすべてのスケジュールされたアクションの記述 | 56 |

| | |
|--|----|
| スケーラブルターゲットに対する 1 つまたは複数のスケジュールされたアクションを記述する | 58 |
| 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ルリングをオフにする | 59 |
| スケジュールされたアクションの削除 | 60 |
| チュートリアル: AWS CLIを使用したスケジュールに基づくスケールリングの開始方法 | 61 |
| ステップ 1: スケーラブルターゲットを登録する | 61 |
| 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 | 63 |
| ステップ 3: スケールリングアクティビティを表示する | 66 |
| ステップ 4: 次のステップ | 69 |
| ステップ 5: クリーンアップ | 70 |
| ターゲット追跡スケールリングポリシー | 72 |
| ターゲット追跡の仕組み | 73 |
| 仕組み | 73 |
| メトリクスを選択する | 75 |
| ターゲット値の定義 | 76 |
| クールダウン期間を定義する | 76 |
| 考慮事項 | 78 |
| 複数のスケールリングポリシー | 78 |
| よく使われるコマンド | 79 |
| 関連リソース | 80 |
| 制限事項 | 80 |
| ターゲット追跡スケールリングポリシーを作成する | 80 |
| スケーラブルターゲットを登録する | 81 |
| ターゲット追跡スケールリングポリシーを作成する | 82 |
| ターゲット追跡スケールリングポリシーを記述する | 84 |
| ターゲット追跡スケールリングポリシーを削除する | 85 |
| Metric Math を使用する | 86 |
| 例: タスクごとの Amazon SQS キューバックログ | 87 |
| 制限事項 | 91 |
| ステップスケールリングポリシー | 92 |
| ステップスケールリングの仕組み | 93 |
| 仕組み | 93 |
| ステップ調整値 | 94 |
| スケールリング調整タイプ | 96 |
| クールダウン期間 | 97 |
| よく使われるコマンド | 98 |

| | |
|---|-----|
| 考慮事項 | 99 |
| 関連リソース | 45 |
| 制限事項 | 99 |
| ステップスケーリングポリシーを作成する | 99 |
| スケーラブルターゲットを登録する | 100 |
| ステップスケーリングポリシーを作成する | 101 |
| スケーリングポリシーを呼び出すアラームを作成する | 104 |
| ステップスケーリングポリシーを記述する | 105 |
| ステップスケーリングポリシーを削除する | 106 |
| 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 | 108 |
| 前提条件 | 109 |
| ステップ 1: スケーラブルターゲットを登録する | 109 |
| 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 | 110 |
| 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 | 114 |
| ステップ 4: 次のステップ | 116 |
| ステップ 5: クリーンアップ | 117 |
| スケーリングを一時停止 | 119 |
| スケーリングアクティビティ | 119 |
| スケーリングアクティビティの一時停止と再開 | 120 |
| 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 | 123 |
| スケーリングアクティビティを再開する | 124 |
| スケーリングアクティビティ | 125 |
| スケーラブルターゲットでスケーリングアクティビティを検索する | 125 |
| スケーリングされていないアクティビティを含める | 126 |
| 理由コード | 128 |
| モニタリング | 131 |
| を使用して をモニタリングする CloudWatch | 132 |
| CloudWatch リソースの使用状況をモニタリングするための メトリクス | 133 |
| 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 | 147 |
| AWS CloudTrail | 150 |
| の Application Auto Scaling 情報 CloudTrail | 150 |
| Application Auto Scaling のログファイルエントリを理解する | 151 |
| | 151 |
| 関連リソース | 153 |
| Amazon EventBridge | 153 |
| Application Auto Scaling イベント | 153 |

| | |
|---|------|
| タグ付けのサポート | 158 |
| タグ付けの例 | 158 |
| セキュリティ用のタグ | 159 |
| タグへのアクセスを制御する | 160 |
| セキュリティ | 162 |
| データ保護 | 163 |
| Identity and Access Management | 163 |
| アクセスコントロール | 164 |
| Application Auto Scaling で IAM が機能する仕組み | 164 |
| AWS マネージドポリシー | 171 |
| サービスリンクロール | 181 |
| アイデンティティベースポリシー例 | 186 |
| トラブルシューティング | 199 |
| アクセス許可の検証 | 200 |
| AWS PrivateLink | 202 |
| インターフェイス VPC エンドポイントを作成する | 202 |
| VPCエンドポイントポリシーを作成する | 203 |
| 耐障害性 | 203 |
| インフラストラクチャセキュリティ | 204 |
| コンプライアンス検証 | 204 |
| クォータ | 207 |
| ドキュメント履歴 | 209 |
| | CCXX |

Application Auto Scaling とは

Application Auto Scaling は、Amazon EC2 以外の個々の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デベロッパーやシステム管理者向けのウェブ AWS 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せん。

- 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 Redis クラスター (レプリケーショングループ) 用の
- 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 エンドポイントバリエーション
- SageMaker 推論コンポーネント
- SageMaker サーバーレス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
- 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 いずれかの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」](#)の」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リングします。
- 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。

Application Auto Scaling と連携

スケール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ている場合は、にサインインして Application Auto Scaling にアクセスします 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。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) — さまざまな のコマンドセットを提供し AWS のサービス、Windows、macOS、Linux でサポートされています。開始するには、[AWS Command Line Interface](#) を参照してください。コマンドのリストについては、「[コマンドAWS CLI 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、幅広い AWS 製品セット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 Windows 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 Cmdlet Reference](#)」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処理、エラー処理など、接続の詳細の多くを処理します。詳細については、「[で構築するツール 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の設定をサポートします。詳細については、「[を使用して Application Auto Scaling リソースを設定する AWS CloudFormation](#)」を参照してください。

にプログラムで接続するには 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、「[シークレットリージョンユーザーガイド Application Auto Scaling エンドポイントとクォータ](#)」AWS 全般のリファレンス。

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 が認識していない、この範囲外の基盤となるリソースに直接変更を加える out-of-band 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。スケール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。スケール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。

スケールリングポリシー

スケールリングポリシーは、特定の CloudWatch メトリクスを追跡するように Application Auto Scaling 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ル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。

自動スケール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発行 CloudWatch してスケールリングポリシーで使用することもできます。

スケールリングアクティビティ間のクールダウン期間は、別のスケール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、スケールリングポリシーが、必要に応じて別のスケール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、スケール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。オートスケーリングは、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 の設定でサポートされている方法に関する情報を含め、Application Auto Scaling で使用できる AWS の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、オート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、オート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、「[を使用して Application Auto Scaling リソースを設定する AWS CloudFormation](#)」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 |  いえ はい |  はい はい |  はい はい |  はい はい |
| Amazon DynamoDB |  はい はい |  はい はい |  はい はい |  はい はい |
| Amazon ECS |  はい はい |  はい はい |  はい はい |  はい はい |
| Amazon ElastiCache |  はい はい |  はい はい |  はい はい |  はい はい |
| Amazon EMR |  はい はい |  はい はい |  はい はい |  はい はい |
| Amazon Keyspaces |  はい はい |  はい はい |  はい はい |  はい はい |
| Lambda |  いえ はい |  はい はい |  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 MSK |  はい |  はい |  はい |  はい |
| Amazon Neptune |  いえ |  はい |  はい |  はい |
| SageMaker |  はい |  はい |  はい |  はい |
| スポットフリート |  はい |  はい |  はい |  はい |
| カスタムリソース |  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、Amazon AppStream 2.0 とスポットフリートのみが ElastiCache、スケジュールされたスケーリングのためのコンソールアクセスを提供します。

Amazon AppStream 2.0 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、およびスケジュールされたスケーリングを使用して、AppStream 2.0 フリートをスケーリングできます。

次の情報は、AppStream 2.0 と Application Auto Scaling の統合に役立ちます。

AppStream 2.0 用に作成されたサービスにリンクされたロール

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、に以下の[サービスリンクロール](#)が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 フリーの Application Auto Scaling への登録

Application Auto Scaling では、AppStream 2.0 フリーのスケール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。

AppStream 2.0 コンソールを使用して自動スケーリングを設定すると、AppStream 2.0 は自動的にスケーラブルターゲットを登録します。

AWS CLI または AWS SDKs のいずれか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

Application Auto Scaling を始めたばかりの場合は、次のドキュメントで AppStream 2.0 リソースのスケールリングに関するその他の有用な情報を確認できます。

「Amazon [AppStream 2.0 管理ガイド](#)」の「[Fleet Auto Scaling for 2.0 AppStream](#)」

Amazon Aurora と Application Auto Scaling

Aurora DB クラスターは、ターゲット追跡スケールリングポリシー、ステップスケールリングポリシー、およびスケジュールされたスケール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、に以下の[サービスリンクロール](#)が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。

AWS CLI または AWS SDKs のいずれか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 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 レプリカの数に登録します。

```
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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントで Aurora 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 RDS ユーザーガイドの「[Aurora レプリカでの Amazon Aurora Auto Scaling の使用](#)」

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、に以下の[サービスリンクロール](#)が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。

AWS CLI またはいずれかの AWS SDKs 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 [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、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"
}
```

エンティティ認識器エンドポイントに対して [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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントで Amazon Comprehend 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 Comprehend デベロッパーガイドの「[Auto scaling with endpoints](#)」

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、に以下の[サービスリンクロール](#)が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オート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。

AWS CLI または AWS SDKs のいずれか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。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、my-table 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 dynamodb \  
  --scalable-dimension dynamodb:table:WriteCapacityUnits \  
  --resource-id table/my-table \  
  --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"  
}
```

テーブルの読み込み容量に対して [register-scalable-target](#) コマンドを呼び出します。以下の例は、最小容量を 5 個の読み取り容量ユニット、最大容量を 10 個の読み取り容量ユニットとして、my-table 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容量を登録します。

```
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"
```

```
}
```

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。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、`my-table-index` と呼ばれる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キャパシティーを登録します。

```
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"  
}
```

グローバルセカンダリインデックスの読み込み容量に対して [register-scalable-target](#) コマンドを呼び出します。以下の例は、最小容量を 5 個の読み取り容量ユニット、最大容量を 10 個の読み取り容量ユニットとして、`my-table-index` と呼ばれる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取り容量を登録します。

```
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 を始めたばかりの場合は、次のドキュメントで DynamoDB リソースのスケーリングに関する追加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 Auto Scaling 設定を評価するDynamoDB](#)」
- 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 Auto Scaling を設定する方法](#)

スケジュールされたスケーリングのチュートリアルは、「」でも確認できます[チュートリアル: AWS CLIを使用したスケジュールに基づくスケーリングの開始方法](#)。このチュートリアルでは、DynamoDB テーブルがスケジュールされた時刻にスケールされるようにスケーリングを設定するための基本的な手順について学びます。

Amazon ECS と Application Auto Scaling

ECS サービスは、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、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、に以下の[サービスリンクロール](#)が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オート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。

AWS CLI または AWS SDKs のいずれか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 を始めたばかりの場合は、以下のドキュメントで Amazon ECS リソースのスケールリングに関するその他の有用な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ルリング」](#)
- [「Amazon Elastic Container Service ベストプラクティスガイド」の「サービスの自動スケールリングの設定」](#)

Note

Amazon ECS デプロイの進行中にスケールアウトプロセスを中断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ルリングとデプロイ」](#)

ElastiCache for Redis と Application Auto Scaling

ターゲット追跡スケールリングポリシーとスケジュールされたスケールリングを使用して、Redis レプリケーショングループ ElastiCache をスケールリングできます。

以下の情報は、Application Auto Scaling ElastiCache との統合に役立ちます。

ElastiCache 向けに作成されたサービスリンクロール

Application Auto Scaling にスケラブルターゲットとして ElastiCache リソースを登録する AWS アカウント と、に以下の[サービスにリンクされたロール](#)が自動的に作成されます。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`

スケーラブルターゲットとしての ElastiCache Redis レプリケーショングループの Application Auto Scaling への登録

Application Auto Scaling では、ElastiCache レプリケーショングループ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、はスケーラブルターゲット ElastiCache を自動的に登録します。

AWS CLI または AWS SDKs のいずれか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:

レプリケーショングループの [register-scalable-target](#) コマンドを呼び出します ElastiCache。以下の例は、最小容量を 1、最大容量を 5 として、`mycluster` 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/mycluster \  
  --min-capacity 1 \  
  --max-capacity 5
```

```
--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"
}
```

以下の例は、最小容量を 1、最大容量を 5 として、mycluster 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希望レプリカ数を登録します。

```
aws application-autoscaling register-scalable-target \
  --service-namespace elasticache \
  --scalable-dimension elasticache:replication-group:Replicas \
  --resource-id replication-group/mycluster \
  --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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントで ElastiCache リソースのスケールリングに関するその他の有用な情報を確認できます。

Amazon [for Redis ユーザーガイドの「Auto Scaling ElastiCache for Redis クラスター ElastiCache」](#)

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、に以下の[サービスリンクロール](#)が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オート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。

AWS CLI または AWS SDKs のいずれか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 [register-scalable-target](#) 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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントで Amazon Keyspaces 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 Keyspaces \(Apache Cassandra 向け\) デベロッパーガイド」の「Amazon Keyspaces Auto Scaling によるスループットキャパシティの管理」](#)

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数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

Lambda 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、に以下の[サービスリンクロール](#)が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。

AWS CLI またはいずれかの AWS SDKs 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 [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、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 を始めたばかりの場合は、以下のドキュメントで Lambda 関数のスケールアップに関するその他の有益な情報を確認できます。

- [「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数の設定AWS Lambda」](#)
- AWS ブログの [「ピーク使用が繰り返される場合の Lambda プロビジョニングされた同時実行のスケジュール」](#)

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ル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、に以下の[サービスリンクロール](#)が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 クラスタストレージの Application Auto Scaling への登録

Application Auto Scaling では、Amazon MSK クラスタ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オート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。

AWS CLI または AWS SDKs のいずれか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 100 GiB、最大容量を 800 GiB として、Amazon MSK クラスタ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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントで Amazon MSK リソースのスケールリングに関するその他の有用な情報を確認できます。

「Amazon Managed Streaming for Apache Kafka デベロッパーガイド」の「[自動スケールリング](#)」

Amazon Neptune と Application Auto Scaling

Neptune 関数は、ターゲット追跡スケールリングポリシーとスケジュールされたスケール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

Neptune リソースをスケ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、に以下の[サービスリンクロール](#)が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。

AWS CLI またはいずれかの AWS SDKs 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 [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、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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントで Neptune 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。

Neptune ユーザーガイドの [Amazon Neptune DB クラスター内のレプリカの数 Auto Scaling](#)

Amazon SageMaker 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、およびスケジュールされたスケーリングを使用して、SageMaker エンドポイントバリエーション、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。

以下の情報は、Application Auto Scaling SageMaker との統合に役立ちます。

SageMaker 向け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 SageMaker リソースを登録する AWS アカウントと、に以下の[サービスにリンクされたロール](#)が自動的に作成されます。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`

スケーラブルターゲットとしての SageMaker エンドポイントバリエーションの Application Auto Scaling への登録

Application Auto Scaling では、SageMaker モデル (バリエーション) 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。スケラブルターゲットは、リソース ID、スケ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 コンソールを使用して自動スケーリングを設定すると、はスケラブルターゲット SageMaker を自動的に登録します。

AWS CLI または AWS SDKs のいずれか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:

製品バリエーションに対して [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エーション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 コンソールを使用して自動スケーリングを設定すると、はスケーラブルターゲット SageMaker を自動的に登録します。

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:

製品バリエーションに対して [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エーション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 を始めたばかりの場合は、「Amazon SageMaker デベロッパーガイド」の「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」を参照してください。

- [Amazon SageMaker 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数を自動的にスケーリングする](#)

- [マルチモデルエンドポイントデプロイの Auto Scaling ポリシーを設定する](#)
- [非同期エンドポイントのオートスケーリング](#)

Note

2023 年に、SageMaker 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。SageMaker 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。推論コンポーネントは、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 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については、AWS ブログの「[Amazon は、基盤モデルのデプロイコストとレイテンシーの削減に役立つ新しい推論機能 SageMaker を追加](#)」および「[Amazon 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 SageMaker する](#)」を参照してください。

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

スポットフリート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、に以下の[サービスにリンクされたロール](#)が自動的に作成されます。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オート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。

AWS CLI または AWS SDKs のいずれか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 [register-scalable-target](#) 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

Application Auto Scaling を始めたばかりの場合は、以下のドキュメントでスポットフリートのスケールリングに関するその他の有用な情報を確認できます。

Amazon EC2 ユーザーガイドの「[スポットフリートの自動スケールリング](#)」

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ルリングポリシー、ステップスケールリングポリシー、およびスケジュールされたスケール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

カスタムリソースをスケ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、に以下の[サービスにリンクされたロール](#)が自動的に作成されます。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。

AWS CLI またはいずれかの AWS SDKs 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 を始めたばかりの場合は、以下のドキュメントでカスタムリソースのスケールリングに関するその他の有用な情報を確認できます。

[GitHub リポジトリ](#)

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

Application Auto Scaling 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あると統合されています。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、複数の AWS アカウント およびリージョンで同じリソースを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取得するには、「[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)。スケーラブルなディメンションを取得するには、「[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 IDs 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、「[で Application Auto Scaling リソースを設定する 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、「[」](#)を参照してください [AWS::ApplicationAutoScaling::ScalingPolicy](#)。

の詳細 AWS CloudFormation

の詳細については [AWS CloudFormation](#)、以下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

- [Application Auto Scaling のスケジュールされたスケーリングの仕組み](#)
- [cron 式を使用して、定期的なスケーリングアクションをスケジュールする](#)
- [Application Auto Scaling のスケジュールされたアクションの例](#)
- [Application Auto Scaling のスケジュールされたスケーリングを管理する](#)
- [チュートリアル: AWS CLIを使用したスケジュールに基づくスケーリングの開始方法](#)

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の仕組みについて説明し、効果的に使用するために理解しておく必要がある重要な考慮事項を紹介します。

内容

- [仕組み](#)
- [考慮事項](#)
- [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](#)
- [関連リソース](#)
- [制限事項](#)

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、「[cron 式を使用して、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、スケーリングを一時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の使用の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。

サンプル AWS リソース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詳しく説明するチュートリアルについては、「[チュートリアル: AWS CLIを使用したスケジュールに基づくスケーリングの開始方法](#)」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。

制限事項

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。

cron 式を使用して、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 AWS CLI を作成するコマンドの例については、「」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 |

| 分 | 時間 | 日 | 月 | 曜日 | 年 | 意味 |
|------|------|---|---|---------|---|---|
| 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。

Application Auto Scaling のスケジュールされたアクションの例

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。

簡略化のため、このトピックの例では、Application Auto Scaling と統合されている一部のサービス用の CLI コマンドを例示しています。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。

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。

内容

- [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
```

Note

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ル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「[レート式](#)」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

定期的なスケール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、「[cron 式を使用して、定期的なスケール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 \  
  --schedule "cron(0 5 * * ? *)" --timezone America/New_York
```

```
--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、「[cron 式を使用して、定期的なスケールアップ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
```

Application Auto Scaling のスケジュールされたスケーリングを管理する

AWS CLI には、スケジュールされたアクションの管理に役立つ他のコマンドがいくつか含まれています。

簡略化のため、このトピックの例では、Application Auto Scaling と統合されている一部のサービス用の CLI コマンドを例示しています。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。

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。

内容

- [指定されたサービスのスケーリングアクティビティを表示する](#)
- [指定されたサービスに対するすべてのスケジュールされたアクションの記述](#)
- [スケーラブルターゲットに対する 1 つまたは複数のスケジュールされたアクションを記述する](#)
- [スケーラブルターゲットに対するスケジュールされたスケーリングをオフにする](#)
- [スケジュールされたアクションの削除](#)

指定されたサービスのスケーリングアクティビティを表示する

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
```

コマンドが正常に完了した場合は、以下のような出力が表示されます。

```
{
  "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ルリングアクティビティを取得するには、`--resource-id` オプションを追加します。

指定されたサービスに対するすべてのスケジュールされたアクションの記述

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
```

正常に完了した場合、この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 1 つまたは複数の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
```

以下は出力例です。

```
{
  "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
    }
  ]
}
```

--scheduled-action-names オプション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。

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。

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 \  
\br/>--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}'
```

成功した場合、このコマンドはスケ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"  
}
```

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

スケジュールされたアクションの削除

スケジュールされたアクションが不要になったら、[delete-scheduled-action](#) コマンドを使用してアクションを削除することができます。

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
```

正常に完了すると、このコマンドはプロンプトに戻ります。

チュートリアル: AWS CLIを使用したスケジュールに基づくスケールリングの開始方法

次のチュートリアルでは、を使用して、というサンプル DynamoDB テーブルをスケールするスケジュールされたアクションの作成を支援することで、スケジュールされたスケールリング AWS CLI を開始する方法を示します TestTable。テストに使用する TestTable テーブルが DynamoDB にまだない場合は、Amazon DynamoDB デベロッパーガイドの[ステップ 1 : DynamoDB テーブルを作成する](#)に示されている create-table コマンドを実行してテーブルを作成できます。

を使用する場合 AWS CLI、コマンドはプロファイル用に設定された AWS リージョン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。

Note

このチュートリアルの一環として AWS 料金が発生する場合があります。[無料利用枠](#)の使用状況をモニタリングするとともに、DynamoDB データベースが使用する読み取りおよび書き込み容量のユニット数に関連付けられたコストを理解しておくようにしてください。

内容

- [ステップ 1: スケーラブルターゲットを登録する](#)
- [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](#)
- [ステップ 3: スケールリングアクティビティを表示する](#)
- [ステップ 4: 次のステップ](#)
- [ステップ 5: クリーンアップ](#)

ステップ 1: スケーラブルターゲットを登録する

スケラブルターゲットとして DynamoDB テーブルを Application Auto Scaling に登録することから始めます。

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

1. まず、[describe-scalable-targets](#) コマンドを使用して、DynamoDB リソースが既に登録されているかどうかをチェックします。これにより、新しいテーブルではない場合に備えて、TestTable テーブルは登録解除されているか確認します。

Linux、macOS、または Unix

```
aws application-autoscaling describe-scalable-targets \  
  --service-namespace dynamodb
```

Windows

```
aws application-autoscaling describe-scalable-targets --service-namespace dynamodb
```

既存のスケーラブルなターゲットがない場合、レスポンスは次のようになります。

```
{  
  "ScalableTargets": []  
}
```

2. 以下の [register-scalable-target](#) コマンドを使用して、TestTable と呼ばれる DynamoDB テーブルの書き込み容量を登録します。希望する最小容量を 5 個の書き込みキャパシティーユニット、希望する最大容量を 10 個の書き込みキャパシティーユニットに設定します。

Linux、macOS、または Unix

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

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb  
  --scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/  
TestTable --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"
}
```

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

Application Auto Scaling では、スケーリングアクションが実行される時刻をスケジュールすることができます。スケーラブルなターゲット、スケジュール、最小キャパシティー、および最大キャパシティーを指定します。指定された時間がくると、Application Auto Scaling がスケーラブルターゲットの最小値と最大値を更新します。現在のキャパシティーが範囲外の場合はスケーリングアクティビティにつながります。

最小キャパシティーと最大キャパシティーへの更新のスケジュールは、スケーリングポリシーを作成する場合にも役立ちます。スケーリングポリシーを使用すると、現在のリソース使用率に基づいてリソースを動的にスケーリングできます。スケーリングポリシーの一般的なガードレールとして、最小キャパシティーと最大キャパシティーに適切な値を設定します。

この演習では、1 回限りのアクションを 2 つ作成し、スケールアウトまたはスケールインできます。

スケジュールされたアクションを作成および表示する

1. 最初のスケジュールされたアクションを作成するには、以下の [put-scheduled-action](#) コマンドを使用します。

--schedule の at コマンドは、指定された将来の日時に 1 回実行されるアクションをスケジュールします。時間は 24 時間形式 (UTC) です。今から約 5 分後にアクションが発生するようにスケジュールします。

指定された日時がくると、Application Auto Scaling が MinCapacity および MaxCapacity の値を更新します。テーブルに現在 5 個の書き込みキャパシティーユニットがあるとする、Application Auto Scaling は MinCapacity にスケールアウトして、そのテーブルが 15~20 書き込みキャパシティーユニットの新しい希望範囲内に収まるようにします。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
```

```
--resource-id table/TestTable \  
--scheduled-action-name my-first-scheduled-action \  
--schedule "at(2019-05-20T17:05:00)" \  
--scalable-target-action MinCapacity=15,MaxCapacity=20
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace dynamodb  
--scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/  
TestTable --scheduled-action-name my-first-scheduled-action --schedule  
"at(2019-05-20T17:05:00)" --scalable-target-action MinCapacity=15,MaxCapacity=20
```

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. Application Auto Scaling がスケールインするために使用する 2 番目のスケジュールされたアクションを作成するには、以下の [put-scheduled-action](#) コマンドを使用します。

今から約 10 分後にアクションが発生するようにスケジュールします。

指定された日時がくると、Application Auto Scaling がテーブルの MinCapacity と MaxCapacity を更新して MaxCapacity にスケールインし、テーブルを元の希望範囲である 5 ~ 10 書き込みキャパシティーユニットに戻します。

Linux、macOS、または Unix

```
aws application-autoscaling put-scheduled-action \  
--service-namespace dynamodb \  
--scalable-dimension dynamodb:table:WriteCapacityUnits \  
--resource-id table/TestTable \  
--scheduled-action-name my-second-scheduled-action \  
--schedule "at(2019-05-20T17:10:00)" \  
--scalable-target-action MinCapacity=5,MaxCapacity=10
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace dynamodb  
--scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/  
TestTable --scheduled-action-name my-second-scheduled-action --schedule  
"at(2019-05-20T17:10:00)" --scalable-target-action MinCapacity=5,MaxCapacity=10
```

3. (オプション) 以下の [describe-scheduled-actions](#) コマンドを使用して、指定されたサービス名前空間に対するスケジュールされたアクションのリストを取得します。

Linux、macOS、または Unix

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

Windows

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

以下は出力例です。

```
{  
  "ScheduledActions": [  
    {  
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",  
      "Schedule": "at(2019-05-20T18:35:00)",  
      "ResourceId": "table/TestTable",  
      "CreationTime": 1561571888.361,  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
east-1:123456789012:scheduledAction:2d36aa3b-cdf9-4565-b290-81db519b227d:resource/  
dynamodb/table/TestTable: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/TestTable",  
      "CreationTime": 1561571946.021,  
      "ScheduledActionARN": "arn:aws:autoscaling:us-  
east-1:123456789012:scheduledAction:2d36aa3b-cdf9-4565-b290-81db519b227d:resource/  
dynamodb/table/TestTable:scheduledActionName/my-second-scheduled-action",  
      "ScalableTargetAction": {  
        "MinCapacity": 5,  
        "MaxCapacity": 10  
      }  
    }  
  ]  
}
```

```
    },
    "ScheduledActionName": "my-second-scheduled-action",
    "ServiceNamespace": "dynamodb"
  }
]
}
```

ステップ 3: スケーリングアクティビティを表示する

このステップでは、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表示してから、DynamoDB がテーブルの書き込みキャパシティーを変更したことを確認します。

スケーリングアクティビティを表示する

1. 選択した時間まで待機し、以下の [describe-scaling-activities](#) コマンドを使用して、スケジュールされたアクションが機能していることを確認します。

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
```

以下は、スケジュールされたアクションの進行中に、スケジュールされた最初のアクションの出力例です。

スケーリングアクティビティは作成日順に並べられ、最新のスケーリングアクティビティが最初に返ります。

```
{
  "ScalingActivities": [
    {
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
      "Description": "Setting write capacity units to 15.",
      "ResourceId": "table/TestTable",
```

```

    "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
    "StartTime": 1561574108.904,
    "ServiceNamespace": "dynamodb",
    "Cause": "minimum capacity was set to 15",
    "StatusMessage": "Successfully set write capacity units to 15. Waiting
for change to be fulfilled by dynamodb.",
    "StatusCode": "InProgress"
  },
  {
    "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
    "Description": "Setting min capacity to 15 and max capacity to 20",
    "ResourceId": "table/TestTable",
    "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"
  }
]
}

```

以下は、スケジュールされたアクションがいずれも実行された後の出力例です。

```

{
  "ScalingActivities": [
    {
      "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
      "Description": "Setting write capacity units to 10.",
      "ResourceId": "table/TestTable",
      "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/TestTable",
    "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/TestTable",
    "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/TestTable",
    "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"
  }
]
```

2. スケジュールされたアクションが正常に実行されたら、DynamoDB コンソールに移動して、使用するテーブルを選択します。[Capacity] (キャパシティー) タブで、[Write capacity units] (書

き込みキャパシティーユニット)を確認します。2番目のスケーリングアクションが実行されると、書き込みキャパシティーユニットは 15 から 10 にスケーリングされます。

[describe-table](#) コマンドを使用して、テーブルの現在の書き込み容量も確認します。出力をフィルタリングするには、`--query` オプションを含めます。の出力フィルタリング機能の詳細については AWS CLI、「ユーザーガイド」の「[からのコマンド出力の制御 AWS CLI](#) AWS Command Line Interface」を参照してください。

Linux、macOS、または Unix

```
aws dynamodb describe-table --table-name TestTable \  
  --query 'Table.[TableName,TableStatus,ProvisionedThroughput]'
```

Windows

```
aws dynamodb describe-table --table-name TestTable --query "Table.  
[TableName,TableStatus,ProvisionedThroughput]"
```

以下は出力例です。

```
[  
  "TestTable",  
  "ACTIVE",  
  {  
    "NumberOfDecreasesToday": 1,  
    "WriteCapacityUnits": 10,  
    "LastIncreaseDateTime": 1561574133.264,  
    "ReadCapacityUnits": 5,  
    "LastDecreaseDateTime": 1561574435.607  
  }  
]
```

ステップ 4: 次のステップ

スケジュールされたスケーリングポリシーとスケーリングポリシーの両方でスケーリングを試す場合は、[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](#) の手順に従います。

ステップ 5 : クリーンアップ

「使用開始」演習が終了したら、関連付けられているリソース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ジュールされたアクションを削除します。後で使用できるように、スケジュールされたアクションを保持する場合は、このステップをスキップできます。

Linux、macOS、または Unix

```
aws application-autoscaling delete-scheduled-action \  
  --service-namespace dynamodb \  
  --scalable-dimension dynamodb:table:WriteCapacityUnits \  
  --resource-id table/TestTable \  
  --scheduled-action-name my-second-scheduled-action
```

Windows

```
aws application-autoscaling delete-scheduled-action --service-namespace dynamodb --  
scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable --  
scheduled-action-name my-second-scheduled-action
```

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 dynamodb \  
  --scalable-dimension dynamodb:table:WriteCapacityUnits \  
  --resource-id table/TestTable
```

Windows

```
aws application-autoscaling deregister-scalable-target --service-namespace dynamodb --scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable
```

DynamoDB テーブルを削除する

以下の [delete-table](#) コマンドを使用して、このチュートリアルで使用したテーブルを削除します。後で使用できるように、テーブルを保持する場合は、このステップをスキップできます。

Linux、macOS、または Unix

```
aws dynamodb delete-table --table-name TestTable
```

Windows

```
aws dynamodb delete-table --table-name TestTable
```

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

ターゲット追跡スケールリングポリシーは、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、メトリクスが CloudWatch ターゲットから逸脱したときにスケールリングイベントをトリガーする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ル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 アラームとスケール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ル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 — メトリクスの数学を使用して、メトリクスを組み合わせたり、既存のメトリクスを活用したり、に公開された独自のカスタムメトリクスを使用したりできます CloudWatch。

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。したがって、容量を 2 倍にすると、メトリクスは 50% 減少します。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](#)
- [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](#)
- [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](#)

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について説明し、ターゲット追跡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](#)
- [メトリクスを選択する](#)
- [ターゲット値の定義](#)
- [クールダウン期間を定義する](#)
- [考慮事項](#)
- [複数のスケーリングポリシー](#)
- [スケーリングポリシーの作成、管理、および削除によく使用されるコマンド](#)
- [関連リソース](#)
- [制限事項](#)

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** — CPU の平均使用率やターゲットあたりの平均リクエスト数など、追跡する CloudWatch メトリクス。

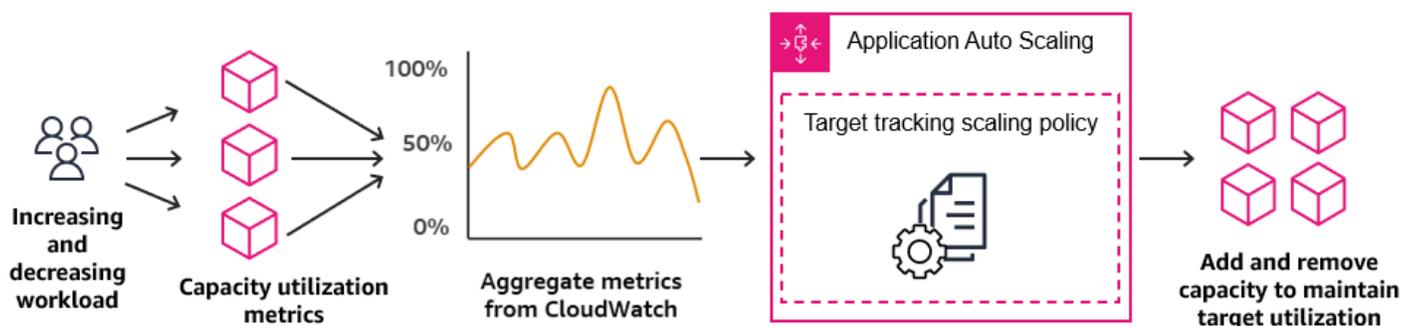
- ターゲット値 — メトリクスのターゲット値 (CPU 使用率 50%、ターゲット 1 分あたり 1000 リクエストなど)。

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ル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ルリングポリシーを作成する場合、[ターゲット追跡スケール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。
- メトリクスが実際の 0 の値を に出力する場合 CloudWatch (例: ALBRequestCountPerTarget)、Application Auto Scaling は、アプリケーションへのトラブルフィックスが長期間続かない場合に 0 にスケールインできます。スケ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。
- スケール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ル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に、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ています。

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 レプリケーショングループのデフォルト値は 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 エンドポイントバリエーション
- SageMaker 推論コンポーネント
- SageMaker サーバーレスで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

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ず。

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、または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、不要になったら削除します。
- 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` に変わります。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 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。 CloudWatch
- 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ない場合のみスケールインされます。

複数のスケールリングポリシーが、スケ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ル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ル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ル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ルリングポリシーをステップスケール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、ステップスケール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ル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、スケール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ルリングとターゲットトラッキングスケール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。

スケールリングポリシーの作成、管理、および削除によく使用されるコマンド

スケールリングポリシーの操作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ラブルターゲット (Application Auto Scaling がスケールリングできるリソース) として登録し、スケールリングを一時停止して再開します。
- [put-scaling-policy](#) 既存のスケラブルターゲットのスケール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ル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。

制限事項

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

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には、次の設定タスク AWS CLI に を使用します。

1. スケーラブルターゲットを登録します。
2. スケーラブルターゲットに対するターゲット追跡スケーリングポリシーを追加します。

簡略化のため、このトピックの例では、Amazon EC2 スポットフリート用の CLI コマンドを例示しています。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。

を使用する場合は AWS CLI、コマンドがプロファイル用に 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。

内容

- [スケーラブルターゲットを登録する](#)
- [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記述する](#)
- [ターゲット追跡スケーリングポリシーを削除する](#)

スケーラブル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
```

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"
}
```

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、以下の例を参考にしてください。

ターゲット追跡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、スケーリングポリシーのターゲット値と事前定義されたメトリクス仕様をホームディレクトリconfig.jsonの という名前の JSON ファイルに保存します。以下は、平均 CPU 使用率を 50% に維持するターゲット追跡設定の例です。

```
$ 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[PredefinedMetric 「仕様」](#)を参照してください。

または、カスタマイズされたメトリクス仕様を作成し、 から各パラメータの値を追加することで、スケーリングにカスタムメトリクスを使用することもできます CloudWatch。以下は、指定されたメトリクスの平均使用率を 100 に維持するターゲット追跡設定の例です。

```
$ 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[CustomizedMetric 「仕様」](#)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
```

成功すると、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 ARNs と名前を返します。

```
{  
  "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"
  }
]
```

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`]"
```

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
}
```

ターゲット追跡スケーリングポリシーを削除する

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。また、Application Auto Scaling がユーザーに代わって作成した CloudWatch アラームも削除されます。

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ルリングポリシーを作成する

Metric Math を使用すると、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いて新しい時系列を作成できます。作成された時系列を CloudWatch コンソールで視覚化し、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 [ユーザーガイド](#)」の「[Metric Math の使用](#)」を参照してください。 CloudWatch

Metric Math の数式には、次の考慮事項が適用されます。

- 使用可能なメトリクスをクエリできます CloudWatch 。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。
- 算術演算子 (+ - * / ^)、統計関数 (AVG や SUM など) 、または が CloudWatch 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、メトリクス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](#)
- [制限事項](#)

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。

$$\text{sum of (number of messages in the queue) / (number of tasks that are currently in the RUNNING state)}$$

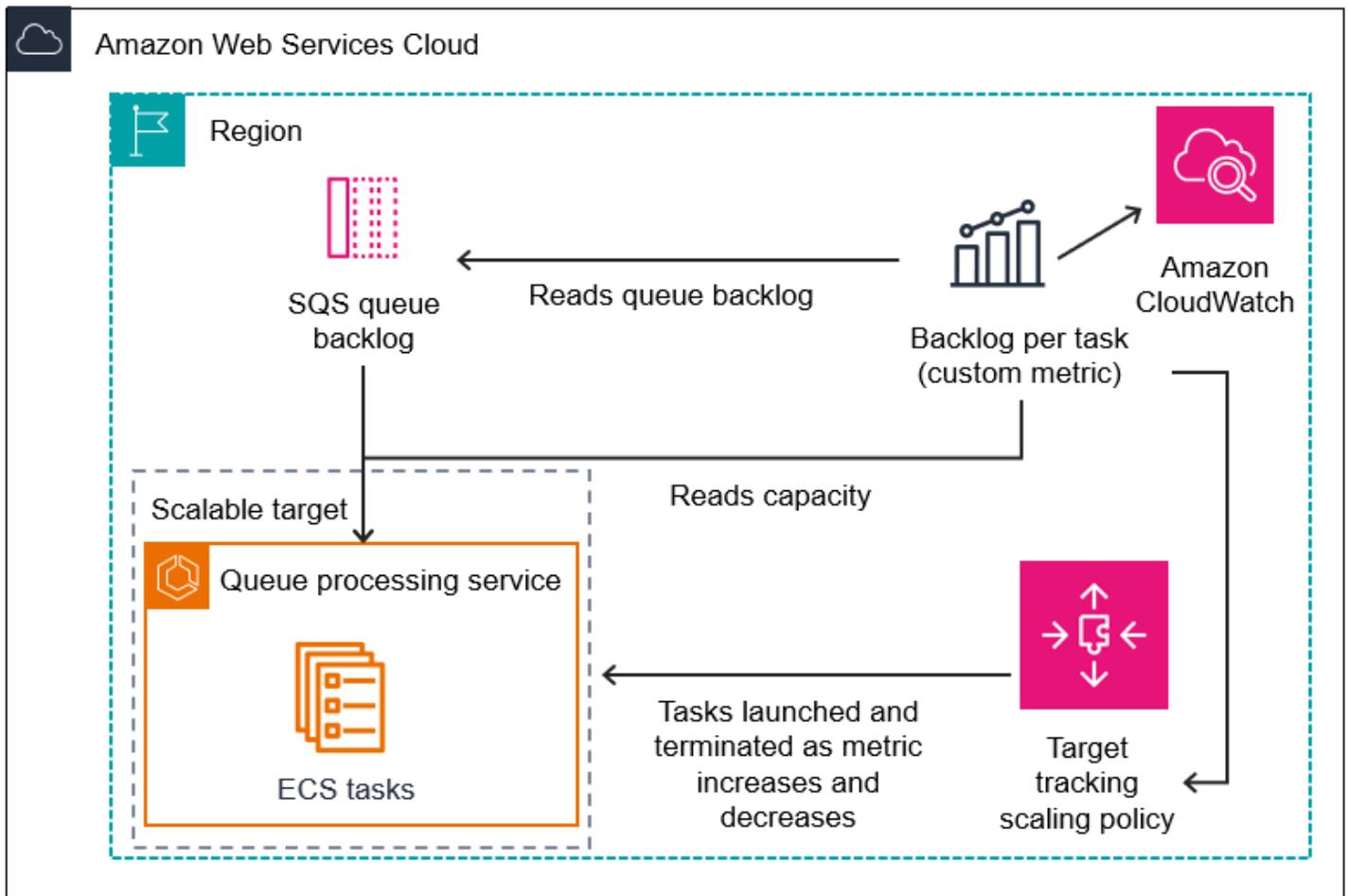
次に、CloudWatch メトリクス情報は次のとおりです。

| ID | CloudWatch メトリクス | 統計 | 間隔 |
|----|---------------------------------|----------------|-----|
| m1 | ApproximateNumberOfMessages表示可能 | 合計 | 1 分 |
| m2 | RunningTaskカウント | [Average] (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) |

次の図は、このメトリクスのアーキテクチャを示しています。



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 リファレンスの[TargetTrackingScalingPolicy](#)「設定」を参照してください。

Note

以下は、メトリクスの名前、名前空間、ディメンション、統計情報の検索に役立つ追加のリソースです CloudWatch。

- AWS サービスの利用可能なメトリクスについては、「Amazon [AWS ユーザーガイド](#)」の [CloudWatch](#) 「メトリクスを発行する のサービス」を参照してください。
CloudWatch
- を使用してメトリクスの正確な CloudWatch 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
```

成功すると、このコマンドはポリシーの Amazon リソースネーム (ARN) と、ユーザーに代わって作成された 2 ARNs を返します。CloudWatch

```
{
  "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

このコマンドでエラーが発生した場合は、[awscli](#) 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。

制限事項

- 最大リクエストサイズは 50 KB です。これは、[PutScalingポリシー](#) 定義で Metric Math を使用する場合の Policy 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アプリケーションの容量を事前に定義された増分でスケールリングします。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) を処理するスケール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では、スケール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ル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ルリングします。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、スケールリングポリシーは、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、スケール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ルリングするのに役立ちます。ただし、安定したスケール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。

ステップスケーリングポリシーは、以下のスケール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 エンドポイントバリエーション
- SageMaker 推論コンポーネント
- SageMaker サーバーレスで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](#)
- [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](#)

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](#)
- [ステップ調整値](#)
- [スケーリング調整タイプ](#)
- [クールダウン期間](#)
- [スケーリングポリシーの作成、管理、および削除によく使用されるコマンド](#)
- [考慮事項](#)
- [関連リソース](#)
- [制限事項](#)

仕組み

ステップスケーリングを使用するには、スケーラブルターゲットのメトリクスをモニタリングする 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。例:

- 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、指定された集計タイプを CloudWatch (未加工のメトリクスデータではなく) からの最新のメトリクスデータポイントに適用します。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% を超えた場合に 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 |

| 下限 | 上限 | 調整 |
|----|------|----|
| 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。

- **ChangeIn容量** — スケーラブルターゲットの現在の容量を指定された値だけ増減します。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在の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在の容量を指定された割合だけ増減します。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。

Note

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。

では `PercentChangeInCapacity`、`MinAdjustmentMagnitude` パラメータを使用してスケーリングする最小量を指定することもできます。例えば、25% 追加するポリシーを作成して、最小数を 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、スケーリングを一時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。

制限事項

- 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

Application Auto Scaling のステップスケーリングポリシーを作成するには、次の設定タスク AWS CLI に を使用します。

1. スケーラブルターゲットを登録します。
2. スケーラブルターゲットに対するステップスケーリングポリシーを追加します。
3. ポリシーの CloudWatch アラームを作成します。

簡略化のため、このトピックの例では、Amazon ECS サービス用の CLI コマンドを例示しています。別のスケ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。

を使用する場合は AWS CLI、コマンドがプロファイル用に 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。

内容

- [スケラブルターゲットを登録する](#)
- [ステップスケーリングポリシーを作成する](#)
- [スケーリングポリシーを呼び出すアラームを作成する](#)
- [ステップスケーリングポリシーを記述する](#)
- [ステップスケーリングポリシーを削除する](#)

スケラブルターゲットを登録する

まだ登録していない場合は、スケ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
```

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、以下の例を参考にして開始してください。

Scale out

スケールアウト (容量の増加) 用のステップスケーリングポリシーを作成するには

1. ホームディレクトリの という名前の JSON ファイルにステップスケーリングポリシー設定を保存するには、次の `cat` コマンド `config.json` を使用します。以下は、次のステップの調整 (CloudWatch アラームしきい値を 70 と想定) に基づいてスケーラブルターゲットの容量 `PercentChangeInCapacity` を増やす調整タイプの設定例です。
 - メトリクスの値が 70 以上 85 未満の場合は、容量を 10% 増やします。
 - メトリクスの値が 85 以上 95 未満の場合は、容量を 20% 増やします。
 - メトリクスの値が 95 以上の場合、容量を 30% 増やす

```
$ 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 [StepScalingPolicyConfiguration](#) の「」を参照してください。

- 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
```

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には、これが必要です。

```
{
  "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 (容量を減らす) 用のステップスケーリングポリシーを作成するには

1. ホームディレクトリの `という名前の JSON ファイルにステップスケーリングポリシー設定を保存するには、次のcatコマンドconfig.jsonを使用します。以下は、次のステップの調整 (CloudWatch アラームしきい値を 50 と想定) に基づいてスケーラブルターゲットの容量ChangeInCapacityを減らす調整タイプの設定例です。`
 - メトリクスの値が 50 以下かつ 40 より大きい場合は、容量を 1 ずつ減らします。
 - メトリクスの値が 40 以下かつ 30 より大きい場合は、容量を 2 ずつ減らします。
 - メトリクスの値が 30 以下の場合は、容量を 3 ずつ減らします。

```
$ 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 [StepScalingPolicyConfiguration](#) の「」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
```

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には、これが必要です。

```
{  
  "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、次の 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。別のメトリクスを指定する CloudWatch が、独自のカスタムメトリクスを使用するには、`でその名前を指定--metric-name`し、`でその名前空間を指定します--namespace`。

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
```

ステップ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 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`]"
```

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
}
]
```

ステップスケーリングポリシーを削除する

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、次のタスクを実行します。

スケーリングポリシーを削除する

以下の [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には

[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
```

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

Important

このチュートリアルに進む前に、以下の概要チュートリアルを確認することをお勧めします。 [チュートリアル: AWS CLIを使用したスケジュールに基づくスケーリングの開始方法](#)

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、容量がなくなってきたときに削除する。
- 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](#)
- [ステップ 1: スケーラブルターゲットを登録する](#)
- [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](#)
- [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](#)
- [ステップ 4: 次のステップ](#)
- [ステップ 5: クリーンアップ](#)

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。
- Application Auto Scaling でスケーラブルターゲットとしてリソースを登録および登録解除するために必要なアクセス許可を付与しました。さらに、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アクセス許可がに付与されました。詳細については、「[Application Auto Scaling の Identity and Access Management](#)」を参照してください。
- このチュートリアルで使用できる非本番環境でサポートされているリソースを作成しました。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、から完全なメトリクス仕様を指定せずに使用できます CloudWatch。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\" } }"
```

成功すると、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 ARNs 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せん。

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せん。

Linux、macOS、または Unix

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

```
--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、または複数を一時的に停止し、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

- [スケーリングアクティビティ](#)
- [スケーリングアクティビティの一時停止と再開](#)

Note

Amazon ECS デプロイの進行中にスケールアウトプロセスを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とデプロイ](#)」

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的に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。これにより、スケーリングポリシーまたは関連する CloudWatch アラームを削除せずに、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にできます。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。これにより、スケーリングポリシーまたは関連する CloudWatch アラームを削除せずに、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にできます。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は、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

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 メトリクスをモニタリングし、しきい値を超えたときに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でスケーリングアクティビティを検索する

特定のスケ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 [ScalingActivity](#) の「」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める

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。

スケールリングされていないアクティビティ (スケール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。

レスポンスにスケール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、失敗したスケール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 [ScalingActivity](#) の「」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。

エントリが大量に重複しないように、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。それ以降のスケーリングされていないアクティビティでは、スケーリングしない理由が変化しない限り、新しいエントリは生成されません。

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 | オート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 | | | |
|-------------------------------------|--|--|--|--|
| | <p>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、スケールアップ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ルアップアクションが開始されます。</p> | | | |
| TargetServicePutResourceAsnscalable | <p>ターゲットサービスにより、リソースは一時的にスケラブルな状態になりました。Application Auto Scaling は、スケールアップポリシーで設定されたオートスケールの条件が満たされると再試行します。</p> | | | |

| 理由 コード | 定義 | | | |
|----------------------------------|---|--|--|--|
| AlreadyAt MaxCapa ty | 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。 | | | |
| AlreadyAt MinCapac ty | 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。 | | | |
| AlreadyAt DesiredC: pacity | 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 のモニタリング

モニタリングは、Application Auto Scaling およびその他の 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 をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行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。は、指定した Amazon S3 バケットのログファイルに情報 CloudTrail を保存します。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 の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向け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 は、ログを分析し、リソースとホストされたアプリケーションのメトリクスをリアルタイムでモニタリングする AWS のに役立ちます。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、リソース使用率 CloudWatch を追跡し、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を受け取ることができます。詳細については、「[を使用してスケーラブルリソースの使用状況をモニタリングする CloudWatch](#)」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

ムを設定できます。詳細については、「Amazon ユーザーガイド」の「[AWS 使用状況メトリクス](#)」を参照してください。 CloudWatch

Amazon EventBridge

Amazon EventBridge は、アプリケーションをさまざまなソースのデータに簡単に接続できるサーバーレスイベントバスサービスです。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 EventBridge 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 を使用して Application Auto Scaling イベントをモニタリングする EventBridge](#)」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、「[の開始方法 AWS Health Dashboard](#)」を参照してください。

を使用してスケーラブルリソースの使用状況をモニタリングする CloudWatch

Amazon を使用すると CloudWatch、スケーラブルなリソース全体で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。を使用して、メトリクス CloudWatch の収集と追跡、アラームの設定、リソースの変更への自動対応を行うことができます AWS。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、次の表に示すメトリクスが に送信されます CloudWatch。では CloudWatch、メトリクスは最初にサービス名前空間によってグループ化され、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によって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

- [CloudWatch リソースの使用状況をモニタリングするための 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](#)

CloudWatch リソースの使用状況をモニタリングするための メトリクス

次の表に、リソースの使用状況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、リソースの設定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 ユーザーガイド CloudWatch](#)」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 : Available Capacity ディメンション: フリート | AppStream 2.0 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 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: DB ClusterId entifier、 ロール (READER) | 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: DB ClusterId entifier、 ロール (READER) | Amazon Aurora でのクラスターレベルのメトリクス |
| Amazon Comprehend | | | |

| | | | |
|-----------------------|----------------|--|---|
| 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 |
| ドキュメント分類のエンドポイント | AWS/Comprehend | 名前 : Inference Utilization ディメンション : EndpointArn | Amazon Comprehend エンドポイントのメトリクス |
| エンティティ認識機能のエンドポイント | AWS/Comprehend | 名前 : Inference Utilization 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 |

| | | | |
|----------------------|------------------------|---|--|
| 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/ ECS | 名前 : MemoryUtilization ディメンション : ClusterName , ServiceName | Amazon ECS のメトリクス |
| 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 Redis メトリクスの |

| 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/ Elast iCache | 名前 : DatabaseCapacityUsageCountedForEvictionPercentage ディメンション : ReplicationGroupId | ElastiCache Redis メトリクスの |
| クラスター (レプリケーショングループ) | AWS/ Elast iCache | 名前: EngineCPUUtilization ディメンション : ReplicationGroupId、ロール (プライマリ) | ElastiCache Redis メトリクスの |

| 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/ Elast iCache | 名前: EngineCPU Utilization ディメン ション: Replicati onGroupId 、ロール (レプリ カ) | ElastiCache Redis メトリクスの |
| Amazon EMR | | | |
| クラスター | AWS/ Elast icMapRedu ce | 名前: YARN MemoryAv: ailableの 割合 ディメン ション: ClusterId | Amazon EMR のメトリクス |
| Amazon Keyspaces | | | |

| 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 : 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 |
| 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 |

| | | | |
|-----------------|-------------------|---|---|
| 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 : 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、リソース | Lambda 関数のメトリクス |
| Amazon MSK | | | |

| 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 |
|-------------|---------------|--|-----------------------------------|
| ブローカストレージ | AWS/ Kafka | 名前 : KafkaData LogsDiskUsed ディメンション: クラスター名 | Amazon MSK のメトリクス |
| ブローカストレージ | AWS/ Kafka | 名前 : KafkaData LogsDiskUsed 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: DB ClusterId entifier、 ロール (READER) | Neptune メトリクス |
| SageMaker | | | |
| エンドポイントバリエーション | AWS/ SageMaker | 名前: InvocationsPerInstance ディメンション: EndpointName, VariantName | 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 |
|--------------------------------|-------------------|---|-------------------------------------|
| 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 |
| 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) | | | |

| 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: 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 |
| 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 |

| | | | |
|-------------|------------------------|---|--|
| 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/ 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](#)の定義済みメトリクスタイプと、対応する CloudWatch メトリクス名を示しています。事前定義された各メトリクスは、基盤となる 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、[CloudWatch リソースの使用状況をモニタリングするためのメトリクス](#) 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 | |
| AppStreamAverageCapacityUtilization | CapacityUtilization |
| Aurora | |
| RDSReaderAverageCPUUtilization | CPUUtilization |
| RDSReaderAverageDatabaseConnections | DatabaseConnections1 |

| 事前定義済みメトリクスタイプ | CloudWatch メトリクス名 |
|---|--|
| Amazon Comprehend | |
| ComprehendInferenceUtilization | InferenceUtilization |
| DynamoDB | |
| DynamoDBReadCapacityUtilization | ProvisionedReadCapacityUnits、 ConsumedReadCapacityUnits2 |
| DynamoDBWriteCapacityUtilization | ProvisionedWriteCapacityUnits、 ConsumedWriteCapacityUnits2 |
| Amazon ECS | |
| ECSServiceAverageCPUUtilization | CPUUtilization |
| ECSServiceAverageMemoryUtilization | MemoryUtilization |
| ALBRequestCountPerTarget | RequestCountPerTarget1 |
| ElastiCache | |
| ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage | DatabaseMemoryUsageCountedForEvict/パーセンテージ |
| ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage | DatabaseCapacityUsageCountedForEvict/パーセンテージ |
| ElastiCachePrimaryEngineCPUUtilization | EngineCPUUtilization |
| ElastiCacheReplicaEngineCPUUtilization | EngineCPUUtilization |
| Amazon Keyspaces | |
| CassandraReadCapacityUtilization | ProvisionedReadCapacityUnits、 ConsumedReadCapacityUnits2 |

| 事前定義済みメトリクスタイプ | CloudWatch メトリクス名 |
|---|--|
| CassandraWriteCapacityUtilization | ProvisionedWriteCapacityUnits、 ConsumedWriteCapacityUnits2 |
| Lambda | |
| LambdaProvisionedConcurrencyUtilization | ProvisionedConcurrency使用率 |
| Amazon MSK | |
| KafkaBrokerStorageUtilization | KafkaDataLogsDisk使用済み |
| Neptune | |
| NeptuneReaderAverageCPUUtilization | CPUUtilization |
| SageMaker | |
| SageMakerVariantInvocationsPerInstance | InvocationsPerインスタンス 1 |
| SageMakerInferenceComponentInvocationsPerCopy | InvocationsPerCopy1 |
| SageMakerVariantProvisionedConcurrencyUtilization | ServerlessProvisionedConcurrencyUtilization |
| スポットフリート | |
| EC2SpotFleetRequestAverageCPUUtilization | CPUUtilization ³ |
| EC2SpotFleetRequestAverageNetworkIn ³ | NetworkIn1 3 |
| EC2SpotFleetRequestAverageNetworkOut ³ | NetworkOut1 3 |

| 事前定義済みメトリクスタイプ | CloudWatch メトリクス名 |
|--------------------------|------------------------|
| ALBRequestCountPerTarget | RequestCountPerTarget1 |

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、事前定義されたメトリクスは、プロビジョニングされたスループットの消費量に基づくスケーリングをサポートする 2 CloudWatch つのメトリクスの集約です。

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、ユーザー AWS CloudTrail、ロール、または Application Auto Scaling API AWS のサービスを使用して によって実行されたアクションを記録するサービスであると統合されています。は、Application Auto Scaling のすべての API コールをイベントとして CloudTrail キャプチャします。キャプチャされた呼び出しには、からの呼び出し AWS Management Console と、Application Auto Scaling API へのコード呼び出しが含まれます。証跡を作成する場合は、Application Auto Scaling の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、Application Auto Scaling に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、[「AWS CloudTrail ユーザーガイド」](#)を参照してください。

の Application Auto Scaling 情報 CloudTrail

CloudTrail アカウントを作成する AWS アカウントと、で が有効になります。Application Auto Scaling アクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴の他の AWS サービスイベントとともにイベントに記録されます。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、[「イベント履歴を含む CloudTrail イベントの表示」](#)を参照してください。

Application Auto Scaling のイベントなど AWS アカウント、 のイベントの継続的な記録については、証跡を作成します。証跡により CloudTrail、 はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、他の Amazon Web Services を設定して、CloudTrail ログで収集されたイベントデータをさらに分析し、それに基づく対応を行うことができます。詳細については、次を参照してください:

- [「証跡作成の概要」](#)
- [CloudTrail がサポートするサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからのログファイルの受信 CloudTrail](#)

すべての Application Auto Scaling アクションは によってログに記録 CloudTrail され、[Application Auto Scaling API リファレンス](#) に記載されています。例えば、PutScalingPolicy、および DescribeScalingPolicies アクションを呼び出すと DeleteScalingPolicy、 CloudTrail ログファイルにエントリが生成されます。

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:

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザーの認証情報を使用し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 サービスによって行われたかどうか。

詳細については、[CloudTrail userIdentity 要素](#)」を参照してください。

Application Auto Scaling のログファイルエントリを理解する

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、DescribeScalableTargetsアクションを示す CloudTrail ログエントリを示しています。

```
{
  "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"
}
```

関連リソース

CloudWatch Logs を使用すると、でキャプチャされた特定のイベントのアラートをモニタリングおよび受信できます CloudTrail。CloudWatch Logs に送信されるイベントは、証跡によってログに記録されるように設定されたイベントであるため、モニタリングするイベントタイプをログに記録するように証跡が設定されていることを確認してください。CloudWatch Logs はログファイル内の情報をモニタリングし、特定のしきい値に達したときに通知できます。高い耐久性を備えたストレージにログデータをアーカイブすることもできます。詳細については、[「ユーザーガイド」の「Amazon CloudWatch Logs ユーザーガイド」](#) および [「Amazon CloudWatch Logs を使用したログファイルのモニタリング CloudTrail」](#) トピックを参照してくださいAWS CloudTrail。

Amazon を使用して Application Auto Scaling イベントをモニタリングする EventBridge

以前 EventBridgeは CloudWatch Events と呼ばれていた Amazon は、Application Auto Scaling に固有のイベントをモニタリングし、他の 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 AWS のサービス。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。

を使用して EventBridge、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 ユーザーガイド」の EventBridge](#) 「Amazon の開始方法 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 で利用できるのは CloudTrail 、最大 にスケーリングされた と を介した API コールに固有のイベントのみです。

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](#)
- [経由の API コールのイベント CloudTrail](#)

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、serviceNameNamespace、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",
    "serviceNameNamespace": "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
  ]
}
}
```

経由の API コールのイベント CloudTrail

証跡は、AWS CloudTrail を使用してイベントをログファイルとして Amazon S3 bucket. CloudTrail log files に配信する設定です。ログファイルにはログエントリが含まれます。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。の使用を開始する方法については CloudTrail、「AWS CloudTrail ユーザーガイド」の「[証跡の作成](#)」を参照してください。

経由で配信されるイベント CloudTrail には、 の値AWS API Call via CloudTrailとしてがありますdetail-type。

次のイベント例は、コンソールユーザーが Application Auto Scaling [RegisterScalableターゲット](#)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"
}
```

すべてのスケラブルターゲットに対するすべての [DeleteScalingPolicy](#) API コールと [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"
      ]
    }
  }
}
```

の使用の詳細については、CloudTrail「」を参照してください [を使用した Application Auto Scaling API コールのログ記録 AWS CloudTrail](#)。

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 および のサンプル値--max-capacityと 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に置き換えます。また、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 \  
  --tags production=true iscontainerbased=true
```

```
--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まれる (または含まない) 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が含まれる (または含まない) 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。

```
{
  "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からメリットを得られます。

セキュリティは、AWS とユーザーの間で共有される責任です。[責任共有モデル](#)ではこれを、クラウドのセキュリティ、およびクラウド内で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 — クラウドで AWS サービスを実行するインフラストラクチャを保護する責任 AWS は AWS にあります。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。コンプライアンス [AWS プログラムコンプライアンス](#) プログラムコンプライアンス プログラムの一環として、サードパーティーの監査者は定期的にセキュリティの有効性を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、[AWS 「コンプライアンスプログラムによる対象範囲内の のサービス」](#) 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 の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](#)
- [Application Auto Scaling の Identity and Access Management](#)
- [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](#)
- [Application Auto Scaling の耐障害性](#)
- [Application Auto Scaling のインフラストラクチャセキュリティ](#)
- [Application Auto Scaling のコンプライアンス検証](#)

Application Auto Scaling でのデータ保護

責任 AWS [共有モデル](#)、Application Auto Scaling のデータ保護に適用されます。このモデルで説明されているように、AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「[データプライバシーのよくある質問](#)」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「[AWS 責任共有モデルおよび GDPR](#)」を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-2](#)」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK を使用して 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 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

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 AWS のサービス 使用できる 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](#)
- [AWS Application Auto Scaling の マネージドポリシー](#)
- [Application Auto Scaling 用のサービスリンクロール](#)
- [Application Auto Scaling のアイデンティティベースポリシー例](#)
- [Application Auto Scaling へのアクセスのトラブルシューティング](#)
- [ターゲットリソースでの Application Auto Scaling API コールのアクセス許可の検証](#)

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のポリシー | はい |
| ポリシーアクション | Yes |
| 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 | No |
| 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 | あり |
| 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を把握するには](#)、IAM ユーザーガイドの [AWS のサービス「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

| | |
|------------------------|-----|
| アイデンティティベースポリシーをサポートする | Yes |
|------------------------|-----|

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の [「IAM ポリシーの作成」](#) を参照してください。

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 | はい |
|----------------|----|

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、「サービス認証リファレンス」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

ポリシーリソースに対するサポート はい

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lement内でワイルドカード (*) を使用します。

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

詳細については、「サービス認証リファレンス」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 はい

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、「サービス認証リファレンス」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 の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 | いいえ |
|-------------------|-----|

Amazon Simple Storage Service AWS などの他のサービスは、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 | No |
|-----------|----|

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

部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限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操作を許可するように ABAC ポリシーを設計します。

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

はい

一部の は、一時的な認証情報を使用してサインインすると機能 AWS のサービスしません。一時的な認証情報 AWS のサービスを使用する などの詳細については、IAM ユーザーガイドの [AWS のサービス](#) 「[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。例えば、会社の Single Sign-On (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「[ロールへの切り替え \(コンソール\)](#)」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、AWS recommends にアクセスできます AWS。これは、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することを推奨しています。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。

サービスロール

| | |
|-----------------|----|
| サービスロールに対するサポート | 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。サービスロールは、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、このロール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。

サービスリンクロール

| | |
|-----------------|----|
| サービスリンクロールのサポート | はい |
|-----------------|----|

サービスにリンクされたロールは、にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービ

スにリンクされたロールは に表示され AWS アカウント、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

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。ユースケース別に[カスタマー マネージドポリシー](#)を定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) が更新されます。は、新しい AWS のサービスが起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、AWS 管理ポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS マネージドポリシー：AppStream 2.0 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingAppStreamFleetPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_AppStreamFleet](#)され、Application Auto Scaling がユーザーに代わって Amazon を呼び出し AppStream CloudWatch、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」：「*」) に対して次のアクションを実行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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingRDSClusterPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_RDSCluster](#) され、Application Auto Scaling がユーザーに代わって Aurora CloudWatch を呼び出し、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」 : 「*」) に対して次のアクションを実行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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingComprehendEndpointPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint](#) され、Application Auto Scaling がユーザーに代わって Amazon Comprehend を呼び出し CloudWatch、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース（「リソース」：「*」）に対して次のアクションを実行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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingDynamoDBTablePolicy](#)

このポリシーは、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 [AWSServiceRoleForApplicationAutoScaling_DynamoDBTable](#) され、Application Auto Scaling がユーザーに代わって DynamoDB and CloudWatch を呼び出し、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース（「リソース」：「*」）に対して次のアクションを実行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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingECSServicePolicy](#)

このポリシーは、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 [AWSServiceRoleForApplicationAutoScaling_ECSService](#) され、Application Auto Scaling がユーザーに代わって Amazon ECS を呼び出し CloudWatch、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース（「リソース」：「*」）に対して次のアクションを実行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: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: ElastiCache および CloudWatch

ポリシー名: [AWSApplicationAutoscalingElastiCacheRGPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG](#) され、Application Auto Scaling がユーザーに代わって ElastiCache を呼び出し CloudWatch、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

アクセス許可ポリシーにより、Application Auto Scaling は指定されたリソースに対して次のアクションを実行できます。

- 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 `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`

- アクション: `cloudwatch:DeleteAlarms`

AWS マネージドポリシー: Amazon Keyspaces および CloudWatch

ポリシー名: [AWSApplicationAutoscalingCassandraTablePolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_CassandraTable](#) され、Application Auto Scaling がユーザーに代わって Amazon Keyspaces を呼び出し CloudWatch 、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

アクセス許可ポリシーにより、Application Auto Scaling は指定されたリソースに対して次のアクションを実行できます。

- アクション: 次のリソース `cassandra:Select` で :
 - `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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingLambdaConcurrencyPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency](#) され、Application Auto Scaling がユーザーに代わって Lambda を呼び出し CloudWatch 、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」 : 「*」) に対して次のアクションを実行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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingKafkaClusterPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_KafkaCluster](#) され、Application Auto Scaling がユーザーに代わって Amazon MSK CloudWatch を呼び出し、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」 : 「*」) に対して次のアクションを実行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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingNeptuneClusterPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_NeptuneCluster](#) され、Application Auto Scaling がユーザーに代わって Neptune CloudWatch と を呼び出し、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

アクセス許可ポリシーにより、Application Auto Scaling は指定されたリソースに対して次のアクションを実行できます。

- 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*:aws: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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingSageMakerEndpointPolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint](#) され、Application Auto Scaling がユーザーに代わって SageMaker を呼び出し CloudWatch、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

アクセス許可ポリシーにより、Application Auto Scaling は指定されたリソースに対して次のアクションを実行できます。

- 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 および CloudWatch

ポリシー名: [AWSApplicationAutoscalingEC2SpotFleetRequestPolicy](#)

このポリシーは、[AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest](#) 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 がユーザーに代わって Amazon EC2 を呼び出し CloudWatch、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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」: 「*」) に対して次のアクションを実行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: カスタムリソースと CloudWatch

ポリシー名: [AWSApplicationAutoScalingCustomResourcePolicy](#)

このポリシーは、 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 [AWSServiceRoleForApplicationAutoScaling_CustomResource](#)、Application Auto Scaling が API Gateway を介して利用可能なカスタムリソースを呼び出し、CloudWatch 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

アクセス許可ポリシーは、Application Auto Scaling がすべての関連リソース (「リソース」: 「*」) に対して次のアクションを実行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に対する Application Auto Scaling の更新

Application Auto Scaling の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。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 |
|---|--|------------------|
| Application Auto Scaling が SageMaker サービスにリンクされたロールにアクセス許可を追加する | このポリシーは、DescribeInferenceComponent および UpdateInferenceComponentRuntimeConfig API アクションを SageMaker 呼び出すアクセス許可を サービスに付与し、今後の統合のための SageMaker リソースの自動スケーリングの互換性をサポートするように | 2023 年 11 月 13 日 |

| 変更 | 説明 | 日付 |
|--|---|------------------------|
| | <p>になりました。このポリシーでは、CloudWatch PutMetric Alarm および 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。</p> | |
| <p>Application Auto Scaling は Neptune ポリシーを追加します</p> | <p>Application Auto Scaling が、Neptune 向けの新しいマネージドポリシーを追加しました。このポリシーは、Application Auto Scaling がユーザーに代わって Neptune CloudWatch と を呼び出し、スケーリングを実行できるようにするサービスにリンクされたロールにアタッチされます。</p> | <p>2021 年 10 月 6 日</p> |
| <p>Application Auto Scaling が Redis ポリシー ElastiCache に追加</p> | <p>Application Auto Scaling に の新しい管理ポリシーが追加されました ElastiCache。このポリシーは、Application Auto Scaling がユーザーに代わって ElastiCache CloudWatch および を呼び出し、スケーリングを実行できるようにするサービスにリンクされたロールにアタッチされます。</p> | <p>2021 年 8 月 19 日</p> |
| <p>Application Auto Scaling が変更の追跡を開始</p> | <p>Application Auto Scaling が AWS マネージドポリシーの変更の追跡を開始しました。</p> | <p>2021 年 8 月 19 日</p> |

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、AWS 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。これは、リンクされたサービスのみが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です。

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、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ますようにします。

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。

```
{
  "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_EC2Service` 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)

以下の [create-service-linked-role](#) CLI コマンドを使用して、Application Auto Scaling サービスリンク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。例えば、AWS Lambda サービスにリンクされたロールを作成するには、`lambda.application-autoscaling.amazonaws.com` を使用します。

```
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「[サービスリンクロールの編集](#)」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「[サービスの削除](#)」を参照してください。

サービスリンクロールは、IAM を使用して削除できます。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 と連携 AWS のサービス する各 のサービスにリンクされたロールの 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:: <i>012345678910</i> :role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet |
| Aurora | arn:aws:iam:: <i>012345678910</i> :role/aws-service-role/rds.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster |
| Comprehend | arn:aws:iam:: <i>012345678910</i> :role/aws-service-role/comprehend.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint |
| DynamoDB | arn:aws:iam:: <i>012345678910</i> :role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable |

| サービス | ARN |
|-------------|--|
| ECS | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/ecs.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService</code> |
| ElastiCache | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/elasticache.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG</code> |
| Keyspaces | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/cassandra.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CassandraTable</code> |
| Lambda | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/lambda.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency</code> |
| MSK | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/kafka.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_KafkaCluster</code> |
| Neptune | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/neptune.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_NeptuneCluster</code> |
| SageMaker | <code>arn:aws:iam:: 012345678910 :role/aws-service-role/sagemaker.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint</code> |

| サービス | ARN |
|-------------|---|
| Spot Fleets | arn:aws:iam:: 012345678910 :role/aws-service-role/ec2.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest |
| カスタムリソース | arn:aws:iam:: 012345678910 :role/aws-service-role/custom-resource.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CustomResource |

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内の [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、内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 に対する API アクションに必要なアクセス許可 CloudWatch](#)
- [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のアクセス許可を持っていることを確認する必要があります CloudWatch (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。

```
{
  "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。

```
{
  "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。

```
{
  "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 に対する API アクションに必要なアクセス許可 CloudWatch

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、エンドユーザーに 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付与する必要があります。次のポリシーを使用して、ターゲット サービスと を操作するために必要な最小限のアクセス許可を付与します CloudWatch。

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 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

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 AppStream 2.0 および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "appstream:DescribeFleets",
        "appstream:UpdateFleet",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch>DeleteAlarms"
      ],
      "Resource": "*"
    }
  ]
}
```

Aurora レプリカ

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 Aurora および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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

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 Amazon Comprehend および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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

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 DynamoDB および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:UpdateTable",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch>DeleteAlarms"
      ],
      "Resource": "*"
    }
  ]
}
```

ECS サービス

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 ECS および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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

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 ElastiCache および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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

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 Amazon EMR および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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

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 Amazon Keyspaces および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select",
        "cassandra:Alter",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch>DeleteAlarms"
      ],
      "Resource": "*"
    }
  ]
}
```

Lambda 関数

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 Lambda アクションと CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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

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 Amazon MSK および CloudWatch API アクションにアクセス許可を付与します。

```

{
  "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

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 Neptune および CloudWatch API アクションにアクセス許可を付与します。

```

{
  "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 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 および CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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)

次のアイデンティティベースのポリシーは、必要なすべてのスポットフリートおよび CloudWatch API アクションにアクセス許可を付与します。

```
{
  "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。このポリシーは、必要なすべての CloudWatch アクションにアクセス許可も付与します。

```
{
  "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、これらは 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ために、ユーザーは CloudWatch アラームを作成および管理するための追加のアクセス許可が必要に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。

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

このポリシーにより、コンソールユーザーは Amazon EC2 コンソールでスケーリングポリシーを表示および変更し、CloudWatch コンソールで CloudWatch アラームを作成および管理できます。

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、必要に応じてアクセス CloudWatch 許可を調整して、機能への CloudWatch 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 「コンソールに必要なアクセス許可」](#) 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 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る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およびのリソースにアクセスするために使用するすべての API アクションにアクセスするためのアクセス許可を持っていることを確認する必要があります CloudWatch。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リングを初めて設定するとき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 を呼び出すアクセス許可を付与する必要があります CloudWatch。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 コールの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 コールの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 発信者がターゲットサービスおよびの 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 CloudWatch。Application Auto Scaling は、リクエストを続行する CloudWatch 前に、ターゲットサービスとの両方に関連付けられたリクエストのアクセス許可を検証します。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。

これがどのように機能するかの一例として、以下の情報は、Application Auto Scaling が Aurora とでアクセス許可の検証を実行する方法に関する詳細を提供します CloudWatch。

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、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) が必要なため、ARN () を構築するには、無効なアカウント ID (12345) とダミー

インスタンス ID (non-existing-db) を使用して「ダミー」リソースを指定する必要があります。arn:aws:rds:us-east-1:12345:db:non-existing-db。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:DescribeDBCluster: オート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:DescribeDBInstance: スケーラブルターゲットに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、DescribeDBInstance 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: DescribeAlarms API を呼び出します。レコードの最大数は 1 に設定されています。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: PutMetricAlarm 上記のように、DeleteAlarms リクエストするパラメータは提供されません。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

を使用して Application Auto Scaling イベントのアラートを受信すると CloudTrail、これらのアラートには、デフォルトでユーザーのアクセス許可を検証するための Application Auto Scaling 呼び出しが含まれます。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にパブリック 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ト管理の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「[AWS のサービスによるアクセス 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 エンドポイントのうち、使用中のものを使用して、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、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティゾーンを提供します。

アベイラビリティゾーンでは、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 [ガインフラストラクチャ AWS](#) を保護する方法については、[AWS 「クラウドセキュリティ」](#) を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の [「Infrastructure Protection」](#) を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で 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 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、[AWS Security Token Service](#) (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

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、[コンプライアンスプログラム AWS のサービスによる対象範囲内のコンプライアンスプログラム](#) を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、[AWS 「コンプライアンスプログラム」](#) を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、[「でのレポートのダウンロード AWS Artifact」](#) の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービスは、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- [セキュリティとコンプライアンスのクイックスタートガイド](#) – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- [アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ](#) – このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

 Note

すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「[HIPAA 対応サービスのリファレンス](#)」を参照してください。

- [AWS コンプライアンスリソース](#) – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- [AWS カスタマーコンプライアンスガイド](#) – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「[デベロッパーガイド](#)」の「[ルールによるリソースの評価](#)」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。AWS Config
- [AWS Security Hub](#) – これにより AWS のサービス、内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「[Security Hub のコントロールリファレンス](#)」を参照してください。
- [Amazon GuardDuty](#) – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。

- [AWS Audit Manager](#) – これにより AWS のサービス、AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Application Auto Scaling のクォータ

には、ごとに、以前 AWS アカウント は制限と呼ばれていた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 |
|--|--|--------|
| リソースタイプあたり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とターゲット追跡ポリシーの両方) | 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 | いいえ |

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 |
|--|---|------------------|
| 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 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 |
| IAM サービスリンクロール許可に対する更新 | Application Auto Scaling ポリシーが <code>AWSApplicationAutoscalingSageMakerEndpointPolicy</code> ポリシーを更新します。詳細については、「 AWS マネージドポリシーに対する Application Auto Scaling 更新 」を参照してください。 | 2023 年 11 月 13 日 |
| SageMaker Serverless プロビジョニング済み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ラブルなターゲットを分類する](#)

これで、Application Auto Scaling スケ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 Metric Math のサポート](#)

ターゲット追跡スケラ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 を使用すると、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いて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ラ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

[ガイドの変更点](#)

Application Auto Scaling ユーザーガイドのトピックは、Application Auto Scaling で AWS CloudShell を使い始めるのに役立ちます。[コマンドラインから「AWS CloudShell を使用して Application Auto Scaling を操作する」](#)を参照してください。

2023 年 2 月 17 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レポートするようになりました](#)

2021 年 8 月 19 日以降、管理ポリシーに対する変更は、[「Application Auto Scaling による AWS 管理ポリシーの更新」](#)のトピックで報告されます。リストされた最初の変更は、ElastiCache for Redis に必要なアクセス許可の追加です。

2021 年 8 月 19 日

[ElastiCache for Redis 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 for Redis 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、[「for ElastiCache Redis」](#)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 でのアクション用の新しい IAM アクセス許可ポリシーの例を追加しました CloudWatch。詳細については、「[AWS CLI または SDK を操作するポリシーの例](#)」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ルリング](#)」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。また、新しい[トピック](#)では、CloudWatch が注意が必要な問題を検出したときに通知をトリガーする方法についても説明します。

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 に送信 EventBridge 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リングします。 | 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 |
| 新規セクション | 「 セットアップ 」セクションが Application Auto Scaling ドキュメントに追加されました。マイナー改善と修正がユーザーガイドに加えられました。 | 2019 年 6 月 28 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 エンドポイントバリエーションのサポートを追加](#)

Application Auto Scaling を使用して、バリエーション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ルリングのサポートを追加 | スケジュールに基づくスケールリングを使用して、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ルリング 」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「 Fleet Auto Scaling for 2.0 」を参照してください。 AppStream | 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。Amazon EC2 | 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。