aws

ユーザーガイド

Amazon EC2 Auto Scaling

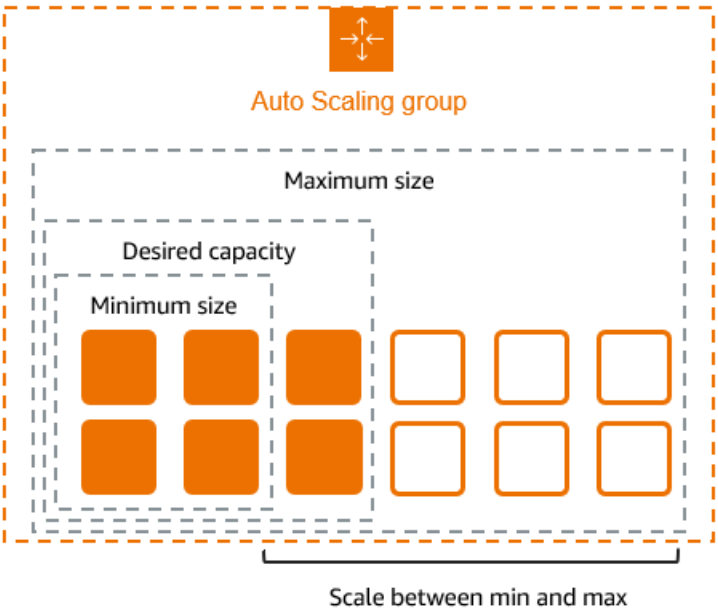
Amazon EC2 Auto Scaling: ユーザーガイド

Table of Contents

Amazon EC2 Auto Scaling とは

Amazon EC2 Auto Scaling は、アプリケーションの負荷を処理するために適切な数の Amazon EC2 インスタンスが使用可能であることを確認するのに役立ちます。Auto Scaling グループと呼ばれる EC2インスタンスのコレクションを作成します。各 Auto Scaling グループでインスタンスの最小数を指定できます。Amazon EC2 Auto Scaling により、グループがこのサイズを下回ることはありません。各 Auto Scaling グループでインスタンスの最大数を指定できます。Amazon EC2 Auto Scaling により、グループがこのサイズを超えることはありません。グループの作成時または作成後の任意の時点で希望する容量を指定すると、Amazon EC2 Auto Scaling はグループがこれだけの数のインスタンスを持つようにします。スケーリングポリシーを指定した場合、Amazon EC2 Auto Scaling はアプリケーションの需要が増減するとインスタンスを起動または終了できます。

例えば、次の Auto Scaling グループで、インスタンス数の最小サイズが 4、希望するキャパシティが 6、最大サイズが 12 であるとします。定義するスケーリングポリシーによって、指定した条件に基づいて、インスタンスの最小数と最大数の間でインスタンス数が調整されます。



Amazon EC2 Auto Scaling の機能

Amazon EC2 Auto Scaling では、EC2インスタンスは Auto Scaling グループに整理されるため、スケーリングと管理の目的で論理単位として扱われます。Auto Scaling グループは、EC2インスタンスの設定テンプレートとして起動テンプレート (または起動設定) を使用します。

Amazon EC2 Auto Scaling の主な機能は次のとおりです。

実行中のインスタンスのヘルスのモニタリング

Amazon EC2 Auto Scaling は、EC2ヘルスチェックを使用してインスタンスのヘルスと可用性を自動的にモニタリングし、終了したインスタンスや障害が発生したインスタンスを置き換えて、希望する容量を維持します。

カスタムヘルスチェック

組み込みのヘルスチェックに加えて、アプリケーションに固有のカスタムヘルスチェックを定義して、期待どおりに応答していることを確認できます。インスタンスがカスタムヘルスチェックに不合格の場合、希望するキャパシティを維持するために自動的に置き換えられます。

アベイラビリティーゾーン間でのキャパシティのバランス調整。

Auto Scaling グループには複数のアベイラビリティーゾーンを指定できます。Amazon EC2 Auto Scaling は、グループのスケーリングに応じて、アベイラビリティーゾーン間でインスタンスを均等に分散します。これにより、1 か所の障害があってもアプリケーションが保護されることで、高可用性と耐障害性が得られます。

複数のインスタンスタイプと購入オプション

単一の Auto Scaling グループで、複数のインスタンスタイプと購入オプション (スポットインスタンスとオンデマンドインスタンス) を起動できるため、スポットインスタンスの使用によってコストを最適化できます。また、リザーブドインスタンスと Savings Plans の割引は、グループ内のオンデマンドインスタンスと併用することで活用できます。

スポットインスタンスの自動交換

グループにスポットインスタンスが含まれている場合、スポットインスタンスが中断されると、Amazon EC2 Auto Scaling は自動的に代替スポット容量をリクエストできます。キャパシティの再調整を通じて、Amazon EC2 Auto Scaling は中断のリスクが高いスポットインスタンスをモニタリングし、事前に置き換えることもできます。

負荷分散

Elastic Load Balancing の負荷分散とヘルスチェックを使用して、正常なインスタンスへのアプリケーショントラフィックの均等な分散を確保できます。インスタンスが起動または終了されるたびに、Amazon EC2 Auto Scaling はロードバランサーからインスタンスを自動的に登録および登録解除します。

スケーラビリティ

Amazon EC2 Auto Scaling には、Auto Scaling グループをスケーリングするためのいくつかの方法も用意されています。自動スケーリングを使用すると、ピーク負荷を処理するキャパシティ

を追加し、需要が低い場合にキャパシティを削除することで、アプリケーションの可用性を維持し、コストを削減できます。必要に応じて、Auto Scaling グループのサイズを手動で調整することもできます。

インスタンスの更新

インスタンスの更新機能は、 AMIまたは 起動テンプレートを更新するときに、ローリング方式でインスタンスを更新するメカニズムを提供します。Canary デプロイと呼ばれる段階的なアプローチを使用して、グループ全体にロールアウトする前に、少数のインスタンスで新しいテンプレートAMIまたは起動テンプレートをテストすることもできます。

ライフサイクルフック

ライフサイクルフックは、新しいインスタンスの起動時またはインスタンスが終了する前に呼び出されるカスタムアクションを定義するのに役立ちます。この機能は、イベント駆動型アーキテクチャの構築に特に役立ちますが、ライフサイクルを通じてインスタンスを管理することもできます。

ステートフルワークロードのサポート

ライフサイクルフックには、シャットダウン時に状態を維持するメカニズムも用意されています。ステートフルアプリケーションの継続性を確保するために、スケールイン保護ポリシーまたはカスタム終了ポリシーを使用して、実行時間が長いプロセスのインスタンスが早期に終了するのを防ぐこともできます。

Amazon EC2 Auto Scaling の利点の詳細については、「」を参照してくださいアプリケーションアーキテクチャに対する Auto Scaling の利点。

Amazon EC2 Auto Scaling の料金

Amazon EC2 Auto Scaling では追加料金は発生しないため、簡単に試して、 AWS アーキテクチャにどのようなメリットがあるかを確認してください。使用した AWS リソース (EC2インスタンス、EBSボリューム、 CloudWatch アラームなど) に対してのみ料金が発生します。

はじめに

使用を開始するには、「最初の Auto Scaling グループを作成する」チュートリアルを最後まで行って Auto Scaling グループを作成し、そのグループ内のインスタンスが終了するときにどのように応答するかを確認してください。

Auto Scaling グループの使用

Auto Scaling グループは、以下のインターフェイスのいずれかを使用して、作成、アクセス、および管理することできます。

- AWS Management Console – Auto Scaling グループへのアクセスに使用できるウェブインターフェイスを提供します。にサインアップしている場合は AWS アカウント、 にサインインし AWS Management Console、ナビゲーションバーの検索ボックスを使用して Auto Scaling グループを検索し、Auto Auto Scaling Auto Scaling グループにアクセスできます。
- AWS Command Line Interface （ AWS CLI ） – 幅広い のセットのコマンドを提供し 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 CLI を使用する準備」を参照してください。詳細については、AWS CLI コマンドリファレンスの「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 コマンドレットリファレンス」を参照してください。
- AWS SDKs – 言語固有のAPIオペレーションを提供し、署名の計算、リクエストの再試行処理、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、「AWS SDKs」を参照してください。
- クエリ 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。クエリの使用はAPI、 にアクセスする最も直接的な方法です AWS のサービス。ただし、この方法では、リクエストに署名するハッシュの生成やエラー処理など、低レベルの詳細な作業をアプリケーションで処理する必要があります。詳細については、「Amazon EC2 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 Auto Scaling グループの作成をサポートします。詳細については、「AWS CloudFormation で Auto Scaling グループを作成する」を参照してください。

プログラムで に接続するには AWS のサービス、 エンドポイントを使用します。Amazon EC2 Auto Scaling への呼び出しのエンドポイントについては、「 エンドポイント」の「Amazon EC2 Auto Scaling エンドポイントとクォータAWS 全般のリファレンス」および「 中国の Amazon Web Services の」の「中国での Amazon Web Services の開始方法」の」を参照してください。

アプリケーションアーキテクチャに対する Auto Scaling の利点

アプリケーションアーキテクチャに Amazon EC2 Auto Scaling を追加するのは、 AWS クラウドの利点を最大化する方法の 1 つです。Amazon EC2 Auto Scaling を使用すると、アプリケーションには次の利点があります。

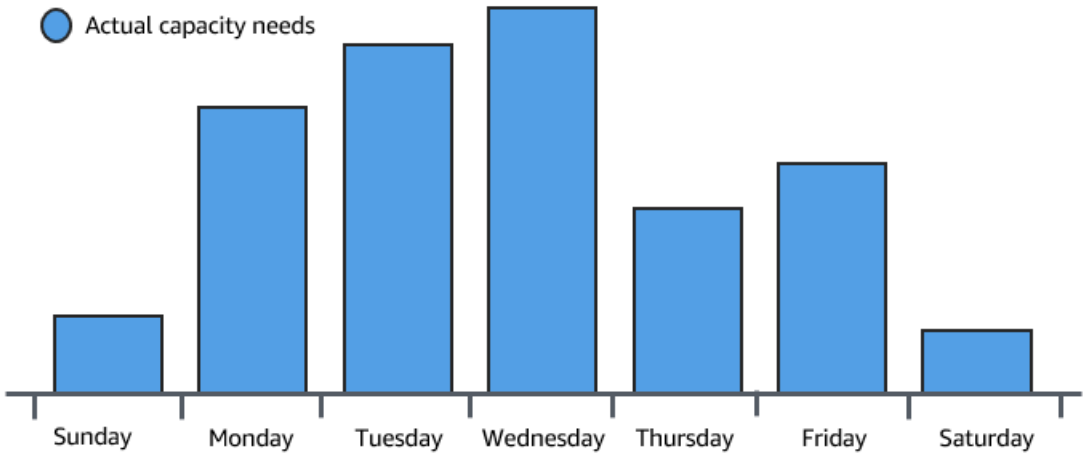
- 耐障害性の向上。Amazon EC2 Auto Scaling は、インスタンスに異常があるタイミングを検出して終了し、インスタンスを起動して置き換えることができます。複数のアベイラビリティーゾーンを使用するように Amazon EC2 Auto Scaling を設定することもできます。あるアベイラビリティーゾーンが使用できなくなった場合、Amazon EC2 Auto Scaling は別のアベイラビリティーゾーンでインスタンスを起動して補正できます。
- 可用性の向上。Amazon EC2 Auto Scaling は、アプリケーションが現在のトラフィック需要を処理するために常に適切な容量を確保できるようにします。
- コスト管理の強化。Amazon EC2 Auto Scaling は、必要に応じて容量を動的に増減できます。使用したEC2インスタンスに対して料金を支払うため、必要なときにインスタンスを起動し、必要でないときにインスタンスを終了することでコストを削減できます。

内容

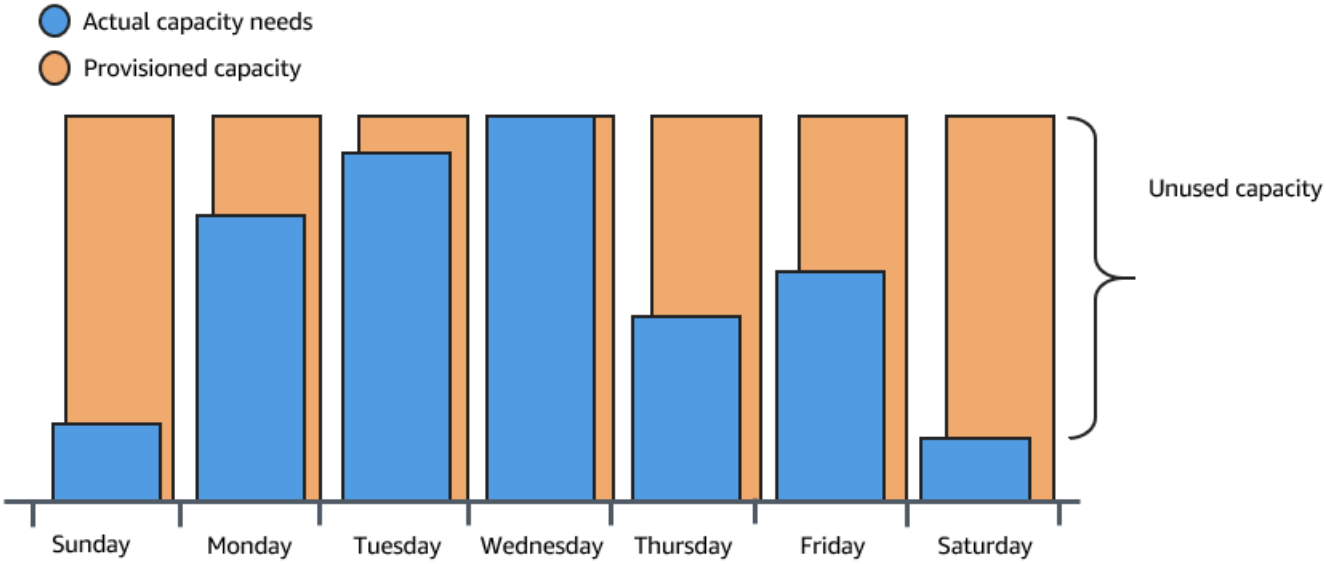
例: 変化する需要に対応する

Amazon EC2 Auto Scaling の利点のいくつかを示すには、 で実行されている基本的なウェブアプリケーションを検討してください AWS。このアプリケーションでは、従業員は会議に使用できる会議室を検索できます。週の始めと終わりには、このアプリケーションの使用量は最小になります。週の中盤では、より多くの従業員が会議をスケジュールしているため、アプリケーションの使用量が大幅に増加します。

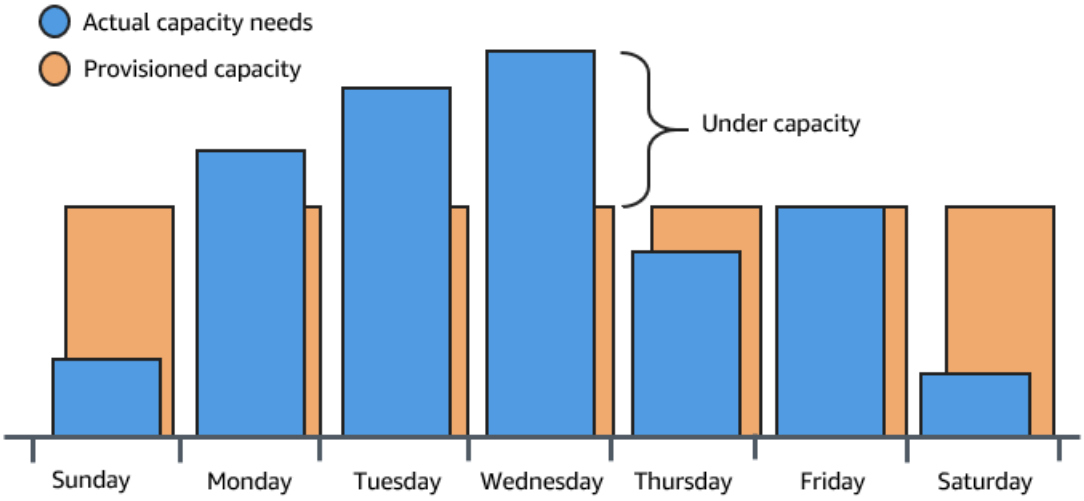
以下のグラフは、1 週間を通じてアプリケーションの処理能力がどのくらい使用されているかを示しています。



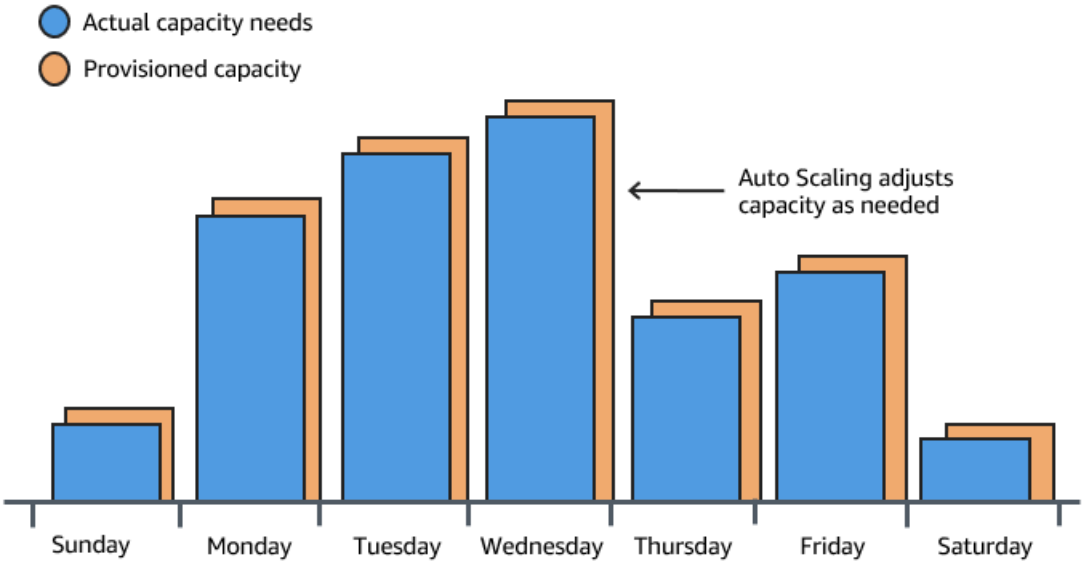
従来、このような処理能力の変化に対応した計画を立てるには 2 通りの方法がありました。最初の方法は、アプリケーションが使用量に応じて十分な処理能力を常に確保できるように、十分な数のサーバーを追加する方法です。ただし、この方法の欠点は、アプリケーションがこの十分な処理能力を必要としない日数が発生することです。余分な処理能力が未使用のままになり、実質的には、アプリケーションの実行を維持するためのコストが増加します。



2 番目の方法は、アプリケーションの平均的な使用量を処理するための十分な処理能力を確保する方法です。この方法では、使用される機会が少ない機器を購入しないため、コストはあまりかかりません。ただし、アプリケーションの使用量が処理能力を超えた場合に、カスタマーエクスペリエンスが低下するというリスクがあります。



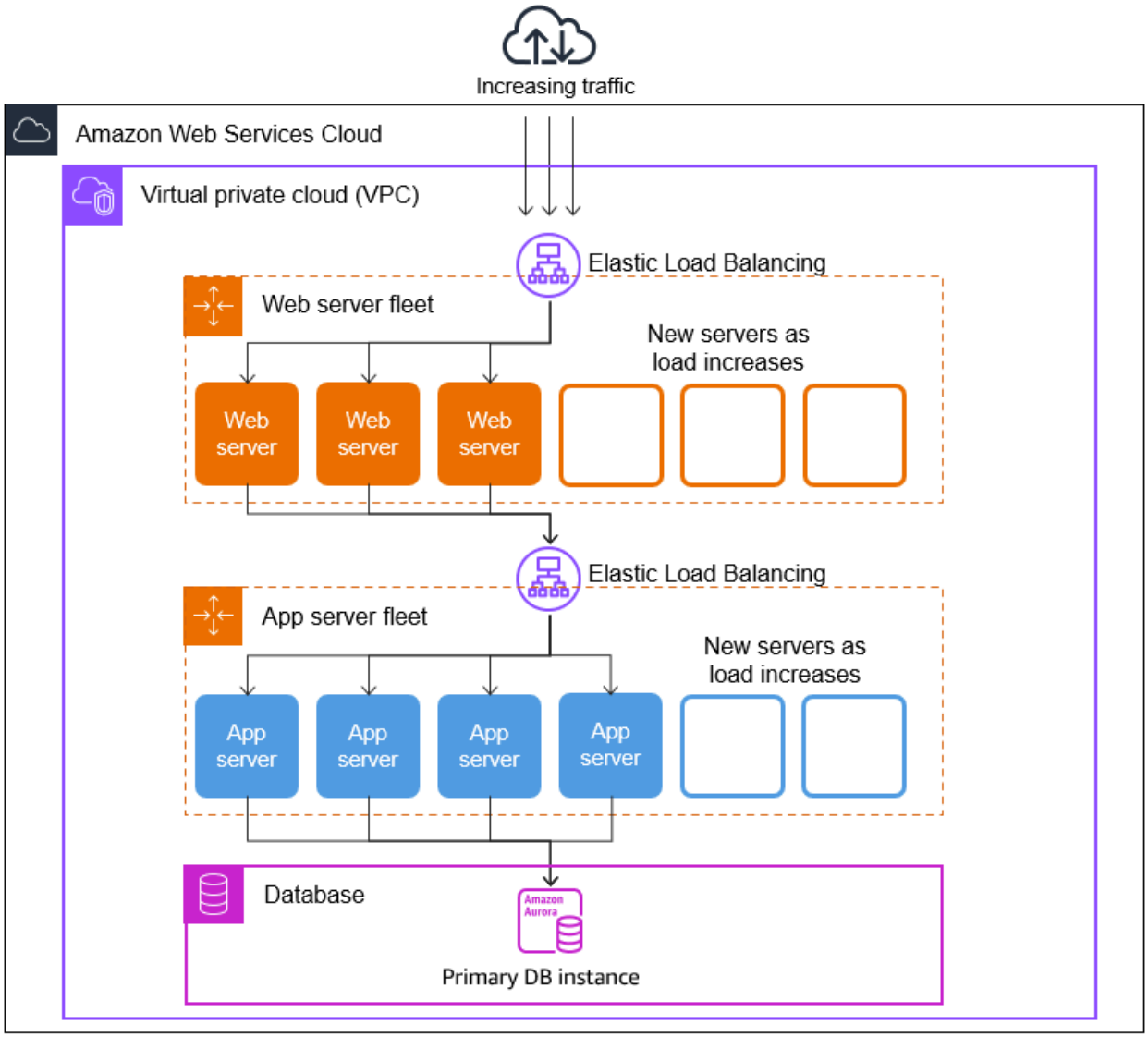
このアプリケーションに Amazon EC2 Auto Scaling を追加することで、3 つ目のオプションを使用できます。必要な場合にのみアプリケーションに新しいインスタンスを追加し、それらのインスタンスが不要になった場合は、インスタンスを終了できます。Amazon EC2 Auto Scaling はEC2インスタンスを使用するため、使用したインスタンスに対してのみ料金を支払う必要があります。この方法により、費用効率が高いアーキテクチャの利用が可能になります。このようなアーキテクチャでは最適なカスタマーエクスペリエンスが提供され、費用を最小限に抑えることができます。



例: ウェブアプリアーキテクチャ

一般的なウェブアプリケーションのシナリオでは、顧客のトラフィックのボリュームに対応するために、アプリケーションの複数のコピーを同時に実行します。アプリケーションのこれらの複数のコピーは、同じEC2インスタンス (クラウドサーバー) でホストされ、それぞれがお客様のリクエストを処理します。

Amazon EC2 Auto Scaling は、ユーザーに代わってこれらのEC2インスタンスの起動と終了を管理します。Auto Scaling グループがEC2インスタンスを起動または終了するタイミングを決定する一連の条件 (Amazon CloudWatch アラームなど) を定義します。Auto Scaling グループをネットワークアーキテクチャに追加することによって、アプリケーションの可用性と耐障害性を向上させることができます。



Auto Scaling グループは必要な数だけ作成できます。例えば、各層のための Auto Scaling グループを作成できます。

Auto Scaling グループ内のインスタンス間のトラフィックを分散させるために、アーキテクチャにロードバランサーを導入できます。詳細については、「Elastic Load Balancing」を参照してください。

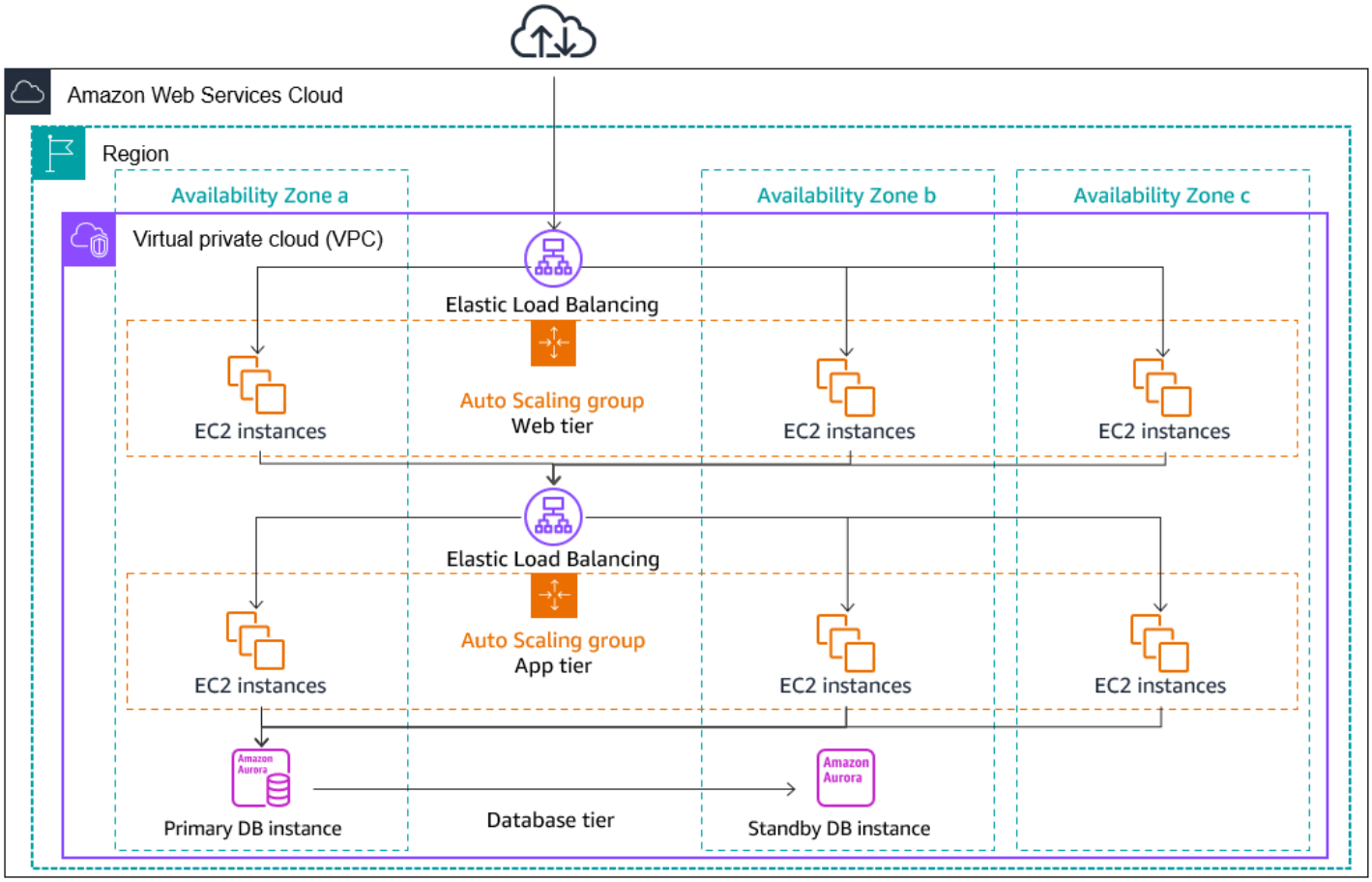
例: 複数のアベイラビリティーゾーン全体にインスタンスを分散させる

アベイラビリティーゾーンは、所定の AWS リージョンにある隔離された場所です。各リージョンには、そのリージョンに対して高可用性を提供するために設計された複数のアベイラビリティーゾーン (AZ) があります。アベイラビリティーゾーンは独立しているため、複数のゾーンを使用するようにアプリケーションを設計すると、アプリケーションの可用性が向上します。詳細については、「Amazon EC2 Auto Scaling のレジリエンス」を参照してください。

アベイラビリティーゾーンは、 AWS リージョン コードの後に文字識別子 (などus-east-1a) が続きます。デフォルトの を使用するのではなく、 VPCおよび サブネットを作成する場合はVPC、各アベイラビリティーゾーンに 1 つ以上のサブネットを定義できます。各サブネットが完全に 1 つのアベイラビリティーゾーン内に含まれている必要があり、1 つのサブネットが複数のゾーンに、またがることはできません。詳細については、「Amazon ユーザーガイド」の「Amazon のVPC仕組み」を参照してください。 VPC

Auto Scaling グループを作成するときは、Auto Scaling グループをデプロイする VPCおよび サブネットを選択する必要があります。Amazon EC2 Auto Scaling は、選択したサブネットにインスタンスを作成します。したがって、各インスタンスは Amazon EC2 Auto Scaling によって選択された特定のアベイラビリティーゾーンに関連付けられます。インスタンスが起動すると、Amazon EC2 Auto Scaling は高可用性と信頼性のためにインスタンスをゾーン間で均等に分散しようとします。

以下の画像は、3 つのアベイラビリティーゾーン間にデプロイされた多層アーキテクチャの概要です。



インスタンスの分散

Amazon EC2 Auto Scaling は、有効な各アベイラビリティーゾーンで同等の数のインスタンスを自動的に維持しようとします。Amazon EC2 Auto Scaling は、インスタンスが最も少ないアベイラビリティーゾーンで新しいインスタンスを起動しようとすることでこれを行います。アベイラビリティーゾーンに複数のサブネットが選択されている場合、Amazon EC2 Auto Scaling は、アベイラビリティーゾーンで使用可能な IP アドレスの数が最も多いサブネットでインスタンスの起動を試みます。ただし、試行が失敗した場合、Amazon EC2 Auto Scaling は成功するまで別のアベイラビリティーゾーンでインスタンスの起動を試みます。

アベイラビリティーゾーンが異常、または利用不能な状況では、アベイラビリティーゾーン間でのインスタンスの分散が不均等になる可能性があります。アベイラビリティーゾーンが回復すると、Amazon EC2 Auto Scaling は Auto Scaling グループを自動的に再調整します。これは、インスタンス数が最も少ない有効なアベイラビリティーゾーンでインスタンスを起動し、その他のゾーンでインスタンスを終了することによって行われます。

アクティビティの再分散

再分散アクティビティは、アベイラビリティーゾーンの再調整とキャパシティ再調整の 2 つのカテゴリに分類されます。

アベイラビリティーゾーンの再調整

特定のアクションが発生すると、Auto Scaling グループはアベイラビリティーゾーン間で不均衡になる可能性があります。Amazon EC2 Auto Scaling は、アベイラビリティーゾーンのバランスを再調整して補正します。再調整アクティビティは、以下のアクションが原因で行われる場合があります。

- Auto Scaling グループに関連付けられたアベイラビリティーゾーンを変更する。
- インスタンスを明示的に終了もしくはデタッチする、またはインスタンスをスタンバイにすることでグループのバランスが悪くなる。
- それまでキャパシティが不足していたアベイラビリティーゾーンが回復し、使用できるキャパシティが増えた。
- これまでスポット価格が上限価格より高かったアベイラビリティーゾーンで、スポット価格が上限価格より低くなった。

再調整時に、Amazon EC2 Auto Scaling は以前のインスタンスを終了する前に新しいインスタンスを起動します。そうすることで、再調整によってアプリケーションのパフォーマンスや可用性が損なわれることがなくなります。

Amazon EC2 Auto Scaling は、以前のインスタンスを終了する前に新しいインスタンスの起動を試みるため、指定された最大容量またはそれに近い状態になると、再調整アクティビティが妨げられたり、完全に停止する可能性があります。

この問題を回避するため、システムは、再調整アクティビティの実行中に指定された最大キャパシティを一時的に超過することができます。デフォルトでは、超過は、10% または 1 インスタンスのどちらか大きい方の範囲で行うことができます。この超過範囲は、グループのキャパシティが最大またはそれに近く、再調整が必要な場合にのみ拡張されます。この追加キャパシティーは、グループの再分散に要する時間にわたってのみ提供されます (通常は数分)。

または、インスタンスメンテナンスポリシーを使用して Auto Scaling グループのしきい値を確立でき、グループはそのしきい値範囲内のキャパシティのみを増減できます。このようにして、グループのバランスが再調整される速度を制御できます。詳細については、「インスタンスメンテナンスポリシー」を参照してください。

キャパシティーの再調整

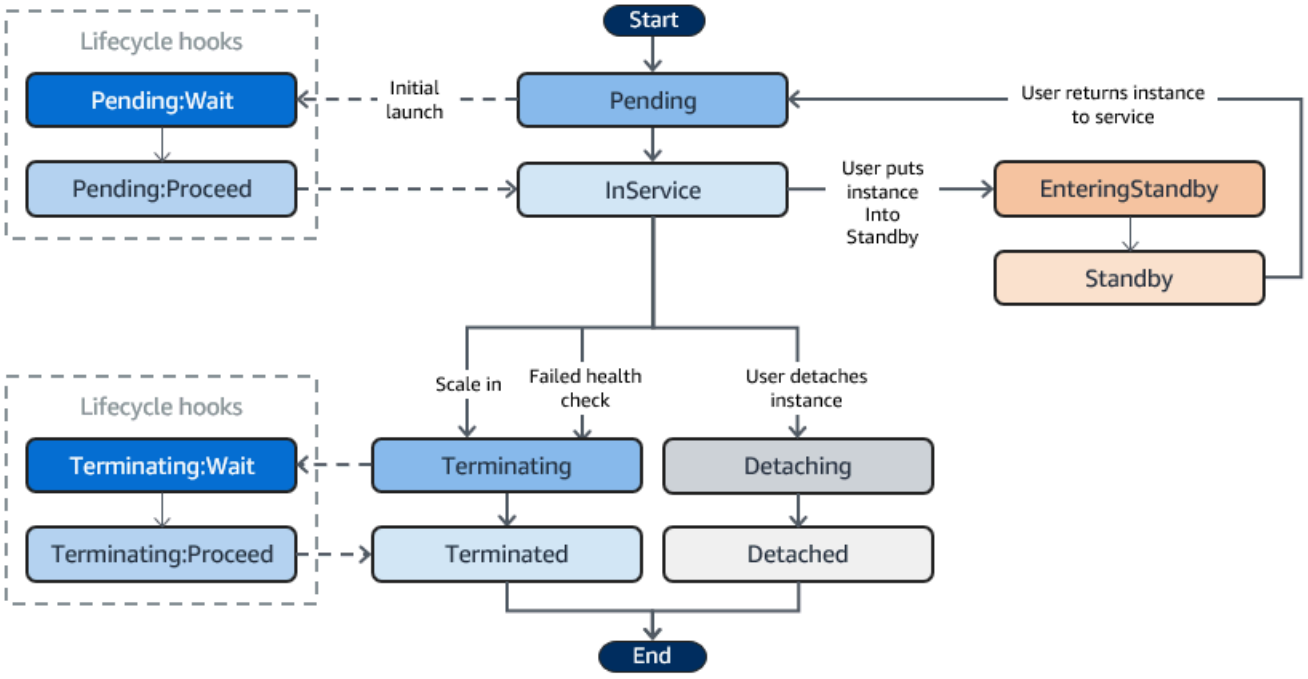
スポットインスタンスを使用するときは、Auto Scaling グループのキャパシティの再調整を有効にできます。これにより、Amazon EC2 Auto Scaling は、スポットインスタンスが中断のリスクが高いことを Amazon がEC2報告するたびに、スポットインスタンスの起動を試みます。新しいインスタンスの起動後、Amazon EC2 Auto Scaling は古いインスタンスを終了します。詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。

Amazon EC2 Auto Scaling インスタンスのライフサイクル

Auto Scaling グループのEC2インスタンスには、他のEC2インスタンスとは異なるパスまたはライフサイクルがあります。Auto Scaling グループがインスタンスを起動すると、ライフサイクルが起動してサービスに組み込まれます。インスタンスを削除するとライフサイクルが終了するか、または Auto Scaling グループがインスタンスをサービスから削除して終了します。

> **Note**
>
> インスタンスが起動すると、サービスを使用していなくてもすぐに課金されます。

次の図は、Amazon EC2 Auto Scaling ライフサイクルのインスタンス状態間の遷移を示しています。



スケールアウト

次のスケールアウトイベントは、Auto Scaling グループにEC2インスタンスを起動してグループにアタッチするように指示します。

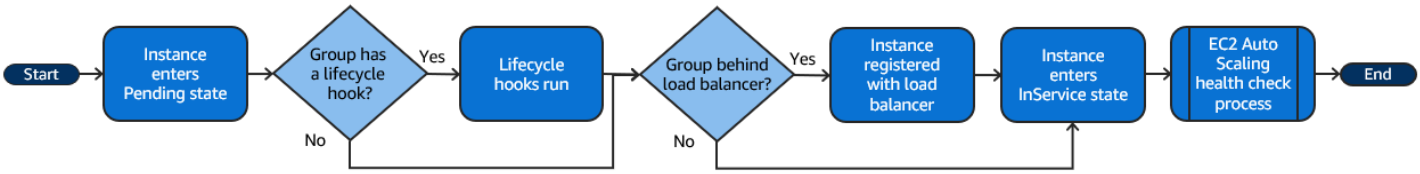
- 手動でグループのサイズを拡大します。詳細については、「既存の Auto Scaling グループの希望するキャパシティを変更する」を参照してください。
- スケーリングポリシーを作成して、指定した需要の増加に基づいてグループのサイズを自動的に拡大します。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。
- スケジュールでスケーリングを設定して、特定の時間でグループのサイズを拡大します。詳細については、「Amazon EC2 Auto Scaling のスケジュールされたスケーリング」を参照してください。

スケールアウトイベントが発生すると、Auto Scaling グループは割り当てられた起動テンプレートを使用して、必要な数のEC2インスタンスを起動します。これらのインスタンスは Pending 状態で起動します。ライフサイクルフックを Auto Scaling グループに追加する場合は、ここでカスタムアクションを実行できます。詳細については、「ライフサイクルフック」を参照してください。

各インスタンスが完全に設定され、Amazon EC2ヘルスチェックに合格すると、Auto Scaling グループにアタッチされ、 InService状態になります。インスタンスは、Auto Scaling グループで必要なキャパシティにカウントされます。

Auto Scaling グループが Elastic Load Balancing ロードバランサーからトラフィックを受信するように設定されている場合、Amazon EC2 Auto Scaling はインスタンスを としてマークする前に、インスタンスをロードバランサーに自動的に登録しますInService。

スケールアウトイベントについて、ロードバランサーにインスタンスを登録する手順の概要を次に示します。



インスタンスが実行中

以下のいずれかが発生するまで、インスタンスは InService 状態のままとなります。

- スケールインイベントが発生し、Amazon EC2 Auto Scaling は Auto Scaling グループのサイズを減らすためにこのインスタンスを終了することを選択します。詳細については、「スケールイン中に終了する Auto Scaling インスタンスを制御する」を参照してください。
- ユーザーがインスタンスを Standby 状態にする 詳細については、「スタンバイを入力し終了します。」を参照してください。
- Auto Scaling グループからインスタンスをデタッチします。詳細については、「Auto Scaling グループからインスタンスをデタッチまたはアタッチする」を参照してください。
- インスタンスが必要な数のヘルスチェックに失敗したため、Auto Scaling グループから削除され、終了されて置き換えられます。詳しくは、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

スケールイン

次のスケールインイベントは、Auto Scaling グループにグループからEC2インスタンスをデタッチして終了するように指示します。

- 手動でグループのサイズを縮小します。詳細については、「既存の Auto Scaling グループの希望するキャパシティを変更する」を参照してください。
- スケーリングポリシーを作成して、指定した需要の縮小に基づいてグループのサイズを自動的に縮小します。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。
- スケジュールでスケーリングを設定して、特定の時間にグループのサイズを縮小します。詳しくは、「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。

作成したそれぞれのスケールアウトイベントについて、対応するスケールインイベントを作成することが重要です。これにより、アプリケーションに割り当てられたリソースが、それらのリソースに対する要求に可能な限り厳密に対応するようになります。

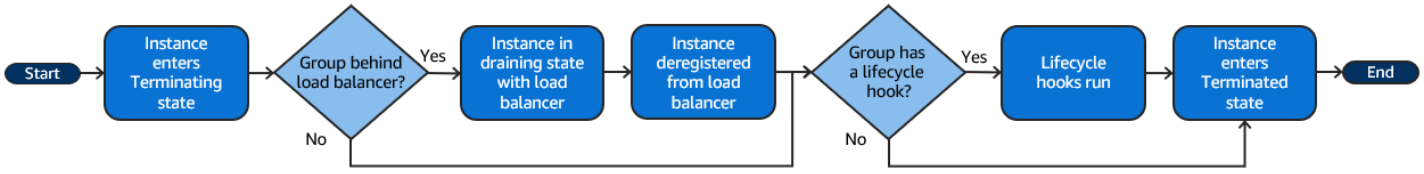
スケールインイベントが発生すると、Auto Scaling グループは 1 つ以上のインスタンスをデタッチします。Auto Scaling グループは終了ポリシーを使用して、終了するインスタンスを決定します。Auto Scaling グループからの終了のプロセス中にあるインスタンスは、Terminating 状態に移行し、稼働状態に戻ることはできません。

Auto Scaling グループが Elastic Load Balancing ロードバランサーからトラフィックを受信するように設定されている場合、Amazon EC2 Auto Scaling は終了インスタンスをロードバランサーから自動的に登録解除します。インスタンスの登録を解除すると、新しいリクエストはすべてロードバラン

サーのターゲットグループ内の他のインスタンスにリダイレクトされ、インスタンスへの既存の接続は登録解除の遅延の期限が切れるまで継続できます。

ライフサイクルフックを Auto Scaling グループに追加する場合は、終了するインスタンスに対してカスタムアクションを実行できます。詳細については、「ライフサイクルフック」を参照してください。最後に、インスタンスは完全に終了し `Terminated` 状態へ移行します。

スケールインイベントについて、ロードバランサーからインスタンスを登録解除する手順の概要を次に示します。



インスタンスのデタッチ

Auto Scaling グループからインスタンスをデタッチできます。インスタンスをデタッチした後で、インスタンスを Auto Scaling グループとは別に管理するか、または別の Auto Scaling グループにアタッチできます。

詳細については、「Auto Scaling グループからインスタンスをデタッチまたはアタッチする」を参照してください。

インスタンスのアタッチ

特定の条件を満たす実行中のEC2インスタンスを Auto Scaling グループにアタッチできます。インスタンスがアタッチされると、Auto Scaling グループの一部として管理されます。

詳細については、「Auto Scaling グループからインスタンスをデタッチまたはアタッチする」を参照してください。

ライフサイクルフック

インスタンスを起動または終了したときにカスタムアクションを実行できるように、ライフサイクルフックを Auto Scaling グループに追加できます。

Amazon EC2 Auto Scaling がスケールアウトイベントに応答すると、1 つ以上のインスタンスを起動します。これらのインスタンスは `Pending` 状態で起動します。Auto Scaling グループに `autoscaling:EC2_INSTANCE_LAUNCHING` ライフサイクルフックを追加すると、インスタンスは `Pending` 状態から `Pending:Wait` 状態に移行します。ライフサイクルアクションを完了したら、

インスタンスは Pending:Proceed 状態に移行します。インスタンスが完全に設定されると、Auto Scaling グループにアタッチされて InService 状態へ移行します。

Amazon EC2 Auto Scaling がスケールインイベントに応答すると、1 つ以上のインスタンスが終了します。これらのインスタンスは Auto Scaling グループからデタッチされ Terminating 状態へ移行します。Auto Scaling グループに autoscaling:EC2_INSTANCE_TERMINATING ライフサイクルフックを追加すると、インスタンスは Terminating 状態から Terminating:Wait 状態に移行します。ライフサイクルアクションを完了したら、インスタンスは Terminating:Proceed 状態に移行します。インスタンスが完全に終了すると、Terminated 状態へ移行します。

詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

スタンバイを入力し終了します。

InService 状態にあるインスタンスを、Standby 状態に移行できます。これによりインスタンスをサービスから削除し、トラブルシューティングや変更を加えてから、サービスに戻すことができます。

Standby状態のインスタンスは引き続き Auto Scaling グループによって管理されます。ただし、このようなインスタンスを稼働状態に戻すまで、それらはアプリケーションのアクティブな部分にはなりません。

詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

Auto Scaling リソースとグループのクォータ

AWS アカウント には、サービスごとに AWS 、以前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。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。

Amazon EC2 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 サービスを選択し、Amazon EC2 Auto Scaling を選択します。

クォータの引き上げをリクエストするには、Service Quotas ユーザーガイド の「クォータ引き上げリクエスト」を参照してください。Service Quotas でクォータがまだ利用できない場合は、[クォータの引き上げをリクエスト] フォームを使用してください。クォータの引き上げは、リクエストされたリージョンに関連付けられます。

Amazon EC2 Auto Scaling リソース

AWS アカウント には、作成できる Auto Scaling グループと起動設定の数に関連する次のクォータがあります。

リソース	デフォルトのクォータ
リージョンあたりの Auto Scaling グループ	500
リージョンごとの起動設定	200

Auto Scaling グループの設定

AWS アカウント には、Auto Scaling グループの設定に関連する次のクォータがあります。変更することはできません。

リソース	クォータ
Auto Scaling グループあたりのスケーリングポリシー	50
Auto Scaling グループあたりのスケジュールされたアクション	125
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
Auto Scaling グループあたりのライフサイクルフック	50
SNS Auto Scaling グループあたりの トピック	10
Auto Scaling グループあたりの Classic Load Balancer	50
Auto Scaling グループごとの Elastic Load Balancing ターゲットグループ	50
VPC Auto Scaling グループあたりのLattice ターゲットグループ	5

Auto Scaling グループのAPIオペレーション

Amazon EC2 Auto Scaling には、Auto Scaling グループをバッチで変更するAPIオペレーションが用意されています。以下は、1 回のオペレーションで許可される項目の最大数 (配列メンバーの最大数) API の制限です。変更することはできません。

操作	配列メンバーの最大数
AttachInstances	20 インスタンス IDs
AttachLoadBalancers	10 個のロードバランサー
AttachLoadBalancerTargetGroups	10 個のターゲットグループ
BatchDeleteScheduledAction	50 個のスケジュールに基づくアクション
BatchPutScheduledUpdateGroupAction	50 個のスケジュールに基づくアクション
DetachInstances	20 インスタンス IDs
DetachLoadBalancers	10 個のロードバランサー
DetachLoadBalancerTargetGroups	10 個のターゲットグループ
EnterStandby	20 インスタンス IDs
ExitStandby	20 インスタンス IDs
SetInstanceProtection	50 インスタンス IDs

Amazon EC2 Auto Scaling のリクエストスロットリング API

Amazon EC2 Auto Scaling APIリクエストは、サービス帯域幅を維持するためにトークンバケットスキームを使用してスロットリングされます。詳細については、「Amazon EC2 Auto Scaling APIリファレンス」のAPI「リクエストレート」を参照してください。

EC2 終了レート

Amazon EC2 Auto Scaling は、Auto Scaling グループがスケールインするときに一度に実行できるEC2インスタンス終了オペレーションの数を動的に決定します。これは、Auto Scaling グループ間で一度に終了されるインスタンスの数にばらつきが見られる可能性があることを意味します。これらの

バリエーションは、Amazon EC2 Auto Scaling がロードバランサーでインスタンスを登録解除する必要があるかどうかなど、外部の考慮事項によって発生します。

その他のサービス

Amazon EC2や Amazon などの他のサービスのクォータはVPC、Auto Scaling グループに影響を与える可能性があります。 Service Quotas を使用して、 内のEC2インスタンスやその他のリソースのクォータを更新できます AWS アカウント。 Service Quotas コンソールでは、使用可能なすべてのサービスクォータを表示し、引き上げをリクエストできます。詳細については、「Service Quotas ユーザーガイド」の「Requesting a quota increase」(クォータ引き上げのリクエスト) を参照してください。

起動テンプレートに固有のクォータについては、「Amazon EC2ユーザーガイド」の「起動テンプレートの制限」を参照してください。

Amazon EC2 Auto Scaling を使用するように設定する

Amazon EC2 Auto Scaling の使用を開始する前に、以下のタスクを完了してください。

タスク

- Amazon EC2 を使用するための準備を整える
- AWS CLI を使用する準備

Amazon EC2 を使用するための準備を整える

Amazon EC2 を使用したことがない場合は、Amazon EC2 のドキュメントで説明されているタスクを完了してください。詳細については、「Amazon EC2 ユーザーガイド」の Amazon EC2 での設定に関するページまたは「Amazon EC2 ユーザーガイド」の Amazon EC2 での設定に関するページを参照してください。

AWS CLI を使用する準備

AWS コマンドラインツールを使用し、システムのコマンドラインでコマンドを発行して Amazon EC2 Auto Scaling および他の AWS タスクを実行できます。

AWS Command Line Interface (AWS CLI) を使用するには、AWS CLI のバージョン 1 または 2 をダウンロードしてインストールしたら、設定します。Amazon EC2 Auto Scaling の同じ機能はバージョン 1 および 2 で利用できます。AWS CLI バージョン 1 をインストールするには、「AWS CLI バージョン 1 ユーザーガイド」の「AWS CLI のインストール、アップデート、アンインストール」を参照してください。AWS CLI バージョン 2 をインストールするには、「AWS CLI バージョン 2 ユーザーガイド」の「AWS CLI の最新バージョンのインストールまたは更新」を参照してください。

AWS CloudShell は、開発環境に AWS CLI のインストールを省き、代わりに AWS Management Console で使用できるようにします。インストールが不要になるだけではなく、認証情報の設定およびリージョンの指定も必要ありません。AWS Management Console セッションはこのコンテキストを AWS CLI に提供します。サポートされている AWS リージョン で AWS CloudShell を使用できます。詳細については、「を使用してコマンドラインから Auto Scaling グループを作成する AWS CloudShell」を参照してください。

詳細については、AWS CLI コマンドリファレンスの「autoscaling」を参照してください。

Amazon EC2 Auto Scaling の使用を開始する

Amazon EC2 Auto Scaling の使用を開始するには、 サービスを紹介するチュートリアルに従ってください。

トピック

- チュートリアル: 最初の Auto Scaling グループを作成する
- チュートリアル: スケーリングとロードバランシングを使用するアプリケーションのセットアップ

Auto Scaling グループのインスタンスのライフサイクルを管理するための特定のツールに焦点を当てた追加のチュートリアルについては、次のトピックを参照してください。

- チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定. このチュートリアルでは、Amazon EventBridge を使用して、Auto Scaling グループのインスタンスで発生するイベントに基づいて Lambda 関数を呼び出すルールを作成する方法を説明します。
- チュートリアル: データスクリプトとインスタンスメタデータを使用してライフサイクル状態を取得する. このチュートリアルでは、インスタンスメタデータサービス (IMDS) を使用してインスタンス自体からアクションを呼び出す方法を説明します。

アプリケーションで使用する Auto Scaling グループを作成する前に、AWS クラウドで実行されるアプリケーションを念入りにレビューするようにしてください。以下の点を考慮します。

- Auto Scaling グループに含めるアベイラビリティーゾーンの数。
- 使用できる既存リソース (セキュリティグループ、Amazon マシンイメージ (AMI) など)。
- キャパシティーをスケーリングするか、または、常時実行中のサーバーを一定数確保します。Amazon EC2 Auto Scaling は両方を同時に実行できます。
- アプリケーションのパフォーマンスと最も関連性が高いメトリクス。
- サーバーの起動とプロビジョニングに要する時間。

アプリケーションの理解が進むにつれて、Auto Scaling アーキテクチャーをより効率的なものにすることができるようになります。

チュートリアル: 最初の Auto Scaling グループを作成する

このチュートリアルでは、AWS Management Console を使用して Amazon EC2 Auto Scaling のハンズオン入門を提供します。EC2 インスタンスと 1 つのインスタンスを含む Auto Scaling グループを定義する起動テンプレートを作成します。Auto Scaling グループを起動した後、インスタンスを終了し、インスタンスがサービスから削除され、置き換えられたことを確認します。一定数のインスタンスを維持するために、Amazon EC2 Auto Scaling は Amazon EC2 のヘルスチェックと到達可能性チェックを自動的に検出して応答します。

AWS にサインアップすると、AWS 無料利用枠を使用して、Amazon EC2 Auto Scaling を無料で開始できます。無料利用枠を使用し、`t2.micro` インスタンスを 12 か月間無料で起動して利用できます (`t2.micro` が利用できないリージョンでは、無料利用枠で `t3.micro` インスタンスを使用できます)。無料利用枠に含まれないインスタンスを起動する場合は、そのインスタンスの通常の Amazon EC2 使用料がかかります。詳細については、「Amazon EC2 料金表」を参照してください。

タスク

- チュートリアルのための準備
- ステップ 1: 起動テンプレートを作成する
- ステップ 2: 単一インスタンスの Auto Scaling グループを作成する
- ステップ 3: Auto Scaling グループを検証する
- ステップ 4: Auto Scaling グループのインスタンスを終了します
- ステップ 5: 次のステップ
- ステップ 6: クリーンアップする

チュートリアルのための準備

このチュートリアルは、EC2 インスタンスの起動について知識があり、キーペアとセキュリティグループを既に作成していることを前提としています。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 でのセットアップ」を参照してください。

Amazon EC2 Auto Scaling の使用を開始するには、AWS アカウント のデフォルトの VPC を使用します。デフォルト VPC には、各アベイラビリティーゾーンのデフォルトのパブリックサブネットと、VPC にアタッチされたインターネットゲートウェイが含まれます。Amazon Virtual Private Cloud (Amazon VPC) コンソールの VPC のページで VPC を表示できます。

ステップ 1: 起動テンプレートを作成する

このステップでは、Amazon EC2 Auto Scaling が自動で作成する EC2 インスタンスのタイプを指定する起動テンプレートを作成します。使用する Amazon Machine Image (AMI) の ID、インスタンスタイプ、キーペア、セキュリティグループなどの情報を含めます。

起動テンプレートを作成するには

1. Amazon EC2 コンソールを開き、[起動テンプレート] ページに移動します。
2. 上部のナビゲーションバーで、[AWS リージョン] を選択します。作成する起動テンプレートと Auto Scaling グループは、指定するリージョンに関連付けられます。
3. [起動テンプレートの作成] を選択します。
4. [起動テンプレート名] を使用する場合、**`my-template-for-auto-scaling`** を入力します。
5. [Auto Scaling ガイダンス] で、チェックボックスを選択します。
6. [Application and OS Images (Amazon Machine Image)] (アプリケーションおよび OS イメージ (Amazon マシンイメージ)) で、[Quick Start] (クイックスタート) リストから Amazon Linux 2 (HVM) のバージョンを選択します。AMI はインスタンスの基本設定テンプレートとして機能します。
7. [インスタンスタイプ] で、指定した AMI と互換性のあるハードウェア設定を選択します。
8. (オプション) [Key pair (login)] (キーペア (ログイン)) で、既存のキーペアを選択します。キーペアは Amazon EC2 インスタンスを SSH に接続するときに使用します。インスタンスへの接続は、このチュートリアルには含まれていません。このため、SSH を使用してインスタンスに接続する予定の場合を除き、キーペアを指定する必要はありません。
9. [Network settings] (ネットワーク設定) で、[Advanced network configuration] (高度なネットワーク設定) を展開し、以下を実行します。

    a. [Add network interface] (ネットワークインターフェイスを追加) を選択して、プライマリネットワークインターフェイスを追加します。

    b. [自動割り当てパブリック IP] で、インスタンスがパブリック IPv4 アドレスを受け取るかどうかを指定します。デフォルトでは、EC2 インスタンスがデフォルトのサブネットで起動された場合、またはパブリック IPv4 アドレスを自動的に割り当てるように設定されたサブネットでインスタンスが起動された場合、Amazon EC2 はパブリック IPv4 アドレスを割り当てます。インスタンスに接続する必要がない場合は、[無効化] を選択します。

    c. [セキュリティグループ ID] で、Auto Scaling グループの VPC として使用する予定の同じ VPC 内のセキュリティグループを選択します。セキュリティグループを指定しないと、インスタンスは VPC のデフォルトのセキュリティグループに自動的に関連付けられます。

d. [終了時に削除] で [はい] を選択すると、インスタンスの削除時にネットワークインターフェイスが削除されます。

10. [起動テンプレートの作成] を選択します。
11. 確認ページで、[Auto Scaling グループの作成] を選択します。

ステップ 2: 単一インスタンスの Auto Scaling グループを作成する

起動テンプレートの作成後に中断したところから続行するには、次の手順に従います。

Auto Scaling グループを作成する

1. [Choose launch template or configuration (起動テンプレートまたは設定の選択)] ページで、Auto Scaling グループの名前を **`my-first-asg`** に入力します。
2. [Next] を選択します。

   [インスタンス起動オプションを選択] ページが表示されます。このページでは、Auto Scaling グループで使用する VPC ネットワーク設定を選択し、オンデマンドインスタンスとスポットインスタンスを起動するためのオプションを選択できます。
3. [Network] (ネットワーク) セクションの [VPC] には、選択した AWS リージョンリージョンのデフォルト設定を維持するか、独自の VPC を選択します。デフォルトの VPC は、インスタンスへのインターネット接続を提供するように自動的に設定されます。この VPC には、リージョンの各アベイラビリティーゾーンのパブリックサブネットが含まれます。
4. [Availability Zones and subnets] (アベイラビリティゾーンとサブネット) で、含める各アベイラビリティーゾーンからサブネットを選択します。複数のアベイラビリティーゾーンのサブネットを使用することで、高可用性を得られます。詳細については、「VPC サブネットを選択する際の考慮事項」を参照してください。
5. [Instance type requirements] (インスタンスタイプの要件) セクションでは、このステップを簡略化するためにデフォルト設定を使用します。(起動テンプレートを上書きしないでください。) このチュートリアルでは、起動テンプレートで指定されたインスタンスタイプを使用して、オンデマンドインスタンスを 1 つだけ起動します。
6. このチュートリアルの残りの部分はデフォルトのままにして、[Skip to review (スキップして確認)] を選択します。

> **Note**
>
> グループの初期サイズは、希望するキャパシティーによって決まります。デフォルト値は 1 インスタンスです。

7. [Review (確認)] ページでグループの情報を確認し、[Auto Scaling グループの作成] を選択します。

ステップ 3: Auto Scaling グループを検証する

Auto Scaling グループを作成し、グループによって EC2 インスタンスが起動されたことを確認する準備が整いました。

> **Tip**
>
> 以下の手順では、Auto Scaling グループについて [Activity history] (アクティビティ履歴) と [Instances] (インスタンス) の各セクションを調べます。どちらのセクションにも、名前付きの列がすでに表示されているはずです。非表示の列を表示する、または表示される行の数を変更するには、各セクションの右上隅にある歯車アイコンを選択して設定モーダルを開き、必要に応じて設定を更新してから、[Confirm] (確認) を選択します。

Auto Scaling グループが EC2 インスタンスを起動したことを確認するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. 作成した Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。使用可能な最初のタブは [詳細] タブで、Auto Scaling グループに関する情報が表示されます。
3. 2 番目のタブ [アクティビティ] を選択します。[アクティビティ履歴] で、Auto Scaling グループに関連付けられているアクティビティの進行状況を表示できます。[ステータス] 列には、インスタンスの現在のステータスが表示されます。インスタンスが起動している間、ステータス列には [`Not yet in service`] と表示されます。ステータスは、インスタンスが起動されると `Successful` に変わります。[Refresh] ボタンを使用して、インスタンスの現在のステータスを表示することもできます。
4. [インスタンス管理] タブの [インスタンス] で、インスタンスのステータスを表示できます。

5. インスタンスが正常に起動したことを確認します。インスタンスの起動には短時間かかります。

   - [ライフサイクル] 列には、インスタンスの状態が表示されます。最初、インスタンスの状態は `Pending` です。インスタンスがトラフィックを受信できるようになったら、そのステータスは `InService` です。
   - [ヘルスステータス] 列には、インスタンスの Amazon EC2 Auto Scaling ヘルスチェックの結果が表示されます。

ステップ 4: Auto Scaling グループのインスタンスを終了します

これらのステップを使用して Amazon EC2 Auto Scaling の機能 (具体的には、必要に応じて新しいインスタンスを起動する方法) を詳しく確認できます。このチュートリアルで作成した Auto Scaling グループの最小サイズは、1 インスタンスです。そのため、実行中のインスタンスを終了する場合、Amazon EC2 Auto Scaling はその代わりとなる新しいインスタンスを起動する必要があります。

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [インスタンス管理] タブの [インスタンス] で、インスタンスの ID を選択します。

   そうすると、Amazon EC2 コンソールの [Instances] (インスタンス) ページが開きます。インスタンスはここで終了できます。
4. [Actions]、[Instance State]、[Terminate] の順に選択します。確認を求めるメッセージが表示されたら、[Yes、Terminate] (はい、終了する) を選択します。
5. ナビゲーションペインの 自動スケーリング で、[Auto Scaling Groups] (Auto Scaling グループ) を選択します。Auto Scaling グループを選択し、[アクティビティ] タブを選択します。

   [インスタンス] ページからインスタンスを終了する場合、インスタンスを終了してから新しいインスタンスが起動するまでに 1 ~ 2 分かかります。アクティビティ履歴で、スケーリングアクティビティが開始すると、最初のインスタンスの削除のエントリおよび新しいインスタンスの起動のエントリが表示されます。新しいエントリが表示されるまで、更新ボタンを使用してください。
6. [インスタンス管理] タブの [インスタンス] セクションには、新しいインスタンスのみが表示されます。

7. ナビゲーションペインの [Instances] (インスタンス) で、[Instances] (インスタンス) を選択します。このページには、終了したインスタンスと実行中の新しいインスタンスの両方が表示されます。

ステップ 5: 次のステップ

作成した基本インフラストラクチャを削除する場合は、次のステップに進みます。それ以外の場合は、ベースとしてこのインフラストラクチャを使用し、次の 1 つ以上を試すことができます。

- Session Manager または SSH を使用した Linux インスタンスへの接続。詳細については、「Amazon EC2 ユーザーガイド」の「Session Manager を使用して Linux インスタンスに接続する」または「SSH を使用して Linux または macOS から Linux インスタンスに接続する」を参照してください。
- Auto Scaling グループの起動インスタンス、または終了インスタンスが変わるたびに通知するよう、Amazon SNS 通知を設定します。詳しくは、「Amazon SNS 通知オプション」を参照してください。
- Auto Scaling グループを手動でスケールして、SNS 通知をテストします。詳細については、「Auto Scaling グループの希望するキャパシティを変更する」を参照してください。

また、ターゲット追跡スケーリングポリシー を読んでおくことで、Auto Scaling の概念に慣れておくこともできます。アプリケーションの負荷が変化した場合、Auto Scaling グループは、グループの希望する容量を最小容量制限と最大容量制限の間で調整することで、自動的にスケールアウト (インスタンスを追加する) およびスケールイン (インスタンスの数を減らして実行する) ことができます。これらの制限の設定に関する詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。

ステップ 6: クリーンアップする

スケーリングインフラストラクチャを削除することも、Auto Scaling グループだけを削除し、後で使用する起動テンプレートを保持することもできます。

AWS無料利用枠外でインスタンスを起動した場合、不要な課金を避けるためにインスタンスを終了する必要があります。インスタンスを終了すると、それに関連付けられたデータも削除されます。

Auto Scaling グループを削除するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。

2. Auto Scaling グループ (`my-first-asg`) の横にあるチェックボックスを選択します。
3. [削除] を選択します。
4. 確認を求められたら、**`delete`** を入力して指定された Auto Scaling グループの削除を確認し、[Delete] (削除) を選択します。

   [Name (名前)] 列のロードアイコンに、Auto Scaling グループが削除されたことが示されます。削除が行われると、[Desired] (必要)、[Min] (最小)、[Max] (最大) 列には、Auto Scaling グループのインスタンス数として `0` と表示されます。インスタンスを終了し、グループを削除するには数分かかります。リストを更新して、現在の状態を確認します。

起動テンプレートを維持する場合は、この手順をスキップします。

起動テンプレートを削除するには

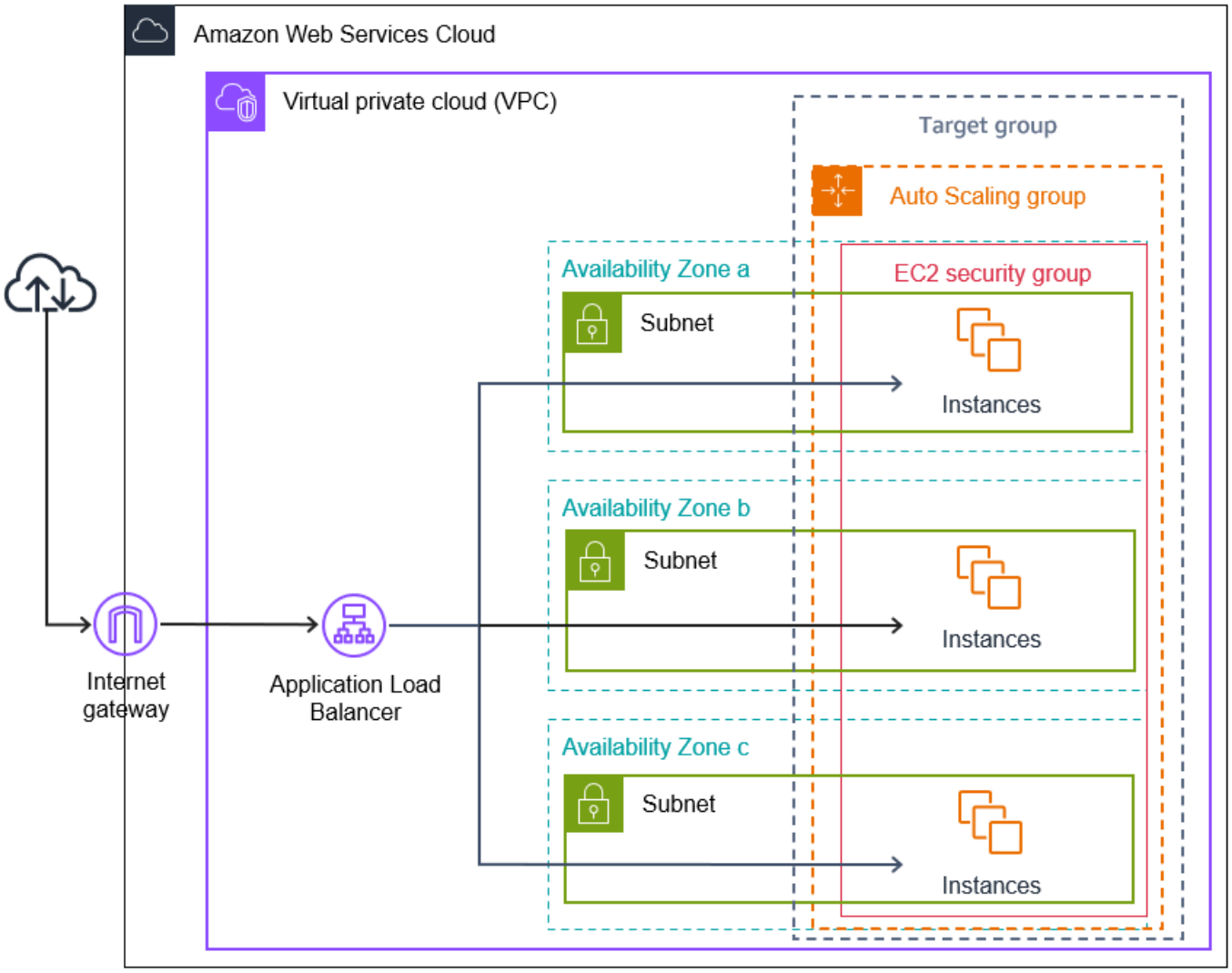
1. Amazon EC2 コンソールの起動テンプレートページを開きます。
2. 起動テンプレートを選択します (`my-template-for-auto-scaling`)。
3. [アクション]、[テンプレートの削除] の順に選択します。
4. 確認を求められたら、**`Delete`** を入力して指定した起動テンプレートの削除を確認し、[Delete] (削除) を選択します。

チュートリアル: スケーリングとロードバランシングを使用するアプリケーションのセットアップ

> **Important**
>
> このチュートリアルに進む前に、「最初の Auto Scaling グループを作成する」チュートリアルで概要を確認することをお勧めします。

Auto Scaling グループを Elastic Load Balancing ロードバランサーに登録すると、負荷分散されたアプリケーションをセットアップできます。Elastic Load Balancing は Amazon EC2 Auto Scaling と連携して、正常な Amazon EC2インスタンス全体に受信トラフィックを分散します。これにより、アプリケーションのスケーラビリティと可用性が向上します。複数のアベイラビリティーゾーン内で Elastic Load Balancing を有効にして、アプリケーションの耐障害性を向上させることができます。

このチュートリアルでは、Auto Scaling グループの作成時に負荷分散されたアプリケーションをセットアップする基本的なステップについて説明します。完了すると、アーキテクチャは次の図表のようになります。



Elastic Load Balancing では、異なる種類のロードバランサーがサポートされています。このチュートリアルでは、Application Load Balancer を使用することをお勧めします。

アーキテクチャへのロードバランサーの導入の詳細については、「Elastic Load Balancing を使用して Auto Scaling グループ内の受信アプリケーショントラフィックを分散する」を参照してください。

タスク

- 前提条件
- ステップ 1: 起動テンプレートまたは起動設定を設定する

前提条件

- ロードバランサーおよびターゲットグループ。Auto Scaling グループに使用するロードバランサーと同じアベイラビリティゾーンを選択していることを確認します。詳細については、Elastic Load Balancing ユーザーガイドの Elastic Load Balancing で使用開始 を参照してください。
- 起動テンプレートまたは起動設定に対応したセキュリティグループ。セキュリティグループは、リスナーポート (通常はHTTPトラフィックのポート 80) と Elastic Load Balancing がヘルスチェックに使用するポートの両方でロードバランサーからのアクセスを許可する必要があります。詳細については、該当するドキュメントを参照してください。
  - Application Load Balancer のユーザーガイド の「ターゲットセキュリティグループ」
  - Network Load Balancer のユーザーガイドの「ターゲットセキュリティグループ」

  オプションで、インスタンスにパブリック IP アドレスがある場合、インスタンスへの接続にSSHトラフィックを許可できます。
- (オプション) アプリケーションにアクセス権を付与する IAMロール AWS。
- (オプション) Amazon EC2インスタンスのソーステンプレートとして定義された Amazon マシンイメージ (AMI) 。作成するには、インスタンスを起動します。IAM ロール (ロールを作成した場合) と、ユーザーデータとして必要な設定スクリプトを指定します。インスタンスに接続し、それをカスタマイズします。例えば、ソフトウェアとアプリケーションのインストール、データのコピー、追加のEBSボリュームのアタッチを行うことができます。インスタンスが正しく設定されたことを確認するために、インスタンスでアプリケーションをテストします。この更新された設定をカスタム として保存しますAMI。後で使用しないインスタンスは、終了できます。この新しいカスタム から起動されるインスタンスAMIには、 の作成時に行ったカスタマイズが含まれますAMI。
- 仮想プライベートクラウド (VPC) 。このチュートリアルでは、デフォルトの を参照しますがVPC、独自の を使用できます。独自の を使用する場合はVPC、作業しているリージョンの各アベイラビリティーゾーンにサブネットがマッピングされていることを確認します。ロードバランサーを作成するには、2 つ以上のパブリックサブネットが必要です。Auto Scaling グループを作成して

ロードバランサーに登録するには、2 つのプライベートサブネットまたは 2 つのパブリックサブネットが必要です。

ステップ 1: 起動テンプレートまたは起動設定を設定する

このチュートリアルでは、起動テンプレートまたは起動設定を使用します。

トピック

起動テンプレートを選択または作成する

使用する起動テンプレートが既に存在している場合は、以下の手順を使用して起動テンプレートを選択します。

既存の起動テンプレートを選択するには

1. Amazon EC2コンソールの起動テンプレートページを開きます。
2. 画面上部のナビゲーションバーで、ロードバランサーが作成されたリージョンを選択します。
3. 起動テンプレートを削除します。
4. Actions、[Auto Scaling グループの作成] を選択します。

または、新しい起動テンプレートを作成するために、次の手順を使用します。

起動テンプレートを作成するには

1. Amazon EC2コンソールの起動テンプレートページを開きます。
2. 画面上部のナビゲーションバーで、ロードバランサーが作成されたリージョンを選択します。
3. [起動テンプレートの作成] を選択します。
4. 名前を入力し、起動テンプレートの最初のバージョンの説明を加えます。
5. アプリケーションイメージと OS イメージ (Amazon マシンイメージ) AMI で、インスタンスの の ID を選択します。利用可能なすべての を検索するかAMIs、履歴またはクイックスタートリストから AMI を選択できます。AMI 必要な が表示されない場合は、詳細を参照AMIsを選択してAMIカタログ全体を参照します。

6. インスタンスタイプで、AMI指定した と互換性のあるインスタンスのハードウェア設定を選択します。
7. (オプション) [キーペア (ログイン)] で、インスタンスに接続するときに使用するキーペアを選択します。
8. [Network settings] (ネットワーク設定) で、[Advanced network configuration] (高度なネットワーク設定) を展開し、以下を実行します。

   a. [Add network interface] (ネットワークインターフェイスを追加) を選択して、プライマリネットワークインターフェイスを追加します。

   b. パブリック IP を自動割り当てするには、インスタンスがパブリックIPv4アドレスを受け取るかどうかを指定します。デフォルトでは、EC2インスタンスがデフォルトのサブネットで起動された場合、またはパブリックIPv4アドレスを自動的にEC2割り当てるように設定されたサブネットで起動された場合、Amazon はパブリックIPv4アドレスを割り当てます。インスタンスに接続する必要がない場合は、グループ内のインスタンスがインターネットから直接トラフィックを受信することのないよう、[無効化] を選択することもできます。この場合、トラフィックの受信はロードバランサーからのみになります。

   c. セキュリティグループ ID には、ロードバランサーVPCと同じ からインスタンスのセキュリティグループを指定します。

   d. [Delete on termination (終了時に削除)] を使用する場合、Yes (はい)を選択してください。これにより、Auto Scaling グループをスケールインしてネットワークインターフェイスがアタッチされているインスタンスを終了するときに、ネットワークインターフェイスが削除されます。

9. (オプション) インスタンスに認証情報を安全に配布するには、詳細、インスタンスIAMプロファイルに、IAMロールの Amazon リソースネーム (ARN) を入力します。
10. (オプション) インスタンスにユーザーデータまたは設定スクリプトを指定するには、[Advanced details]、[User data] に設定スクリプトを貼り付けます。
11. [起動テンプレートの作成] を選択します。
12. 確認ページで、[Auto Scaling グループの作成] を選択します。

起動設定を作成または選択する

> **Note**
>
> 投資計画がないレガシー機能であるため、新しいアプリケーションで起動設定を使用することはお勧めしません。また、2023 年 6 月 1 日以降に作成された新しいアカウントには、

コンソールから新しい起動設定を作成することはできません。詳細については、「Auto Scaling 起動設定」を参照してください。

既存の起動設定を選択するには

1. Amazon EC2コンソールの起動設定ページを開きます。
2. 上部のナビゲーションバーで、ロードバランサーが作成されたリージョンを選択します。
3. 起動設定を選択します。
4. Actions、[Auto Scaling グループの作成] を選択します。

または、新しい起動設定を作成するために、次の手順を使用します。

起動設定を作成するには

1. Amazon EC2コンソールの起動設定ページを開きます。確認を求めるプロンプトが表示されたら、[起動設定を表示] を選択して、[起動設定] ページを表示することを確認します。
2. 上部のナビゲーションバーで、ロードバランサーが作成されたリージョンを選択します。
3. [Create launch configuration (起動設定の作成)] を選択して、起動設定の名前を入力します。
4. Amazon マシンイメージ (AMI) の場合は、検索条件としてインスタンスAMIの の ID を入力します。
5. [インスタンスタイプ] では、インスタンスのハードウェア設定を選択します。
6. [Additional configuration (追加設定)] 以下のフィールドに注意してください。

   a. (オプション) EC2インスタンスIAMプロファイルなど、インスタンスに認証情報を安全に配布するには、IAMロールを選択します。詳細については、「Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール」を参照してください。

   b. (オプション) インスタンスにユーザーデータまたは設定スクリプトを指定するには、[Advanced details (高度な詳細)]、[User data] に設定スクリプトを貼り付けます。

   c. (オプション) [Advanced details (高度な詳細)] の [IP アドレスタイプ] は、デフォルト値のままにします。Auto Scaling グループを作成するときに、デフォルトの のデフォルトサブネットなど、パブリック IP アドレス属性が有効になっているサブネットを使用して、Auto Scaling グループのインスタンスにパブリック IP アドレスを割り当てることができます VPC。または、インスタンスに接続する必要がない場合は、グループ内のインスタンスがインターネットから直接トラフィックを受信することのないよう、[パブリック IP アドレス

をどのインスタンスにも割り当てないでください] を選択することもできます。この場合、トラフィックの受信はロードバランサーからのみになります。

7. セキュリティグループで、ロードバランサーVPCと同じ から既存のセキュリティグループを選択します。新しいセキュリティグループの作成オプションを選択したままにすると、Linux を実行している Amazon EC2インスタンスに対してデフォルトのSSHルールが設定されます。Windows を実行する Amazon EC2インスタンスには、デフォルトのRDPルールが設定されます。
8. [Key pair (login) (キーペア (ログイン)] で、[Key pair options (キーペアのオプション)] の下にあるオプションを選択します。

   Amazon EC2インスタンスのキーペアをすでに設定している場合は、ここで選択できます。

   Amazon EC2インスタンスのキーペアがまだない場合は、新しいキーペアを作成して認識可能な名前を付けます。[Download key pair (キーペアのダウンロード)] を選択し、コンピュータにダウンロードします。

   > **Important**
   >
   > インスタンスに接続する必要がある場合は、[Proceed without a key pair (キーペアなしで続行する)] を選択しないでください。

9. 確認チェックボックスをオンにし、[Create launch configuration (起動設定の作成)] を選択します。
10. 新しい起動設定の名前の横にあるチェックボックスを選択し、アクション,Auto Scaling グループの作成を選択してください。

ステップ 2: Auto Scaling グループを作成する

起動テンプレートまたは起動設定を作成または選択した後、以下の手順に従って中断していた作業を続行します。

Auto Scaling グループを作成する

1. [Choose launch template or configuration (起動テンプレートまたは起動設定を選択する)] ページで [Auto Scaling グループ名] にAuto Scaling グループの名前を入力します。

2. [Launch template only] [起動テンプレート] で、スケールアウト時に Auto Scaling グループで使用する起動テンプレートのバージョン (デフォルト、最新、または特定のバージョン) を選択します。
3. [Next (次へ)] を選択します。

   インスタンス起動オプションの選択ページが表示され、Auto Scaling グループで使用するVPC ネットワーク設定を選択したり、オンデマンドインスタンスとスポットインスタンスを起動するオプション (起動テンプレートを選択した場合) を選択したりできます。
4. Network セクションの でVPC、ロードバランサーVPCに使用した を選択します。デフォルトのを選択した場合VPC、インスタンスにインターネット接続を提供するように自動的に設定されます。これには、リージョンの各アベイラビリティーゾーンのパブリックサブネットVPCが含まれます。
5. [Availability Zones and subnets] (アベイラビリティゾーンとサブネット) で、ロードバランサーがあるアベイラビリティーゾーンに基づいて、含める各アベイラビリティーゾーンから 1 つ以上のサブネットを選択します。詳細については、「VPC サブネットを選択する際の考慮事項」を参照してください。
6. [起動テンプレートのみ] [Instance type requirements] (インスタンスタイプの要件) セクションで、デフォルト設定を使用して、この手順を簡略化します。(起動テンプレートを上書きしないでください。) このチュートリアルでは、起動テンプレートで指定されたインスタンスタイプを使用して、オンデマンドインスタンスのみを起動します。
7. [Next] (次へ) を選択して、[Configure advanced options] (詳細オプションの設定) ページに移動します。
8. グループを既存のロードバランサーにアタッチするには、[Load balancing] (ロードバランシング) セクションで [Attach to an existing load balancer] (既存のロードバランサーにアタッチする) を選択します。[Choose from your load balancer target groups] (ロードバランサーのターゲットグループから選択する) または [Choose from Classic Load Balancers] (Classic Load Balancer から選択する) も選択できます。その後、作成した Application Load Balancer または Network Load Balancer のターゲットグループの名前を選択するか、Classic Load Balancer の名前を選択できます。
9. (オプション) [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Elastic Load Balancing のヘルスチェックをオンにする] を選択します。
10. (オプション) [ヘルスチェックの猶予期間] に秒単位で時間を入力します。この時間は、Amazon EC2 Auto Scaling がインスタンスが `InService` 状態になった後にインスタンスのヘルスステータスをチェックするまでに待機する必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

11. Auto Scaling グループの設定が完了したら、[Skip to review (スキップして確認)] を選択します。
12. [確認] ページで、Auto Scaling グループの詳細を確認します。[Edit] を選択して、変更を加えることができます。完了したら、[Auto Scaling グループの作成] を選択します。

ロードバランサーがアタッチされた Auto Scaling グループを作成すると、ロードバランサーは新しいインスタンスがオンラインになると自動的に登録します。この時点ではインスタンスが 1 つしかないため、登録するインスタンスは多くありません。ただし、グループの希望キャパシティーを更新することで、インスタンスを追加できるようになりました。手順については step-by-step、「」を参照してくださいAuto Scaling グループの希望するキャパシティを変更する。

ステップ 3: ロードバランサーがアタッチされたことを確認する

ロードバランサーがアタッチされたことを確認するには

1. Amazon EC2コンソールの Auto Scaling グループページで、Auto Scaling グループの横にあるチェックボックスをオンにします。
2. [詳細] タブの [Load balancing] には、アタッチされているロードバランサーターゲットグループまたは Classic Load Balancer が表示されます。
3. [アクティビティ]タブのアクティビティ履歴で、インスタンスが正常に起動したことを確認できます。[Status] 列は Auto Scaling グループがインスタンスを正常に起動したかどうかを表示します。インスタンスの起動に失敗した場合は、Amazon EC2 Auto Scaling の問題のトラブルシューティング にインスタンスの起動に関する一般的な問題のトラブルシューティングのヒントがあります。
4. [インスタンス管理] タブの [インスタンス] で、インスタンスがトラフィックを受け取る準備ができたことを確認できます。当初、インスタンスの状態は `Pending` です。インスタンスがトラフィックを受信できるようになったら、そのステータスは `InService` です。ヘルスステータス列には、インスタンスの Amazon EC2 Auto Scaling ヘルスチェックの結果が表示されます。インスタンスが正常とマークされていても、ロードバランサーは、ロードバランサーのヘルスチェックに合格したインスタンスにのみ、トラフィックを送信します。
5. インスタンスがロードバランサーに登録されていることを確認します。Amazon EC2コンソールのターゲットグループページを開きます。ターゲットグループを選択し、[ターゲット] タブを選択します。インスタンスの状態が `initial` の場合、おそらく登録中であるか、まだヘルスチェック中です。インスタンスの状態が `healthy` になると、使用できる状態です。

ステップ 4: 次のステップ

このチュートリアルを完了したので、さらに詳しい内容に進むことができます。

- Amazon EC2 Auto Scaling は、Auto Scaling グループが使用するヘルスチェックのステータスに基づいて、インスタンスが正常かどうかを決定します。ロードバランサーのヘルスチェックを有効にして、インスタンスがヘルスチェックに失敗した場合、Auto Scaling グループではインスタンスが異常と見なされ、これが置き換えられます。詳細については、「ヘルスチェック」を参照してください。
- 同じリージョン内の追加のアベイラビリティーゾーンにアプリケーションを拡張して、サービス中断が発生した場合の耐障害性を向上させることができます。詳細については、「アベイラビリティーゾーンを追加する」を参照してください。
- Auto Scaling グループを設定することで、ターゲット追跡スケーリングポリシーを使用できます。これにより、インスタンスの需要の変化に応じて、インスタンスの数が自動的に増減します。これにより、グループはアプリケーションが受信するトラフィック量の変化に対応できます。詳細については、「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。

ステップ 5 : クリーンアップ

このチュートリアル用に作成したリソースを使用し終えたら、不要な料金の発生を回避するため、クリーンアップを検討してください。

Auto Scaling グループを削除するには

1. Amazon EC2コンソールの Auto Scaling グループページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [削除] を選択します。
4. 確認を求められたら、**`delete`** を入力して指定された Auto Scaling グループの削除を確認し、[Delete] (削除) を選択します。

   [Name (名前)] 列のロードアイコンに、Auto Scaling グループが削除されたことが示されます。削除が行われると、[Desired] (必要)、[Min] (最小)、[Max] (最大) 列には、Auto Scaling グループのインスタンス数として `0` と表示されます。インスタンスを終了し、グループを削除するには数分かかります。リストを更新して、現在の状態を確認します。

起動テンプレートを維持する場合は、この手順をスキップします。

起動テンプレートを削除するには

1. Amazon EC2コンソールの起動テンプレートページを開きます。
2. 起動テンプレートを選択します。
3. [アクション]、[テンプレートの削除] の順に選択します。
4. 確認を求められたら、**Delete** を入力して指定した起動テンプレートの削除を確認し、[Delete] (削除) を選択します。

起動設定を維持する場合は、以下の手順をスキップします。

起動設定を削除するには

1. Amazon EC2コンソールの起動設定ページを開きます。
2. 起動設定を選択します。
3. [Actions]、[Delete launch configuration] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

将来使用できるように、ロードバランサーを保持する場合は、次の手順をスキップします。

ロードバランサーを削除するには

1. Amazon EC2コンソールのロードバランサーページを開きます。
2. ロードバランサーを選択してから、[Actions (アクション)]、[Delete (削除)] の順に選択します。
3. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

ターゲットグループを削除するには

1. Amazon EC2コンソールのターゲットグループページを開きます。
2. ターゲットグループを選択し、[Actions (アクション)]、[Delete (削除)] を選択します。
3. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

関連リソース

を使用すると AWS CloudFormation、テンプレートファイルを使用してリソースのコレクションを 1 つのユニット (スタック) としてまとめて作成および削除することで、 AWS インフラストラクチャ

のデプロイを予測どおりに繰り返し作成およびプロビジョニングできます。詳細については、AWS CloudFormation ユーザーガイドをご参照ください。

スタックテンプレートを使用して、Auto Scaling グループと Application Load Balancer をプロビジョニングする方法のチュートリアルは、「AWS CloudFormation ユーザーガイド」の「チュートリアル: スケーラブルなロードバランシングウェブサーバーの作成」を参照してください。このチュートリアルとサンプルテンプレートは、実際のニーズに合わせて、同様のテンプレートを作成する際の開始点として使用できます。

Auto Scaling 起動テンプレート

起動テンプレートは、インスタンス設定情報を指定する起動設定と似ています。これには、Amazon マシンイメージ (AMI) の ID、インスタンスタイプ、キーペア、セキュリティグループ、およびEC2 インスタンスの起動に使用されるその他のパラメータが含まれます。ただし、起動設定の代わりに起動テンプレートを定義すると、複数のバージョンの起動テンプレートを使用することができます。

起動テンプレートのバージョン管理では、パラメータのフルセットのサブセットを作成できます。その後、再使用して、同じ起動テンプレートの他のバージョンを作成できます。たとえば、 AMIまたは ユーザーデータスクリプトを使用せずに、基本設定を定義する起動テンプレートを作成できます。起動テンプレートを作成したら、新しいバージョンを作成し、テスト用のアプリケーションの最新バージョンを持つ AMIおよび ユーザーデータを追加できます。これにより、起動テンプレートのバージョンが 2 つになります。基本構成を保存すると、必要な一般構成パラメータを維持するのに役立ちます。基本設定から起動テンプレートの新しいバージョンを、必要に応じて作成することができます。アプリケーションのテストに使用されたバージョンも、不要になったら削除することができます。

最新の機能や改善点にアクセスできるように、起動テンプレートを使用することをお勧めします。起動設定を使用する場合、すべての Amazon EC2 Auto Scaling 機能が利用できるわけではありません。たとえば、スポットインスタンスとオンデマンドインスタンスの両方を起動する Auto Scaling グループや、複数のインスタンスタイプを指定する Auto Scaling グループを作成することはできません。これらの機能を設定するには、起動テンプレートを使用する必要があります。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。

起動テンプレートを使用すると、Amazon の新しい機能を使用することもできますEC2。これには、Systems Manager パラメータ (AMI ID)、現行世代のEBSプロビジョンドIOPSボリューム (io2)、EBSボリュームのタグ付け、T2 Unlimited インスタンス、キャパシティ予約、Capacity Blocks、、Dedicated Hosts などです。

起動テンプレートを作成するときは、すべてのパラメータはオプションです。ただし、起動テンプレートで が指定されていない場合AMI、Auto Scaling グループの作成AMI時に を追加することはできません。を指定してもインスタンスタイプAMIを指定しない場合は、Auto Scaling グループの作成時に 1 つ以上のインスタンスタイプを追加できます。

内容

- Auto Scaling グループの起動テンプレートを作成する
- 詳細設定を使用して起動テンプレートを作成する
- Auto Scaling グループを起動テンプレートに移行する
- AWS CloudFormation スタックを起動テンプレートに移行する
- を使用した起動テンプレートの作成と管理の例 AWS CLI
- 起動テンプレートAMIIDsで の代わりに AWS Systems Manager パラメータを使用する

起動テンプレートを操作するためのアクセス許可

このセクションの手順は、起動テンプレートを使用するために必要なアクセス許可が既に付与されていることを前提としています。管理者がアクセス許可を付与する方法の詳細については、「Amazon EC2ユーザーガイド」の「アクセスIAM許可を使用して起動テンプレートへのアクセスを制御する」を参照してください。

起動テンプレートで指定したリソースを使用および作成するための十分な許可がない場合、Auto Scaling グループに起動テンプレートを指定しようとすると、起動テンプレートを使用する権限がないというエラーが表示されます。詳細については、「Amazon EC2 Auto Scaling のトラブルシューティング: テンプレートの起動」を参照してください。

起動テンプレートを使用して `CreateAutoScalingGroup`、、`UpdateAutoScalingGroup`および`RunInstances`APIオペレーションを呼び出すことができるIAMポリシーの例については、「」を参照してくださいAuto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する。

API 起動テンプレートでサポートされる オペレーション

起動テンプレートでサポートされているAPIオペレーションのリストについては、「Amazon リファレンス」の「Amazon EC2アクション」を参照してください。 EC2 API

Auto Scaling グループの起動テンプレートを作成する

起動テンプレートを使用して Auto Scaling グループを作成する前に、Amazon マシンイメージ () の ID など、インスタンスを起動するための設定情報を含む起動テンプレートを作成する必要がありますAMI。

新しい起動テンプレートを作成するには、次の手順を使用します。

内容

- 起動テンプレートを作成する (コンソール)
- デフォルトのネットワークインターフェイス設定を変更する (コンソール)
- ストレージ設定の変更 (コンソール)
- 既存のインスタンスから起動テンプレートを作成する (コンソール)
- 関連リソース
- 制限

> **Important**
>
> 起動テンプレートパラメータは、起動テンプレート作成の際には完全には検証されません。パラメータに誤った値を指定した場合、またはサポートされているパラメータの組み合わせを使用しない場合、この起動テンプレートを使用してインスタンスは起動できません。パラメータに正しい値を指定したこと、およびサポートされているパラメータの組み合わせを使用していることを確認します。たとえば、Arm ベースの AWS Graviton または Graviton2 でインスタンスを起動するにはAMI、Arm 互換インスタンスタイプを指定する必要があります。詳細については、「Amazon EC2ユーザーガイド」の「起動テンプレートの制限」を参照してください。

起動テンプレートを作成する (コンソール)

以下の手順では、基本的な起動テンプレートを設定する方法を説明します。

- インスタンスを起動する Amazon マシンイメージ (AMI) を指定します。
- AMI 指定した と互換性のあるインスタンスタイプを選択します。
- インスタンスに接続するときに使用するキーペアを指定します。たとえば、 を使用しますSSH。
- 1 つ以上のセキュリティグループを追加して、インスタンスへのネットワークアクセスを許可します。
- 各インスタンスに追加のボリュームをアタッチするかどうかを指定します。
- インスタンスおよびボリュームにカスタムタグ (キーバリューペア) を追加する。

起動テンプレートを作成するには

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。

2. ナビゲーションペインで、[インスタンス] の [テンプレートの起動] を選択します。
3. [起動テンプレートの作成] を選択します。名前を入力し、起動テンプレートの最初のバージョンの説明を加えます。
4. （オプション) Auto Scaling ガイダンスでチェックボックスをオンにすると、Amazon EC2 Auto Scaling で使用するテンプレートの作成に役立つガイダンスが Amazon EC2に提供されます。
5. [Launch template contents] (起動テンプレートのコンテンツ) で各必須フィールドに入力し、必要に応じてオプションフィールドにも入力します。

   a. アプリケーションおよび OS イメージ (Amazon マシンイメージ) : (必須) インスタンスAMI の の ID を選択します。利用可能なすべての を検索するかAMIs、履歴またはクイックスタートリストから AMI を選択できます。AMI 必要な が表示されない場合は、詳細を参照 AMIsを選択してAMIカタログ全体を参照します。

      カスタム を選択するにはAMI、まずカスタマイズされたインスタンスAMIから を作成する必要があります。詳細については、「Amazon ユーザーガイド」の「Amazon EBS-backed の作成AMI」を参照してください。 EC2

   b. インスタンスタイプで、AMI指定した と互換性のある 1 つのインスタンスタイプを選択します。

      あるいは、属性ベースのインスタンスタイプの選択を使用するには、[詳細]、[インスタンスタイプの属性を指定] を選択し、次のオプションを指定します。

      - の数vCPUs: の最小数と最大数を入力しますvCPUs。制限を設定しない場合は、最小値に 0 を入力し、最大値を空白のままにします。
      - [Amount of memory (MiB)] (メモリの量 (MiB)): メモリの最小量と最大量を MiB 単位で入力します。制限を設定しない場合は、最小値に 0 を入力し、最大値を空白のままにします。
      - [Optional instance type attributes] (オプションのインスタンスタイプ属性) を展開して [Add attribute] (属性の追加) を選択し、目的のキャパシティーを満たすために使用できるインスタンスのタイプをさらに絞り込みます。各属性の詳細については、「Amazon EC2APIリファレンスInstanceRequirementsRequest」の「」を参照してください。
      - 結果のインスタンスタイプ: vCPUs、メモリ、ストレージなど、指定されたコンピューティング要件に一致するインスタンスタイプを表示できます。
      - インスタンスタイプを除外するには、[属性の追加] を選択します。[属性] リストから [除外されたインスタンスタイプ] を選択します。[Attribute value] (属性値) リストから、除外したいインスタンスタイプを選択します。

c. [キーペア (ログイン)]: [キーペア名] で既存のキーペアを選択するか、[新しいキーペアの作成] を選択して新しいキーペアを作成します。詳細については、「Amazon ユーザーガイド」の「Amazon EC2キーペアと Linux インスタンス」を参照してください。 EC2

d. [ネットワーク設定]: [ファイアウォール (セキュリティグループ)] で 1 つ以上のセキュリティグループを使用するか、空白のままにして、ネットワークインターフェイスの一部として 1 つ以上のセキュリティグループを設定します。詳細については、「Amazon ユーザーガイド」の「Linux インスタンス用の Amazon EC2 セキュリティグループ」を参照してください。 EC2

   起動テンプレートでセキュリティグループを指定しない場合、Amazon は Auto Scaling グループがインスタンスを起動VPCする のデフォルトのセキュリティグループEC2を使用します。デフォルトでは、このセキュリティグループは外部ネットワークからのインバウンドトラフィックを許可しません。詳細については、「Amazon VPCユーザーガイド」の「 のデフォルトのセキュリティグループVPCs」を参照してください。

e. 次のいずれかを行います。

   - デフォルトのネットワークインターフェイス設定を変更します。例えば、サブネットの自動割り当てパブリックIPv4アドレス設定を上書きするパブリックIPv4アドレス指定機能を有効または無効にできます。詳細については、「デフォルトのネットワークインターフェイス設定を変更する (コンソール)」を参照してください。
   - デフォルトのネットワークインターフェイス設定を維持するには、このステップをスキップします。

f. 次のいずれかを行います。

   - ストレージ設定を変更します。詳細については、「ストレージ設定の変更 (コンソール)」を参照してください。
   - デフォルトのストレージ設定を保持するには、このステップをスキップします。

g. [Resource tags] (リソースタグ) には、キーと値の組み合わせを入力してタグを指定します。起動テンプレートでインスタンスタグを指定して、Auto Scaling グループのタグをそのインスタンスに伝播することを選択した場合、すべてのタグがマージされます。起動テンプレートのタグと Auto Scaling グループのタグに同じタグキーが指定されている場合、グループのタグ値が優先されます。

6. 詳細設定を構成します (オプション)。たとえば、アプリケーションが他の AWS リソースにアクセスするときに使用できる IAMロールを選択するか、インスタンスの起動後に一般的な自動設定タスクを実行するために使用できるインスタンスユーザーデータを指定できます。詳細については、「詳細設定を使用して起動テンプレートを作成する」を参照してください。

7. 起動テンプレートを作成する準備ができたら、[起動テンプレートを作成] を選択します。
8. Auto Scaling グループを作成するには、[confirmation (確認)] ページで[Auto Scaling グループの作成] を選択します。

デフォルトのネットワークインターフェイス設定を変更する (コンソール)

ネットワークインターフェイスは、 VPCおよびインターネット内の他のリソースへの接続を提供します。詳細については、「Amazon VPC を使用して Auto Scaling インスタンスのネットワーク接続を提供する」を参照してください。

このセクションでは、デフォルトのネットワークインターフェイス設定を変更する方法について説明します。例えば、サブネットの自動割り当てパブリックIPv4アドレス設定をデフォルトにするのではなく、各インスタンスにパブリックIPv4アドレスを割り当てるかどうかを定義できます。

考慮事項と制限事項

デフォルトのネットワークインターフェイス設定を変更する場合、次の考慮事項と制約事項に留意してください。

- セキュリティグループは、テンプレートの [セキュリティグループ] セクションではなく、ネットワークインターフェイスの一部として設定する必要があります。両方の場所でセキュリティグループを指定することはできません。
- 既存のネットワークインターフェイス ID を指定した場合、起動できるインスタンスは 1 つだけです。これを行うには、 AWS CLI または SDK を使用して Auto Scaling グループを作成する必要があります。グループを作成するときは、アベイラビリティーゾーンを指定する必要がありますが、サブネット ID は指定しないでください。また、デバイスインデックスが 0 の場合にのみ、既存のネットワークインターフェイスを指定できます。
- 複数のネットワークインターフェイスを指定した場合、パブリックIPv4アドレスを自動割り当てすることはできません。また、ネットワークインターフェイス間で重複するデバイスインデックスを指定することもできません。プライマリとセカンダリの両方のネットワークインターフェイスは同じサブネットに存在します。
- インスタンスが起動すると、各ネットワークインターフェイスにプライベートアドレスが自動的に割り当てられます。アドレスは、インスタンスが起動されるサブネットCIDRの範囲から取得されます。VPC または サブネットのCIDRブロック (または IP アドレス範囲) を指定する方法については、「Amazon VPC ユーザーガイド」を参照してください。

デフォルトのネットワークインターフェイス設定を変更するには

1. [ネットワーク設定] で [高度なネットワーク設定] を展開します。
2. [ネットワークインターフェイスを追加] を選択して、プライマリネットワークインターフェイスを設定します。以下のフィールドに注意してください。

   a. デバイスインデックス: プライマリネットワークインターフェイス (eth0) に変更を適用するには、デフォルト値の 0 のままにします。

   b. ネットワークインターフェイス: インスタンスの起動時に Amazon EC2 Auto Scaling が新しいネットワークインターフェイスを自動的に作成するように、デフォルト値の新しいインターフェイスを保持します。デバイスインデックスが 0 の既存の利用可能なネットワークインターフェイスを選択できます。その場合、Auto Scaling グループは 1 つのインスタンスに制限されます。

   c. 説明: 分かりやすい名前を入力します (オプション)。

   d. サブネット: デフォルトの [[起動テンプレートの設定に含めない] のままにします。

      がネットワークインターフェイスのサブネットAMIを指定すると、エラーが発生します。回避策として、[Auto Scaling ガイダンス] をオフにすることが推奨されます。この変更を行った後は、エラーメッセージは表示されません。ただし、サブネットが指定されている場所に関係なく、Auto Scaling グループのサブネット設定が優先され、上書きすることはできません。

   e. パブリック IP の自動割り当て: デバイスインデックスが 0 のネットワークインターフェイスがパブリックIPv4アドレスを受け取るかどうかを変更します。デフォルトでは、デフォルトサブネットのインスタンスはパブリックIPv4アドレスを受け取りますが、デフォルト以外のサブネットのインスタンスはパブリックアドレスを受け取りません。[Enable] または [Disable] を選択すると、これによりサブネットのデフォルト設定がオーバーライドされます。

   f. セキュリティグループ: ネットワークインターフェイスに対して 1 つ以上のセキュリティグループを選択します。各セキュリティグループは、Auto Scaling グループがインスタンスを起動VPCする 用に設定する必要があります。詳細については、「Amazon ユーザーガイド」の「Linux インスタンス用の Amazon EC2 セキュリティグループ」を参照してください。 EC2

   g. 終了時に削除: インスタンスが終了したときにネットワークインスタンスを削除するには、[はい] を選択します。ネットワークインスタンスを維持するには、[いいえ] を選択します。

   h. Elastic Fabric Adapter: ハイパフォーマンスコンピューティングと機械学習のユースケースをサポートするには、ネットワークインターフェイスを Elastic Fabric Adapter ネット

ワークインターフェイスに変更します。詳細については、「Amazon EC2ユーザーガイド」の「Elastic Fabric Adapter」を参照してください。

i. ネットワークカードのインデックス: デバイスインデックスが 0 のネットワークカードにプライマリネットワークインターフェイスをアタッチするには [0] を選択します。このオプションが使用できない場合は、デフォルト値 ([起動テンプレートに含めない]) のままにします。特定のネットワークカードへのネットワークインターフェイスのアタッチは、サポートされているインスタンスタイプでのみ使用できます。詳細については、「Amazon EC2ユーザーガイド」の「ネットワークカード」を参照してください。

j. ENA Express: ENA Express をサポートするインスタンスタイプでは、ENAExpress を有効にするには Enable、無効にするにはDisableを選択します。詳細については、「Amazon EC2 ユーザーガイド」の「Linux インスタンスでの ENA Express によるネットワークパフォーマンスの向上」を参照してください。

k. ENA Express UDP: ENA Express を有効にすると、オプションでUDPトラフィックに使用できます。有効を選択して ENA Express を有効にするUDPか、無効を選択して無効にします。

3. セカンダリネットワークインターフェイスを追加するには、ネットワークインターフェイスを追加するを選択します。

ストレージ設定の変更 (コンソール)

Amazon EBS-backed AMIまたは instance store-backed から起動されたインスタンスのストレージ設定を変更できますAMI。インスタンスにアタッチする追加のEBSボリュームを指定することもできます。AMI には、ルートボリューム (ボリューム 1 (ルート)) を含む 1AMI つ以上のストレージボリュームが含まれます。

ストレージ設定を変更するには

1. [Configure storage] (ストレージの設定) でボリュームのサイズまたはタイプを変更します。

   ボリュームのサイズに指定した値がボリュームのタイプの制限外である場合、またはスナップショットのサイズより小さい場合は、エラーメッセージが表示されます。この問題に対処するために、このメッセージには、フィールドに入力できる最小値または最大値が示されます。

   Amazon EBS-backed に関連付けられたボリュームのみAMIが表示されます。instance store-backed から起動されたインスタンスのストレージ設定に関する情報を表示するにはAMI、インスタンスストアボリュームセクションから詳細を表示を選択します。

すべてのEBSボリュームパラメータを指定するには、右上隅の詳細ビューに切り替えます。

2. アドバンストオプションで、変更するボリュームを展開し、次のようにボリュームを構成します。

   a. ストレージタイプ: インスタンスに関連付けるボリューム (EBS またはエフェメラル) のタイプ。インスタンスストア (エフェメラル) ボリュームタイプは、それをサポートするインスタンスタイプを選択した場合にのみ使用できます。詳細については、「Amazon ユーザーガイド」の「Amazon EBSボリューム」および「Amazon EC2 ユーザーガイド」の「Amazon EC2インスタンスストア」を参照してください。 EBS

   b. [Device name] (デバイス名): ボリュームで利用できるデバイス名の一覧から選択します。

   c. [Snapshot] (スナップショット): ボリュームの作成元となるスナップショットを選択します。[Snapshot] (スナップショット) フィールドにテキストを入力して、利用できる共有スナップショットとパブリックスナップショットを検索することもできます。

   d. サイズ (GiB): EBSボリュームの場合は、ストレージサイズを指定できます。無料利用枠の対象となる AMIおよび インスタンスを選択した場合は、無料利用枠内に収まるようにするには、合計ストレージの 30 GiB 未満を維持する必要があります。詳細については、「Amazon EBSユーザーガイド」の「 EBSボリュームのサイズと設定の制限」を参照してください。

   e. ボリュームタイプ: EBSボリュームの場合は、ボリュームタイプを選択します。詳細については、「Amazon ユーザーガイド」の「Amazon EBSボリュームタイプ」を参照してください。 EBS

   f. IOPS: プロビジョンド IOPS SSD (`io1` および `io2`) または汎用 SSD (`gp3`) ボリュームタイプを選択した場合は、ボリュームがサポートできる 1 秒あたりの I/O オペレーションの数 (IOPS) を入力できます。これは、io1、io2、gp3 ボリュームに必要です。gp2、st1、sc1、またはスタンダードボリュームではサポートされていません。

   g. 終了時に削除: EBSボリュームの場合は、はいを選択してインスタンスの終了時にボリュームを削除するか、いいえを選択してボリュームを維持します。

   h. 暗号化： インスタンスタイプがEBS暗号化をサポートしている場合は、はいを選択してボリュームの暗号化を有効にできます。このリージョンでデフォルトで暗号化を有効にした場合、暗号化は有効になります。詳細については、「Amazon ユーザーガイド」の「Amazon EBS暗号化」および「デフォルトで暗号化を有効にする」を参照してください。 EBS

      以下の表に示すように、このパラメータを設定した場合のデフォルトの効果は、選択したボリュームのソースによって異なります。いずれの場合も、指定された を使用するためのアクセス許可が必要です AWS KMS key。

暗号化の結果

Encrypted パラメータの設定	ボリュームのソース	デフォルトの暗号化状態	メモ
いいえ	新しい (空の) ボリューム	暗号化されていない*	該当なし
	所有する暗号化されていないスナップショット	暗号化されていない*	
	お客様が所有する暗号化されたスナップショット	同じキーで暗号化されている	
	お客様と共有されている暗号化されていないスナップショット	暗号化されていない*	
	お客様と共有されている暗号化されたスナップショット	デフォルトKMSキーで暗号化	
あり	新しいボリューム	デフォルトKMSキーで暗号化	デフォルト以外のKMSキーを使用するには、KMSキーパラメータの値を指定します。
	所有する暗号化されていないスナップショット	デフォルトKMSキーで暗号化	
	お客様が所有する暗号化されたスナップショット	同じキーで暗号化されている	
	お客様と共有されている暗号化されていないスナップショット	デフォルトKMSキーで暗号化	
	お客様と共有されている暗号化されたスナップショット	デフォルトKMSキーで暗号化	

* 暗号化がデフォルトで有効になっている場合、新しく作成されたすべてのボリューム (Encrypted パラメータが Yes に設定されているかどうかにかかわらず) は、デフォルトの KMS キーを使用して暗号化されます。暗号化パラメータとKMSキーパラメータの両方を設定する場合は、デフォルト以外のKMSキーを指定できます。

i. KMS キー: 暗号化に「はい」を選択した場合は、ボリュームの暗号化に使用するカスタマーマネージドキーを選択する必要があります。 このリージョンでデフォルトの暗号化を有効にした場合は、自動的にデフォルトのカスタマーマネージド型キーが選択されます。別のキーを選択するか、 を使用して以前に作成したカスタマーマネージドキーARNの を指定できます AWS Key Management Service。

3. この起動テンプレートによって起動されたインスタンスにアタッチする追加のボリュームを指定するには、[新しいボリュームを追加] を選択します。

既存のインスタンスから起動テンプレートを作成する (コンソール)

既存のインスタンスから起動テンプレートを作成するには

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインの [Instances] (インスタンス) で、[Instances] (インスタンス) を選択します。
3. インスタンスを選び、[Actions (アクション)]、[Image and templates (イメージとテンプレート)]、[Create Template from Instance (インスタンスからテンプレートを作成)] の順に選択します。
4. 名前と説明を入力します。
5. [Auto Scaling ガイダンス] で、[チェックボックス] を選択します。
6. 必要に応じて設定を調整し、[起動テンプレートの作成] を選択します。
7. Auto Scaling グループを作成するには、[confirmation (確認)] ページで[Auto Scaling グループの作成] を選択します。

関連リソース

AWS CloudFormation スタックYAMLテンプレートで起動テンプレートを宣言する方法を理解するために使用できるいくつかの JSONおよび テンプレートスニペットが用意されています。詳細につい

ては、「 AWS CloudFormation ユーザーガイド」のAWS「 : EC2:::LaunchTemplate」および「 セクションを使用して起動テンプレートを作成する AWS CloudFormation」を参照してください。

起動テンプレートの詳細については、「Amazon EC2 ユーザーガイド」の「起動テンプレートからのインスタンスの起動」を参照してください。

制限

- 起動テンプレートではサブネットを指定できますが、起動テンプレートを使用して Auto Scaling グループを作成するだけであれば、サブネットを指定する必要はありません。起動テンプレートでサブネットを指定して、Auto Scaling グループのサブネットを指定することはできません。Auto Scaling グループのサブネットは、Auto Scaling グループの独自のリソース定義から取得されます。
- ユーザー定義のネットワークインターフェースに関するその他の制限については、デフォルトのネットワークインターフェイス設定を変更する (コンソール)を参照してください。

詳細設定を使用して起動テンプレートを作成する

このトピックでは、 AWS Management Consoleの詳細設定を使用して起動テンプレートを作成する方法について説明します。

詳細設定を使用して起動テンプレートを作成するには

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインの [インスタンス] で、[起動テンプレート]、[起動テンプレートの作成] を順に選択します。
3. 次のトピックの説明に従って、起動テンプレートを設定します。
   - 必須の設定
   - 詳細設定
4. [起動テンプレートの作成] を選択します。

必須の設定

起動テンプレートを作成するときは、次の必須の設定を含める必要があります。

起動テンプレートの名前

起動テンプレートを説明する一意の名前を入力します。

アプリケーションと OS イメージ (Amazon マシンイメージ)

使用する Amazon マシンイメージ (AMI) を選択します。AMI 使用する を検索または参照できます。最適なスケーリング効率を得るには、アプリケーションコードでインスタンスを起動するようにAMI完全に設定され、起動時に変更が必要なカスタム を選択します。

インスタンスタイプ

と互換性のあるインスタンスタイプを選択しますAMI。Auto Scaling グループ独自のリソース定義に埋め込まれている複数のインスタンスタイプを使用する予定の場合は、起動テンプレートへのインスタンスタイプの追加をスキップできます。インスタンスタイプは、混合インスタンスグループを作成する予定がない場合にのみ必要です。

詳細設定

詳細設定はオプションです。詳細設定を設定しない場合、特定の機能はインスタンスに追加されません。

[高度な詳細] セクションを展開して、詳細設定を表示します。次のセクションでは、Auto Scaling グループの起動テンプレートを作成する場合に注目すべき、最も役に立つ詳細設定について説明します。詳細については、「Amazon EC2ユーザーガイド」の「詳細」を参照してください。

IAM インスタンスプロファイル

インスタンスプロファイルには、使用するIAMロールが含まれています。Auto Scaling グループがEC2インスタンスを起動すると、関連付けられたIAMロールで定義されたアクセス許可が、インスタンスで実行されているアプリケーションに付与されます。詳細については、「Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール」を参照してください。

終了保護

この機能を有効にすると、ユーザーは Amazon EC2コンソール、CLIコマンド、APIおよびオペレーションを使用してインスタンスを終了できなくなります。終了保護は、偶発的な終了に対する追加の保護手段を提供します。Amazon EC2 Auto Scaling がインスタンスを終了することを妨げるものではありません。Amazon EC2 Auto Scaling が終了できるインスタンスを制御するには、「」を参照してくださいインスタンスのスケールイン保護を使用してインスタンスの終了を制御する。

詳細 CloudWatch モニタリング

EC2 インスタンスの詳細モニタリングを有効にして、1 分間隔でメトリクスデータを Amazon CloudWatch に送信できます。デフォルトでは、EC2インスタンスはメトリクスデータを 5 分間隔で CloudWatch に送信します。別途 料金がかかります。詳細については、「Auto Scaling インスタンスのモニタリングを設定する」を参照してください。

クレジット仕様

Amazon EC2は、TT2, T3T3a などのバーストパフォーマンスインスタンスを提供します。これにより、アプリケーションは必要に応じてベースラインCPUパフォーマンスを超えてバーストできます。デフォルトでは、これらのインスタンスはCPU使用がスロットリングされる前に期間限定でバーストする可能性があります。オプションで、無制限モードを有効にして、インスタンスがベースラインを超えて必要な期間バーストできるようにすることができます。これにより、アプリケーションは必要に応じて高いCPUパフォーマンスを維持できます。追加料金が適用される場合があります。詳細については、「Amazon EC2 ユーザーガイド」のAuto Scaling グループを使用してバーストパフォーマンスインスタンスを無制限で起動する」を参照してください。

プレイスメントグループ名

プレイスメントグループを指定し、クラスターまたはパーティション戦略を使用すると、インスタンスが AWS データセンター内で物理的に配置される方法に影響を与えることができます。小規模な Auto Scaling グループの場合は、スプレッド戦略を使用することもできます。詳細については、「Amazon EC2ユーザーガイド」の「プレイスメントグループ」を参照してください。

Auto Scaling グループでプレイスメントグループを使用する場合、いくつかの考慮事項があります。

- プレイスメントグループが起動テンプレートと Auto Scaling グループの両方で指定されている場合、Auto Scaling グループのプレイスメントグループが優先されます。
- では AWS CloudFormation、起動テンプレートでプレイスメントグループを定義する場合は注意してください。Amazon EC2 Auto Scaling は、指定されたプレイスメントグループにインスタンスを起動します。ただし、Auto Scaling グループUpdatePolicyで を使用する場合、CloudFormation はこれらのインスタンスからシグナルを受信しません (ただし、これは将来変更される可能性があります) 。

購入オプション

スポットインスタンスをオンデマンド料金を上限とするスポット料金でリクエストする場合は [スポットインスタンスのリクエスト] を選択し、スポットインスタンスのデフォルト設定を変更する場合は [カスタマイズ] を選択します。Auto Scaling グループでは、終了日なしのワンタイムリクエストを指定する必要があります (デフォルト)。詳細については、「耐障害性に優れた柔軟

なアプリケーションのためにスポットインスタンスをリクエストする」を参照してください。この設定は特殊な状況では便利ですが、通常は指定せず、代わりに混合インスタンスグループを作成することをお勧めします。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。

起動テンプレートでスポットインスタンスリクエストを指定した場合、混合インスタンスグループを作成することはできません。混合インスタンスグループでスポットインスタンスをリクエストする起動テンプレートを使用しようとすると、次のように `Incompatible launch template: You cannot use a launch template that is set to request Spot Instances (InstanceMarketOptions) when you configure an Auto Scaling group with a mixed instances policy. Add a different launch template to the group and try again.` というエラーメッセージが表示されます。

Capacity Reservation

キャパシティ予約を使用すると、特定のアベイラビリティーゾーンの Amazon EC2インスタンスのキャパシティを任意の期間予約できます。詳細については、「Amazon EC2ユーザーガイド」の「オンデマンドキャパシティ予約」を参照してください。

インスタンスを起動する場合の設定を以下から選択できます。

- 任意のオープンキャパシティ予約 ([オープン])
- 特定のキャパシティ予約 ([ID 別のターゲット])
- キャパシティ予約のグループ ([グループ別のターゲット])

特定のキャパシティ予約をターゲットにするには、起動テンプレートのインスタンスタイプが予約のインスタンスタイプと一致する必要があります。Auto Scaling グループを作成するときは、キャパシティ予約と同じアベイラビリティーゾーンを使用します。 AWS リージョン 選択した に応じて、代わりにキャパシティブロックをターゲットにすることを選択できます。詳細については、「使用アイテム Capacity Blocks 機械学習ワークロード用の」を参照してください。

キャパシティ予約のグループをターゲットにするには、「キャパシティ予約を使用して特定のアベイラビリティーゾーンでキャパシティを予約する 」を参照してください。キャパシティ予約のグループをターゲットにすることで、キャパシティを複数のアベイラビリティーゾーンに分散させ、耐障害性を向上させることができます。

テナンシー

Amazon EC2 には、EC2インスタンスのテナンシーに 3 つのオプションがあります。

- 共有 ([共有]) – 複数の AWS アカウント で同じ物理ハードウェアを共有できます。これは、インスタンスを起動する際のデフォルトのテナンシーオプションです。

- ハードウェア専有インスタンス ([専用]) – インスタンスはシングルテナントハードウェアで実行されます。他の AWS のお客様が同じ物理サーバーを共有することはありません。詳細については、「Amazon EC2ユーザーガイド」の「ハードウェア専有インスタンス」を参照してください。
- Dedicated Hosts ([専有ホスト]) – インスタンスはお客様専用の物理サーバー上で実行されます。Dedicated Hosts を使用すると、ハードウェア専有要件を持つ独自のライセンス (BYOL) をに持ち込みEC2、コンプライアンスのユースケースを満たすことが容易になります。このオプションを選択した場合は、[テナンシーのホストリソースグループ] でホストリソースグループを指定する必要があります。詳細については、「Amazon EC2ユーザーガイド」の「Dedicated Hosts」を参照してください。

Dedicated Hosts のサポートは、ホストリソースグループを指定した場合にのみ使用できます。特定のホスト ID をターゲットにしたり、ホストのプレイスメントアフィニティを使用したりすることはできません。

- ホスト ID を指定する起動テンプレートを使用しようとすると、エラーメッセージ「`Incompatible launch template: Tenancy host ID is not supported for Auto Scaling.`」が表示されます。
- ホストのプレイスメントアフィニティを指定する起動テンプレートを使用しようとすると、エラーメッセージ「`Incompatible launch template: Auto Scaling does not support host placement affinity.`」が表示されます。

テナンシーのホストリソースグループ

を使用すると AWS License Manager、独自のライセンスを に持ち込み AWS 、一元管理できます。ホストリソースグループとは、特定の License Manager ライセンス設定にリンクされている Dedicated Hosts のグループのことです。ホストリソースグループを使用すると、ソフトウェアライセンスのニーズに合った Dedicated Hosts でEC2インスタンスを簡単に起動できます。Dedicated Hosts を事前に手動で割り当てる必要はありません。必要に応じて自動的に作成されます。AMI をライセンス設定に関連付けると、一度に 1 つのホストリソースグループにのみ関連付けAMIることができます。詳細については、「License Manager ユーザーガイド」の「AWS License Managerのホストリソースグループ」を参照してください。

ライセンス設定

この設定では、テナンシーを Dedicated Hosts に制限することなく、インスタンスのライセンス設定を指定できます。ライセンス設定を使用すると、インスタンスにデプロイされたソフトウェアライセンスの追跡が可能になるため、ライセンスの使用状況とコンプライアンスをモニタリングできます。詳細については、「License Manager ユーザーガイド」の「セルフマネージドライセンスの作成」を参照してください。

アクセス可能なメタデータ

インスタンスメタデータサービスのHTTPエンドポイントへのアクセスを有効または無効にするかどうかを選択できます。デフォルトでは、HTTPエンドポイントは有効になっています。エンドポイントを無効にすると、インスタンスメタデータへのアクセスはオフになります。条件を指定して、HTTPエンドポイントが有効になっていIMDSv2る場合にのみ を要求できます。詳細については、「Amazon EC2ユーザーガイド」の「インスタンスメタデータオプションの設定」を参照してください。

メタデータのバージョン

インスタンスメタデータをリクエストするときに、インスタンスメタデータサービスバージョン 2 (IMDSv2) の使用を要求できます。値を指定しない場合、デフォルトは IMDSv1と の両方をサポートしますIMDSv2。詳細については、「Amazon EC2ユーザーガイド」の「インスタンスメタデータオプションの設定」を参照してください。

メタデータトークンレスポンスのホップ制限

メタデータトークンに許容されるネットワークホップ数を設定できます。値を指定していない場合、デフォルトで 1 が適用されます。詳細については、「Amazon EC2ユーザーガイド」の「インスタンスメタデータオプションの設定」を参照してください。

ユーザーデータ

シェルスクリプトまたは Cloud-init ディレクティブをユーザーデータとして指定することで、起動時にインスタンスをカスタマイズして設定を完了できます。ユーザーデータは、インスタンスの初回起動時に実行されるため、起動時にアプリケーション、依存関係、またはカスタマイズを自動的にインストールできます。詳細については、「Amazon EC2 ユーザーガイド」の「起動時に Linux インスタンスでコマンドを実行する」を参照してください。

大量のダウンロードや複雑なスクリプトがある場合、インスタンスが使用可能になるまでの時間が長くなります。この場合、インスタンスが完全にプロビジョニングされるまで、`InService` 状態に到達するのを遅らせるようにライフサイクルフックを設定する必要があります。Auto Scaling グループへのライフサイクルフックの追加について詳しくは、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

耐障害性に優れた柔軟なアプリケーションのためにスポットインスタンスをリクエストする

起動テンプレートには、終了日や使用期間を指定せずにスポットインスタンスをリクエストできるオプションがあります。Amazon EC2スポットインスタンスは、EC2オンデマンド料金と比較して大幅

な割引で利用できる予備の容量です。スポットインスタンス は、アプリケーションを実行する時間に柔軟性がある場合や、アプリケーションを中断できる場合に、費用効率の高い選択肢です。スポットインスタンスをリクエストする起動テンプレート作成の詳細については、「詳細設定を使用して起動テンプレートを作成する」を参照してください。

> **Important**
>
> スポットインスタンスは通常、オンデマンドインスタンスを補完するために使用されます。このシナリオでは、スポットインスタンスの起動に使用されるものと同じ設定を、Auto Scaling グループの設定内で指定できます。Auto Scaling グループの一部として設定を指定する場合、特定の数のオンデマンドインスタンスを起動した後のみにスポットインスタンスを起動するようにリクエストできます。その後、グループのスケーリングに応じてオンデマンドインスタンスとスポットインスタンスの組み合わせを継続して起動するようにリクエストできます。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。

このトピックでは、Auto Scaling グループ自体ではなく起動テンプレートにより設定を指定することで、Auto Scaling グループ内にスポットインスタンスのみを起動する方法について説明します。このトピックの情報は、起動設定を使用してスポットインスタンスをリクエストする Auto Scaling グループにも適用されます。違いは、起動設定には上限価格が必須ですが、起動テンプレートの場合、この設定はオプションとなることです。

起動テンプレートを作成してスポットインスタンスのみを起動する場合は、次の考慮事項に留意してください。

- スポット料金 起動するスポットインスタンスには、現在のスポット料金のみが課金されます。この料金は、需要と供給の長期的な傾向に基づいて時間の経過とともに緩やかに変動します。詳細については、「Amazon EC2 ユーザーガイド」の「スポットインスタンス」および「料金と削減額」を参照してください。
- 上限価格を設定する。起動テンプレートではオプションとして、スポットインスタンスの時間あたりの上限価格を指定することができます。上限価格が現在のスポット料金を超える場合、Amazon EC2 スポットサービスは容量が利用可能な場合、リクエストをすぐに受理します。スポットインスタンスの料金が、お客様の Auto Scaling グループで実行中のインスタンスに設定された上限価格を上回った場合には、インスタンスが終了されます。

> **Warning**
>
> 低すぎる上限価格が設定されるなどの理由で、スポットインスタンスを取得できない場合には、アプリケーションが実行されないことがあります。利用可能なスポットインスタンスを、可能な限り長期にわたり活用するには、上限価格をオンデマンド料金に近い値に設定します。

- アベイラビリティーゾーン間でのバランシング。複数のアベイラビリティーゾーンを指定すると、Amazon EC2 Auto Scaling は指定されたゾーンにスポットリクエストを分散します。1 つのアベイラビリティーゾーンで上限価格が低すぎてリクエストが受理されない場合、Amazon EC2 Auto Scaling は他のゾーンでリクエストが受理されたかどうかを確認します。その場合、Amazon EC2 Auto Scaling は失敗したリクエストをキャンセルし、リクエストが受理されたアベイラビリティーゾーン全体に再分散します。リクエストが受理されていないアベイラビリティーゾーンの料金が、将来のリクエストが成功するのに十分なだけ下がった場合、Amazon EC2 Auto Scaling はすべてのアベイラビリティーゾーン間で再調整します。
- スポットインスタンスの終了 スポットインスタンスは任意のタイミングで終了できます。Amazon EC2スポットサービスは、スポットインスタンスの可用性または料金が変わると、Auto Scaling グループのスポットインスタンスを終了できます。スケーリングまたはヘルスチェックを実行する場合、Amazon EC2 Auto Scaling はオンデマンドインスタンスを終了するのと同じ方法でスポットインスタンスを終了することもできます。インスタンスが終了された際には、そのためのストレージは削除されます。
- 必要とされるキャパシティーの維持 スポットインスタンスが終了すると、Amazon EC2 Auto Scaling はグループの希望する容量を維持するために、別のスポットインスタンスの起動を試みます。現在のスポット料金が上限価格未満の場合は、スポットインスタンスが起動されます。スポットインスタンスのリクエストが正常に処理されなかった場合は、その試行が繰り替えされます。
- 上限価格の変更 上限価格を変更するには、起動テンプレートを新規で作成するか、既存の起動テンプレートを新しい上限価格で更新します。その上で、このテンプレートを Auto Scaling グループに関連付けます。これらのインスタンスに使用する起動テンプレートで指定された上限価格が、現在のスポット料金より高い限り、既存のスポットインスタンスが実行され続けます。上限価格を指定しない場合、オンデマンド料金がデフォルトの上限価格となります。

使用アイテム Capacity Blocks 機械学習ワークロード用の

Capacity Blocks は、短期間の機械学習 (ML) ワークロードをサポートするために、需要の高いGPU インスタンスを将来の日付で予約するのに役立ちます。

の概要 Capacity Blocks とその仕組みについては、「」を参照してください。 Capacity Blocks「Amazon EC2ユーザーガイド」の「 ML 用」。

の使用を開始するには Capacity Blocksでは、特定のアベイラビリティーゾーンにキャパシティ予約を作成します。Capacity Blocks は、単一のアベイラビリティーゾーンでtargetedキャパシティ予約として配信されます。起動テンプレートを作成するときは、キャパシティブロックの予約 ID とインスタンスタイプを指定します。次に、作成した起動テンプレートとキャパシティブロックのアベイラビリティーゾーンを使用するように Auto Scaling グループを更新します。キャパシティブロック予約が開始されたら、スケジュールされたスケーリングを使用して、キャパシティブロック予約と同じ数のインスタンスを起動します。

> **Important**
>
> Capacity Blocks は、特定の Amazon EC2インスタンスタイプおよび でのみ使用できます AWS リージョン。詳細については、「Amazon EC2 ユーザーガイド」の「前提条件」を参照してください。

内容

操作のガイドライン

以下は、Auto Scaling グループでキャパシティブロックを使用するに従うべき操作の基本的なガイドラインです。

- キャパシティブロックの予約終了時刻の 30 分以上前に、Auto Scaling グループをゼロにスケールインします。Amazon EC2は、キャパシティブロックの終了時刻の 30 分前にまだ実行中のインスタンスをすべて終了します。
- スケジュールされたスケーリングを使用して、適切な予約時刻にスケールアウト (インスタンスの追加) およびスケールイン (インスタンスの削除) することをお勧めします。詳細については、「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。
- 必要に応じてライフサイクルフックを追加し、スケールインの際にインスタンス内でアプリケーションを正常にシャットダウンします。Amazon がキャパシティブロックの予約終了時刻の 30 分

前にインスタンスの強制終了EC2を開始する前に、ライフサイクルアクションが完了するまで十分な時間を確保してください。詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

- Auto Scaling グループが、予約期間全体を通して正しいバージョンの起動テンプレートを指定していることを確認してください。`$Default` または `$Latest` バージョンではなく、特定のバージョンの起動テンプレートを指定することをお勧めします。

> **Note**
>
> 予約が終了するまでキャパシティブロックインスタンスを実行したままにして Amazon が EC2それを再利用した場合、Auto Scaling グループのスケーリングアクティビティは、キャパシティブロックの最後に意図的に再利用された場合でも`taken out of service in response to an EC2 health check that indicated it had been terminated or stopped`、「」と表示されます。同様に、Amazon EC2 Auto Scaling は、ヘルスチェックに失敗したインスタンスの場合と同じ方法でインスタンスの置き換えを試みます。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

起動テンプレートでキャパシティブロックを指定する

Auto Scaling グループの特定のキャパシティブロックをターゲットとする起動テンプレートを作成するには、次のいずれかの方法を使用します。

Console

起動テンプレートでキャパシティブロックを指定するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. 上部のナビゲーションバーで、キャパシティブロックを作成した AWS リージョン を選択します。
3. ナビゲーションペインで、[インスタンス] の [テンプレートの起動] を選択します。
4. [起動テンプレートの作成] を選択し、起動テンプレートを作成します。必要に応じて、Amazon マシンイメージ (AMI) の ID、インスタンスタイプ、およびその他の起動テンプレート設定を含めます。
5. [高度な詳細] セクションを展開して、詳細設定を表示します。

6. [購入オプション] で、[キャパシティブロック] を選択します。
7. [キャパシティ予約] で [ID 別のターゲット] を選択し、[キャパシティ予約 - ID ごとのターゲット] で、既存のキャパシティブロックのキャパシティ予約 ID を選択します。
8. 完了したら、[起動テンプレートの作成] を選択します。

起動テンプレートを使用した Auto Scaling グループの作成について詳しくは、「起動テンプレートを使用して Auto Scaling グループを作成する」を参照してください。

AWS CLI

起動テンプレートでキャパシティブロックを指定するには (AWS CLI)

次のcreate-launch-templateコマンドを使用して、既存のキャパシティブロック予約 ID を指定する起動テンプレートを作成します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

```
aws ec2 create-launch-template --launch-template-name my-template-for-capacity-block
  \
  --version-description AutoScalingVersion1 --region us-east-2 \
  --launch-template-data file://config.json
```

> **Tip**
>
> このコマンドでエラーが発生した場合は、 を AWS CLI ローカルで最新バージョンに更新していることを確認してください。

config.json の内容。

```
{
    "ImageId": "ami-04d5cc9b88example",
    "InstanceType": "p4d.24xlarge",
    "SecurityGroupIds": [
        "sg-903004f88example"
    ],
    "KeyName": "MyKeyPair",
    "InstanceMarketOptions": {
        "MarketType": "capacity-block"
    },
```

```
    "CapacityReservationSpecification": {
        "CapacityReservationTarget": {
            "CapacityReservationId": "cr-02168da1478b509e0"
        }
    }
}
```

以下は出力例です。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b724example",
        "LaunchTemplateName": "my-template-for-capacity-block",
        "CreateTime": "2023-10-27T15:12:44.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

次のdescribe-launch-template-versionsコマンドを使用して、起動テンプレートに関連付けられたキャパシティブロック予約 ID を確認できます。

```
aws ec2 describe-launch-template-versions --launch-template-names my-template-for-capacity-block \
  --region us-east-2
```

以下は、キャパシティブロックの予約を指定する起動テンプレートの出力例です。

```
{
    "LaunchTemplateVersions": [
        {
            "LaunchTemplateId": "lt-068f72b724example",
            "LaunchTemplateName": "my-template-for-capacity-block",
            "VersionNumber": 1,
            "CreateTime": "2023-10-27T15:12:44.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersion": true,
            "LaunchTemplateData": {
                "ImageId": "ami-04d5cc9b88example",
                "InstanceType": "p5.48xlarge",
```

```
                "SecurityGroupIds": [
                    "sg-903004f88example"
                ],
                "KeyName": "MyKeyPair",
                "InstanceMarketOptions": {
                    "MarketType": "capacity-block"
                },
                "CapacityReservationSpecification": {
                    "CapacityReservationTarget": {
                        "CapacityReservationId": "cr-02168da1478b509e0"
                    }
                }
            }
        }
    ]
}
```

制限

- サポート対象 Capacity Blocks は、Auto Scaling グループに互換性のある設定がある場合にのみ使用できます。混合インスタンスグループおよびウォームプールはサポートされていません。
- 一度にターゲットにできるキャパシティブロックは 1 つだけです。

関連リソース

- P5 インスタンスを使用するための前提条件と推奨事項については、「Amazon EC2 ユーザーガイド」のP5 インスタンスの開始方法」を参照してください。
- Amazon が の使用EKSをサポート Capacity Blocks は、Amazon EKSクラスターで短期間の機械学習 (ML) ワークロードをサポートします。詳細については、「」を参照してくださいCapacity Blocks「Amazon EKSユーザーガイド」の「 ML 用」。
- 以下を使用できます..。Capacity Blocks サポートされているインスタンスタイプとリージョンを持つ 。ただし、オンデマンドキャパシティ予約では、その他のインスタンスタイプとリージョンのキャパシティを柔軟に予約できます。オンデマンドキャパシティ予約オプションの使用方法を示すチュートリアルについては、「キャパシティ予約を使用して特定のアベイラビリティーゾーンでキャパシティを予約する 」を参照してください。

Auto Scaling グループを起動テンプレートに移行する

2023 年以降、2022 年 12 月 31 日以降にリリースされた新しい Amazon EC2インスタンスタイプCreateLaunchConfigurationで を呼び出すことはできません。詳細については、「Auto Scaling 起動設定」を参照してください。

Auto Scaling グループを起動設定から起動テンプレートに移行するには、次の手順を参照してください。

> **Important**
>
> 続行する前に、起動テンプレートを操作するために必要な許可があることを確認してください。詳細については、「起動テンプレートを操作するためのアクセス許可」を参照してください。

ステップ 1: 起動設定を使用する Auto Scaling グループを検索する

起動設定をまだ使用している Auto Scaling グループがあるかどうかを確認するには、 を使用して次のdescribe-auto-scaling-groupsコマンドを実行します AWS CLI。を *REGION*に置き換えます AWS リージョン。

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

以下は出力例です。

```
[
    {
 "AutoScalingGroupName": "group-1",
 "AutoScalingGroupARN": "arn",
 "LaunchConfigurationName": "my-launch-config",
 "MinSize": 1,
 "MaxSize": 5,
 "DesiredCapacity": 2,
 "DefaultCooldown": 300,
 "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
```

```
        ],
  "LoadBalancerNames": [],
  "TargetGroupARNs": [],
  "HealthCheckType": "EC2",
  "HealthCheckGracePeriod": 300,
  "Instances": [
            {
                "ProtectedFromScaleIn": false,
                "AvailabilityZone": "us-west-2a",
                "LaunchConfigurationName": "my-launch-config",
                "InstanceId": "i-05b4f7d5be44822a6",
                "InstanceType": "t3.micro",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService"
            },
            {
                "ProtectedFromScaleIn": false,
                "AvailabilityZone": "us-west-2b",
                "LaunchConfigurationName": "my-launch-config",
                "InstanceId": "i-0c20ac468fa3049e8",
                "InstanceType": "t3.micro",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService"
            }
  ],
  "CreatedTime": "2023-03-09T22:15:11.611Z",
  "SuspendedProcesses": [],
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "EnabledMetrics": [],
  "Tags": [
            {
   "ResourceId": "group-1",
   "ResourceType": "auto-scaling-group",
   "Key": "environment",
   "Value": "production",
   "PropagateAtLaunch": true
            }
        ],
  "TerminationPolicies": [
      "Default"
  ],
  "NewInstancesProtectedFromScaleIn": false,
  "ServiceLinkedRoleARN": "arn",
        "TrafficSources": []
```

```
    },

    ... additional groups ...

]
```

あるいは、Auto Scaling グループ名、それぞれの起動設定およびタグの名前のみを出力に表示するには、次のコマンドを実行します。

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`].{AutoScalingGroupName:
 AutoScalingGroupName, LaunchConfigurationName: LaunchConfigurationName, Tags: Tags}'
```

出力例を次に示します。

```
[
    {
        "AutoScalingGroupName": "group-1",
        "LaunchConfigurationName": "my-launch-config",
        "Tags": [
            {
                "ResourceId": "group-1",
                "ResourceType": "auto-scaling-group",
                "Key": "environment",
                "Value": "production",
                "PropagateAtLaunch": true
            }
        ]
    },

    ... additional groups ...

]
```

フィルタリングの詳細については、「 AWS Command Line Interface ユーザーガイド」のAWS CLI「出力のフィルタリング」を参照してください。

ステップ 2: 起動設定を起動テンプレートにコピーする

次のステップを実行して、起動設定を起動テンプレートにコピーできます。その後、それを Auto Scaling グループに追加できます。

複数の起動設定をコピーすると、同じ名前の起動テンプレートが作成されます。コピープロセス中に起動テンプレートに付けられた名前を変更するには、起動設定を 1 つずつコピーする必要があります。

> **Note**
>
> コピー機能は、コンソールでのみ使用できます。

起動テンプレートへ起動設定をコピーするには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. 左のナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。
3. ページの上部付近にある [起動設定] を選択します。確認を求めるプロンプトが表示されたら、[起動設定を表示] を選択して、[起動設定] ページを表示することを確認します。
4. コピーする起動設定を選択し、[Copy to launch template (起動テンプレートにコピー)、Copy selected (選択した範囲をコピー)] を選択します。これにより、新しい起動テンプレートが、選択した起動設定と同じ名前およびオプションでセットアップされます。
5. [New launch template name (新しい起動テンプレート名)] では、起動設定の名前 (デフォルト) を使用する、または、新しい名前を入力します。起動テンプレート名は一意である必要があります。
6. (オプション) [新しいテンプレートを使用して Auto Scaling グループを作成] を選択します。

   このステップをスキップして、起動設定のコピーを完了することができます。新しい Auto Scaling グループを作成する必要はありません。
7. [コピー] を選択します。

すべての起動設定を起動テンプレートにコピーするには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインの [Auto Scaling] で、[Launch Configurations (起動設定)] を選択します。
3. Copy to launch template (起動テンプレートにコピー)、Copy all (すべてコピー)を選択します。これにより、現在のリージョンの各起動設定が、同じ名前およびオプションで新しい起動テンプレートにコピーされます。
4. [コピー] を選択します。

ステップ 3: 起動テンプレートを使用するように Auto Scaling グループを更新する

起動テンプレートを作成すると、それを Auto Scaling グループに追加する準備が整います。

起動テンプレートを使用するように Auto Scaling グループを更新するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの隣にあるチェックボックスを選択します。

   ページ下部に分割ウィンドウが開き、選択したグループの情報が表示されます。
3. [詳細] タブで、[起動設定]、[編集] の順に選択します。
4. 起動テンプレートに切り替えるを選択します。
5. [Launch Template (起動テンプレート)] では、起動テンプレートを選択します。
6. [Version (バージョン)] では、必要に応じて起動テンプレートのバージョンを選択します。起動テンプレートのバージョンを作成したら、スケールアウト時に Auto Scaling グループで起動テンプレートのデフォルトバージョンを使用するか最新バージョンを使用するかを選択できます。
7. [Update] (更新) を選択します。

起動テンプレートを使用するように Auto Scaling グループを更新するには (AWS CLI)

次のupdate-auto-scaling-groupコマンドは、指定された Auto Scaling グループを更新して、指定された起動テンプレートの初期バージョンを使用します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
```

CLI コマンドを使用して Auto Scaling グループを更新して起動テンプレートを使用するその他の例については、「」を参照してください起動テンプレートを使用するように Auto Scaling グループを更新する。

ステップ 4: インスタンスを置き換える

起動設定を起動テンプレートに置き換えると、新しいインスタンスは、その新しい起動テンプレートを使用するようになります。既存のインスタンスは影響を受けません。

既存のインスタンスを更新するには、一度にいくつかのインスタンスを手動で置き換えるのではなく、インスタンスの更新を開始して Auto Scaling グループ内のインスタンスを置き換えることができます。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。グループが大きい場合、インスタンスの更新は特に便利です。

あるいは、自動スケーリングを許可して、グループの終了ポリシーに基づき既存のインスタンスを新しいインスタンスに徐々に置き換えたり、インスタンスを終了したりすることもできます。手動での終了により、グループが必要とするキャパシティーを維持するため、強制的にAuto Scaling グループが新しいインスタンスを起動します。詳細については、「Amazon EC2ユーザーガイド」の「インスタンスの終了」を参照してください。

追加情報

詳細については、 AWS 「 コンピューティングブログ」の「Amazon EC2 Auto Scaling は起動設定EC2に新機能のサポートを追加しなくなります」を参照してください。

起動設定から起動テンプレートに AWS CloudFormation スタックを移行する方法を説明するトピックについては、「」を参照してくださいAWS CloudFormation スタックを起動テンプレートに移行する。

AWS CloudFormation スタックを起動テンプレートに移行する

既存の AWS CloudFormation スタックテンプレートを起動設定から起動テンプレートに移行できます。これを行うには、起動テンプレートを既存のスタックテンプレートに直接追加した後、そのスタックテンプレート内で、起動テンプレートに Auto Scaling グループを関連付けます。その後、変更したテンプレートを使用してスタックを更新します。

起動テンプレートに移行する場合、このトピックでは、 CloudFormation スタックテンプレートの起動設定を起動テンプレートとして書き換える手順を提供することで、時間を節約できます。起動設定を起動テンプレートに移行する方法の詳細については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

トピック

- 起動設定を使用している Auto Scaling グループを検索する
- 起動テンプレートを使用するようにスタックを更新する
- スタックリソースの更新時の動作
- 移行を追跡する

- 起動設定のマッピングリファレンス

起動設定を使用している Auto Scaling グループを検索する

起動設定を使用している Auto Scaling グループを見つけるには

- 次のdescribe-auto-scaling-groupsコマンドを使用して、指定したリージョンで起動設定を使用している Auto Scaling グループの名前を一覧表示します。`--filters` スタックに関連付けられたグループに結果を絞り込むオプションを含めます CloudFormation（`aws:cloudformation:stack-name`タグキーでフィルタリング）。

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --filters Name=tag-key,Values=aws:cloudformation:stack-name \
  --query 'AutoScalingGroups[?LaunchConfigurationName!
=`null`].AutoScalingGroupName'
```

出力例を次に示します。

```
[
    "{stack-name}-group-1",
    "{stack-name}-group-2",
    "{stack-name}-group-3"
]
```

で Auto Scaling グループを検索して、出力を移行およびフィルタリングするためのその他の便利な AWS CLI コマンドを見つけることができますAuto Scaling グループを起動テンプレートに移行する。

> **Important**
>
> スタックリソースの名前AWSEBに が含まれている場合は、 を通じて作成されたことを意味します AWS Elastic Beanstalk。この場合、Beanstalk 環境を更新して、起動設定を削除して起動テンプレートに置き換えるように Elastic Beanstalk に指示する必要があります。

起動テンプレートを使用するようにスタックを更新する

このセクションのステップに従い、 その手順を実行します。

- 起動設定のプロパティを、起動テンプレートの同等のプロパティに対し記述し直します。
- 新しい起動テンプレートを Auto Scaling グループと関連付けます。
- これらの更新内容をデプロイします。

スタックテンプレートを変更してスタックを更新するには

1. 「AWS CloudFormation ユーザーガイド」の「スタックテンプレートの変更」で説明されている、一般的なスタックテンプレートの変更と同じ手順に従います。
2. 起動設定を起動テンプレートとして記述し直します。次の例を参照してください。

   例: シンプルな起動設定

```
---
Resources:
  myLaunchConfig:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId: ami-02354e95b3example
      InstanceType: t3.micro
      SecurityGroups:
        - !Ref EC2SecurityGroup
      KeyName: MyKeyPair
      BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: 150
            DeleteOnTermination: true
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource myASG
 --region ${AWS::Region}
```

   例: 同等の起動テンプレート

```
---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
```

```
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        SecurityGroupIds:
          - Ref! EC2SecurityGroup
        KeyName: MyKeyPair
        BlockDeviceMappings:
          - DeviceName: /dev/xvda
            Ebs:
              VolumeSize: 150
              DeleteOnTermination: true
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash -x
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
myASG --region ${AWS::Region}
```

Amazon がEC2サポートするすべてのプロパティのリファレンス情報については、「 AWS CloudFormation ユーザーガイド」のAWS「 : EC2::LaunchTemplate : 」を参照してください。

起動テンプレートには、!Sub ${AWS::StackName}-launch-template 値が指定された LaunchTemplateName プロパティが含まれていることに注意してください。これは、起動テンプレート名にスタック名を含めたい場合に必須です。

3. **IamInstanceProfile** プロパティが起動設定に存在する場合は、それを 構造に変換し、インスタンスプロファイルの名前または ARN を指定する必要があります。例については、AWS「:::EC2:::LaunchTemplate」を参照してください。
4. 起動設定に **AssociatePublicIpAddress**、**InstanceMonitoring**、または **PlacementTenancy** プロパティがある場合は、これらを構造体に変換する必要があります。例については、AWS「:::EC2::LaunchTemplate」を参照してください。

   ただし、Auto Scaling グループに使用したサブネットの MapPublicIpOnLaunch プロパティの値が、起動設定の AssociatePublicIpAddress プロパティの値と一致する場合は例外です。この場合は、AssociatePublicIpAddress プロパティを無視できます。AssociatePublicIpAddress プロパティは、 MapPublicIpOnLaunch プロパティを上書きして、インスタンスが起動時にパブリックIPv4アドレスを受け取るかどうかを変更する場合にのみ使用されます。

5. **SecurityGroups** プロパティから、セキュリティグループを起動テンプレートの 2 つの場所のどちらかにコピーすることができます。通常は、SecurityGroupIds プロパティにセキュリティグループをコピーします。ただし、起動テンプレート内に AssociatePublicIpAddress プロパティを指定する NetworkInterfaces 構造体を作成する場合は、代わりにネットワークインターフェイスの Groups プロパティに対し、セキュリティグループをコピーする必要があります。

6. を **NoDevice**に設定して起動設定にBlockDeviceMapping構造が存在する場合はtrue、起動テンプレートNoDeviceで に空の文字列を指定して、Amazon EC2 がデバイスを省略するようにする必要があります。

7. 起動設定内に **SpotPrice** プロパティが含まれている場合、起動テンプレートではそれを省略することをお勧めします。スポットインスタンスは現在のスポット料金で起動します。この価格は、オンデマンド料金を超えることはありません。

   スポットインスタンスをリクエストする際には、下記のように相互に排他的な 2 つのオプションが存在します。

   - 1 つ目は、起動テンプレート内で InstanceMarketOptions 構造体を使用することです (非推奨)。詳細については、 AWS CloudFormation ユーザーガイドのAWS「 : EC2:::LaunchTemplate InstanceMarketOptions」を参照してください。
   - 2 つ目は、Auto Scaling グループに MixedInstancesPolicy 構造体を追加することです。これを行うと、リクエストの方法に、より多くの選択肢が得られるようになります。起動テンプレートのスポットインスタンスリクエストでは、Auto Scaling グループごとに複数のインスタンスタイプを選択することはできません。ただし、インスタンスポリシーを組み合わせることで、Auto Scaling グループごとに複数のインスタンスタイプを選択できます。インスタンスタイプに複数の選択肢があると、スポットインスタンスリクエストに対するメリットが得られます。詳細については、「 ユーザーガイド」のAWS「 : AutoScaling::AutoScalingGroup MixedInstancesPolicy」を参照してください。 AWS CloudFormation

8. AWS::::AutoScaling:::AutoScalingGroupリソースから **LaunchConfigurationName**プロパティを削除します。その代わりに、起動テンプレートを追加します。

   次の例では、Ref 組み込み関数は論理 ID を持つ AWS::EC2::LaunchTemplate: リソースの ID を取得しますmyLaunchTemplate。GetAtt 関数は、 Versionプロパティの起動テンプレートの最新バージョン番号 (など1) を取得します。

   例: インスタンスポリシーの組み合わせなし

```
---
```

```
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref myLaunchTemplate
        Version: !GetAtt myLaunchTemplate.LatestVersionNumber
      ...
```

例: インスタンスポリシーの組み合わせあり

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref myLaunchTemplate
            Version: !GetAtt myLaunchTemplate.LatestVersionNumber
      ...
```

Amazon EC2 Auto Scaling がサポートするすべてのプロパティのリファレンス情報については、「 AWS CloudFormation ユーザーガイド」のAWS「::AutoScaling::AutoScalingGroup：」を参照してください。

9. これらの更新をデプロイする準備ができたら、 CloudFormation変更されたスタックテンプレートでスタックを更新する手順に従います。詳細については、「AWS CloudFormation ユーザーガイド」の「スタックテンプレートの変更」を参照してください。

スタックリソースの更新時の動作

CloudFormation は、指定した更新されたテンプレートと、以前のバージョンのスタックテンプレートで説明したリソース設定の変更を比較して、スタックリソースを更新します。変更されていないリソース設定は、更新プロセス中も影響を受けません。

CloudFormation は Auto Scaling グループの UpdatePolicy 属性をサポートします。更新中に UpdatePolicyが に設定されている場合AutoScalingRollingUpdate、この手順のステップを実行した後、 はInServiceインスタンスを CloudFormation 置き換えます。UpdatePolicy が に

設定されている場合AutoScalingReplacingUpdate、 は Auto Scaling グループとそのウォームプール (存在する場合) を CloudFormation 置き換えます。

Auto Scaling グループのUpdatePolicy属性を指定しなかった場合、起動テンプレートが正しいかどうかはチェックされますが、Auto Scaling グループのインスタンス全体に変更がデプロイ CloudFormation されることはありません。すべての新しいインスタンスは、起動テンプレートを使用するようになります。ただし、既存のインスタンスは、起動した当初の起動設定のままで実行を継続します (起動設定が存在しない場合は除く)。この例外となるのは、組み合わせのインスタンスポリシーを追加するなどして、購入オプションを変更した場合です。この場合、新しい購入オプションに適合するために、Auto Scaling グループにより、徐々に既存のインスタンスが新しいインスタンスに置き換えられます。

起動設定から起動テンプレートに移行するために変更をロールバックする必要がある場合は、必ずロールバック操作をテストしてください。

移行を追跡する

移行を追跡するには

1. AWS CloudFormation コンソールで、更新したスタックを選択し、[Events] (イベント) タブを選択してスタックイベントを表示します。
2. イベントリストを最新のイベントで更新するには、 CloudFormation コンソールで更新ボタンを選択します。
3. スタックの更新中は、リソースが更新されるたびに複数のイベントが表示されます。起動テンプレートを作成しようとしたときに、[ステータスの理由] 列に例外が表示され問題が発生したことが示された場合は、「Amazon EC2 Auto Scaling のトラブルシューティング: テンプレートの起動」で可能性のある原因を参照してください。
4. (オプション) UpdatePolicy 属性の使用に応じて、Amazon EC2コンソールの Auto Scaling グループページから Auto Scaling グループの進行状況をモニタリングできます。Auto Scaling グループを選択します。[Activity] (アクティビティ) タブで、[Activity history] (アクティビティ履歴) の下の [Status] (ステータス) 列に、Auto Scaling グループがインスタンスを正常に起動あるいは終了したか、または、依然としてスケーリングが進行中なのかが表示されます。
5. スタックの更新が完了すると、 は UPDATE_COMPLETEスタックイベント CloudFormation を発行します。詳細については、「AWS CloudFormation ユーザーガイド」の「スタック更新の進行状況の監視」を参照してください。

6. スタックの更新が完了したら、Amazon EC2コンソールの起動テンプレートページと起動設定ページを開きます。新しい起動テンプレートが作成され、起動設定が削除されていることが確認できます。

起動設定のマッピングリファレンス

参考までに、次の表は、AWS::::::::AutoScalingLaunchConfiguration::::::::::::::::::AWS::::::::::::::EC2::::::LaunchTemplate::::::::::::::

起動設定のソースプロパティ	起動テンプレートのターゲットプロパティ
AssociatePublicIpAddress	NetworkInterfaces.AssociatePublicIpAddress
BlockDeviceMappings	BlockDeviceMappings
ClassicLinkVPCId	利用不可
ClassicLinkVPCSecurityGroups	利用不可
EbsOptimized	EbsOptimized
IamInstanceProfile	IamInstanceProfile.Arn または IamInstanceProfile.Name を入力します。両方を入力することはできません。
ImageId	ImageId
InstanceId	InstanceId
InstanceMonitoring	Monitoring.Enabled
InstanceType	InstanceType
KernelId	KernelId
KeyName	KeyName
LaunchConfigurationName	LaunchTemplateName

起動設定のソースプロパティ	起動テンプレートのターゲットプロパティ
MetadataOptions	MetadataOptions
PlacementTenancy	Placement.Tenancy
RamDiskId	RamDiskId
SecurityGroups	SecurityGroupIds または NetworkInterfaces.Groups を入力します。両方を入力することはできません。
SpotPrice	InstanceMarketOptions.SpotOptions.MaxPrice
UserData	UserData

1 EC2-Classic は使用できないため、 ClassicLinkVPCIdおよび ClassicLinkVPCSecurityGroupsプロパティは起動テンプレートでは使用できません。

を使用した起動テンプレートの作成と管理の例 AWS CLI

起動テンプレートは、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または を使用して作成および管理できますSDKs。このセクションでは、 から Amazon EC2 Auto Scaling の起動テンプレートを作成および管理する例を示します AWS CLI。

内容

- 既存のネットワークインターフェイスを指定する
- 複数のネットワークインターフェイスを作成する
- 起動テンプレートを管理する
- 起動テンプレートを使用するように Auto Scaling グループを更新する

使用例

```
{
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "VersionDescription": "test description",
    "LaunchTemplateData": {
        "ImageId": "ami-04d5cc9b88example",
        "InstanceType": "t2.micro",
        "SecurityGroupIds": [
            "sg-903004f88example"
        ],
        "KeyName": "MyKeyPair",
        "Monitoring": {
            "Enabled": true
        },
        "Placement": {
            "Tenancy": "dedicated"
        },
        "CreditSpecification": {
            "CpuCredits": "unlimited"
        },
        "MetadataOptions": {
            "HttpTokens": "required",
            "HttpPutResponseHopLimit": 1,
            "HttpEndpoint": "enabled"
        }
    }
}
```

基本的な起動テンプレートを作成する

基本的な起動テンプレートを作成するには、次のように create-launch-template コマンドを使用し、以下の変更を加えます。

- を、インスタンスを起動する AMI の ID `ami-04d5cc9b88example`に置き換えます。

- を、指定した と互換性のあるインスタンスタイプ`t2.micro`に置き換えAMIます。

この例では、 という名前の起動テンプレートを作成します*my-template-for-auto-scaling*。この起動テンプレートによって作成されたインスタンスがデフォルトの で起動された場合VPC、デフォルトでパブリック IP アドレスを受け取ります。インスタンスがデフォルト以外の で起動された場合VPC、デフォルトではパブリック IP アドレスは受信されません。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

JSON形式のパラメータの引用の詳細については、「 ユーザーガイド」の「文字列での引用符の使用 AWS CLI」を参照してください。 AWS Command Line Interface

または、設定ファイルで JSON形式のパラメータを指定することもできます。

次の例では、起動テンプレートパラメータ値の設定ファイルを参照して、基本的な起動テンプレートを作成します。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data file://config.json
```

`config.json` の内容：

```
{
    "ImageId":"ami-04d5cc9b88example",
    "InstanceType":"t2.micro"
}
```

起動時にインスタンスにタグ付けするタグを指定する

次の例では、起動時にタグ (例: `purpose=webserver`) をインスタンスに追加します。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"purpose","Value":"webserver"}]}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.
```

> **Note**
>
> 起動テンプレートでインスタンスタグを指定して、Auto Scaling グループのタグをそのインスタンスに伝播することを選択した場合、すべてのタグがマージされます。起動テンプレートのタグと Auto Scaling グループのタグに同じタグキーが指定されている場合、グループのタグ値が優先されます。

インスタンスに渡す IAMロールを指定する

次の例では、起動時にインスタンスに渡すIAMロールに関連付けられたインスタンスプロファイルの名前を指定します。詳細については、「Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール」を参照してください。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data '{"IamInstanceProfile":{"Name":"my-instance-
profile"},"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

パブリック IP アドレスを割り当てる

次のcreate-launch-template例では、デフォルト以外の で起動されたインスタンスにパブリックアドレスを割り当てるように起動テンプレートを設定しますVPC。

> **Note**
>
> ネットワークインターフェイスを指定するときは、Auto Scaling グループGroupsがインスタンスを起動VPCする のセキュリティグループに対応する の値を指定します。Auto Scaling グループのプロパティとしてサブネットVPCを指定します。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","Instanc
```

起動時にインスタンスを設定するユーザーデータスクリプトを指定する

次の例では、起動時にインスタンスを設定する base64-encoded 文字列としてユーザーデータスクリプトを指定します。create-launch-template コマンドには、base64 でエンコードされたユーザーデータが必要です。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data
 '{"UserData":"IyEvYmluL2Jhc...","ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

ブロックデバイスマッピングを指定する

次のcreate-launch-template例では、ブロックデバイスマッピングを使用して起動テンプレートを作成します。22 ギガバイトのEBSボリュームが にマッピングされます/dev/xvdcz。/dev/xvdcz ボリュームは汎用 SSD (gp2) ボリュームタイプを使用し、アタッチされているインスタンスを終了すると削除されます。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"BlockDeviceMappings":[{"DeviceName":"/dev/xvdcz","Ebs":
{"VolumeSize":22,"VolumeType":"gp2","DeleteOnTermination":true}}],"ImageId":"ami-04d5cc9b88exam
```

外部ベンダーからソフトウェアライセンスを取得するための Dedicated Hosts を指定する

ホストテナンシーを指定すると、ホストリソースグループと License Manager ライセンス構成を指定して、外部ベンダーから適格なソフトウェアライセンスを取得できます。次に、次のcreate-launch-templateコマンドを使用して、EC2インスタンスでライセンスを使用できます。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"Placement":
{"Tenancy":"host","HostResourceGroupArn":"arn"},"LicenseSpecifications":
[{"LicenseConfigurationArn":"arn"}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"
```

既存のネットワークインターフェイスを指定する

次のcreate-launch-template例では、既存のネットワークインターフェイスを使用するようにプライマリネットワークインターフェイスを設定します。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"NetworkInterfaceId":"eni-
b9a5ac93","DeleteOnTermination":false}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.mi
```

複数のネットワークインターフェイスを作成する

次のcreate-launch-template例では、セカンダリネットワークインターフェイスを追加します。プライマリネットワークインターフェイスのデバイスインデックスは 0 で、セカンダリネットワークインターフェイスのデバイスインデックスは 1 です。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true},{"DeviceIndex":1,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","Instanc
```

複数のネットワークカードと Elastic Fabric Adapters (EFAs) をサポートするインスタンスタイプを使用する場合は、次のcreate-launch-templateコマンドEFAを使用してセカンダリネットワークカードにセカンダリインターフェイスを追加し、 を有効にできます。詳細については、「Amazon EC2 ユーザーガイド」の「起動テンプレートEFAへの の追加」を参照してください。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"NetworkCardIndex":0,"DeviceIndex":0,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true},
{"NetworkCardIndex":1,"DeviceIndex":1,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true}],"ImageId":"ami-09d95
```

> **Warning**
>
> p4d.24xlarge インスタンスタイプは、このセクションの他の例よりも高いコストが発生します。P4d インスタンスの料金の詳細については、「Amazon EC2 P4d インスタンスの料金」を参照してください。

> **Note**
>
> 同じサブネットから複数のネットワークインターフェイスをインスタンスにアタッチすると、特に Amazon Linux 以外のバリアントを使用するインスタンスでは、非対称ルーティングが発生する場合があります。このタイプの設定が必要な場合は、OS 内でセカンダリネットワークインターフェイスを設定する必要があります。例については、 AWS ナレッジセンターの「Ubuntu EC2インスタンスでセカンダリネットワークインターフェイスを機能させるにはどうすればよいですか？」を参照してください。

起動テンプレートを管理する

AWS CLI には、起動テンプレート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。

内容

- 起動テンプレートをリストして記述する
- 起動テンプレートのバージョンの作成
- 起動テンプレートのバージョンの削除
- 起動テンプレートの削除

起動テンプレートをリストして記述する

起動テンプレートに関する情報を取得するには、 describe-launch-templatesと の 2 つの AWS CLI コマンドを使用できますdescribe-launch-template-versions。

describe-launch-templates コマンドを使用すると、作成した起動テンプレートのリストを取得できます。オプションを使用すると、起動テンプレート名、作成時間、タグキー、またはタグとキーバリューの組み合わせの結果をフィルタリングできます。このコマンドは、起動テンプレート識別子、最新バージョン、デフォルトバージョンなど、起動テンプレートに関する概要情報を返します。

次の例では、指定された起動テンプレートの概要を示します。

```
aws ec2 describe-launch-templates --launch-template-names my-template-for-auto-scaling
```

以下に、応答の例を示します。

```
{
    "LaunchTemplates": [
        {
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "CreateTime": "2020-02-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersionNumber": 1,
            "LatestVersionNumber": 1
        }
    ]
}
```

出力を 1 つの起動テンプレートだけに制限する `--launch-template-names` オプションを使用しない場合は、すべての起動テンプレートの情報が返されます。

次のdescribe-launch-template-versionsコマンドは、指定された起動テンプレートのバージョンを説明する情報を提供します。

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b729example
```

以下に、応答の例を示します。

```
{
    "LaunchTemplateVersions": [
        {
            "VersionDescription": "version1",
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "LaunchTemplateData": {
                "TagSpecifications": [
                    {
                        "ResourceType": "instance",
                        "Tags": [
```

```
                        {
                            "Key": "purpose",
                            "Value": "webserver"
                        }
                    ]
                }
            ],
            "ImageId": "ami-04d5cc9b88example",
            "InstanceType": "t2.micro",
            "NetworkInterfaces": [
                {
                    "DeviceIndex": 0,
                    "DeleteOnTermination": true,
                    "Groups": [
                        "sg-903004f88example"
                    ],
                    "AssociatePublicIpAddress": true
                }
            ]
        },
        "DefaultVersion": true,
        "CreateTime": "2020-02-28T19:52:27.000Z"
    }
]
}
```

起動テンプレートのバージョンの作成

次のcreate-launch-template-versionコマンドは、起動テンプレートのバージョン 1 に基づいて新しい起動テンプレートのバージョンを作成し、別の AMI ID を指定します。

```
aws ec2 create-launch-template-version --launch-template-id lt-068f72b729example --
version-description version2 \
  --source-version 1 --launch-template-data "ImageId=ami-c998b6b2example"
```

起動テンプレートのデフォルトバージョンを設定するには、 modify-launch-template コマンドを使用します。

起動テンプレートのバージョンの削除

次のdelete-launch-template-versionsコマンドは、指定された起動テンプレートのバージョンを削除します。

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b729example --
versions 1
```

起動テンプレートの削除

起動テンプレートが不要になった場合は、次のdelete-launch-templateコマンドを使用して削除できます。起動テンプレートを削除すると、すべてのバージョンが削除されます。

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b729example
```

起動テンプレートを使用するように Auto Scaling グループを更新する

update-auto-scaling-group コマンドを使用して、既存の Auto Scaling グループに起動テンプレートを追加できます。

最新バージョンの起動テンプレートを使用するように Auto Scaling グループを更新する

次のupdate-auto-scaling-groupコマンドは、指定された Auto Scaling グループを更新して、指定された起動テンプレートの最新バージョンを使用します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateId=lt-068f72b729example,Version='$Latest'
```

特定バージョンの起動テンプレートを使用するように Auto Scaling グループを更新する

次のupdate-auto-scaling-groupコマンドは、指定された Auto Scaling グループを更新して、指定された起動テンプレートの特定のバージョンを使用します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

起動テンプレートAMIIDsで の代わりに AWS Systems Manager パラメータを使用する

このセクションでは、Amazon マシンイメージ (AMI) ID を参照する AWS Systems Manager パラメータを指定する起動テンプレートを作成する方法について説明します。同じ に保存されているパ

ラメータ AWS アカウント、別の から共有されているパラメータ AWS アカウント、または によってAMI管理されているパブリックのパブリックパラメータを使用できます AWS。

Systems Manager パラメータを使用すると、AMIID が変更されるたびに新しい起動テンプレートや新しいバージョンの起動テンプレートを作成するAMIIDsことなく、新しい を使用するように Auto Scaling グループを更新できます。これらは、 AMIが最新のオペレーティングシステムまたはソフトウェア更新で更新された場合など、定期的に変更IDsされる可能性があります。

の一機能である Parameter Store を使用して、独自の Systems Manager パラメータを作成、更新、 AWS Systems Managerまたは削除できます。起動テンプレートで Systems Manager パラメータを使用する前に、作成する必要があります。開始するには、データ型 でパラメータを作成しaws:ec2:image、その値に の ID を入力しますAMI。AMI ID の形式は ami-*<identifier>*です。たとえば、 ですami-123example456。正しい AMI ID は、インスタンスタイプと Auto Scaling グループを起動 AWS リージョン するインスタンスタイプによって異なります。

AMI ID の有効なパラメータの作成の詳細については、「Systems Manager パラメータの作成」を参照してください。

のパラメータを指定する起動テンプレートを作成する AMI

のパラメータを指定する起動テンプレートを作成するにはAMI、次のいずれかの方法を使用します。

Console

AWS Systems Manager パラメータを使用して起動テンプレートを作成するには

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインで、[起動テンプレート]、[起動テンプレートの作成] の順に選択します。
3. [起動テンプレート名] に、起動テンプレートのわかりやすい名前を入力します。
4. アプリケーションイメージと OS イメージ (Amazon マシンイメージ) で、「その他の を参照するAMIs」を選択します。
5. 検索バーの右側にある矢印ボタンを選択したら、[カスタム値を指定 / Systems Manager パラメータ] を選択します。
6. [カスタム値または Systems Manager のパラメータを指定] ダイアログボックスで、次の手順を実行します。

    a. AMI ID または Systems Manager パラメータ文字列には、次のいずれかの形式を使用して Systems Manager パラメータ名を入力します。

パブリックパラメータを参照するには:

- **resolve:ssm:*public-parameter***

同じアカウントに保存されているパラメータを参照するには:

- **resolve:ssm:*parameter-name***
- **resolve:ssm:*parameter-name*:*version-number***
- **resolve:ssm:*parameter-name*:*label***

別の AWS アカウントから共有されたパラメータを参照するには:

- **resolve:ssm:*parameter-ARN***
- **resolve:ssm:*parameter-ARN*:*version-number***
- **resolve:ssm:*parameter-ARN*:*label***

b. [Save] を選択します。

7. 必要に応じて、他の起動テンプレート設定を実行してから [起動テンプレートを作成] を選択します。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。

AWS CLI

Systems Manager パラメータを指定する起動テンプレートを作成するには、次のいずれかのコマンド例を使用します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

例: AWS所有のパブリックパラメータを指定する起動テンプレートを作成する

`resolve:ssm:`*public-parameter* という構文を使用します。ただし、`resolve:ssm` は標準のプレフィクスで *public-parameter* はパブリックパラメータのパスおよび名前です。

この例では、起動テンプレートは AWSが提供するパブリックパラメータを使用して、プロファイル用に AWS リージョン 設定された AMIで最新の Amazon Linux 2 を使用してインスタンスを起動します。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
 --version-description version1 \
```

```
  --launch-template-data file://config.json
```

config.json の内容:

```
{
    "ImageId":"resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    "InstanceType":"t2.micro"
}
```

以下に、応答の例を示します。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-089c023a30example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2022-12-28T19:52:27.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

例: 同じアカウントに保存されているパラメータを指定する起動テンプレートを作成する

resolve:ssm:*parameter-name* という構文を使用します。ただし、resolve:ssm は標準のプレフィックスで *parameter-name* は Systems Manager のパラメータ名です。

次の例では、 という名前の既存の Systems Manager パラメータから AMI ID を取得する起動テンプレートを作成します*golden-ami*。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling \
  --launch-template-data file://config.json
```

config.json の内容:

```
{
    "ImageId":"resolve:ssm:golden-ami",
    "InstanceType":"t2.micro"
}
```

パラメータのデフォルトバージョンを指定しない場合、最新バージョンです。

次の例では、*golden-ami* パラメータの特定のバージョンを参照しています。この例では、*golden-ami* パラメータのバージョン *3* を使用していますが、有効なバージョン番号であればどれでも使用できます。

```
{
    "ImageId":"resolve:ssm:golden-ami:3",
    "InstanceType":"t2.micro"
}
```

次の同様の例では、*golden-ami* パラメータの特定のバージョンにマップするパラメータラベル *prod* を参照しています。

```
{
    "ImageId":"resolve:ssm:golden-ami:prod",
    "InstanceType":"t2.micro"
}
```

以下は出力例です。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b724example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2022-12-27T17:11:21.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

例: 別の から共有されるパラメータを指定する起動テンプレートを作成する AWS アカウント

次の構文を使用します: 。ここでresolve:ssm:*parameter-ARN*、 resolve:ssmは標準プレフィックスで、 *parameter-ARN*は Systems Manager パラメータARNの です。

次の例では、 ARNの を使用して、既存の Systems Manager パラメータから AMI ID を取得する起動テンプレートを作成します*arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter*。

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
 --version-description version1 \
  --launch-template-data file://config.json
```

config.json の内容:

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter",
    "InstanceType":"t2.micro"
}
```

パラメータのデフォルトバージョンを指定しない場合、最新バージョンです。

次の例では、*MyParameter* パラメータの特定のバージョンを参照しています。この例では、*MyParameter* パラメータのバージョン *3* を使用していますが、有効なバージョン番号であればどれでも使用できます。

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter:3",
    "InstanceType":"t2.micro"
}
```

次の同様の例では、*MyParameter* パラメータの特定のバージョンにマップするパラメータラベル *prod* を参照しています。

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter:prod",
    "InstanceType":"t2.micro"
}
```

以下に、応答の例を示します。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-00f93d4588example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2024-01-08T12:43:21.000Z",
```

```
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

起動テンプレートで Parameter Store からのパラメータを指定するには、指定するパラメータに対する ssm:GetParameters アクセス許可が必要です。起動テンプレートを使用するすべてのユーザーには、パラメータ値を検証するための ssm:GetParameters アクセス許可も必要です。詳細については、「 AWS Systems Manager ユーザーガイド」のIAM「ポリシーを使用した Systems Manager パラメータへのアクセスの制限」を参照してください。

起動テンプレートが正しい AMI ID を取得することを確認する

describe-launch-template-versions コマンドを使用し、 --resolve-aliasオプションを含めてパラメータを実際の AMI ID に解決します。

```
aws ec2 describe-launch-template-versions --launch-template-name my-template-for-auto-scaling \
   --versions $Default --resolve-alias
```

この例では、 の AMI ID を返しますImageId。この起動テンプレートを使用してインスタンスを起動すると、AMIID は に解決されますami-0ac394d6a3example。

```
{
    "LaunchTemplateVersions": [
        {
            "LaunchTemplateId": "lt-089c023a30example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreateTime": "2022-12-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersion": true,
            "LaunchTemplateData": {
                "ImageId": "ami-0ac394d6a3example",
                "InstanceType": "t2.micro",
            }
        }
    ]
}
```

関連リソース

起動テンプレートで Systems Manager パラメータを指定する方法の詳細については、「Amazon EC2 ユーザーガイド」のAMI「ID の代わりに Systems Manager パラメータを使用する」を参照してください。

Systems Manager パラメータの使用に関する詳細は、Systems Manager のドキュメントの次のリファレンス資料を参照してください。

- パラメータバージョンおよびラベルを作成するには、「パラメータバージョンの使用」および「パラメータラベルの操作」を参照してください。
- Amazon でサポートされているAMIパブリックパラメータを検索する方法については EC2、AMI「パブリックパラメータの呼び出し」を参照してください。
- 他の AWS アカウントと、または を介してパラメータを共有する方法については AWS Organizations、「共有パラメータの使用」を参照してください。
- パラメータが正常に作成されたかどうかのモニタリングについては、「Amazon マシンイメージのネイティブパラメータのサポートIDs」を参照してください。

制限

Systems Manager パラメータを使用する場合は、次の制限事項に注意してください。

- Amazon EC2 Auto Scaling は、パラメータAMIIDsとしての の指定のみをサポートしています。
- Systems Manager パラメータを指定する起動テンプレートを使用した、属性ベースのインスタンスタイプ選択による混合インスタンスグループの作成または更新はサポートされていません。
- Auto Scaling グループが Systems Manager パラメータを指定する起動テンプレートを使用している場合、希望する設定で、あるいはスキップマッチングを使用してインスタンスの更新を開始することはできません。
- Auto Scaling グループを作成または更新する呼び出しごとに、Amazon EC2 Auto Scaling は起動テンプレートの Systems Manager パラメータを解決します。アドバンストパラメータまたはより高いスループット制限を使用している場合、Parameter Store への頻繁な呼び出し (`GetParameters`オペレーション) により、Parameter Store APIとのやり取りごとに料金が発生するため、Systems Manager のコストが増加する可能性があります。詳細については、AWS Systems Manager の料金を参照してください。

Auto Scaling 起動設定

> **Important**
>
> 起動設定に関する情報は、起動設定から起動テンプレートにまだ移行していないお客様向けに提供しています。Auto Scaling グループを移行してテンプレートを起動する方法については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

起動設定は、Auto Scaling グループがインスタンスを起動するために使用するEC2インスタンス設定テンプレートです。起動設定を作成する場合は、インスタンスの情報を指定します。Amazon マシンイメージの ID (AMI) 、インスタンスタイプ、キーペア、1 つ以上のセキュリティグループ、ブロックデバイスマッピングを含めます。以前にEC2インスタンスを起動したことがある場合は、インスタンスを起動するために同じ情報を指定しました。

複数の Auto Scaling グループについて起動設定を指定できます。ただし、1 つの Auto Scaling グループに指定できる起動設定は 1 つだけであり、グループを作成した後で起動設定を変更することはできません。Auto Scaling グループの起動設定を変更するには、起動設定を作成し、それを使用して Auto Scaling グループを更新する必要があります。

内容

「起動設定を作成する」

> **Important**
>
> 2022 年 12 月 31 日以降にリリースされた新しい Amazon EC2インスタンスタイプ`CreateLaunchConfiguration`で を呼び出すことはできません。 また、2023 年 6 月 1 日以降に作成された新しいアカウントには、コンソールから新しい起動設定を作成することはできません。2024 年 10 月 1 日以降、新しいアカウントは、コンソール、、API、CLIおよび を使用して新しい起動設定を作成できなくなります CloudFormation。起動テンプレートに移行すると、現在または将来的に新しい起動設定を作成する必要がなくなります。Auto

Scaling グループを移行してテンプレートを起動する方法については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

このトピックでは、起動設定を作成する方法について説明します。

起動設定を作成すると、以後、その設定を変更することはできません。代わりに、新しい起動設定を作成する必要があります。

新しい起動設定を既存の Auto Scaling グループに関連付けるには、「Auto Scaling グループの起動設定を変更する」を参照してください。新しい Auto Scaling グループを作成するには、「起動設定を使用して Auto Scaling グループを作成する」を参照してください。

内容

「起動設定を作成する」

起動設定を作成するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. 上部のナビゲーションバーで、 AWS リージョンを選択します。
3. 左のナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。
4. ページの上部付近にある [起動設定] を選択します。確認を求めるプロンプトが表示されたら、[起動設定を表示] を選択して、[起動設定] ページを表示することを確認します。
5. [Create launch configuration (起動設定の作成)] を選択して、起動設定の名前を入力します。
6. Amazon マシンイメージ (AMI) で、 を選択しますAMI。特定の を見つけるにはAMI、適切な を見つけAMI、その ID をメモし、検索条件として ID を入力します。

   Amazon Linux 2 の ID を取得するにはAMI：

   a. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
   b. 左側のナビゲーションペインの [インスタンス] で、[インスタンス] を選択し、[インスタンスを起動] を選択します。

c. Amazon マシンイメージの選択ページのクイックスタートタブで、Amazon Linux 2 () AMI の横にある の ID を書き留めます。 AMI HVM

7. [インスタンスタイプのタイプ] で、インスタンスのハードウェア設定を選択します。

8. [Additional configuration (追加設定)] の下にある、次のフィールドに注意してください:

a. (オプション) [Purchasing option (購入オプション)] で、[Request Spot Instances (スポットインスタンスのリクエスト)] を選択し、オンデマンド価格を上額とするスポット料金でスポットインスタンスをリクエストできます。オプションで、スポットインスタンスのインスタンス時間ごとに上限価格を指定できます。

> **Note**
>
> スポットインスタンスは、アプリケーションを実行する時間に柔軟性がある場合や、アプリケーションを中断できる場合に、オンデマンドインスタンスと比べて費用効率の高い選択肢です。詳細については、「耐障害性に優れた柔軟なアプリケーションのためにスポットインスタンスをリクエストする」を参照してください。

b. (オプション) IAMインスタンスプロファイルで、インスタンスに関連付けるロールを選択します。詳細については、「Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール」を参照してください。

c. (オプション) モニタリング で、詳細モニタリングを有効に CloudWatch して、インスタンスが 1 分間隔でメトリクスデータを Amazon に発行できるようにするかどうかを選択します。別途 料金がかかります。詳細については、「Auto Scaling インスタンスのモニタリングを設定する」を参照してください。

d. (オプション) [Advanced details (詳細情報)]、[User data (ユーザーデータ)] で、起動中にインスタンスを設定するユーザーデータ、またはインスタンスの起動後に設定スクリプトを実行するユーザーデータを指定できます。

e. (オプション) [Advanced details (詳細情報)]、[IP address type (IP アドレスタイプ)]で、パブリック IP アドレスをグループのインスタンスに割り当てるかどうか選択します。値を設定しない場合、デフォルトでは、インスタンスが起動されるサブネットの自動割り当てパブリック IP 設定が使用されます。

9. (オプション) [Storage (volumes) (ストレージボリューム)] で、追加のストレージが必要ない場合、このセクションをスキップできます。それ以外の場合は、 で指定されたボリュームに加えてインスタンスにアタッチするボリュームを指定するにはAMI、新しいボリュームを追加を選択します。次に、デバイス、スナップショット、サイズ、ボリュームタイプ、、スループットIOPS、終了時に削除、および暗号化に必要なオプションと関連する値を選択します。

10. [Security groups (セキュリティグループ)] で、グループのインスタンスに関連付けるセキュリティグループを作成または選択します。新しいセキュリティグループの作成オプションを選択したままにすると、Linux を実行している Amazon EC2インスタンスに対してデフォルトのSSHルールが設定されます。Windows を実行している Amazon EC2インスタンスには、デフォルトのRDPルールが設定されています。
11. [Key pair (login) (キーペア (ログイン)] で、[Key pair options (キーペアのオプション)] の下にあるオプションを選択します。

    Amazon EC2インスタンスのキーペアをすでに設定している場合は、ここで選択できます。

    Amazon EC2インスタンスのキーペアがまだない場合は、新しいキーペアを作成して認識可能な名前を付けます。[Download key pair (キーペアのダウンロード)] を選択し、コンピュータにダウンロードします。

    > **Important**
    >
    > インスタンスに接続する必要がある場合は、[Proceed without a key pair (キーペアなしで続行する)] を選択しないでください。

12. 確認チェックボックスをオンにし、[Create launch configuration (起動設定の作成)] を選択します。

既存の起動設定から起動設定を作成するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. 上部のナビゲーションバーで、 AWS リージョンを選択します。
3. 左のナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。
4. ページの上部付近にある [起動設定] を選択します。確認を求めるプロンプトが表示されたら、[起動設定を表示] を選択して、[起動設定] ページを表示することを確認します。
5. 起動設定を選択し、[Actions]、[Copy launch configuration] の順に選択します。これにより、新しい起動設定は元の設定と同じオプションで設定されますが、名前に「コピー」が追加されます。
6. [Copy Launch Configuration] ページで、必要に応じて設定オプションを編集し、[Create launch configuration] を選択します。

コマンドラインを使用して起動設定を作成するには

以下のコマンドのいずれかを使用できます。

- create-launch-configuration (AWS CLI)
- 新規 -ASLaunchConfiguration (AWS Tools for Windows PowerShell)

インスタンスメタデータオプションの設定

Amazon EC2 Auto Scaling は、起動設定でのインスタンスメタデータサービス (IMDS) の設定をサポートしています。これにより、起動設定を使用して、インスタンスメタデータをリクエストするためのセッション指向のメソッドであるインスタンスメタデータサービスバージョン 2 (IMDSv2) を要求するように EC2 Auto Scaling グループの Amazon インスタンスを設定するオプションが提供されます。の利点の詳細については、IMDSv2EC2「 インスタンスメタデータサービスに多層防御を追加するための機能強化に関する AWS ブログ」のこの記事を参照してください。

IMDSv2 と IMDSv1 (デフォルト) の両方をサポートするIMDSか、 の使用を要求するように を設定できますIMDSv2。 AWS CLI または のいずれかを使用して SDKsを設定する場合はIMDS、最新バージョンの AWS CLI または を使用して の使用SDKを要求する必要がありますIMDSv2。

起動設定は次の設定を指定できます。

- インスタンスメタデータをリクエストするときに IMDSv2 の使用を要求する
- PUT レスポンスのホップ制限を指定する
- インスタンスメタデータへのアクセスを無効にする

インスタンスメタデータサービスの設定の詳細については、「Amazon EC2ユーザーガイド」の「インスタンスメタデータサービスの設定」を参照してください。

起動設定でIMDSオプションを設定するには、次の手順に従います。起動設定を作成したら、それを Auto Scaling グループに関連付けることができます。起動設定を既存の Auto Scaling グループに関連付けると、既存の起動設定は Auto Scaling グループとの関連付けが解除され、新しい起動設定で指定したIMDSオプションを使用するには、既存のインスタンスの置き換えが必要になります。詳細については、「Auto Scaling グループの起動設定を変更する」を参照してください。

起動設定IMDSで を設定するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。

2. 上部のナビゲーションバーで、 AWS リージョンを選択します。
3. 左のナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。
4. ページの上部付近にある [起動設定] を選択します。確認を求めるプロンプトが表示されたら、[起動設定を表示] を選択して、[起動設定] ページを表示することを確認します。
5. [Create launch configuration (起動設定の作成)] を選択し、通常の方法で起動設定を作成します。Amazon マシンイメージ (AMI) の ID、インスタンスタイプ、オプションでキーペア、1 つ以上のセキュリティグループ、インスタンスの追加EBSボリュームまたはインスタンスストアボリュームを含めます。
6. この起動設定に関連付けられているすべてのインスタンスのインスタンスメタデータオプションを設定するには、[Additional configuration (追加設定)] の下にある [Advanced details (詳細情報)] で次の作業を行います。

   a. メタデータアクセス可能の場合は、インスタンスメタデータサービスのHTTPエンドポイントへのアクセスを有効または無効にするかどうかを選択します。デフォルトでは、HTTPエンドポイントは有効になっています。エンドポイントを無効にすると、インスタンスメタデータへのアクセスはオフになります。HTTP エンドポイントが有効になっているIMDSv2場合にのみ、 が要求する条件を指定できます。

   b. メタデータバージョンでは、インスタンスメタデータをリクエストするときに、インスタンスメタデータサービスバージョン 2 (IMDSv2) の使用を要求できます。値を指定しない場合、デフォルトは IMDSv1と の両方をサポートしますIMDSv2。

   c. [Metadata token response hop limit] で、メタデータトークンに許容されるネットワークホップ数を設定できます。値を指定していない場合、デフォルトは 1 です。
7. 完了したら、[Create launch configuration (起動設定の作成)] を選択します。

を使用して起動設定IMDSv2で の使用を要求するには AWS CLI

を `--metadata-options`に設定して、次のcreate-launch-configurationコマンドを使用しますHttpTokens=required。HttpTokens の値を指定する場合は、HttpEndpoint も有効にする必要があります。メタデータの取得リクエストでは、セキュアトークンヘッダーが必須に設定されているため、インスタンスメタデータをリクエストIMDSv2するときに の使用を要求するようにインスタンスをオプトインします。

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imdsv2 \
  --image-id ami-01e24be29428c15b2 \
  --instance-type t2.micro \
```

```
    ...
  --metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

インスタンスメタデータへのアクセスを無効にするには

インスタンスメタデータへのアクセスを無効にするには、次のcreate-launch-configurationコマンドを使用します。modify-instance-metadata-options コマンドを使用して、後でアクセスをオンに戻すことができます。

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imds-disabled \
  --image-id ami-01e24be29428c15b2 \
  --instance-type t2.micro \
    ...
  --metadata-options "HttpEndpoint=disabled"
```

EC2 インスタンスを使用して起動設定を作成する

実行中のEC2インスタンスの属性を使用して起動設定を作成するオプションもあります。

起動設定を最初から作成することと、既存のEC2インスタンスから起動設定を作成することには違いがあります。起動設定をゼロから作成するときは、イメージ ID、インスタンスタイプ、オプションリソース (ストレージデバイスなど) 、オプション設定 (モニタリングなど) を指定します。実行中のインスタンスから起動設定を作成すると、Amazon EC2 Auto Scaling は指定されたインスタンスから起動設定の属性を取得します。属性は、AMIインスタンスが起動された のブロックデバイスマッピングからも取得され、起動後に追加された追加のブロックデバイスはすべて無視されます。

実行中のインスタンスを使用して起動設定を作成する場合、同じリクエストの一部として指定することで、次の属性を上書きできます。AMI、ブロックデバイス、キーペア、インスタンスプロファイル、インスタンスタイプ、カーネル、インスタンスモニタリング、プレイスメントテナンシー、ラムディスク、セキュリティグループ、スポット (最大) 料金、ユーザーデータ、インスタンスにパブリック IP アドレスがあるかどうか、インスタンスが EBS最適化されているかどうか。

> **Note**
>
> 指定されたインスタンスに、起動設定で現在サポートされていないプロパティがある場合、Auto Scaling グループによって起動されたインスタンスは、元のEC2インスタンスと同じではない可能性があります。

> **Important**
>
> 指定されたインスタンスを起動するAMIために使用される がまだ存在している必要があります。

トピック

- EC2 インスタンスから起動設定を作成する (AWS CLI)
- インスタンスから起動設定を作成し、ブロックデバイスをオーバーライドする (AWS CLI)
- 起動設定を作成し、インスタンスタイプをオーバーライドする (AWS CLI)

EC2 インスタンスから起動設定を作成する (AWS CLI)

インスタンスから、そのインスタンスと同じ属性を使用して起動設定を作成するには、次の create-launch-configuration コマンドを使用します。起動後に追加されたブロックデバイスはすべて無視されます。

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance --instance-id i-a8e09d9c
```

次のdescribe-launch-configurationsコマンドを使用して起動設定を記述し、その属性がインスタンスの属性と一致することを確認できます。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance
```

以下に、応答の例を示します。

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-05355a6c",
```

```
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "t1.micro",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

インスタンスから起動設定を作成し、ブロックデバイスをオーバーライドする (AWS CLI)

デフォルトでは、Amazon EC2 Auto Scaling は指定したEC2インスタンスの属性を使用して起動設定を作成します。ただし、ブロックデバイスは、インスタンスではなく、インスタンスの起動AMIに使用される から取得されます。起動設定にブロックデバイスを追加するには、起動設定のブロックデバイスマッピングをオーバーライドします。

次のcreate-launch-configurationコマンドを使用して、EC2インスタンスを使用するが、カスタムブロックデバイスマッピングを使用して起動設定を作成します。

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-bdm --instance-id i-a8e09d9c \
  --block-device-mappings "[{\"DeviceName\":\"/dev/sda1\",\"Ebs\":{\"SnapshotId\":\"snap-3decf207\"}},{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":\"snap-eed6ac86\"}}]"
```

次のdescribe-launch-configurationsコマンドを使用して起動設定を記述し、カスタムブロックデバイスマッピングを使用していることを確認します。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-bdm
```

以下の応答例は、起動設定の詳細を表しています。

```
{
```

```
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-c49c0dac",
            "CreatedTime": "2015-01-07T14:51:26.065Z",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "SnapshotId": "snap-3decf207"
                    }
                },
                {
                    "DeviceName": "/dev/sdf",
                    "Ebs": {
                        "SnapshotId": "snap-eed6ac86"
                    }
                }
            ],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8637d3e3"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-bdm",
            "KernelId": null,
            "RamdiskId": null,
            "InstanceType": "t1.micro",
            "AssociatePublicIpAddress": true
        }
    ]
}
```


起動設定を作成し、インスタンスタイプをオーバーライドする (AWS CLI)

デフォルトでは、Amazon EC2 Auto Scaling は指定したEC2インスタンスの属性を使用して起動設定を作成します。要件によっては、インスタンスの属性をオーバーライドし、必要な値を使用する場合があります。たとえば、インスタンスタイプをオーバーライドできます。

次のcreate-launch-configurationコマンドを使用して、EC2インスタンスを使用して起動設定を作成しますが、インスタンスとは異なるインスタンスタイプ (などt2.medium) を使用します (など) t2.micro。

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-changetype \
  --instance-id i-a8e09d9c --instance-type t2.medium
```

次のdescribe-launch-configurationsコマンドを使用して起動設定を記述し、インスタンスタイプが上書きされたことを確認します。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-changetype
```

以下の応答例は、起動設定の詳細を表しています。

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-05355a6c",
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-changetype",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "t2.medium",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

Auto Scaling グループの起動設定を変更する

> **Important**
>
> 起動設定に関する情報は、起動設定から起動テンプレートにまだ移行していないお客様向けに提供しています。Auto Scaling グループを移行してテンプレートを起動する方法については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

このトピックでは、別の起動設定を Auto Scaling グループに関連付ける方法について説明します。

起動設定を変更すると、以後、新しいインスタンスは新しい設定オプションを使用して起動されますが、既存のインスタンスは影響を受けません。詳細については、「Auto Scaling インスタンスの更新」を参照してください。

Auto Scaling グループの起動設定を置き換えるには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. 左のナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. [Details] (詳細) タブで、[Launch configuration] (起動設定)、[Edit] (編集) の順に選択します。
5. [起動設定] で、起動設定を選択します。
6. 完了したら、[更新] を選択します。

コマンドラインを使用して Auto Scaling グループの起動設定を変更するには

以下のコマンドのいずれかを使用できます。

- update-auto-scaling-group (AWS CLI)
- 更新 -ASAutoScalingGroup (AWS Tools for Windows PowerShell)

「Auto Scaling グループ」

> **Note**
>
> Auto Scaling グループを初めて使用する場合は、チュートリアル「最初の Auto Scaling グループを作成する」の手順を実行して使用を開始し、グループ内のインスタンスが終了したときに Auto Scaling グループがどのように応答するかを確認します。

Auto Scaling グループには、自動スケーリングと管理の目的で論理グループとして扱われるEC2 インスタンスのコレクションが含まれています。Auto Scaling グループでは、ヘルスチェックの置き換えやスケーリングポリシーなどの Amazon EC2 Auto Scaling 機能を使用することもできます。Auto Scaling グループ内のインスタンス数の維持と自動スケーリングの両方が、Amazon EC2 Auto Scaling サービスの中核的な機能です。

Auto Scaling グループのサイズは、希望するキャパシティーとして設定したインスタンスの数によって異なります。手動でまたは自動スケーリングを使用して、需要に合わせてサイズを調整できます。

Auto Scaling グループは起動時に、希望するキャパシティーに対応するために十分な数のインスタンスを起動します。また、グループのインスタンスに対して定期的にヘルスチェックを実行することで、このインスタンス数を維持します。Auto Scaling グループは、インスタンスに異常が発生した場合でも、一定数のインスタンスを維持し続けます。インスタンスに異常が生じた場合、グループは異常のあるインスタンスを終了し、そのインスタンスを置き換える別のインスタンスを起動します。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

スケーリングポリシーを使用して、状況の変化に合わせてグループのインスタンスの数を動的に増減することができます。スケーリングポリシーが有効になっている場合、Auto Scaling グループは、指定された最小キャパシティ値と最大キャパシティ値の間で、グループの希望するキャパシティを調整し、必要に応じてインスタンスを起動または終了します。スケーリングはスケジュールに従って行うこともできます。詳細については、「スケーリング方法を選択する」を参照してください。

Auto Scaling グループを作成する際に、オンデマンドインスタンス、スポットインスタンス、またはその両方を起動するかどうかを選択できます。Auto Scaling グループに複数の購入オプションを指定できるのは、launch template。 詳細については、「」を参照してください複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ。

スポットインスタンスでは、オンデマンド価格と比べて大幅な割引価格で、使われていない EC2 容量を購入できます。詳細については、「Amazon EC2 スポットインスタンス」を参照してください。スポットインスタンスとオンデマンドインスタンスの主な違いは次のとおりです。

- スポットインスタンスの料金は需要に応じて変化する
- Amazon は、スポットインスタンスの可用性または料金が変更されたときに、個々のスポットインスタンスを終了EC2できます。

スポットインスタンスが削除されると、Auto Scaling グループはグループの希望するキャパシティを維持するために、代わりのインスタンスを起動しようとします。

インスタンスの起動時に複数のアベイラビリティーゾーンを指定した場合は、希望するキャパシティがこれらのアベイラビリティーゾーンに分散されます。スケーリングアクションが発生すると、Amazon EC2 Auto Scaling は指定したすべてのアベイラビリティーゾーンのバランスを自動的に維持します。

内容

起動テンプレートを使用して Auto Scaling グループを作成する

起動テンプレートを作成した場合は、EC2インスタンスの設定テンプレートとして起動テンプレートを使用する Auto Scaling グループを作成できます。起動テンプレートは、インスタンスの AMI ID、

インスタンスタイプ、キーペア、セキュリティグループ、ブロックデバイスマッピングなどの情報を指定します。起動テンプレートの作成の詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。

Auto Scaling グループを作成するための十分な許可が必要です。また、Amazon EC2 Auto Scaling がユーザーに代わってアクションを実行するために使用するサービスにリンクされたロールがまだ存在しない場合は、作成するための十分なアクセス許可が必要です。管理者がアクセス許可を付与するためのリファレンスとして使用できるIAMポリシーの例については、アイデンティティベースのポリシーの例「」および「」を参照してくださいAuto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する。

内容

- 起動テンプレートを使用して Auto Scaling グループを作成する
- Amazon EC2 起動ウィザードを使用して Auto Scaling グループを作成する
- 複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ

起動テンプレートを使用して Auto Scaling グループを作成する

Auto Scaling グループを作成するときは、Amazon EC2 インスタンス、インスタンスのアベイラビリティーゾーンと VPC サブネット、希望する容量、および最小容量と最大容量の制限を設定するために必要な情報を指定する必要があります。

Auto Scaling グループによって起動される Amazon EC2 インスタンスを設定するには、起動テンプレートまたは起動設定を指定します。次の手順では、起動テンプレートを使用して Auto Scaling グループを作成する方法を説明します。

前提条件

- 起動テンプレートを作成しておく必要があります。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。

起動テンプレートを使用して Auto Scaling グループを作成するには (コンソール)

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. 画面上部のナビゲーションバーで、起動テンプレートの作成時に使用した AWS リージョン ものと同じ を選択します。

3. [Auto Scaling グループの作成] を選択します。
4. Choose launch template or configuration のページで、以下を実行します。
   a. Auto Scaling グループ名に　Auto Scaling グループの名前を入力します。
   b. [起動テンプレート] で、既存の起動テンプレートを選択します。
   c. [起動テンプレートのバージョン] で、スケールアウト時に Auto Scaling グループで使用する起動テンプレートのバージョン (デフォルト、最新、または特定のバージョン) を選択します。
   d. 起動テンプレートが、使用する予定のすべてのオプションをサポートしていることを確認し、[次へ] を選択します。
5. インスタンス起動オプションの選択ページで、複数のインスタンスタイプを使用していない場合は、インスタンスタイプ要件セクションをスキップして、起動テンプレートで指定された EC2 インスタンスタイプを使用できます。

   複数のインスタンスタイプを使用する場合は、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。
6. Network の VPC で、VPC を選択します。Auto Scaling グループは、起動テンプレートで指定したセキュリティグループと同じ VPC で作成する必要があります。
7. Availability Zones とサブネットで、指定した VPC で 1 つ以上のサブネットを選択します。複数のアベイラビリティーゾーンのサブネットを使用することで、高可用性を得られます。詳細については、「VPC サブネットを選択する際の考慮事項」を参照してください。
8. アベイラビリティーゾーンのディストリビューションでは、ディストリビューション戦略を選択します。詳細については、「Auto Scaling グループのアベイラビリティーゾーンディストリビューション」を参照してください。
9. 指定したインスタンスタイプを使用して起動テンプレートを作成した場合は、次の手順に進み、起動テンプレートのインスタンスタイプを使用する Auto Scaling グループを作成できます。

   または、起動テンプレートでインスタンスタイプが指定されていない場合や、Auto Scaling に複数のインスタンスタイプを使用する場合は、[Override launch template] (起動テンプレートを上書きする) オプションを選択できます。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。
10. [Next] (次へ) を選択して、次のステップに進みます。

    または、残りはデフォルトのままにして、[Skip to Review (確認をスキップ)] を選択できます。
11. (オプション) 他のサービスとの統合ページで、次のオプションを設定し、次へを選択します。

a. （オプション) Amazon Application Recovery Controller (ARC) ゾーンシフトで、ゾーンシフトを有効にするを選択します。

b. ヘルスチェックの動作で、異常を無視または異常を置き換えを選択します。詳細については、「Auto Scaling グループのゾーンシフトの仕組み」を参照してください。

c. （オプション) ヘルスチェック、その他のヘルスチェックタイプで、Amazon EBS ヘルスチェックを有効にするを選択します。詳細については、「ヘルスチェックを使用して、Amazon EBS ボリュームに障害がある Auto Scaling インスタンスをモニタリングする」を参照してください。

d. (オプション) [ヘルスチェックの猶予期間] に秒単位で時間を入力します。この時間は、Amazon EC2 Auto Scaling が `InService` 状態になった後にインスタンスのヘルスステータスをチェックするまでの待機時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

e. 「追加設定」「モニタリング」で、 CloudWatch グループメトリクス収集を有効にするかどうかを選択します。これらのメトリクスは、終了インスタンス数や保留中のインスタンスの数など、潜在的な問題の指標となる測定値を提供します。詳細については、「Auto Scaling グループとインスタンスの CloudWatch メトリクスを監視する」を参照してください。

f. [デフォルトのインスタンスのウォームアップを有効にする] でこのオプションを選択し、アプリケーションのウォームアップ時間を選択します。スケーリングポリシーを持つ Auto Scaling グループを作成する場合、デフォルトのインスタンスウォームアップ機能により、動的スケーリングに使用される Amazon CloudWatch メトリクスが改善されます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

12. (オプション) [Configure group size and scaling policies (グループサイズとスケーリングポリシーの設定)] ページで、次のオプションを設定し、[次へ] を選択します。

a. [グループサイズ] の [希望する容量] に、起動するインスタンスの初期数を入力します。

b. [スケーリング] セクションの [スケーリング制限] で、[希望する容量] の新しい値が [最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に希望する新しい容量の値に引き上げられます。これらの制限は、必要に応じて変更できます。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。

c. [自動スケーリング] で、ターゲット追跡スケーリングポリシーを作成するかどうかを選択します。このポリシーは、Auto Scaling グループの作成後に作成することもできます。

[ターゲット追跡スケーリングポリシー] を選択した場合は、「ターゲット追跡スケーリングポリシーを作成する」の指示に従ってポリシーを作成してください。

d. [インスタンスメンテナンスポリシー] で、インスタンスメンテナンスポリシーを作成するかどうかを選択します。このポリシーは、Auto Scaling グループの作成後に作成することもできます。ポリシーを作成するには、「インスタンスメンテナンスポリシーを設定する」の手順を実行してください。

e. [Instance scale-in protection (インスタンスのスケールイン保護)] で、インスタンスのスケールイン保護を有効にするかどうかを選択します。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

13. (オプション) 通知を受け取るには、[通知の追加] を選択し、通知を設定してから [次へ] を選択します。詳細については、「Amazon EC2 Auto Scaling の Amazon SNS 通知オプション」を参照してください。
14. (オプション) タグを追加するには、[タグの追加] を選択し、各タグのタグキーと値を指定し、[次へ] を選択します。詳細については、「Auto Scaling グループとインスタンスにタグを付ける」を参照してください。
15. [Review (レビュー)]ページで、[Create Auto Scaling group (Auto Scaling グループを作成)] を選択します。

コマンドラインを使用して Auto Scaling グループを作成するには

以下のコマンドのいずれかを使用できます。

- create-auto-scaling-group (AWS CLI)
- New-ASAutoScalingGroup (AWS Tools for Windows PowerShell)

Amazon EC2 起動ウィザードを使用して Auto Scaling グループを作成する

次の手順は、Amazon EC2 コンソールでインスタンス起動ウィザードを使用して Auto Scaling グループを作成する方法を示しています。このオプションでは、特定の設定が詳細に入力されている起動テンプレートが、[Launch instance] (インスタンスの作成) ウィザードに自動的に入力されます。

> **Note**
>
> ウィザードは、Auto Scaling グループに指定したインスタンス数を入力しません。起動テンプレートには Amazon マシンイメージ (AMI) ID とインスタンスタイプのみを入力しま

す。[Create Auto Scaling group] (Auto Scaling グループの作成) ウィザードで、起動するインスタンスの数を指定します。
AMI は、インスタンスの設定に必要な情報を提供します。同じ設定の複数のインスタンスが必要な場合、1 つの AMI から複数のインスタンスを起動できます。Auto Scaling グループに属するインスタンスを再起動した場合にインスタンスが終了しないように、アプリケーションがすでにインストールされているカスタム AMI を使用することをお勧めします。Amazon EC2 Auto Scaling でカスタム AMI を使用するには、まずカスタマイズされたインスタンスから AMI を作成し、次に AMI を使用して Auto Scaling グループの起動テンプレートを作成する必要があります。

前提条件

- Auto Scaling グループを作成する AWS リージョン のと同じ にカスタム AMI を作成しておく必要があります。詳細については、「Amazon AMI ユーザーガイド」の「単語の作成」を参照してください。 EC2

カスタム AMI をテンプレートとして使用する

このセクションでは、Amazon EC2 起動ウィザードを使用して、起動テンプレートにカスタム AMI を自動的に入力します。また、起動テンプレートを最初から設定する場合や、起動テンプレートに設定できるパラメータの詳細については、起動テンプレートを作成する (コンソール) を参照してください。

カスタム AMI をテンプレートとして使用するには

1. EC2 で Amazon https://console.aws.amazon.com/ec2/ コンソールを開きます。
2. 画面上部のナビゲーションバーに、現在の AWS リージョン が表示されます。Auto Scaling グループを起動するリージョンを選択します。
3. ナビゲーションペインで、[インスタンス] を選択します。
4. [Launch instance] (インスタンスを起動) を選択してから、以下を実行します。

    a. [Name and tags] (名前とタグ) では、[Name] (名前) を空のままにしておきます。名前は、起動テンプレートの作成に使用されるデータの 1 部ではありません。

    b. アプリケーションイメージと OS イメージ (Amazon マシンイメージ) で、より多くの AMIsを参照 を選択して、AMIカタログ全体を参照します。

    c. My AMIs を選択し、作成したAMIを見つけて、Select を選択します。

d. [Instance type] (インスタンスタイプ) でインスタンスタイプを選択します。

> **Note**
>
> AMI を作成したときと同じインスタンスタイプを選択するか、より強力なインスタンスタイプを選択します。

e. 画面の右側にある [Summary] (概要) で、[Number of instances] (インスタンス数) に任意の数を入力します。ここに入力する数値は重要ではありません。起動するインスタンスの数は、Auto Scaling グループの作成時に指定します。

インスタンス数 フィールドの下に、複数のインスタンスを起動するときは、EC2 Auto Scaling を検討してくださいというメッセージが表示されます。

f. EC2 Auto Scaling ハイパーリンクを検討するテキストを選択します。

g. Auto Scaling グループを起動するの確認ダイアログで、「起動テンプレートの作成」ページに進み、インスタンス起動ウィザードで選択した AMI とインスタンスタイプが既に入力されている「続行」を選択します。

[Continue] (続行) を選択すると、[Create launch template] (起動テンプレートの作成) ページが開きます。この手順に従って、起動テンプレートの作成を完了します。

起動テンプレートを作成するには

1. [Launch template name and description] (起動テンプレート名と説明) で、新しい起動テンプレートの名前と説明を入力します。
2. (オプション) キーペア (ログイン) のキーペア名で、SSH などを使用してインスタンスに接続するときに使用する、以前に作成したキーペアの名前を選択します。
3. (オプション) [Network settings] (セットワーク設定) の [Security groups] (セキュリティグループ) では、以前作成したセキュリティグループを 1 つ、または複数選択します。
4. (オプション) [Configure storage] (ストレージを設定) で、ストレージ設定を更新します。デフォルトのストレージ設定は、AMI とインスタンスタイプによって決まります。
5. 起動テンプレートの設定が終わったら、[Create launch template] (起動テンプレートの作成) を選択します。
6. 確認ページで、[Auto Scaling グループの作成] を選択します。

Auto Scaling グループを作成する

> **Note**
>
> このトピックの残りの部分では、Auto Scaling グループ作成の基本的な手順について説明します。Auto Scaling グループに設定できるパラメータの詳細については、「起動テンプレートを使用して Auto Scaling グループを作成する」を参照してください。

[Create Auto Scaling group] (Auto Scaling グループの作成) を選択すると、[Create Auto Scaling group] (Auto Scaling グループの作成) ウィザードが開きます。Auto Scaling グループを作成するには、次の手順を実行します。

Auto Scaling グループを作成する

1. [Choose launch template or configuration (起動テンプレートまたは設定の選択)] ページで、Auto Scaling グループの名前を入力します。
2. 作成した起動テンプレートは、既に選択されています。

   [起動テンプレートのバージョン] で、スケールアウト時に Auto Scaling グループで使用する起動テンプレートのバージョン (デフォルト、最新、または特定のバージョン) を選択します。
3. [Next] (次へ) を選択して、次のステップに進みます。
4. インスタンス起動オプションの選択ページで、複数のインスタンスタイプを使用していない場合は、インスタンスタイプ要件セクションをスキップして、起動テンプレートで指定された EC2 インスタンスタイプを使用できます。

   複数のインスタンスタイプを使用する場合は、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。
5. Network の VPC で、VPC を選択します。Auto Scaling グループは、起動テンプレートで指定したセキュリティグループと同じ VPC で作成する必要があります。

   > **Tip**
   >
   > 起動テンプレートでセキュリティグループを指定しなかった場合、インスタンスは指定した VPC のデフォルトのセキュリティグループで起動されます。デフォルトでは、このセキュリティグループは外部ネットワークからのインバウンドトラフィックを許可しません。

6. Availability Zones とサブネットで、指定した VPC で 1 つ以上のサブネットを選択します。
7. アベイラビリティーゾーンのディストリビューションでは、ディストリビューション戦略を選択します。詳細については、「Auto Scaling グループのアベイラビリティーゾーンディストリビューション」を参照してください。
8. [Next] (次へ) を2 回選択すると、[Configure group size and scaling policies] (グループサイズとスケーリングポリシーの設定) ページへ移動します。
9. [グループサイズ] で、[希望する容量] (Auto Scaling グループの作成直後に起動するインスタンスの初期数) を定義します。
10. [スケーリング] セクションの [スケーリング制限] で、[希望する容量] の新しい値が [最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に希望する新しい容量の値に引き上げられます。これらの制限は、必要に応じて変更できます。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。
11. [Skip to review] を選択します。
12. [Review (レビュー)]ページで、[Create Auto Scaling group (Auto Scaling グループを作成)] を選択します。

次のステップ

アクティビティ履歴を表示することによって、Auto Scaling グループが正しく作成されたことを確認できます。[Activity] (アクティビティ) タブの [Activity history] (アクティビティ履歴) では、[Status] (ステータス) 列に、Auto Scaling グループがインスタンスを正常に起動したかどうかが表示されます。インスタンスの起動に失敗したり、起動してもすぐに終了したりする場合は、次のトピックで、考えられる原因と解決方法を参照してください。

- Amazon EC2 Auto Scaling のトラブルシューティング: EC2インスタンスの起動失敗
- Amazon EC2 Auto Scaling のトラブルシューティング: AMI問題
- Amazon EC2 Auto Scaling の異常なインスタンスをトラブルシューティングする

必要に応じて、Auto Scaling グループと同じリージョンに、ロードバランサーをアタッチできるようになりました。詳細については、「Elastic Load Balancing を使用して Auto Scaling グループ内の受信アプリケーショントラフィックを分散する 」を参照してください。

複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ

1 つの Auto Scaling グループ内で、オンデマンドインスタンスとスポットインスタンスのフリートを起動してオートスケールできます。スポットインスタンスの使用で割引を受けるだけでなく、リザーブドインスタンスまたは Savings Plans を使用して、通常のオンデマンドインスタンスの料金に対する割引の適用を受けることができます。これらの要因は、EC2インスタンスのコスト削減を最適化し、アプリケーションに必要なスケールとパフォーマンスを実現するのに役立ちます。

スポットインスタンスは、EC2オンデマンド料金と比較して大幅な割引で利用できる予備の容量です。スポットインスタンス は、アプリケーションを実行する時間に柔軟性がある場合や、アプリケーションを中断できる場合に、費用効率の高い選択肢です。これらは、フォールトトレラントで柔軟性があるさまざまなアプリケーションに使用できます。例としては、ステートレスウェブサーバー、APIエンドポイント、ビッグデータおよび分析アプリケーション、コンテナ化されたワークロード、CI/CD pipelines, high performance and high throughput computing (HPC/HTC) 、レンダリングワークロード、その他の柔軟なワークロードなどがあります。

詳細については、「Amazon EC2ユーザーガイド」の「インスタンス購入オプション」を参照してください。

トピック

混合インスタンスグループを作成するための設定の概要

このトピックでは、Auto Scaling の混合インスタンスグループを作成するための概要とベストプラクティスについて説明します。

内容

概要

混合インスタンスグループを作成するには、次の 2 つのオプションがあります。

- 属性ベースのインスタンスタイプの選択 — コンピューティング要件を定義して、特定のインスタンス属性に基づいてインスタンスタイプを自動的に選択します。
- インスタンスタイプの手動選択 — ワークロードに適したインスタンスタイプを手動で選択します。

Manual selection

次のステップでは、インスタンスタイプを手動で選択して混合インスタンスグループを作成する方法を説明します。

1. EC2 インスタンスを起動するためのパラメータを含む起動テンプレートを選択します。起動テンプレートのパラメータはオプションですが、Amazon マシンイメージ (AMI) ID が起動テンプレートにない場合、Amazon EC2 Auto Scaling はインスタンスを起動できません。
2. 起動テンプレートをオーバーライドするオプションを選択します。
3. ワークロードに合ったインスタンスタイプを手動で選択します。
4. 起動するオンデマンドインスタンスとスポットインスタンスの割合を指定します。
5. Amazon EC2 Auto Scaling がオンデマンドキャパシティとスポットキャパシティをどのように満たすかを決定する配分戦略を、可能なインスタンスタイプから選択します。
6. インスタンスを起動するアベイラビリティーゾーンとVPCサブネットを選択します。
7. グループの初期サイズ (希望するキャパシティ) と、グループの最小サイズおよび最大サイズを指定します。

オーバーライドは、起動テンプレートで宣言されたインスタンスタイプをオーバーライドし、Auto Scaling グループ独自のリソース定義に埋め込まれている複数のインスタンスタイプを使用するために必要です。使用可能なインスタンスタイプの詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプ」を参照してください。

インスタンスタイプごとに次のオプションパラメータを設定することもできます。

- `LaunchTemplateSpecification` — 必要に応じて、別の起動テンプレートをインスタンスタイプに割り当てることができます。このオプションは現在、コンソールからは利用できません。詳細については、「インスタンスタイプに異なる起動テンプレートを使用する」を参照してください。
- `WeightedCapacity` — グループ内の残りのインスタンスとの関係で、どのくらいのインスタンスが希望する容量にカウントされるかを決定します。1 つのインスタンスタイプに `WeightedCapacity` の値を指定する場合は、すべてのインスタンスタイプに `WeightedCapacity` の値を指定する必要があります。デフォルトでは、各インスタンスは希望するキャパシティに対して 1 個としてカウントされます。詳細については、「インスタンスの重みを使用するように Auto Scaling グループを設定する」を参照してください。

Attribute-based selection

Amazon EC2 Auto Scaling が特定のインスタンス属性に基づいてインスタンスタイプを自動的に選択できるようにするには、次のステップを使用してコンピューティング要件を指定して混合インスタンスグループを作成します。

1. EC2 インスタンスを起動するためのパラメータを含む起動テンプレートを選択します。起動テンプレートのパラメータはオプションですが、Amazon マシンイメージ (AMI) ID が起動テンプレートにない場合、Amazon EC2 Auto Scaling はインスタンスを起動できません。
2. 起動テンプレートをオーバーライドするオプションを選択します。
3. vCPUs やメモリ要件など、コンピューティング要件に合ったインスタンス属性を指定します。
4. 起動するオンデマンドインスタンスとスポットインスタンスの割合を指定します。
5. Amazon EC2 Auto Scaling がオンデマンドキャパシティとスポットキャパシティをどのように満たすかを決定する配分戦略を、可能なインスタンスタイプから選択します。
6. インスタンスを起動するアベイラビリティーゾーンとVPCサブネットを選択します。
7. グループの初期サイズ (希望するキャパシティ) と、グループの最小サイズおよび最大サイズを指定します。

オーバーライドは、起動テンプレートで宣言されたインスタンスタイプをオーバーライドし、コンピューティング要件を記述するインスタンス属性のセットを使用するために必要です。サポートされている属性については、「Amazon EC2 Auto Scaling APIリファレンスInstanceRequirements」の「」を参照してください。あるいは、既にインスタンス属性が定義されている起動テンプレートを使用することもできます。

必要に応じて、オーバーライド構造内で `LaunchTemplateSpecification` パラメータを設定して、一連のインスタンス要件に別の起動テンプレートを割り当てることもできます。このオプションは現在、コンソールからは利用できません。詳細については、「Amazon EC2 Auto Scaling APIリファレンスLaunchTemplateOverrides」の「」を参照してください。

デフォルトでは、Auto Scaling グループの希望するキャパシティとしてインスタンスの数を設定します。

または、希望する容量の値をメモリの数 vCPUs またはメモリ量に設定することもできます。これを行うには、 `CreateAutoScalingGroup`APIオペレーションで `DesiredCapacityType`プロパティを使用するか、 で希望する容量タイプのドロップダウンフィールドを使用します AWS Management Console。これは、インスタンスの重みに代わる便利な方法です。

インスタンスタイプの柔軟性

可用性を高めるには、複数のインスタンスタイプにアプリケーションをデプロイします。キャパシティ要件を満たすために複数のインスタンスタイプを使用するのがベストプラクティスです。これにより、選択したアベイラビリティーゾーンに十分なインスタンス容量がない場合に、Amazon EC2 Auto Scaling は別のインスタンスタイプを起動できます。

スポットインスタンスでインスタンス容量が不十分な場合、Amazon EC2 Auto Scaling は他のスポットインスタンスプールからの起動を試み続けます。(使用するプールは、インスタンスタイプと配分戦略の選択によって決まります)。Amazon EC2 Auto Scaling は、オンデマンドインスタンスの代わりにスポットインスタンスを起動することで、スポットインスタンスのコスト削減を活用できます。

ワークロードごとに少なくとも 10 個のインスタンスを使用して柔軟性を持たせることをお勧めします。インスタンスタイプを選択する際には、最も人気のある新しいインスタンスタイプに限定しないでください。旧世代のインスタンスタイプを選択すると、オンデマンドの顧客からの需要が少ないため、スポット中断が少なくなる傾向があります。

アベイラビリティーゾーンの柔軟性

Auto Scaling グループが複数のアベイラビリティーゾーンにまたがるようにすることを強くお勧めします。複数のアベイラビリティーゾーンを使用すると、ゾーン間で自動的にフェイルオーバーして回復力を高めるアプリケーションを設計できます。

追加の利点として、単一のアベイラビリティーゾーンのグループと比較して、より深い Amazon EC2キャパシティープールにアクセスできます。キャパシティは各アベイラビリティーゾーンのインスタンスタイプごとに個別に変動するため、多くの場合、より多くのコンピューティングキャパシティを得ることができ、インスタンスタイプとアベイラビリティーゾーンの両方を柔軟に選択できます。

複数のアベイラビリティーゾーンの使用の詳細については、「例: 複数のアベイラビリティーゾーン全体にインスタンスを分散させる」を参照してください。

最大スポット料金

AWS CLI または を使用して Auto Scaling グループを作成する場合SDK、 SpotMaxPriceパラメータを指定できます。SpotMaxPrice パラメータは、お客様がスポットインスタンス時間について支払ってもよいと考える最大料金を決定します。

オーバーライド (またはグループレベルで "DesiredCapacityType": "vcpu" もしくは "DesiredCapacityType": "memory-mib") で WeightedCapacity パラメータを指定すると、最大料金はインスタンス全体の最大料金ではなく、最大単価を表します。

最大料金を指定しないことを強くお勧めします。低すぎる上限価格が設定されるなどの理由で、スポットインスタンスを取得できない場合には、アプリケーションが実行されないことがあります。上限料金を指定しない場合、デフォルトの上限料金はオンデマンド価格となります。起動するスポットインスタンスのスポット価格のみを支払います。スポットインスタンスによって提供される大幅な割引は、そのまま受けることができます。これらの割引を実現できるのは、スポット料金モデルを使用して適用されるスポット料金が安定しているためです。詳細については、「Amazon EC2ユーザーガイド」の「料金と削減額」を参照してください。

プロアクティブなキャパシティの再調整

ユースケースで可能な場合は、キャパシティの再調整をお勧めします。容量の再調整は、実行中のスポットインスタンスが 2 分間のスポットインスタンス中断通知を受け取る前に、新しいスポットインスタンスでフリートを事前に拡張することにより、ワークロードの可用性を維持するのに役立ちます。

キャパシティの再調整が有効になっている場合、Amazon EC2 Auto Scaling は、再調整に関する推奨事項を受け取ったスポットインスタンスを事前に置き換えようとします。これは、中断のリスクが低い新しいスポットインスタンスにワークロードを再調整する機会を提供します。

詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。

スケーリングの動作

混合インスタンスグループを作成すると、デフォルトでオンデマンドインスタンスが使用されます。スポットインスタンスを使用するには、オンデマンドインスタンスとして起動されるグループの割合を変更する必要があります。オンデマンドインスタンスの割合には、0 から 100 までの任意の数を指定できます。

オプションで、開始時のオンデマンドインスタンスのベース数を指定することもできます。その場合、Amazon EC2 Auto Scaling は、グループがスケールアウトしたときにオンデマンドインスタンスの基本容量を起動するまで、スポットインスタンスの起動を待機します。ベースキャパシティーを超える場合、オンデマンドインスタンスの割合を使用して、起動するオンデマンドインスタンスとスポットインスタンスの数が決まります。

Amazon EC2 Auto Scaling は、割合を同等のインスタンス数に変換します。結果が小数になる場合、オンデマンドインスタンスを優先して次の整数に切り上げられます。

次の表は、サイズが増減した場合における Auto Scaling グループの動作について示しています。

例: スケーリング動作

購入オプション	各購入オプションのグループのサイズと実行中のインスタンスの数			
	10	20	30	40
例 1: 10 を基準、50/50% オンデマンド/スポット				
オンデマンドインスタンス (ベース量)	10	10	10	10

購入オプション	各購入オプションのグループのサイズと実行中のインスタンスの数			
オンデマンドインスタンス	0	5	10	15
スポットインスタンス	0	5	10	15
例 2: 0 を基準、0/100% オンデマンド/スポット				
オンデマンドインスタンス (ベース量)	0	0	0	0
オンデマンドインスタンス	0	0	0	0
スポットインスタンス	10	20	30	40
例 3: 0 を基準、60/40% オンデマンド/スポット				
オンデマンドインスタンス (ベース量)	0	0	0	0
オンデマンドインスタンス	6	12	18	24
スポットインスタンス	4	8	12	16

購入オプション	各購入オプションのグループのサイズと実行中のインスタンスの数			
例 4: 0 を基準、100/0% オンデマンド/スポット				
オンデマンドインスタンス (ベース量)	0	0	0	0
オンデマンドインスタンス	10	20	30	40
スポットインスタンス	0	0	0	0
例 5: 12 を基準、0/100% オンデマンド/スポット				
オンデマンドインスタンス (ベース量)	10	12	12	12
オンデマンドインスタンス	0	0	0	0
スポットインスタンス	0	8	18	28

グループのサイズが大きくなると、Amazon EC2 Auto Scaling は指定されたアベイラビリティーゾーン間で容量の均等なバランスを試みます。次に、指定された配分戦略に従ってインスタンスタイプを起動します。

グループのサイズが小さくなると、Amazon EC2 Auto Scaling はまず 2 つのタイプ (スポットまたはオンデマンド) のうち、終了する必要があるタイプを特定します。その後、指定されたアベイラビリティーゾーン全体でバランスの取れた方法でインスタンスを終了することを試みます。また、優

先的に、割り当て戦略に近い方法でインスタンスを終了します。終了ポリシーの詳細については、「Amazon EC2 Auto Scaling の終了ポリシーを設定する」を参照してください。

リージョン別のインスタンスタイプの可用性

EC2 インスタンスタイプの可用性は、 によって異なります AWS リージョン。例えば、最新世代のインスタンスタイプは、特定のリージョンではまだ利用できない可能性があります。リージョンによってインスタンスの可用性が異なるため、オーバーライドする複数のインスタンスタイプをリージョンで利用できない場合、プログラムでリクエストを行う際に問題が発生する可能性があります。リージョンで利用できない複数のインスタンスタイプを使用すると、リクエストが完全に失敗する可能性があります。この問題を解決するには、異なるインスタンスタイプでリクエストを再試行し、各インスタンスタイプがリージョンで利用可能であることを確認してください。ロケーションによって提供されるインスタンスタイプを検索するには、 describe-instance-type-offerings コマンドを使用します。詳細については、「Amazon ユーザーガイド」の「Amazon EC2インスタンスタイプの検索」を参照してください。 EC2

関連リソース

スポットインスタンスのベストプラクティスの詳細については、「Amazon EC2ユーザーガイド」のEC2「スポットのベストプラクティス」を参照してください。

制限

混合インスタンスポリシーを使用して Auto Scaling グループにオーバーライドを追加した後、`UpdateAutoScalingGroup`API呼び出しでオーバーライドを更新できますが、削除することはできません。オーバーライドを完全に削除するには、まず Auto Scaling グループを切り替え、混合インスタンスポリシーではなく起動テンプレートまたは起動設定を使用する必要があります。その後、オーバーライドなしで混合インスタンスポリシーを再度追加できます。

複数のインスタンスタイプの配分戦略

複数のインスタンスタイプを使用する場合、Amazon EC2 Auto Scaling がオンデマンドキャパシティとスポットキャパシティをどのように満たすかは、可能なインスタンスタイプから管理します。これを実行するには、割り当て戦略を指定します。

混合インスタンスグループのベストプラクティスを確認するには、「混合インスタンスグループを作成するための設定の概要」を参照してください。

内容

- スポットインスタンス

- オンデマンドインスタンス
- 配分戦略と重みの仕組み

スポットインスタンス

Amazon EC2 Auto Scaling では、スポットインスタンスに次の配分戦略が用意されています。

`price-capacity-optimized` (推奨)

価格とキャパシティで最適化された配分戦略では、価格とキャパシティの両方を考慮して、中断される可能性が最も低く、価格ができるだけ低いスポットインスタンスプールを選択します。

使用を開始する際には、この戦略をお勧めします。詳細については、 AWS ブログの price-capacity-optimizedEC2「スポットインスタンスの配分戦略の紹介」を参照してください。

`capacity-optimized`

Amazon EC2 Auto Scaling は、起動するインスタンスの数に最適な容量を持つプールからスポットインスタンスをリクエストします。

スポットインスタンスでは、料金は需要と供給の長期的な傾向に基づいて時間の経過とともに緩やかに変動します。ただし、キャパシティはリアルタイムで変動します。`capacity-optimized`戦略では、リアルタイムのキャパシティーデータを調べ、可用性の最も高いプールを予測することで、そのプールから スポットインスタンス を自動的に起動します。これにより、作業の再開とチェックポイントに伴う中断コストが高くなる可能性があるワークロードの中断を最小限に抑えることができます。最初に特定のインスタンスタイプを起動する可能性を高めるには、`capacity-optimized-prioritized` を使用します。

`capacity-optimized-prioritized`

起動テンプレートのオーバーライドのリスト内のインスタンスタイプの順序を、優先順位の高いものから順番に (リストの最初から最後まで) 設定することもできます。Amazon EC2 Auto Scaling は、ベストエフォートベースでインスタンスタイプの優先順位を尊重しますが、最初に容量を最適化します。これは、中断の可能性を最小限に抑える必要があるワークロードに適したオプションですが、適切なインスタンスタイプを設定することも重要です。オンデマンド配分戦略を `prioritized` に設定した場合、オンデマンドキャパシティを満たす際にも同じ優先順位が適用されます。

`lowest-price` (非推奨)

Amazon EC2 Auto Scaling は、最低価格のプール設定に指定した N 個のスポットプール全体で、アベイラビリティーゾーン内の最低価格のプールを使用してスポットインスタンスをリクエス

トします。例えば、4 つのインスタンスタイプと 4 つのアベイラビリティーゾーンを指定した場合、Auto Scaling グループは最大 16 個のスポットプールにアクセスできます。(各アベイラビリティーゾーンに 4 個ずつ)。配分戦略で 2 つのスポットプール (N=2) を指定した場合、Auto Scaling グループは、アベイラビリティゾーンあたり 2 つの最低価格のプールを引き出し、スポットキャパシティを満たすことができます。

この戦略では、インスタンスの価格のみが考慮され、キャパシティの可用性は考慮されないため、中断率が高くなる可能性があります。

Amazon EC2 Auto Scaling は、指定した N 個のプールからスポットインスタンスを描画するよう努めます。ただし、希望する容量を満たす前にプールのスポット容量が不足した場合、Amazon EC2 Auto Scaling は次に安い価格のプールを利用してリクエストを満たし続けます。希望するキャパシティを満たすために、指定した N 個を超えるプールからスポットインスタンスが割り当てられることがあります。同様に、ほとんどのプールにスポットキャパシティーがない場合には、指定した N 個より少ないプールから希望するキャパシティのすべてが割り当てられることがあります。

> **Note**
>
> AMD SEV-SNP をオンにして起動するようにスポットインスタンスを設定すると、選択したインスタンスタイプのオンデマンド時間料金の 10% に相当する追加の時間単位使用料が課金されます。配分戦略で価格を入力として使用している場合、Amazon EC2 Auto Scaling にはこの追加料金は含まれず、スポット料金のみが使用されます。

オンデマンドインスタンス

Amazon EC2 Auto Scaling には、オンデマンドインスタンスに使用できる以下の配分戦略が用意されています。

`lowest-price`

Amazon EC2 Auto Scaling は、現在のオンデマンド料金に基づいて、各アベイラビリティーゾーンで最低価格のインスタンスタイプを自動的にデプロイします。

希望するキャパシティを満たすために、各アベイラビリティーゾーンで複数のタイプのオンデマンドインスタンスを受け取る場合があります。これは、リクエストするキャパシティの大きさによって異なります。

`prioritized`

オンデマンド容量を満たす場合、Amazon EC2 Auto Scaling は、起動テンプレートのオーバーライドのリスト内のインスタンスタイプの順序に基づいて、最初に使用するインスタンスタイプを決定します。例えば、3 つの起動テンプレートの優先度を `c5.large`、`c4.large`、`c3.large` と指定するとします。オンデマンドインスタンスが起動すると、Auto Scaling グループは、`c5.large`、`c4.large`、`c3.large` の順にオンデマンドキャパシティを満たします。

オンデマンドインスタンスの優先順位を管理する場合は、次の点を考慮してください。

- Savings Plans またはリザーブドインスタンスを利用して使用料を前払いすることで、オンデマンドインスタンスの大幅な割引を受けることができます。詳細については、Amazon のEC2 料金ページを参照してください。
- リザーブドインスタンス では、Amazon EC2 Auto Scaling が一致するインスタンスタイプを起動した場合、通常のオンデマンドインスタンス料金の割引料金が適用されます。したがって、`c4.large` の未使用のリザーブドインスタンスがある場合、インスタンスタイプの優先順位を設定して、リザーブドインスタンスの最も高い優先順位を `c4.large` インスタンスタイプに付与できます。`c4.large` インスタンスが起動すると、リザーブドインスタンスの料金が発生します。
- Savings Plans では、Amazon EC2 Instance Savings Plans または Compute Savings Plans Savings Plans を使用する場合、通常のオンデマンドインスタンス料金の割引料金が適用されます。Savings Plans では、インスタンスタイプの優先順位をより柔軟に設定することができます。Savings Plans の対象となるインスタンスタイプであれば、任意の順序で優先順位を設定できます。また、インスタンスタイプの全体の順序を変えることも可能で、その場合でも Savings Plans の割引料金が適用されます。Savings Plans の詳細については、Savings Plans ユーザーガイドを参照してください。

配分戦略と重みの仕組み

オーバーライド (またはグループレベルでは `"DesiredCapacityType": "vcpu"` もしくは `"DesiredCapacityType": "memory-mib"`) で `WeightedCapacity` パラメータを指定すると、配分戦略は他の Auto Scaling グループの場合とまったく同じように機能します。

違いは、`lowest-price` または `price-capacity-optimized` 戦略を選択した場合、アベイラビリティーゾーンにおけるユニットあたりの料金が最も安いインスタンスプールから、インスタンスが取得されることのみです。詳細については、「インスタンスの重みを使用するように Auto Scaling グループを設定する」を参照してください。

例えば、さまざまな量の を持つ複数のインスタンスタイプを持つ Auto Scaling グループがあるとしますvCPUs。スポットとオンデマンドの配分戦略として `lowest-price` を使うとします。各インスタンスタイプの vCPU カウントに基づいて重みを割り当てることを選択した場合、Amazon EC2 Auto Scaling は、フルフィルメント時に割り当てられた重み値 (v ごとなどCPU) ごとに最低価格のインスタンスタイプを起動します。スポットインスタンスの場合、これは v あたりの最低スポット料金を意味しますCPU。 オンデマンドインスタンスの場合、これは v あたりのオンデマンド料金が最も低いことを意味しますCPU。

属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する

混合インスタンスグループのインスタンスタイプを手動で選択する代わりに、コンピューティング要件を記述するインスタンス属性のセットを指定できます。Amazon EC2 Auto Scaling がインスタンスを起動するとき、Auto Scaling グループで使用されるインスタンスタイプは、必要なインスタンス属性と一致する必要があります。これは 属性ベースのインスタンスタイプの選択 と呼ばれます。

このアプローチは、コンテナやビッグデータ、CI/CD など、柔軟にインスタンスタイプを使用するワークロードとフレームワークに最適です。

属性ベースのインスタンスタイプを選択すると、次の利点があります。

- スポットインスタンスに最適な柔軟性 – Amazon EC2 Auto Scaling は、スポットインスタンスを起動するための幅広いインスタンスタイプから選択できます。これは、インスタンスタイプに柔軟に対応するというスポットのベストプラクティスを満たしているため、Amazon EC2 Spot サービスは必要な量のコンピューティング容量を見つけて割り当てる可能性が高くなります。
- 適切なインスタンスタイプを簡単に使用 - 利用可能なインスタンスタイプの数が多いため、ワークロードに適したインスタンスタイプを見つけるには時間がかかることがあります。インスタンス属性を指定すると、インスタンスタイプにはワークロードに必要な属性が自動的に設定されます。
- 新しいインスタンスタイプの自動的な使用 — Auto Scaling グループでは、リリースされた時点で、新しい世代のインスタンスタイプを使用できます。要件に一致し、かつ Auto Scaling グループのために選択した割り当て戦略にマッチする場合には、新しい世代のインスタンスタイプが自動的に使用されます。

トピック

- 属性ベースのインスタンスタイプ選択の仕組み
- 料金保護
- パフォーマンス保護

- 前提条件
- 属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する (コンソール)
- 属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する (AWS CLI)
- 設定例
- インスタンスタイプをプレビューする
- 関連リソース

属性ベースのインスタンスタイプ選択の仕組み

属性ベースのインスタンスタイプの選択では、特定のインスタンスタイプのリストの代わりに、以下のようなインスタンスに必要なインスタンス属性のリストが提供されます。

- vCPU 数 – インスタンスあたりの vCPUs の最小数と最大数。
- メモリ — インスタンスあたりのメモリの最小および最大 GiBs 。
- ローカルストレージ — ローカルストレージに EBS ボリュームとインスタンスストアボリュームのどちらを使用するか。
- バースト可能なパフォーマンス – T4g、T3a、T3、および T2 タイプを含む T インスタンスファミリーを使用するかどうか。

インスタンス要件を定義するには、多くのオプションを使用できます。各オプションの説明とデフォルト値については、Amazon InstanceRequirements Auto Scaling Word APIリファレンスのEC2」を参照してください。

Auto Scaling グループがインスタンスを起動する必要がある場合、指定した属性に一致し、そのアベイラビリティーゾーンで使用できるインスタンスタイプが検索されます。その後、配分戦略によって、起動する一致するインスタンスタイプが決定されます。属性ベースのインスタンスタイプの選択ではデフォルトで、Auto Scaling グループが予算のしきい値を超えるインスタンスタイプを起動できないようにする料金の保護機能が有効になっています。

デフォルトでは、Auto Scaling グループの希望する容量を設定する際に、インスタンスの数を測定単位として使用します。つまり、各インスタンスが 1 つの単位としてカウントされます。

または、希望する容量の値を vCPUs の数またはメモリ量に設定することもできます。これを行うには、 の「希望する容量タイプ」ドロップダウンフィールド、 AWS Management Console ま

または CreateAutoScalingGroupまたは API オペレーションの UpdateAutoScalingGroup DesiredCapacityTypeプロパティを使用します。Amazon EC2 Auto Scaling は、必要な vCPU またはメモリ容量を満たすために必要なインスタンスの数を起動します。例えば、希望するキャパシティータイプとして vCPUs を使用し、それぞれ 2 vCPUs のインスタンスを使用する場合、希望するキャパシティーを 10 vCPUs にすると、5 つのインスタンスが起動します。これは、インスタンスの重みに代わる便利な方法です。

料金保護

料金保護を使用すると、Auto Scaling グループによって起動された EC2 インスタンスに対して支払う上限価格を指定できます。料金の保護は、指定した属性に適合した場合でも、コストが高すぎると思われるインタンスタイプについては Auto Scaling グループで使用できないようにする機能です。

料金の保護はデフォルトで有効になっており、オンデマンドインスタンスとスポットインスタンスにおける料金のしきい値は異なります。Amazon EC2 Auto Scaling が新しいインスタンスを起動する必要がある場合、関連するしきい値を超える料金のインスタンスタイプは起動されません。

トピック

- オンデマンド料金の保護
- スポット料金の保護
- 料金の保護をカスタマイズする

オンデマンド料金の保護

オンデマンドインスタンスの場合、指定のオンデマンド料金よりも高い割合で支払うことができる最大のオンデマンド料金を定義します。指定のオンデマンド料金は、指定された属性を持つ現行世代の C、M、または R インスタンスタイプ中で最も安いです。

オンデマンド料金の保護の値が明確に定義されていない場合、指定のオンデマンド料金よりも 20% 高いデフォルトのオンデマンド料金が最大料金として使用されます。

スポット料金の保護

デフォルトでは、Amazon EC2 Auto Scaling は最適なスポットインスタンス料金保護を自動的に適用し、幅広いインスタンスタイプから一貫して選択します。料金保護を手動で設定することもできます。ただし、Amazon EC2 Auto Scaling に任せることで、スポット容量が満たされる可能性を高めることができます。

料金保護は、次のいずれかのオプションを使用して手動で指定できます。料金保護を手動で設定する場合は、最初のオプションを使用することをお勧めします。

- 指定のオンデマンド料金の割合 — 指定のオンデマンド料金は、指定された属性を持つ現行世代の C、M、または R インスタンスタイプ中で最も安いです。
- 指定のスポット料金よりも高い割合 — 指定のスポット料金は、指定された属性を持つ現行世代の C、M、または R インスタンスタイプの中で最も安いです。スポット料金および料金の保護のしきい値は変動する可能性があるため、このオプションの使用は推奨されません。

料金の保護をカスタマイズする

料金保護のしきい値は、Amazon EC2 Auto Scaling コンソールまたは AWS CLI または SDKs を使用してカスタマイズできます。

- コンソールで、[追加のインスタンス属性] の [オンデマンド料金の保護] および [スポット料金の保護] 設定を使用します。
- InstanceRequirements 構造で、オンデマンドインスタンスの料金保護しきい値を指定するには、 `OnDemandMaxPricePercentageOverLowestPrice` プロパティを使用します。スポットインスタンス料金の保護のしきい値を指定するには、`MaxSpotPriceAsPercentageOfOptimalOnDemandPrice` または `SpotMaxPricePercentageOverLowestPrice` プロパティのいずれかを使用します。

希望する容量タイプ (`DesiredCapacityType`) を vCPUs またはメモリ GiB に設定すると、インスタンスあたりの料金ではなく、vCPU 単位またはメモリ単位の料金に基づいて料金保護が適用されます。

料金保護をオフにすることもできます。料金保護のしきい値を指定しない場合は、999999 などの高いパーセンテージ値を指定します。

> **Note**
>
> 現行世代の C、M、または R インスタンスタイプが指定の属性と一致しない場合でも、料金の保護は適用されます。一致するものが見つからなかった場合、指定の料金は現行世代のインスタンスタイプの中で最も安価なものになります。それもない場合は、属性に一致する前世代のインスタンスタイプの中で最も安価なものになります。

パフォーマンス保護

パフォーマンス保護は、Auto Scaling グループが、指定されたパフォーマンスベースラインに類似または上回るインスタンスタイプを使用するようにする機能です。パフォーマンス保護を使用する

には、ベースラインリファレンスとしてインスタンスファミリーを指定します。指定されたインスタンスファミリーの機能によって、許容可能な最低レベルのパフォーマンスが確立されます。Auto Scaling がインスタンスタイプを選択するときは、指定した属性とパフォーマンスベースラインを考慮します。パフォーマンスベースラインを下回るインスタンスタイプは、他の指定された属性と一致していても、自動的に選択から除外されます。これにより、選択したすべてのインスタンスタイプが、指定されたインスタンスファミリーによって確立されたベースラインと同等以上のパフォーマンスを提供します。Auto Scaling はこのベースラインを使用してインスタンスタイプの選択をガイドしますが、選択したインスタンスタイプがすべてのアプリケーションのベースラインを常に超える保証はありません。

現在、この機能はベースラインパフォーマンス係数として CPU パフォーマンスのみをサポートしています。指定されたインスタンスファミリーの CPU パフォーマンスがパフォーマンスベースラインとして機能し、選択したインスタンスタイプがこのベースラインと類似しているか、それを超えていることを確認します。同じ CPU プロセッサを持つインスタンスファミリーは、ネットワークまたはディスクのパフォーマンスが異なっていても、同じフィルタリング結果になります。例えば、ベースライン参照c6iとして c6inまたは を指定すると、両方のインスタンスファミリーが同じ CPU プロセッサを使用するため、パフォーマンスベースのフィルタリング結果が同じになります。

サポートされていないインスタンスファミリー

以下のインスタンスファミリーはパフォーマンス保護をサポートしていません。

- c1
- g3 | g3s
- hpc7g
- m1 | m2
- mac1 | mac2 | mac2-m1ultra | mac2-m2 | mac2-m2pro
- p3dn | p4d | p5
- t1
- u-12tb1 | u-18tb1 | u-24tb1 | u-3tb1 | u-6tb1 | u-9tb1 | u7i-12tb | u7in-16tb | u7in-24tb | u7in-32tb

サポートされているインスタンスファミリーを指定してパフォーマンス保護を有効にすると、返されるインスタンスタイプは、上記のサポートされていないインスタンスファミリーを除外します。

例: CPU パフォーマンスベースラインの設定

次の例では、インスタンス要件は、インスタンスファミリーと同じパフォーマンスの CPU コアを持つc6iインスタンスタイプで を起動することです。これにより、CPU の数など、他の指定されたインスタンス要件を満たしている場合でも、パフォーマンスの低い vCPUs プロセッサを持つインスタンスタイプが除外されます。例えば、指定したインスタンス属性に 4 vCPUs と 16 GB のメモリが含まれている場合、これらの属性を持つが、CPU のパフォーマンスが低いインスタンスタイプは選択から除外c6iされます。

```
"BaselinePerformanceFactors": {
        "Cpu": {
            "References": [
                {
                    "InstanceFamily": "c6i"
                }
            ]
        }
```

考慮事項

パフォーマンス保護を使用する場合は、次の点を考慮してください。

- インスタンスタイプまたはインスタンス属性のいずれかを指定できますが、両方を同時に指定することはできません。
- リクエスト設定では、最大 4 つの InstanceRequirements 構造を指定できます。

前提条件

- 起動テンプレートを作成する。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。
- 起動テンプレートがまだスポットインスタンスをリクエストしていないことを確認します。

属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する (コンソール)

属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成するには、次の手順を実行します。ステップを効率的に進めるために、いくつかのオプションのセクションは省略されています。

ほとんどの汎用ワークロードでは、必要な vCPUs とメモリの数を指定するだけで十分です。高度なユースケースでは、ストレージタイプ、ネットワークインターフェイス、CPU 製造元、アクセラレータータイプなどの属性を指定できます。

混合インスタンスグループのベストプラクティスを確認するには、「混合インスタンスグループを作成するための設定の概要」を参照してください。

混合インスタンスグループを作成するには

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. 画面の上部のナビゲーションバーで、起動テンプレートを作成したときに使用したのと同じ AWS リージョン を選択します。
3. [Auto Scaling グループの作成] を選択します。
4. [起動テンプレートまたは起動設定を選択する] ページで [Auto Scaling グループ名] にAuto Scaling グループの名前を入力します。
5. 起動テンプレートを選択するには、以下の手順を実行します。

   a. [起動テンプレート] で、既存の起動テンプレートを選択します。

   b. [起動テンプレートのバージョン] で、スケールアウト時に Auto Scaling グループで使用する起動テンプレートのバージョン (デフォルト、最新、または特定のバージョン) を選択します。

   c. 起動テンプレートが、使用する予定のすべてのオプションをサポートしていることを確認し、[次へ] を選択します。
6. [インスタンス起動オプションを選択] ページで、次を実行します。

   a. [Instance type requirements] (インスタンスタイプの要件) で、[Override launch template] (起動テンプレートを上書きする) を選択します。

   > **Note**
   >
   > vCPUs やメモリなどのインスタンス属性のセットが既に含まれている起動テンプレートを選択した場合は、インスタンス属性が表示されます。これらの属性は Auto Scaling グループのプロパティに追加され、Amazon EC2 Auto Scaling コンソールからいつでも更新できます。

   b. 「インスタンス属性を指定する」で、まず your vCPUs とメモリの要件を入力します。

- vCPUs には、目的の最小数と最大数を入力しますvCPUs。制限なしを指定するには、[No minimum] (最小値なし)、[No maximum] (最大値なし)、または両方を選択します。
- [Memory (GiB)] (メモリ (GiB)) に、希望する最小値と最大値を入力します。制限なしを指定するには、[No minimum] (最小値なし)、[No maximum] (最大値なし)、または両方を選択します。

c. (オプション) [Additional instance attributes] (その他のインスタンス属性) では、オプションで 1 つ以上の属性を指定して、コンピューティング要件をより詳細に表現できます。追加の属性は、リクエストにさらに制約を追加します。

d. [一致するインスタンスタイプをプレビュー] を展開して、指定した属性を持つインスタンスタイプを表示します。

e. [インスタンスの購入オプション] の [インスタンスの分散] で、オンデマンドインスタンスとスポットインスタンスとして起動するグループの割合をそれぞれ指定します。アプリケーションが、ステートレスでフォールトトレラントであり、中断されるインスタンスを扱える場合は、より高い割合のスポットインスタンスを指定できます。

f. (オプション) スポットインスタンスの割合を指定するときは、[オンデマンドベースキャパシティを含める] を選択してから、オンデマンドインスタンスによって満たされる必要がある Auto Scaling グループの最小初期キャパシティを指定します。ベースキャパシティーを超える場合は、[Instances distribution] (インスタンスの分散) 設定を使用して、起動するオンデマンドインスタンスとスポットインスタンスの数を決定します。

g. [Allocation strategies] (配分戦略) の[Lowest price] (最低価格) は、[On-Demand allocation strategy] (オンデマンドの配分戦略) によって自動的に選択され、変更できません。

h. [Spot allocation strategy] (スポット配分戦略) で、配分戦略を選択します。デフォルトでは、[Price capacity optimized] (価格のキャパシティの最適化) が選択されています。[Lowest price] (最低価格) はデフォルトでは非表示になっており、[Show all strategies] (すべての戦略を表示) を選択した場合にのみ表示されます。[最低料金] を選択した場合は、[最低料金のプール] に、分散する最低料金のプールの数を入力します。

i. [容量の再分散] で、容量の再分散を有効にするか無効にするかを選択します。キャパシティの再調整を使用すると、スポットインスタンスがスポットの中断によって終了に近づいたときに自動的に応答します。詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。

j. Network の VPC で、VPC を選択します。Auto Scaling グループは、起動テンプレートで指定したセキュリティグループと同じ VPC で作成する必要があります。

k. アベイラビリティーゾーンとサブネットでは、指定した VPC で 1 つ以上のサブネットを選択します。複数のアベイラビリティーゾーンのサブネットを使用することで、高可用性を得られます。詳細については、「VPC サブネットを選択する際の考慮事項」を参照してください。

l. [次へ]、[次へ] を選択します。

7. [Configure group size and scaling policies] (グループサイズとスケーリングポリシーを設定する) ステップでは、以下の手順を実行します。

   a. 希望する容量をインスタンス以外のユニットで測定するには、[グループサイズ] および [希望する容量タイプ] で適切なオプションを選択します。単位、vCPUs、メモリ GiB がサポートされています。デフォルトでは、Amazon EC2 Auto Scaling はインスタンス数に変換される単位を指定します。

   b. [希望する容量] で、Auto Scaling グループの初期サイズを設定します。

   c. [スケーリング] セクションの [スケーリング制限] で、[希望する容量] の新しい値が [最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に希望する新しい容量の値に引き上げられます。これらの制限は、必要に応じて変更できます。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。

8. [Skip to review] を選択します。

9. [Review (レビュー)]ページで、[Create Auto Scaling group (Auto Scaling グループを作成)] を選択します。

属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する (AWS CLI)

コマンドラインを使用して混合インスタンスグループを作成するには

以下のいずれかのコマンドを使用します。

- create-auto-scaling-group (AWS CLI)
- New-ASAutoScalingGroup (AWS Tools for Windows PowerShell)

設定例

を使用して、属性ベースのインスタンスタイプを選択して Auto Scaling グループを作成するには AWS CLI、次の create-auto-scaling-group コマンドを使用します。

次のインスタンス属性が指定されています。

- VCpuCount – インスタンスタイプには、4 vCPUs以上 8 vCPUs以下が必要です。
- MemoryMiB – インスタンスタイプには最低 16,384 MiB のメモリが必要です。
- CpuManufacturers – インスタンスタイプにはインテル製の CPU が必要です。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

次は、config.json ファイルの例です。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacityType": "units",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [{
                "InstanceRequirements": {
                    "VCpuCount": {"Min": 4, "Max": 8},
                    "MemoryMiB": {"Min": 16384},
                    "CpuManufacturers": ["intel"]
                }
            }]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "price-capacity-optimized"
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 4,
    "DesiredCapacityType": "units",
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

希望する容量の値を vCPUs の数またはメモリ量として設定するには、 ファイル"DesiredCapacityType": "memory-mib"で "DesiredCapacityType": "vcpu"または を指定します。希望するキャパシティータイプのデフォルトは units で、これはインスタンスの数を、希望するキャパシティー値として設定します。

YAML

または、次の create-auto-scaling-group コマンドを使用して Auto Scaling グループを作成することもできます。これは、Auto Scaling グループの唯一のパラメータとして YAML ファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

次は、config.yaml ファイルの例です。

```
---
AutoScalingGroupName: my-asg
DesiredCapacityType: units
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceRequirements:
        VCpuCount:
          Min: 2
          Max: 4
        MemoryMiB:
          Min: 2048
        CpuManufacturers:
        - intel
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: price-capacity-optimized
MinSize: 0
MaxSize: 100
DesiredCapacity: 4
DesiredCapacityType: units
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

希望する容量の値を vCPUs の数またはメモリ量として設定するには、 ファイルDesiredCapacityType: memory-mibで DesiredCapacityType: vcpuまたは を指定します。希望するキャパシティータイプのデフォルトは units で、これはインスタンスの数を、希望するキャパシティー値として設定します。

インスタンスタイプをプレビューする

インスタンスを起動することなくコンピューティング要件に一致するインスタンスタイプをプレビューでき、必要に応じて要件を調整できます。Amazon EC2 Auto Scaling コンソールで Auto Scaling グループを作成すると、インスタンスタイプのプレビューが、インスタンス起動オプションの選択ページの一致するインスタンスタイプのプレビューセクションに表示されます。

または、 AWS CLI または Word を使用して Amazon EC2 WordGetInstanceTypesFromInstanceRequirements API を呼び出すことで、インスタンスタイプをプレビューすることもできますSDK。Auto Scaling グループの作成または更新のリクエストの中で、正しい形式で InstanceRequirements パラメーターを渡します。詳細については、「Amazon EC2 ユーザーガイド」の「指定された属性を持つインスタンスタイプのプレビュー」を参照してください。

関連リソース

属性ベースのインスタンスタイプの選択の詳細については、 AWS ブログのEC2 Auto Scaling と EC2 フリートの属性ベースのインスタンスタイプの選択」を参照してください。

AWS CloudFormationを使用して Auto Scaling グループを作成する際に、属性ベースのインスタンスタイプの選択を宣言できます。詳細については、「AWS CloudFormation ユーザーガイド」の「Auto Scaling テンプレートスニペット」セクションのサンプルスニペットを参照してください。

インスタンスタイプを手動で選択して混合インスタンスグループを作成する

このトピックでは、インスタンスタイプを手動で選択して、単一の Auto Scaling グループで複数のインスタンスタイプを起動する方法を示します。

インスタンスタイプを選択する基準としてインスタンス属性を使用する場合は、「属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する」を参照してください。

内容

- 混合インスタンスグループを作成する (AWS CLI)
- 設定例

前提条件

- 起動テンプレートを作成する。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。
- 起動テンプレートがまだスポットインスタンスをリクエストしていないことを確認します。

混合インスタンスグループを作成する (コンソール)

次の手順を実行して、グループが起動できるインスタンスタイプを手動で選択し、混合インスタンスグループを作成します。ステップを効率的に進めるために、いくつかのオプションのセクションは省略されています。

混合インスタンスグループのベストプラクティスを確認するには、「混合インスタンスグループを作成するための設定の概要」を参照してください。

混合インスタンスグループを作成するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 画面の上部のナビゲーションバーで、起動テンプレートを作成したときに使用したのと同じ AWS リージョン を選択します。
3. [Auto Scaling グループの作成] を選択します。
4. [起動テンプレートまたは起動設定を選択する] ページで [Auto Scaling グループ名] にAuto Scaling グループの名前を入力します。
5. 起動テンプレートを選択するには、以下の手順を実行します。
   a. [起動テンプレート] で、既存の起動テンプレートを選択します。
   b. [起動テンプレートのバージョン] で、スケールアウト時に Auto Scaling グループで使用する起動テンプレートのバージョン (デフォルト、最新、または特定のバージョン) を選択します。
   c. 起動テンプレートが、使用する予定のすべてのオプションをサポートしていることを確認し、[Next] (次へ) を選択します。
6. [インスタンス起動オプションを選択] ページで、次を実行します。

a. [Instance type requirements] (インスタンスタイプの要件) で、[Override launch template] (起動テンプレートを上書きする) を選択してから、[Manually add instance types] (インスタンスタイプを手動で追加する) を選択します。

b. インスタンスタイプを選択します。まずはレコメンデーションを使用できます。デフォルトでは、[Family and generation flexible] (ファミリーと世代が柔軟) が選択されています。

   - インスタンスタイプの順序を変更するには、矢印を使用します。優先順位付けをサポートする配分戦略を選択した場合、インスタンスタイプの順序によって起動の優先順位が設定されます。
   - インスタンスタイプを削除するには、[X] を選択します。
   - (オプション) [重み] 列のボックスで、各インスタンスタイプに相対的な重みを割り当てます。これを行うには、そのタイプのインスタンスがグループの希望するキャパシティにカウントされるユニット数を入力します。インスタンスタイプが異なる v、メモリCPU、ストレージ、またはネットワーク帯域幅機能を提供する場合、これを行うと便利です。詳細については、「インスタンスの重みを使用するように Auto Scaling グループを設定する」を参照してください。

     [サイズが柔軟] のレコメンデーションを使用することを選択した場合は、このセクションに含まれるすべてのインスタンスタイプに自動的に重みの値が設定されることに留意してください。重みを指定したくない場合は、すべてのインスタンスタイプについて [Weight] (重み) 列のボックスをクリアしてください。

c. [Instance purchase options] (インスタンスの購入オプション) の [Instances distribution] (インスタンスの分散) で、オンデマンドインスタンスとスポットインスタンスとして起動されるグループの割合をそれぞれ指定します。アプリケーションが、ステートレスでフォールトトレラントであり、中断されるインスタンスを扱える場合は、より高い割合のスポットインスタンスを指定できます。

d. (オプション) スポットインスタンスの割合を指定するときは、[オンデマンドベースキャパシティを含める] を選択してから、オンデマンドインスタンスによって満たされる必要がある Auto Scaling グループの最小初期キャパシティを指定します。ベースキャパシティーを超える場合は、[Instances distribution] (インスタンスの分散) 設定を使用して、起動するオンデマンドインスタンスとスポットインスタンスの数を決定します。

e. [Allocation strategies] (配分戦略) の [On-Demand allocation strategy] (オンデマンドの配分戦略) で、配分戦略を選択します。インスタンスタイプを手動で選択すると、デフォルトで [Prioritized] (高い優先順位で設定済み) が選択されます。

f. [Spot allocation strategy] (スポット配分戦略) で、配分戦略を選択します。デフォルトでは、[Price capacity optimized] (価格のキャパシティの最適化) が選択されています。[Lowest price] (最低価格) はデフォルトでは非表示になっており、[Show all strategies] (すべての戦略を表示) を選択した場合にのみ表示されます。

- [最低料金] を選択した場合は、[最低料金のプール] に、分散する最低料金のプールの数を入力します。
- キャパシティ最適化を選択した場合は、オプションでインスタンスタイプの優先順位付けチェックボックスをオンにして、Amazon EC2 Auto Scaling がインスタンスタイプがリストされている順序に基づいて最初に起動するインスタンスタイプを選択できるようにします。

g. [容量の再分散] で、容量の再分散を有効にするか無効にするかを選択します。キャパシティの再調整を使用すると、スポットインスタンスがスポットの中断によって終了に近づいたときに自動的に応答します。詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。

h. Network で、 に VPCを選択しますVPC。Auto Scaling グループは、起動テンプレートで指定したセキュリティグループVPCと同じ に作成する必要があります。

i. アベイラビリティーゾーンとサブネットでは、指定した で 1 つ以上のサブネットを選択しますVPC。複数のアベイラビリティーゾーンのサブネットを使用することで、高可用性を得られます。詳細については、「VPC サブネットを選択する際の考慮事項」を参照してください。

j. [次へ]、[次へ] を選択します。

7. [Configure group size and scaling policies] (グループサイズとスケーリングポリシーを設定する) ステップでは、以下の手順を実行します。

a. [グループサイズ] の [希望する容量] に、起動するインスタンスの初期数を入力します。

デフォルトでは、希望する容量はインスタンスの数として表されます。インスタンスタイプに重みを割り当てた場合は、この値を の数など、重みの割り当てに使用したのと同じ測定単位に変換する必要がありますvCPUs。

b. [スケーリング] セクションの [スケーリング制限] で、[希望する容量] の新しい値が [最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に希望する新しい容量の値に引き上げられます。これらの制限は、必要に応じて変更できます。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。

8. [Skip to review] を選択します。

9. [Review (レビュー)]ページで、[Create Auto Scaling group (Auto Scaling グループを作成)] を選択します。

混合インスタンスグループを作成する (AWS CLI)

コマンドラインを使用して混合インスタンスグループを作成するには

以下のいずれかのコマンドを使用します。

- create-auto-scaling-group (AWS CLI)
- 新規 -ASAutoScalingGroup (AWS Tools for Windows PowerShell)

設定例

次の設定例は、さまざまなスポット割り当て戦略を使用して混合インスタンスグループを作成する方法を示しています。

> **Note**
>
> これらの例は、 JSONまたは でフォーマットされた設定ファイルを使用する方法を示していますYAML。 AWS CLI バージョン 1 を使用する場合は、 JSON形式の設定ファイルを指定する必要があります。 AWS CLI バージョン 2 を使用する場合は、 YAMLまたは のいずれかの形式の設定ファイルを指定できますJSON。

例

- 例 1: capacity-optimized 割り当て戦略を使用して スポットインスタンス を起動する
- 例 2: capacity-optimized-prioritized 割り当て戦略を使用して スポットインスタンス を起動する
- 例 3: 2つのプール間での lowest-price 配分戦略を使用してスポットインスタンスを起動する
- 例 4: price-capacity-optimized 配分戦略を使用して スポットインスタンス を起動する

例 1: **`capacity-optimized`** 割り当て戦略を使用して スポットインスタンス を起動する

次のcreate-auto-scaling-groupコマンドは、以下を指定する Auto Scaling グループを作成します。

- オンデマンドインスタンスとして起動するグループの割合 (0) と開始時のオンデマンドインスタンスのベース数 (1)

- 優先度に従って起動するインスタンスタイプ (c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)
- インスタンスを起動するサブネット (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。それぞれが異なるアベイラビリティーゾーンに対応します。
- 起動テンプレート (my-launch-template) とそのバージョン ($Default)。

Amazon EC2 Auto Scaling がオンデマンド容量を満たす場合、最初にc5.largeインスタンスタイプを起動します。スポットインスタンスは、スポットインスタンスのキャパシティーに基づいて、各アベイラビリティーゾーンの最適なスポットプールから取得されます。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
```

```
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

または、次のcreate-auto-scaling-groupコマンドを使用して Auto Scaling グループを作成することもできます。これは、Auto Scaling グループの唯一のパラメータとしてYAMLファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml ファイルには次のコンテンツが含まれます。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
```

```
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

例 2: **capacity-optimized-prioritized** 割り当て戦略を使用して スポットインスタンス を起動する

次のcreate-auto-scaling-groupコマンドは、以下を指定する Auto Scaling グループを作成します。

- オンデマンドインスタンスとして起動するグループの割合 (0) と開始時のオンデマンドインスタンスのベース数 (1)
- 優先度に従って起動するインスタンスタイプ (c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)
- インスタンスを起動するサブネット (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。それぞれが異なるアベイラビリティーゾーンに対応します。
- 起動テンプレート (my-launch-template) とそのバージョン ($Latest)。

Amazon EC2 Auto Scaling がオンデマンド容量を満たす場合、最初にc5.largeインスタンスタイプを起動します。Amazon EC2 Auto Scaling がスポットキャパシティーを達成しようとすると、ベストエフォートベースでインスタンスタイプの優先順位が優先されます。ただし、最初にキャパシティを最適化します。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
```

```
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized-prioritized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

または、次のcreate-auto-scaling-groupコマンドを使用して Auto Scaling グループを作成することもできます。これは、Auto Scaling グループの唯一のパラメータとしてYAMLファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml ファイルには次のコンテンツが含まれます。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized-prioritized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

例 3: 2つのプール間での **lowest-price** 配分戦略を使用してスポットインスタンスを起動する

次のcreate-auto-scaling-groupコマンドは、以下を指定する Auto Scaling グループを作成します。

- オンデマンドインスタンスとして起動するグループの割合 (50)。(これは、開始時のオンデマンドインスタンスのベース数を指定するものではありません)。
- 優先度に従って起動するインスタンスタイプ (c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large

- インスタンスを起動するサブネット (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。それぞれが異なるアベイラビリティーゾーンに対応します。
- 起動テンプレート (my-launch-template) とそのバージョン ($Latest)。

Amazon EC2 Auto Scaling がオンデマンド容量を満たす場合、最初にc5.largeインスタンスタイプを起動します。スポットキャパシティーの場合、Amazon EC2 Auto Scaling は、各アベイラビリティーゾーンの 2 つの最低価格のプールでスポットインスタンスを均等に起動しようとします。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
```

```
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "lowest-price",
            "SpotInstancePools": 2
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

または、次のcreate-auto-scaling-groupコマンドを使用して Auto Scaling グループを作成することもできます。これは、Auto Scaling グループの唯一のパラメータとして YAML ファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml ファイルには次のコンテンツが含まれます。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
```

```
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: lowest-price
    SpotInstancePools: 2
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

例 4: **price-capacity-optimized** 配分戦略を使用して スポットインスタンス を起動する

次のcreate-auto-scaling-groupコマンドは、以下を指定する Auto Scaling グループを作成します。

- オンデマンドインスタンスとして起動するグループの割合 (30)。(これは、開始時のオンデマンドインスタンスのベース数を指定するものではありません)。
- 優先度に従って起動するインスタンスタイプ (c5.large、c5a.large、m5.large、m5a.large、c4.large、m4.large、c3.large、m3.large)
- インスタンスを起動するサブネット (subnet-5ea0c127、subnet-6194ea3b、subnet-c934b782)。それぞれが異なるアベイラビリティーゾーンに対応します。
- 起動テンプレート (my-launch-template) とそのバージョン ($Latest)。

Amazon EC2 Auto Scaling がオンデマンドキャパシティーの達成を試みると、まずc5.largeインスタンスタイプを起動します。スポットキャパシティーの場合、Amazon EC2 Auto Scaling は可能な限り低価格で、起動するインスタンスの数に最適なキャパシティーを持つスポットインスタンスプールからスポットインスタンスを起動しようとします。

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
```

```
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 30,
            "SpotAllocationStrategy": "price-capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

または、次のcreate-auto-scaling-groupコマンドを使用して Auto Scaling グループを作成することもできます。これは、Auto Scaling グループの唯一のパラメータとして YAML ファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

config.yaml ファイルには次のコンテンツが含まれます。

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 30
    SpotAllocationStrategy: price-capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

インスタンスの重みを使用するように Auto Scaling グループを設定する

複数のインスタンスタイプを使用する場合、各インスタンスタイプに関連付けるユニット数を指定し、同じ測定単位を持つグループの容量を指定できます。この容量の仕様オプションは重みと呼ばれます。

例えば、少なくとも 8 vCPUs ~ 15 GiB の で最高のパフォーマンスを発揮するコンピューティング集約型アプリケーションを実行するとしますRAM。をベースユニットc5.2xlargeとして使用すると、次のいずれかのEC2インスタンスタイプがアプリケーションのニーズを満たします。

インスタンスタイプの例

インスタンスタイプ	vCPU	メモリ (GiB)
c5.2xlarge	8	16
c5.4xlarge	16	32
c5.12xlarge	48	96
c5.18xlarge	72	144
c5.24xlarge	96	192

デフォルトでは、すべてのインスタンスタイプは、サイズにかかわらず同じ重みを持ちます。つまり、Amazon EC2 Auto Scaling がラージインスタンスタイプを起動するかスモールインスタンスタイプを起動するかにかかわらず、各インスタンスは Auto Scaling グループの希望する容量に対して同じ数をカウントします。

ただし重みでは、各インスタンスタイプに関連付けるユニットを数値で指定します。例えば、インスタンスのサイズが異なる場合、c5.2xlarge インスタンスには 2 の重みを付け、c5.4xlarge (2 倍大きい) インスタンスには 4 の重みを付けます。次に、Amazon EC2 Auto Scaling がグループをスケーリングすると、これらの重みは、各インスタンスが希望する容量にカウントされる単位数に変換されます。

重みは、Amazon EC2 Auto Scaling が起動することを選択するインスタンスタイプを変更しません。代わりに、割り当て戦略によって変更されます。詳細については、「複数のインスタンスタイプの配分戦略」を参照してください。

> **Important**
>
> 各インスタンスタイプのメモリ数 vCPUs またはメモリ量を使用して、希望する容量を満たすように Auto Scaling グループを設定するには、属性ベースのインスタンスタイプの選択を使用することをお勧めします。DesiredCapacityType パラメータを設定すると、このパラメータに設定した値に基づいて、各インスタンスタイプに関連付けるユニット数が自動的

に指定されます。詳細については、「属性ベースのインスタンスタイプの選択を使用して混合インスタンスグループを作成する」を参照してください。

内容

考慮事項

このセクションでは、重みを効果的に実装するための重要な考慮事項について説明します。

- アプリケーションのパフォーマンスニーズに合ったインスタンスタイプをいくつか選択します。Auto Scaling グループの希望する容量に対して、機能に基づきカウントされる各インスタンスタイプの重みを決定します。これらの重みは、現在および今後のインスタンスに適用されます。
- 重みの範囲を大きくすることは避けてください。たとえば、インスタンスタイプの重みに 1 を指定し、次に大きいインスタンスタイプの重みに 200 を指定しないでください。最小の重みと最大の重みの差も極端であってはなりません。重みの違いが大きいと、コストパフォーマンスの最適化に悪影響を及ぼす可能性があります。
- グループの希望する容量をインスタンスではなくユニットで指定します。例えば、v CPUベースの重みを使用する場合は、必要なコア数と最小値と最大値を設定します。
- 重みと希望するキャパシティを設定して、希望するキャパシティが最も大きい重みの少なくとも 2〜3 倍になるようにします。

既存のグループを更新する際には、次の点に注意してください。

- 既存のグループに重みを追加する際は、現在使用中のすべてのインスタンスタイプの重みを含めます。
- 重みを追加または変更すると、Amazon EC2 Auto Scaling は新しい重み値に基づいて希望する容量に達するようにインスタンスを起動または終了します。
- インスタンスタイプを削除した場合、そのタイプのインスタンスを実行すると、定義されなくなった場合でも最新の重みが維持されます。

インスタンスの重みの動作

インスタンスの重みを使用すると、Amazon EC2 Auto Scaling は次のように動作します。

- 現在のキャパシティーは、希望するキャパシティーと同じかそれ以上になります。起動されたインスタンスが残りの希望する容量ユニットを超える場合、現在の容量が希望する容量を超える可能性があります。例えば、2 つのインスタンスタイプ `c5.2xlarge` と `c5.12xlarge` を指定し、`c5.2xlarge` にインスタンスの重み 2 を割り当て、`c5.12xlarge` にインスタンスの重み 12 を割り当てるとします。希望する容量を満たすためのユニットが 5 つ残っており、Amazon EC2 Auto Scaling が をプロビジョニングすると`c5.12xlarge`、希望する容量は 7 ユニット超過します。
- インスタンスを起動する場合、Amazon EC2 Auto Scaling は、アベイラビリティーゾーン間で容量を分散し、希望する容量を超えるよりも配分戦略を優先します。
- Amazon EC2 Auto Scaling は、任意の配分戦略を使用して、アベイラビリティーゾーン間のバランスを維持するために最大容量制限を超える可能性があります。Amazon EC2 Auto Scaling によって適用されるハード制限は、希望する容量と最大の重みを加えたものです。

重みを使用するように Auto Scaling グループを設定する

次の AWS CLI の例に示すように、重みを使用するように Auto Scaling グループを設定できます。コンソールを使用する手順については、「インスタンスタイプを手動で選択して混合インスタンスグループを作成する」を参照してください。

重みを使用するように新しい Auto Scaling グループを設定するには (AWS CLI)

create-auto-scaling-group コマンドを使用します。例えば、次のコマンドは新しい Auto Scaling グループを作成し、次を指定して重みを割り当てます。

- オンデマンドインスタンスとして起動するグループの割合 (`0`)
- 各アベイラビリティーゾーンのスポットインスタンスの配分戦略 (`capacity-optimized`)
- 優先度に従って起動するインスタンスタイプ (`m4.16xlarge`、`m5.24xlarge`)
- インスタンスタイプ間の相対的なサイズ差 (vCPUs) に対応するインスタンスの重み (16、24)
- インスタンスを起動するサブネット (`subnet-5ea0c127`、`subnet-6194ea3b`、`subnet-c934b782`)。それぞれ異なるアベイラビリティーゾーンに対応
- 起動テンプレート (`my-launch-template`) とそのバージョン (`$Latest`)

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "m4.16xlarge",
                    "WeightedCapacity": "16"
                },
                {
                    "InstanceType": "m5.24xlarge",
                    "WeightedCapacity": "24"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 160,
    "MaxSize": 720,
    "DesiredCapacity": 480,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

重みを使用するように既存の Auto Scaling グループを設定するには (AWS CLI)

update-auto-scaling-group コマンドを使用します。例えば、ここで示すコマンドは、次を指定して既存の Auto Scaling グループのインスタンスタイプに重みを割り当てます。

- 優先度に従って起動するインスタンスタイプ (c5.18xlarge、c5.24xlarge、c5.2xlarge、c5.4xlarge)

- インスタンスタイプ間の相対的なサイズ差 (vCPUs) に対応するインスタンスの重み (18、24、2、4)
- 増やす新しい希望するキャパシティー (最大重みよりも大きい)

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
    "AutoScalingGroupName": "my-existing-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "Overrides": [
                {
                    "InstanceType": "c5.18xlarge",
                    "WeightedCapacity": "18"
                },
                {
                    "InstanceType": "c5.24xlarge",
                    "WeightedCapacity": "24"
                },
                {
                    "InstanceType": "c5.2xlarge",
                    "WeightedCapacity": "2"
                },
                {
                    "InstanceType": "c5.4xlarge",
                    "WeightedCapacity": "4"
                }
            ]
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 100
}
```

コマンドラインを使用して重みを検証するには

以下のいずれかのコマンドを使用します。

- describe-auto-scaling-groups (AWS CLI)
- 取得 -ASAutoScalingGroup (AWS Tools for Windows PowerShell)

ユニット時間あたりのスポット料金の例

次の表では、米国東部 (バージニア北部) の異なるアベイラビリティーゾーンでのスポットインスタンスの時間単位の使用料金と、同じリージョンでのオンデマンドインスタンスの使用料金を比較しています。ここで示している料金は例であり、現在の料金ではありません。これらはインスタンス時間単位のコストです。

例: インスタンス時間単位のスポット料金

インスタンスタイプ	us-east-1a	us-east-1b	us-east-1c	オンデマンド料金
c5.2xlarge	0.180 USD	0.191 USD	0.170 USD	0.34 USD
c5.4xlarge	0.341 USD	0.361 USD	0.318 USD	0.68 USD
c5.12xlarge	0.779 USD	0.777 USD	0.777 USD	2.04 USD
c5.18xlarge	1.207 USD	1.475 USD	1.357 USD	3.06 USD
c5.24xlarge	1.555 USD	1.555 USD	1.555 USD	4.08 USD

インスタンスの重みにより、ユニット時間あたりの使用料金に基づいてコストを評価できます。時間単位の使用料金はインスタンスタイプの料金をその単位となる時間数で割って決定できます。オンデマンドインスタンスの場合、時間単位の使用料金は、1 つのインスタンスタイプをデプロイするときと、異なるサイズの同じインスタンスタイプをデプロイするときで同じです。一方、時間単位のスポット料金はスポットプールによって異なります。

次の例は、ユニット時間あたりのスポット料金の計算がインスタンスの重みでどのように機能するかを示しています。計算が容易になるように、スポットインスタンスを us-east-1a でのみ起動するとします。ユニット時間あたりの料金を次の表に示します。

例: ユニット時間あたりのスポット料金

インスタンスタイプ	us-east-1a	インスタンスの重み	ユニット時間あたりの価格
c5.2xlarge	0.180 USD	2	0.090 USD
c5.4xlarge	0.341 USD	4	0.085 USD
c5.12xlarge	0.779 USD	12	0.065 USD
c5.18xlarge	1.207 USD	18	0.067 USD
c5.24xlarge	1.555 USD	24	0.065 USD

インスタンスタイプに異なる起動テンプレートを使用する

複数のインスタンスタイプを使用するだけでなく、複数の起動テンプレートを使用することもできます。

例えば、コンピューティング集約型アプリケーション用に Auto Scaling グループを設定し、C5、C5a、C6g のインスタンスタイプを混在させたいとします。ただし、C6g インスタンスは 64 ビット Arm アーキテクチャに基づく AWS Graviton プロセッサを搭載し、C5 および C5a インスタンスは 64 ビット Intel x86 プロセッサで実行されます。C5 および C5a インスタンスAMIsの は、これらの各インスタンスで動作しますが、C6g インスタンスでは動作しません。この問題を解決するには、C6g インスタンス用に別の起動テンプレートを使用します。C5 インスタンスと C5a インスタンスには同じ起動テンプレートを引き続き使用できます。

このセクションでは、 を使用して AWS CLI 、複数の起動テンプレートの使用に関連するタスクを実行する手順について説明します。現在、この機能は、 AWS CLI または を使用する場合にのみ使用できSDK、 コンソールからは利用できません。

内容

- 複数の起動テンプレートを使用するように Auto Scaling グループを設定する
- 関連リソース

複数の起動テンプレートを使用するように Auto Scaling グループを設定する

次の例に示すように、複数の起動テンプレートを使用するように Auto Scaling グループを設定できます。

複数の起動テンプレートを使用するように新しい Auto Scaling グループを設定するには (AWS CLI)

create-auto-scaling-group コマンドを使用します。例えば、次のコマンドは新しい Auto Scaling グループを作成します。`c5.large`、、および `c6g.large`インスタンスタイプを指定し`c5a.large`、`c6g.large`インスタンスタイプの新しい起動テンプレートを定義して、適切な AMI が Arm インスタンスの起動に使用されるようにします。Amazon EC2 Auto Scaling は、インスタンスタイプの順序を使用して、オンデマンド容量を満たすときに最初に使用するインスタンスタイプを決定します。

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` ファイルには次のコンテンツが含まれます。

```
{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
    "LaunchTemplate":{
      "LaunchTemplateSpecification":{
        "LaunchTemplateName":"my-launch-template-for-x86",
        "Version":"$Latest"
      },
      "Overrides":[
        {
          "InstanceType":"c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        }
      ]
    },
```

```
    "InstancesDistribution":{
      "OnDemandBaseCapacity": 1,
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize":1,
  "MaxSize":5,
  "DesiredCapacity":3,
  "VPCZoneIdentifier":"subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags":[ ]
}
```

複数の起動テンプレートを使用するように既存の Auto Scaling グループを設定するには (AWS CLI)

update-auto-scaling-group コマンドを使用します。例えば、次のコマンドは、*my-launch-template-for-arm* という名前の起動テンプレートを、*my-asg* という名前の Auto Scaling グループの *c6g.large* インスタンスタイプに割り当てます。

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

config.json ファイルには次のコンテンツが含まれます。

```
{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
    "LaunchTemplate":{
      "Overrides":[
        {
          "InstanceType":"c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        }
      ]
```

```
    }
  }
}
```

Auto Scaling グループの起動テンプレートを確認するには

以下のいずれかのコマンドを使用します。

- describe-auto-scaling-groups (AWS CLI)
- 取得 -ASAutoScalingGroup (AWS Tools for Windows PowerShell)

関連リソース

AWS re:Post のテンプレートで、属性ベースのインスタンスタイプの選択を使用して複数の起動 AWS CloudFormation テンプレートを指定する例を示します。

起動設定を使用して Auto Scaling グループを作成する

> **Important**
>
> 2022 年 12 月 31 日以降にリリースされた新しい Amazon EC2インスタンスタイプ`CreateLaunchConfiguration`で を呼び出すことはできません。 また、2023 年 6 月 1 日以降に作成された新しいアカウントには、コンソールから新しい起動設定を作成することはできません。2024 年 10 月 1 日以降、新しいアカウントは、コンソール、、API、CLIおよび を使用して新しい起動設定を作成できなくなります CloudFormation。起動テンプレートに移行すると、現在または将来的に新しい起動設定を作成する必要がなくなります。Auto Scaling グループを移行してテンプレートを起動する方法については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

起動設定またはEC2インスタンスを作成した場合は、EC2インスタンスの設定テンプレートとして起動設定を使用する Auto Scaling グループを作成できます。起動設定では、インスタンスの AMI ID、インスタンスタイプ、キーペア、セキュリティグループ、ブロックデバイスマッピングなどの情報を指定します。起動設定の作成については、「「起動設定を作成する」」を参照してください。

Auto Scaling グループを作成するための十分な許可が必要です。また、Amazon EC2 Auto Scaling がユーザーに代わってアクションを実行するために使用するサービスにリンクされたロールがまだ存在

しない場合は、作成するための十分なアクセス許可が必要です。管理者がアクセス許可を付与するためのリファレンスとして使用できるIAMポリシーの例については、「」を参照してくださいアイデンティティベースのポリシーの例。

内容

- 起動設定を使用して Auto Scaling グループを作成する
- を使用して既存のインスタンスから Auto Scaling グループを作成する AWS CLI

起動設定を使用して Auto Scaling グループを作成する

> **Important**
>
> 起動設定に関する情報は、起動設定から起動テンプレートにまだ移行していないお客様向けに提供しています。Auto Scaling グループを移行してテンプレートを起動する方法については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

Auto Scaling グループを作成する際、Amazon EC2 インスタンスを設定するために必要な情報、そのインスタンスのアベイラビリティーゾーンと VPC サブネット、希望するキャパシティー、キャパシティー制限の最小値と最大値を指定する必要があります。

次の手順では、起動設定を使用して Auto Scaling グループを作成する方法を説明します。起動設定は作成後に変更することはできませんが、Auto Scaling グループの起動設定を置き換えることはできます。詳細については、「Auto Scaling グループの起動設定を変更する」を参照してください。

前提条件

- 起動設定を作成しておく必要があります。詳細については、「「起動設定を作成する」」を参照してください。

起動設定 (コンソール) を使用して Auto Scaling グループを作成するには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 画面の上部のナビゲーションバーで、起動設定を作成したときに使用したのと同じ AWS リージョン を選択します。
3. [Auto Scaling グループの作成] を選択します。

4. [起動テンプレートまたは起動設定を選択する] ページで [Auto Scaling グループ名] にAuto Scaling グループの名前を入力します。
5. 起動設定を選択するには、次の手順を実行します。

   a. [Launch template] (起動テンプレート) で、[Switch to launch configuration] (起動設定に切り替える) を選択します。

   b. [起動設定] で、既存の起動設定を選択します。

   c. 起動設定が、使用する予定のすべてのオプションをサポートしていることを確認し、[次へ] を選択します。
6. (設定の構成)、[Configure instance launch options] (インスタンス起動オプションの設定) ページの [Network] (ネットワーク) にある [VPC] で、VPC を選択します。Auto Scaling グループは、起動設定で指定したセキュリティグループと、同じ VPC 内に作成する必要があります。
7. (サブネット)、[Availability Zones and subnets] (アベイラビリティーゾーンとサブネット) で、指定した VPC 内のサブネットを 1 つ以上選択します。複数のアベイラビリティーゾーンのサブネットを使用することで、高可用性を得られます。詳細については、「VPC サブネットを選択する際の考慮事項」を参照してください。
8. [Next] を選択します。

   または、残りはデフォルトのままにして、[Skip to Review (確認をスキップ)] を選択できます。
9. (オプション) [詳細オプションの設定] ページで、次のオプションを設定し、[次へ] を選択します。

   a. (オプション) [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Amazon EBS ヘルスチェックをオンにする] を選択します。詳細については、「ヘルスチェックを使用して、Amazon EBS ボリュームに障害がある Auto Scaling インスタンスをモニタリングする」を参照してください。

   b. (オプション) [ヘルスチェックの猶予期間] に秒単位で時間を入力します。これは、インスタンスが `InService` 状態になった後で、Amazon EC2 Auto Scaling がインスタンスのヘルスステータスのチェックを待つ必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

   c. [Additional settings] (追加設定)、[Monitoring] (モニタリング) で、CloudWatch グループメトリクスの収集を有効にするかどうかを選択します。これらのメトリクスは、終了インスタンス数や保留中のインスタンスの数など、潜在的な問題の指標となる測定値を提供します。詳細については、「Auto Scaling グループとインスタンスの CloudWatch メトリクスを監視する」を参照してください。

d. [デフォルトのインスタンスのウォームアップを有効にする] でこのオプションを選択し、アプリケーションのウォームアップ時間を選択します。スケーリングポリシーを持つ Auto Scaling グループを作成している場合は、インスタンスのデフォルトウォームアップ機能によって、動的スケーリングに使用される Amazon CloudWatch メトリクスが改善されます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

10. (オプション) [Configure group size and scaling policies (グループサイズとスケーリングポリシーの設定)] ページで、次のオプションを設定し、[次へ] を選択します。

    a. [グループサイズ] の [希望する容量] に、起動するインスタンスの初期数を入力します。

    b. [スケーリング] セクションの [スケーリング制限] で、[希望する容量] の新しい値が [最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に希望する新しい容量の値に引き上げられます。これらの制限は、必要に応じて変更できます。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。

    c. [自動スケーリング] で、ターゲット追跡スケーリングポリシーを作成するかどうかを選択します。このポリシーは、Auto Scaling グループの作成後に作成することもできます。

    [ターゲット追跡スケーリングポリシー] を選択した場合は、「ターゲット追跡スケーリングポリシーを作成する」の指示に従ってポリシーを作成してください。

    d. [インスタンスメンテナンスポリシー] で、インスタンスメンテナンスポリシーを作成するかどうかを選択します。このポリシーは、Auto Scaling グループの作成後に作成することもできます。ポリシーを作成するには、「インスタンスメンテナンスポリシーを設定する」の手順を実行してください。

    e. [Instance scale-in protection (インスタンスのスケールイン保護)] で、インスタンスのスケールイン保護を有効にするかどうかを選択します。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

11. (オプション) 通知を受け取るには、[通知の追加] を選択し、通知を設定してから [次へ] を選択します。詳細については、「Amazon EC2 Auto Scaling の Amazon SNS 通知オプション」を参照してください。

12. (オプション) タグを追加するには、[タグの追加] を選択し、各タグのタグキーと値を指定し、[次へ] を選択します。詳細については、「Auto Scaling グループとインスタンスにタグを付ける」を参照してください。

13. [Review (レビュー)]ページで、[Create Auto Scaling group (Auto Scaling グループを作成)] を選択します。

コマンドラインを使用して Auto Scaling グループを作成するには

以下のコマンドのいずれかを使用できます。

- create-auto-scaling-group (AWS CLI)
- New-ASAutoScalingGroup (AWS Tools for Windows PowerShell)

を使用して既存のインスタンスから Auto Scaling グループを作成する AWS CLI

> **Important**
>
> 起動設定に関する情報は、起動設定から起動テンプレートにまだ移行していないお客様向けに提供しています。Auto Scaling グループを移行してテンプレートを起動する方法については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

Auto Scaling グループを初めて作成する場合は、コンソールを使用して既存のEC2インスタンスから起動テンプレートを作成することをお勧めします。次に、起動テンプレートを使用して新しい Auto Scaling グループを作成します。詳しい手順については、「Amazon EC2 起動ウィザードを使用して Auto Scaling グループを作成する」を参照してください。

次の手順は、他のインスタンスを起動するためのベースとして使用する、既存のインスタンスを指定して Auto Scaling グループを作成する方法を示しています。Amazon マシンイメージ (AMI) ID、EC2インスタンスタイプ、キーペア、セキュリティグループなど、インスタンスを作成するには複数のパラメータが必要です。この情報はすべて、Amazon EC2 Auto Scaling がスケーリングが必要な場合にユーザーに代わってインスタンスを起動するためにも使用されます。この情報は、起動テンプレートまたは起動設定のいずれかに保存されます。

既存のインスタンスを使用する場合、Amazon EC2 Auto Scaling は、同時に作成された起動設定に基づいてインスタンスを起動する Auto Scaling グループを作成します。新しい起動設定には Auto Scaling グループと同じ名前がついていて、インスタンスからの特定の設定についての詳細が含まれます。

次の設定詳細は、特定のインスタンスから起動設定にコピーされます。

- AMI ID
- インスタンスタイプ

- キーペア
- セキュリティグループ
- IP アドレスタイプ (パブリックまたはプライベート)
- IAM 該当する場合は、 インスタンスプロファイル
- モニタリング (true または false)
- EBS 最適化 (true または false)
- で起動する場合のテナンシー設定 VPC (共有または専用)
- 該当する場合は、カーネル ID とRAMディスク ID
- ユーザーデータ (指定された場合)
- スポット (最大) 料金

VPC サブネットとアベイラビリティーゾーンは、識別されたインスタンスから Auto Scaling グループ独自のリソース定義にコピーされます。

特定されたインスタンスがプレイスメントグループ内にある場合、新しい Auto Scaling グループは、特定されたインスタンスと同じプレイスメントグループ内でインスタンスを起動します。起動設定ではプレイスメントグループの指定が許可されないため、プレイスメントグループは新しい Auto Scaling グループの `PlacementGroup` 属性にコピーされます。

次の設定の詳細は、特定のインスタンスからはコピーされません。

- ストレージ: ブロックデバイス (EBS ボリュームとインスタンスストアボリューム) は、識別されたインスタンスからコピーされません。代わりに、 の作成の一部として作成されたブロックデバイスマッピングによって、使用するデバイスがAMI決まります。
- ネットワークインターフェイスの数: ネットワークインターフェイスは、特定のインスタンスからコピーされません。代わりに、Amazon EC2 Auto Scaling はデフォルト設定を使用して、プライマリネットワークインターフェイス (eth0) である 1 つのネットワークインターフェイスを作成します。
- インスタンスメタデータオプション: アクセス可能なメタデータ、メタデータのバージョン、およびトークンレスポンスのホップ制限の設定は、特定のインスタンスからコピーされません。代わりに、Amazon EC2 Auto Scaling はデフォルト設定を使用します。詳細については、「インスタンスメタデータオプションの設定」を参照してください。
- ロードバランサー: 識別されたインスタンスが 1 つ以上のロードバランサーに登録されている場合、このロードバランサーの情報は新しい Auto Scaling グループのロードバランサーあるいはターゲットグループ属性にコピーされません。

- タグ: 特定されたインスタンスにタグが指定されている場合、そのタグは新しい Auto Scaling グループの Tags 属性にはコピーされません。

前提条件

EC2 インスタンスは、次の基準を満たしている必要があります。

- インスタンスは他の Auto Scaling グループのメンバーではありません。
- インスタンスが running 状態であること。
- インスタンスの起動にAMI使用された がまだ存在している必要があります。

EC2 インスタンスから Auto Scaling グループを作成する (AWS CLI)

次の手順では、 CLI コマンドを使用してEC2インスタンスから Auto Scaling グループを作成する方法を示します。

この手順では、Auto Scaling グループにインスタンスを追加しません。Auto Scaling グループを作成した後、アタッチするインスタンスに attach-instances コマンドを実行する必要があります。

開始する前に、Amazon EC2コンソールまたは describe-instances コマンドを使用してEC2インスタンスの ID を見つけます。

現在のインスタンスをテンプレートとして使用するには

- 次のcreate-auto-scaling-groupコマンドを使用して、EC2インスタンス my-asg-from-instanceから Auto Scaling グループ を作成しますi-123456789abcdefg0。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \
  --instance-id i-123456789abcdefg0 --min-size 1 --max-size 2 --desired-capacity 2
```

Auto Scaling グループがインスタンスを起動したことを確認するには

- 次のdescribe-auto-scaling-groupsコマンドを使用して、Auto Scaling グループが正常に作成されたことを確認します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg-from-instance
```

次のレスポンスの例は、グループの希望するキャパシティーが 2 であり、グループには実行中のインスタンスが 2 つあり、起動設定の名前が my-asg-from-instance であることを示しています。

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg-from-instance",
      "AutoScalingGroupARN":"arn",
      "LaunchConfigurationName":"my-asg-from-instance",
      "MinSize":1,
      "MaxSize":2,
      "DesiredCapacity":2,
      "DefaultCooldown":300,
      "AvailabilityZones":[
        "us-west-2a"
      ],
      "LoadBalancerNames":[],
      "TargetGroupARNs":[],
      "HealthCheckType":"EC2",
      "HealthCheckGracePeriod":0,
      "Instances":[
        {
          "InstanceId":"i-34567890abcdef012",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        },
        {
          "InstanceId":"i-012345abcdefg6789",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        }
      ],
      "CreatedTime":"2020-10-28T02:39:22.152Z",
```

```
      "SuspendedProcesses":[ ],
      "VPCZoneIdentifier":"subnet-0abc1234",
      "EnabledMetrics":[ ],
      "Tags":[ ],
      "TerminationPolicies":[
        "Default"
      ],
      "NewInstancesProtectedFromScaleIn":false,
      "ServiceLinkedRoleARN":"arn",
      "TrafficSources":[]
    }
  ]
}
```

起動設定を表示するには

- 起動設定の詳細を表示するには、次のdescribe-launch-configurationsコマンドを使用します。

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-asg-from-instance
```

出力例を次に示します。

```
{
  "LaunchConfigurations":[
    {
      "LaunchConfigurationName":"my-asg-from-instance",
      "LaunchConfigurationARN":"arn",
      "ImageId":"ami-234567890abcdefgh",
      "KeyName":"my-key-pair-uswest2",
      "SecurityGroups":[
        "sg-12abcdefgh3456789"
      ],
      "ClassicLinkVPCSecurityGroups":[ ],
      "UserData":"",
      "InstanceType":"t2.micro",
      "KernelId":"",
      "RamdiskId":"",
      "BlockDeviceMappings":[ ],
      "InstanceMonitoring":{
        "Enabled":true
      },
```

```
      "CreatedTime":"2020-10-28T02:39:22.321Z",
      "EbsOptimized":false,
      "AssociatePublicIpAddress":true
    }
  ]
}
```

インスタンスを終了するには

- 必要がなくなった場合は、インスタンスを終了できます。以下の terminate-instances コマンドで、インスタンスを終了しますi-123456789abcdefg0。

```
aws ec2 terminate-instances --instance-ids i-123456789abcdefg0
```

Amazon EC2インスタンスを終了した後は、インスタンスを再起動できません。削除後、ボリュームに含まれるデータは消去され、ボリューム自体はどのインスタンスにもアタッチできなくなります。インスタンスの終了の詳細については、「Amazon EC2ユーザーガイド」の「インスタンスの終了」を参照してください。

Auto Scaling グループを更新する

Auto Scaling グループの詳細は更新することができます。Auto Scaling グループの名前を更新したり、変更したりすることはできません AWS リージョン。

Auto Scaling グループ (コンソール) を更新するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループを選択すると、[詳細]、[アクティビティ]、[自動スケーリング]、[インスタンス管理]、[モニタリング]、[インスタンス更新] のタブでグループに関する情報が表示されます。
3. 目的の設定エリアのタブを選択し、必要に応じて設定を更新します。編集する各設定について、[更新] を選択して Auto Scaling グループの設定に変更を保存します。

   - [詳細] タブ

     以下は Auto Scaling グループの一般的な設定です。これらのルールは、Auto Scaling グループの作成時と同じ方法で編集および管理できます。

詳細設定セクションには、[終了ポリシー]、[クールダウン]、[一時停止プロセス]、[最大インスタンス有効期間]など、グループの作成時には使用できないオプションがいくつかあります。Auto Scaling グループのプレイスメントグループとサービスにリンクされたロールは表示することはできますが、編集することはできません。

グループが Elastic Load Balancing リソースに関連付けられている場合は、アベイラビリティーゾーンを変更する前に アベイラビリティーゾーンを追加する を参照してください。ロードバランサーの制限によっては、グループのアベイラビリティーゾーンへの変更をロードバランサーのアベイラビリティーゾーンに適用できない場合があります。

- [アクティビティ] タブ
  - アクティビティ通知 – Amazon SNS通知
- [自動スケーリング] タブ
  - [動的スケーリングポリシー] – 動的スケーリングポリシー
  - [予測スケーリングポリシー] – 予測スケーリングポリシー
  - [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
- [インスタンス管理] タブ
  - [ライフサイクルフック]—— ライフサイクルフック
  - [ウォームプール] – ウォームプール
- [モニタリング] タブ
  - このタブには 1 つのオプションのみがあり、CloudWatchグループメトリクスの収集を有効または無効にできます。

コマンドラインを使用して Auto Scaling グループを更新するには

以下のコマンドのいずれかを使用できます。

- update-auto-scaling-group (AWS CLI)
- 更新 -ASAutoScalingGroup (AWS Tools for Windows PowerShell)

Auto Scaling インスタンスの更新

新しい起動テンプレートまたは起動設定を Auto Scaling グループに関連付けると、すべての新しいインスタンスに更新された設定が適用されます。既存のインスタンスは、最初に起動された構成で実行され続けます。既存のインスタンスに変更を適用するには、次のオプションがあります。

- 古いインスタンスを置き換えるためにインスタンスの更新を開始します。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。
- スケーリングアクティビティが、終了ポリシーに基づいて古いインスタンスを新しいインスタンスに徐々に置き換えるのを待ちます。
- 手動で終了させて Auto Scaling グループに置き換えます。

> **Note**
>
> 以下のインスタンス属性を起動テンプレートまたは起動設定の一部として指定することで変更できます。
>
> - Amazon マシンイメージ (AMI)
> - ブロックデバイス
> - キーペア
> - インスタンスタイプ
> - セキュリティグループ
> - ユーザーデータ
> - モニタリング
> - IAM インスタンスプロファイル
> - プレースメントテナンシー
> - kernel
> - ラムディスク
> - インスタンスにパブリック IP アドレスがあるかどうか
> - アベイラビリティーゾーンの分散戦略

Auto Scaling グループとインスタンスにタグを付ける

タグは、 AWS リソースに割り当てる、または AWS に割り当てるカスタム属性ラベルです。各 タグは 2 つの部分で構成されます:

- タグキー (例: `costcenter`、`environment` または `project`)
- タグ値として知られるオプションのフィールド (例: `111122223333` または `production`)

タグは、以下のことに役立ちます。

- AWS コストを追跡します。 AWS Billing and Cost Management ダッシュボードでこれらのタグをアクティブ化します。 AWS はタグを使用してコストを分類し、毎月のコスト配分レポートを配信します。詳細については、AWS Billing ユーザーガイド の「コスト配分タグの使用」を参照してください。
- タグに基づいて、Auto Scaling グループへのアクセスを制御します。IAM ポリシーの条件を使用して、そのグループのタグに基づいて Auto Scaling グループへのアクセスを制御できます。詳細については、「セキュリティ用のタグ」を参照してください。
- 追加したタグに基づく Auto Scaling グループのフィルタリングと検索。詳細については、「タグを使用して Auto Scaling グループをフィルタリングする」を参照してください。
- AWS リソースを特定して整理します。多くの はタグ付け AWS のサービス をサポートしているため、異なる サービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。

新規または既存の Auto Scaling グループにタグを付けることができます。Auto Scaling グループから起動するEC2インスタンスにタグを伝播することもできます。

タグは Amazon EBSボリュームには伝達されません。Amazon EBSボリュームにタグを追加するには、起動テンプレートでタグを指定します。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。

タグは、 AWS Management Console、 AWS CLI、または を使用して作成および管理できます SDKs。

内容

タグの命名と使用制限

タグには以下のようなベーシック制限があります。

- リソースあたりのタグの最大数は 50 です。
- 単一の呼び出しを使用して追加または削除できるタグの最大数は 25 です。
- キーの最大長は Unicode 文字で 128 文字です。
- 値の最大長は Unicode 文字で 256 文字です。
- タグのキーと値は大文字と小文字が区別されます。ベストプラクティスとして、タグを大文字にするための戦略を決定し、その戦略をすべてのリソースタイプにわたって一貫して実装します。
- タグ名または値に `aws:` プレフィックスを使用しないでください。このプレフィックスは AWS 用に予約されています。このプレフィックスが含まれるタグの名前または値は編集または削除できません。これらはリソースクォータあたりのタグに対して計算されません。

EC2 インスタンスのタグ付けライフサイクル

EC2 インスタンスにタグを付けることを選択した場合、タグは以下のように管理されます。

- Auto Scaling グループがインスタンスを起動すると、リソースの作成後ではなく作成中に、インスタンスにタグが追加されます。
- Auto Scaling グループでは、キーに `aws:autoscaling:groupName`、値に Auto Scaling グループ名が使用され、インスタンスにタグが自動的に追加されます。
- 起動テンプレートでインスタンスタグを指定し、グループのタグをそのインスタンスに伝播することを選択した場合、すべてのタグがマージされます。起動テンプレートのタグと Auto Scaling グループのタグに同じタグキーが指定されている場合、グループのタグ値が優先されます。
- 既存のインスタンスをアタッチするときに、Auto Scaling グループはタグをインスタンスに追加し、同じタグキーを持つ既存のタグを上書きします。また、キーが `aws:autoscaling:groupName` で、値が Auto Scaling グループ名のタグも追加されます。
- Auto Scaling グループからインスタンスからデタッチした場合は、`aws:autoscaling:groupName` タグのみが削除されます。

Auto Scaling グループにタグを付ける

Auto Scaling グループにタグを追加する際、Auto Scaling グループで起動するインスタンスに追加するかどうかを指定できます。タグを変更する場合は、変更後にその Auto Scaling グループで起動さ

れたインスタンスには更新されたタグのバージョンが追加されます。Auto Scaling グループのタグを作成または変更しても、これらの変更内容は既に Auto Scaling グループで実行中のインスタンスには加えられません。

内容

タグの追加または変更 (コンソール)

Auto Scaling グループの作成時にタグを付けるには

Amazon EC2コンソールを使用して Auto Scaling グループを作成する場合、Auto Scaling グループの作成ウィザードのタグの追加ページでタグキーと値を指定できます。Auto Scaling グループで起動されるインスタンスにタグを付けるには、[Tag New Instances (新しいインスタンスのタグ付け)] オプションが選択されたままにしてください。タグを付けない場合は、このオプションの選択を解除できます。

既存の Auto Scaling グループのタグを追加または変更するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[タグ]、[編集] の順に選択します。
4. 既存のタグを変更するには、[Key] と [Value] フィールドを編集します。
5. 新しいタグを追加するには、[Add tag] を選択し、[Key] と [Value] フィールドを選択します。[Tag New Instances (新しいインスタンスにタグ付けする)] を選択したままにして Auto Scaling グループで起動されるインスタンスに自動的にタグを追加することも、選択解除して追加しないこともできます。
6. タグの追加を完了したら、[保存] を選択します

タグの追加または変更 (AWS CLI)

次の例は、 を使用して Auto Scaling グループを作成するときにタグ AWS CLI を追加し、既存の Auto Scaling グループのタグを追加または変更する方法を示しています。

Auto Scaling グループの作成時にタグを付けるには

create-auto-scaling-group コマンドを使用して新しい Auto Scaling グループを作成し、**environment=production**Auto Scaling グループに などのタグを追加します。タグは、Auto Scaling グループで起動されるインスタンスにも追加されます。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config --min-size 1 --max-size 3 \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --tags Key=environment,Value=production,PropagateAtLaunch=true
```

既存の Auto Scaling グループのタグを作成または変更するには

create-or-update-tags コマンドを使用して、タグを作成または変更します。例えば、以下のコマンドは **Name=my-asg** および **costcenter=cc123** タグを追加します。この変更後、それらのタグは Auto Scaling グループ内で起動されるすべてのインスタンスに追加されます。いずれかのキーを持つタグがすでに存在する場合、既存のタグは置き換えられます。Amazon EC2コンソールは、各インスタンスの表示名をNameキーに指定された名前 (大文字と小文字を区別) に関連付けます。

```
aws autoscaling create-or-update-tags \
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-
asg,PropagateAtLaunch=true \
  ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=costcenter,Value=cc123,PropagateAtLaunch=true
```

Auto Scaling グループのタグを記述する (AWS CLI)

特定の Auto Scaling グループに適用されているタグを表示する場合は、次のいずれかのコマンドを使用できます。

- describe-tags – Auto Scaling グループ名を指定して、指定したグループのタグのリストを表示します。

  ```
  aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
  ```

  以下に、応答の例を示します。

  ```
  {
      "Tags": [
          {
              "ResourceType": "auto-scaling-group",
  ```

```
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "production",
            "Key": "environment"
        }
    ]
}
```

- describe-auto-scaling-groups – Auto Scaling グループ名を指定して、タグを含む指定されたグループの属性を表示します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下に、応答の例を示します。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            ...
            "Tags": [
                {
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "production",
                    "Key": "environment"
                }
            ],
            ...
        }
    ]
```

```
}
```

タグの削除

Auto Scaling グループに関連付けられたタグは、いつでも削除できます。

内容

- タグの削除 (コンソール)
- タグの削除 (AWS CLI)

タグの削除 (コンソール)

タグを削除するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[タグ]、[編集] の順に選択します。
4. タグの横にある [削除] を選択します。
5. [Update] (更新) を選択します。

タグの削除 (AWS CLI)

delete-tags コマンドを使用してタグを削除します。例えば、以下のコマンドは **environment** キーを持つタグを削除します。

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=environment"
```

タグのキーは指定する必要がありますが、値を指定する必要はありません。指定した値が正しくない場合、タグは削除されません。

セキュリティ用のタグ

タグを使用して、リクエスタ (IAM ユーザーやロールなど) に特定の Auto Scaling グループを作成、変更、または削除するためのアクセス許可があることを確認します。次の条件キーを 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 Auto Scaling グループ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、`autoscaling:ResourceTag/`*`tag-key`*`: `*`tag-value` を使用します。
- リクエストに特定のタグが含める (または含めない) ことを要求するには、`aws:RequestTag/`*`tag-key`*`: `*`tag-value` を使用します。
- リクエストに特定のタグキーを含める (または含めない) ことを要求するには、`aws:TagKeys [`*`tag-key`*`, ...]` を使用します。

例えば、以下の例に示すように、キー **environment** および 値 **production** を持つタグを含むすべての Auto Scaling グループへのアクセスを拒否できます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"autoscaling:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

条件キーを使用して Auto Scaling グループへのアクセスを制御する方法の詳細については、「Amazon EC2 Auto Scaling と IAM の連携方法」を参照してください。

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 (IAM ユーザーやロールなど) に Auto Scaling グループのタグを追加、変更、または削除するためのアクセス許可があることを確認します。

次のIAMポリシー例では、Auto Scaling グループから **temporary**キーを持つタグのみを削除するアクセス許可をプリンシパルに付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "autoscaling:DeleteTags",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary"] }
            }
        }
    ]
}
```

Auto Scaling グループに指定されたタグに制約を適用するIAMポリシーのその他の例については、「」を参照してください使用できるタグキーとタグ値を制御する。

> **Note**
>
> ユーザーによる Auto Scaling グループに対するタグ付け (または、タグの削除) 操作の実行を制限するポリシーを持っていても、ユーザーはインスタンスの起動後にタグを手動で変更できます。EC2 インスタンスのタグへのアクセスを制御する例については、「Amazon EC2 ユーザーガイド」の「例: リソースのタグ付け」を参照してください。

タグを使用して Auto Scaling グループをフィルタリングする

次の例は、 describe-auto-scaling-groups コマンドでフィルターを使用して、特定のタグを持つ Auto Scaling グループを記述する方法を示しています。タグによるフィルタリングは、 AWS CLI または に制限されておりSDK、コンソールからは利用できません。

フィルタリングの考慮事項

- 1 つのリクエストで複数のフィルターと複数のフィルタの値を指定できます。
- フィルターの値にワイルドカードを使用することはできません。
- フィルターの値は大文字と小文字が区別されます。

例: 特定のタグキーと値のペアを持つ Auto Scaling グループを記述する

次のコマンドは、**environment=production** のタグキーと値のペアを持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-value,Values=production
```

以下に、応答の例を示します。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            ...
            "Tags": [
                {
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "production",
                    "Key": "environment"
                }
            ],
            ...
        },

    ... additional groups ...
```

```
    ]
}
```

または tag:**<key>** フィルターを使用して、タグを指定することもできます。例えば、次のコマンドは、**environment=production** のタグキーと値のペアを持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。このフィルターは、次の形式になります: Name=tag:**<key>**,Values=**<value>** (**<key>** および **<value>** はタグキーと値のペアを表します。)

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production
```

--query オプションを使用して AWS CLI 出力をフィルタリングすることもできます。次の例は、前のコマンドの AWS CLI 出力をグループ名、最小サイズ、最大サイズ、および希望する容量属性のみに制限する方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production \
  --query "AutoScalingGroups[].{AutoScalingGroupName: AutoScalingGroupName, MinSize:
 MinSize, MaxSize: MaxSize, DesiredCapacity: DesiredCapacity}"
```

以下に、応答の例を示します。

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "MinSize": 0,
        "MaxSize": 10,
        "DesiredCapacity": 1
    },

    ... additional groups ...

]
```

フィルタリングの詳細については、「 AWS Command Line Interface ユーザーガイド」のAWS CLI「出力のフィルタリング」を参照してください。

例: 指定したタグキーと一致するタグを持つ Auto Scaling グループを記述する

次のコマンドは、タグ値に関係なく **environment** タグを持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment
```

例: 指定したタグキーのセットに一致するタグを持つ Auto Scaling グループを記述する

次のコマンドは、タグ値に関係なく **environment** および **project** のタグを持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-key,Values=project
```

例: 指定したタグキーのうち少なくとも 1 つに一致するタグを持つ Auto Scaling グループを記述する

次のコマンドは、タグ値に関係なく **environment** または **project** のタグを持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment,project
```

例: 指定したタグ値を持つ Auto Scaling グループを記述する

次のコマンドは、タグキーに関係なくタグ値 **production** を持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production
```

例: 指定したタグ値のセットを持つ Auto Scaling グループを記述する

次のコマンドは、タグキーに関係なくタグ値 **production** および **development** を持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production Name=tag-value,Values=development
```

例: 指定したタグ値の少なくとも 1 つに一致するタグを持つ Auto Scaling グループを記述する

次のコマンドは、タグキーに関係なくタグ値 **production** または **development** を持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production,development
```

例: 複数のタグキーおよびタグ値が一致するタグを持つ Auto Scaling グループを記述する

フィルターを組み合わせてカスタム AND and OR より複雑なフィルタリングを行うロジック。

次のコマンドは、特定のタグのセットを持つ Auto Scaling グループのみを表示するように、結果をフィルタリングする方法を示しています。1 つのタグキーは **environment** AND タグ値は (**production** OR **development**) AND もう 1 つのタグキーは **costcenter** AND タグ値は ですcc123。

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production,development
 Name=tag:costcenter,Values=cc123
```

インスタンスメンテナンスポリシー

インスタンスの更新やヘルスチェックプロセスなどのイベントが発生したときに、インスタンスの置き換え動作を制御するために、Auto Scaling グループのインスタンスメンテナンスポリシーを設定できます。

例えば、少数のインスタンスを含む Auto Scaling グループがあるとします。ヘルスチェックで障害のあるインスタンスが検出された際、インスタンスを終了して置き換えることによって発生する可能性のあるサービスの中断を避けたいと考えています。インスタンスメンテナンスポリシーを使用すると、Amazon EC2 Auto Scaling が最初に新しいインスタンスを起動し、それが完全に準備できるのを待ってから、異常なインスタンスを終了するようにすることができます。

インスタンスメンテナンスポリシーは、複数のインスタンスが同時に置き換えられた場合に発生する可能性のある中断を最小限に抑えるのに役立ちます。ポリシーの最小正常率と最大正常率のパラメータを設定すると、Auto Scaling グループはインスタンスを置き換えるときに、最小値と最大値の範囲内でしかキャパシティを増減できません。範囲を大きくすると、同時に置き換えることができるインスタンスの数が増えます。

内容

Auto Scaling グループのインスタンスメンテナンスポリシー

このトピックでは、使用可能なオプションの概要と、インスタンスメンテナンスポリシーの作成時に考慮すべき点を説明します。

内容

概要

Auto Scaling グループのインスタンスメンテナンスポリシーを作成すると、そのポリシーはインスタンスの置き換えを引き起こす Amazon EC2 Auto Scaling イベントに影響を与えます。これにより、同じ Auto Scaling グループ内でより一貫した置き換え動作が得られます。また、必要に応じて、可用性やコストに合わせてグループを最適化することもできます。

コンソールでは、次の設定オプションを使用できます。

- 終了前の起動 — 既存のインスタンスを終了する前に、新しいインスタンスをプロビジョニングする必要があります。このアプローチは、コスト削減よりも可用性を重視するアプリケーションに適しています。
- 終了と起動 — 新しいインスタンスは、既存のインスタンスが終了すると同時にプロビジョニングされます。このアプローチは、可用性よりもコスト削減を優先するアプリケーションに適しています。また、インスタンスを置き換える場合でも、現在利用可能な容量を超える容量を起動する必要がないアプリケーションにも適しています。
- カスタムポリシー – このオプションを使用すると、インスタンスを置き換えするときに必要なキャパシティの最小値と最大値の範囲をカスタムで指定してポリシーを設定できます。このアプローチは、コストと可用性の最適なバランスを重視する場合に効果的です。

Auto Scaling グループはデフォルトでインスタンスメンテナンスポリシーが設定されていないため、インスタンスのメンテナンスイベントが発生した場合、デフォルトの動作に従って処理されます。デフォルトの動作については、次の表で説明します。

インスタンスメンテナンスイベントのデフォルトの動作

イベント	説明	デフォルトの動作
ヘルスチェックの失敗	インスタンスがヘルスチェックに失敗すると自動的に発生します。Amazon EC2 Auto Scaling は、ヘルスチェックに失敗したインスタンスを置き換えます。ヘルスチェックの失敗の原因については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。	終了と起動。
インスタンスの更新	インスタンスの更新を開始すると実行されます。インスタンスの更新では、設定に応じてインスタンスを 1 つずつ、複数ずつ、またはすべてを一度に置き換えられます。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。	終了と起動。
最大インスタンス有効期間	インスタンスが Auto Scaling グループに指定したインスタンスの最大有効期間に達すると、自動的に実行されます。Amazon EC2 Auto Scaling は、インスタンスの最大有効期間に達したインスタ	終了と起動。

イベント	説明	デフォルトの動作
	ンスを置き換えます。詳細については、「インスタンスの最大存続期間に基づいて Auto Scaling インスタンスを置き換える」を参照してください。	

イベント	説明	デフォルトの動作
リバランシング	グループのバランスが崩れるような根本的な変更があった場合に自動的に実行されます。Amazon EC2 Auto Scaling は、次の状況でグループのバランスを調整します。 • 以前にキャパシティが不足していたアベイラビリティーゾーンが回復した場合、またはグループから可用性ゾーンを追加または削除した場合。この場合、Auto Scaling グループはアベイラビリティーゾーン間で均等にバランスを取るよう試みます。詳細については、「アクティビティの再分散」を参照してください。 • Auto Scaling グループでキャパシティの再調整を有効にし、スポットインスタンスの可用性が変化すると、既存のスポットインスタンスが中断される前に新しいスポットインスタンスを起動しようとします。詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。 • Auto Scaling グループを更新すると、混合インスタン	終了前の起動。 Amazon EC2 Auto Scaling は、最大キャパシティの 10% を超えて、グループのサイズを増やすことができます。ただし、キャパシティの再調整を使用している場合、これらの制限を超えることができるのは、希望するキャパシティの最大 10% のみです。

イベント	説明	デフォルトの動作
	スポリシーの更新時に選択した新しい購入オプションに合わせて既存のインスタンスが徐々に置き換わります。詳細については、「Auto Scaling グループを更新する」を参照してください。	

Amazon EC2 Auto Scaling は、以下の状況では引き続きデフォルトで終了および起動されます。したがって、これらの状況のいずれかが発生すると、グループのキャパシティがインスタンスメンテナンスポリシーの下限しきい値を下回る可能性があります。

- 例えば、人為的な操作などにより、インスタンスが予期せず終了した場合。Amazon EC2 Auto Scaling は、実行されなくなったインスタンスを直ちに置き換えます。詳細については、「Amazon EC2 ヘルスチェック」を参照してください。
- Amazon EC2 Auto Scaling が代替インスタンスを起動する前に、Amazon EC2 がスケジュールされたイベントの一環としてインスタンスを再起動、停止、またはリタイアした場合。予これらのイベントの詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの予定されたイベント」を参照してください。
- Amazon EC2 スポットサービスがスポットインスタンスの中断を開始し、スポットインスタンスが強制的に終了した場合。

スポットインスタンスでは、Auto Scaling グループでキャパシティの再調整を有効にした場合、インスタンスに、スポット中断を開始する前に起動した別のスポットプールからの保留中のインスタンスが既にある可能性があります。キャパシティの再調整の仕組みの詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。

ただし、スポットインスタンスは常に利用できるとは限らず、2分前に通知されて停止される可能性があるため、新しいインスタンスの起動前にインスタンスが中断された場合、インスタンスメンテナンスポリシーの下限しきい値を超える可能性があります。

重要な概念

開始する前に、以下の主要概念と用語を理解してください。

希望するキャパシティ

希望するキャパシティは、Auto Scaling グループの作成時のキャパシティを表します。また、グループにスケーリング条件がアタッチされていない場合に、グループが維持しようとするキャパシティでもあります。

インスタンスメンテナンスポリシー

インスタンスメンテナンスポリシーは、インスタンスのメンテナンスイベントが発生した場合に、既存のインスタンスが終了する前に新しいインスタンスがプロビジョニングされるかどうかを制御します。また、Auto Scaling グループが同時に複数のインスタンスを置き換えるために希望するキャパシティをどの程度下回り、超えるかも決まります。

最大正常率

最大正常率は、インスタンスを置き換えるときに Auto Scaling グループが増やすことができる、希望するキャパシティの割合です。これは、ワークロードをサポートするために、稼働中で正常な状態、または起動中のインスタンスが、グループ内で占める割合の最大値を表します。コンソールでは、終了前の起動オプションまたは カスタムポリシーオプションのいずれかを使用すると、最大正常率を設定できます。有効な値は 100～200% です。

最小正常率

最小正常率とは、インスタンスのメンテナンスや更新を行う際に、常に稼働し、正常な状態を維持するために必要なキャパシティの割合です。インスタンスは、最初のヘルスチェックが正常に完了し、指定されたウォームアップ時間が経過すると、正常と見なされ、すぐに使用できます。コンソールで、終了と起動オプションまたはカスタムポリシーオプションのいずれかを使用すると、最小正常率を設定できます。有効な値は 0～100% です。

> **Note**
>
> インスタンスをより速く置き換えるには、最小正常率を低く設定します。ただし、正常なインスタンスが十分にない場合、可用性が低下する可能性があります。複数のインスタンスが置き換えられる状況で可用性を維持するために、適切な値を選択することをお勧めします。

インスタンスのウォームアップ

インスタンスが `InService` 状態になった後に初期化する時間が必要な場合は、Auto Scaling グループのデフォルトのインスタンスウォームアップを有効にします。デフォルトのインスタンスウォーム

アップ機能を使うと、インスタンスの準備が整う前に、インスタンスを最小正常率の計算から除外できます。これにより、Amazon EC2 Auto Scaling は、既存のインスタンスを停止する前に、ワークロードに対応できるだけの新しいインスタンスが準備できるまで待つため、サービスの中断を防ぎます。

デフォルトのインスタンスウォームアップ機能を有効にすることで、動的スケーリングに利用される Amazon CloudWatch メトリクスの精度が向上し、より正確なスケーリングが可能になります。Auto Scaling グループにスケーリングポリシーが設定されている場合、グループがスケールアウトするとき、インスタンスが初期化を完了する前に CloudWatch メトリクスにカウントされないように、同じデフォルトのウォームアップ期間が使用されます。

詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

ヘルスチェックの猶予期間

Amazon EC2 Auto Scaling は、Auto Scaling グループが使用するヘルスチェックのステータスに基づいてインスタンスが正常かどうかを判断します。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

これらのヘルスチェックをできるだけ早く開始するには、グループのヘルスチェック猶予期間をあまり長く設定しないでください。ただし、Elastic Load Balancing ヘルスチェックがターゲットがリクエストを処理できるかどうかを判断するのに十分な長さに設定します。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

Auto Scaling グループをスケールする

インスタンスメンテナンスポリシーは、インスタンスメンテナンスイベントにのみ適用されます。グループが手動または自動でスケーリングされるのを防ぐことはできません。

Auto Scaling グループに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がアタッチされている場合、インスタンスメンテナンスイベントの発生中に並行して実行されることがあります。この場合、グループの希望するキャパシティが増減する可能性がありますが、定義したスケーリング制限内に限られます。これらの制限の詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。

シナリオ例

一般的なシナリオでは、インスタンスのメンテナンスポリシーと希望するキャパシティは次のようになります。

- 最小正常率 = 90%
- 最大正常率 = 120%
- 希望するキャパシティ = 100

インスタンスのメンテナンスイベント実行中は、Auto Scaling グループのインスタンス数が 90 ~ 120 の間で変動することがあります。イベント後、グループはインスタンス数を 100 に戻します。

ウォームプールを持つ Auto Scaling グループでインスタンスメンテナンスポリシーを使用する場合、正常率の最小値と最大値は Auto Scaling グループとウォームプールに個別に適用されます。

例えば、これが設定であるとします。

- 最小正常率 = 90%
- 最大正常率 = 120%
- 希望するキャパシティ = 100
- ウォームプールサイズ = 10

インスタンスの更新を開始してグループのインスタンスをリサイクルする場合、Amazon EC2 Auto Scaling は最初に Auto Scaling グループのインスタンスを置き換え、次にウォームプール内のインスタンスを置き換えます。Amazon EC2 Auto Scaling が Auto Scaling グループ内のインスタンスの置き換えの処理を行っている間、グループ内のインスタンス数は90から120の間で変動する可能性があります。グループでの置換が完了すると、Amazon EC2 Auto Scaling はウォームプール内のインスタンスの置き換えを開始します。これが発生している間、ウォームプールのインスタンス数が 9 ~ 12 個になる可能性があります。

Auto Scaling グループにインスタンスメンテナンスポリシーを設定する

Auto Scaling グループを作成する際、インスタンスメンテナンスポリシーを作成できます。既存のグループに対して作成することもできます。

Auto Scaling グループにインスタンスメンテナンスポリシーを設定することで、インスタンスメンテナンスポリシーを上書きしない限り、インスタンス更新機能の最小正常率および最大正常率のパラメータ値を指定する必要がなくなります。

コンソールでは、Amazon EC2 Auto Scaling に開始に役立つオプションが用意されています。

内容

インスタンスメンテナンスポリシーを設定する

Auto Scaling グループにインスタンスメンテナンスポリシーを設定するには、次のいずれかの方法で行います。

Console

新しいグループにインスタンスメンテナンスポリシーを設定するには (コンソール)

1. 起動テンプレートを使用して Auto Scaling グループを作成する の指示に従い、ステップ 11 までの手順の各ステップを完了します。
2. [グループサイズとスケーリングポリシーを設定] ページの [希望するキャパシティ] に、起動するインスタンスの初期数を入力します。
3. [スケーリング] セクションの [スケーリング制限] で、[希望する容量] の新しい値が [最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に希望する新しい容量の値に引き上げられます。これらの制限は、必要に応じて変更できます。
4. [自動スケーリング] で、ターゲット追跡スケーリングポリシーを作成するかどうかを選択します。このポリシーは、Auto Scaling グループの作成後に作成することもできます。

   [ターゲット追跡スケーリングポリシー] を選択した場合は、「ターゲット追跡スケーリングポリシーを作成する」の指示に従ってポリシーを作成してください。
5. インスタンスメンテナンスポリシーセクションで、使用可能なオプションのいずれかを選択します。

   - 終了前の起動: 既存のインスタンスを終了する前に、新しいインスタンスをプロビジョニングする必要があります。これは、コスト削減よりも可用性を重視するアプリケーションに適しています。
   - 終了と起動: 新しいインスタンスは、既存のインスタンスが終了すると同時にプロビジョニングされます。これは、可用性よりもコスト削減を優先するアプリケーションに適しています。これは、現在利用可能なキャパシティを超えて処理能力を増やす必要のないアプリケーションにも適しています。
   - カスタムポリシー : このオプションを使用すると、インスタンスを置き換えするときに必要なキャパシティの最小値と最大値のカスタム範囲を指定してポリシーを設定できます。これにより、コストと可用性の適切なバランスを実現できます。

6. [正常率を設定]には、次のフィールドの 1 つまたは両方に値を入力します。有効なフィールドは、前のステップで選択したオプションによって異なります。

   - 最小 : インスタンスの置き換えを続行するために必要な最小正常率を設定します。
   - 最大 : インスタンスを置き換えを実行できる最大正常率を設定します。

7. [希望するキャパシティーセクションに基づいて置き換え中にキャパシティーを表示] セクションを展開し、最小と最大の値がグループにどのように適用されるかを確認します。使用される正確な値は、希望するキャパシティ値によって異なります。これは、グループがスケールすると変化します。

8. 起動テンプレートを使用して Auto Scaling グループを作成する のステップを続行します。

AWS CLI

新しいグループにインスタンスメンテナンスポリシーを設定するには (AWS CLI)

create-auto-scaling-group コマンドに `--instance-maintenance-policy` オプションを追加します。次の例では、*my-asg* という名前の新しい Auto Scaling グループにインスタンスメンテナンスポリシーを設定します。

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --default-instance-warmup 20 \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": 90,
      "MaxHealthyPercentage": 120
    }' \
  --vpc-zone-identifier "subnet-5e6example,subnet-613example,subnet-c93example"
```

Console

既存のグループにインスタンスメンテナンスポリシーを設定するには (コンソール)

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。

2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. [詳細] タブで、[インスタンスメンテナンスポリシー] 、[編集] の順に選択します。
5. グループにインスタンスメンテナンスポリシーを設定するには、使用可能なオプションのいずれかを選択します。

   - 終了前の起動: 既存のインスタンスを終了する前に、新しいインスタンスをプロビジョニングする必要があります。これは、コスト削減よりも可用性を重視するアプリケーションに適しています。
   - 終了と起動: 新しいインスタンスは、既存のインスタンスが終了すると同時にプロビジョニングされます。これは、可用性よりもコスト削減を優先するアプリケーションに適しています。これは、現在利用可能なキャパシティを超えて処理能力を増やす必要のないアプリケーションにも適しています。
   - カスタムポリシー : このオプションを使用すると、インスタンスを置き換えするときに必要なキャパシティの最小値と最大値のカスタム範囲を指定してポリシーを設定できます。これにより、コストと可用性の適切なバランスを実現できます。
6. [正常率を設定]には、次のフィールドの 1 つまたは両方に値を入力します。有効なフィールドは、前のステップで選択したオプションによって異なります。

   - 最小 : インスタンスの置き換えを続行するために必要な最小正常率を設定します。
   - 最大 : インスタンスを置き換えを実行できる最大正常率を設定します。
7. [希望するキャパシティーセクションに基づいて置き換え中にキャパシティーを表示] セクションを展開し、最小と最大の値がグループにどのように適用されるかを確認します。使用される正確な値は、希望するキャパシティ値によって異なります。これは、グループがスケールすると変化します。
8. [Update] (更新) を選択します。

AWS CLI

既存のグループにインスタンスメンテナンスポリシーを設定するには (AWS CLI)

update-auto-scaling-group コマンドに --instance-maintenance-policy オプションを追加します。次の例では、指定された Auto Scaling グループにインスタンスメンテナンスポリシーを設定します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": 90,
      "MaxHealthyPercentage": 120
    }'
```

インスタンスメンテナンスポリシーを削除する

Auto Scaling グループでインスタンスメンテナンスポリシーの使用を停止する場合は、削除できます。

Console

インスタンスメンテナンスポリシーを削除するには (コンソール)

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. [詳細] タブで、[インスタンスメンテナンスポリシー] 、[編集] の順に選択します。
5. インスタンスメンテナンスポリシーなし を選択します。
6. [Update] (更新) を選択します。

AWS CLI

インスタンスメンテナンスポリシーを削除するには (AWS CLI)

update-auto-scaling-group コマンドに --instance-maintenance-policy オプションを追加します。以下は、指定された Auto Scaling グループからインスタンスメンテナンスポリシーを削除する例を示しています。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": -1,
      "MaxHealthyPercentage": -1
    }'
```

Amazon EC2 Auto Scaling のライフサイクルフック

Amazon EC2 Auto Scaling は、Auto Scaling グループにライフサイクルフックを追加する機能を提供します。これらのフックにより、Auto Scaling インスタンスライフサイクルのイベントを認識し、対応するライフサイクルイベントが発生したときにカスタムアクションを実行するソリューションを作成できます。ライフサイクルフックでは、インスタンスが次の状態に移行する前に、ライフサイクルアクションの完了を待つ時間 (デフォルトでは 1 時間) が指定されます。

Auto Scaling インスタンスでライフサイクルフックを使用する例として、以下を実行します。

- スケールアウト イベントが発生すると、新しく起動したインスタンスはスタートアップ シーケンスを完了し、待機状態に移行します。インスタンスが待機状態の間に、アプリケーションに必要なソフトウェアパッケージをダウンロードしてインストールするスクリプトを実行して、トラフィックの受信をスタートする前に、インスタンスの準備がすべて完了していることを確認できます。スクリプトがソフトウェアのインストールを終了すると、complete-lifecycle-actionコマンドを実行して続行します。
- スケールイン イベントが発生すると、ライフサイクルフックによってインスタンスが終了される前に一時停止され、Amazon EventBridge を使用して通知が送信されます。インスタンスが待機状態の間に、AWS Lambda関数を呼び出したり、インスタンスに接続して、インスタンスが完全に終了される前にログやその他のデータをダウンロードできます。

ライフサイクルフックの一般的な使用方法は、インスタンスが Elastic Load Balancing に登録されるタイミングを制御することです。Auto Scaling グループに起動ライフサイクルフックを追加すると、ライフサイクルフックの最後にロードバランサーに登録される前に、ブートストラップスクリプトが正常に完了していて、インスタンス上のアプリケーションがトラフィックを受け入れる準備ができていることを確認できます。

内容

- 関連リソース
- Auto Scaling グループでのライフサイクルフックの仕組み
- ライフサイクルフックを Auto Scaling グループに追加するための準備
- インスタンスメタデータを使用してターゲットライフサイクル状態を取得する
- Auto Scaling グループにライフサイクル フックを追加する
- Auto Scaling グループでライフサイクルアクションを完了する
- チュートリアル: データスクリプトとインスタンスメタデータを使用してライフサイクル状態を取得する
- チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定

ライフサイクルフックの可用性

次の表は、さまざまなシナリオで利用できるライフサイクルフックを示しています。

イベント	インスタンスの起動または終了[1]	インスタンスの最大存続期間: 置き換えインスタンス	インスタンスの更新: 置き換えインスタンス	キャパシティの再調整: 置き換えインスタンス	ウォームプール: ウォームプールに出入りするインスタンス
インスタンスの起動	✓	✓	✓	✓	✓
インスタンスの削除	✓	✓	✓	✓	✓

[1] 自動的に開始されたか、または手動で開始されたかにかかわらず (SetDesiredCapacity もしくは TerminateInstanceInAutoScalingGroup オペレーションを呼び出す場合など)、すべての起動と終了に適用されます。インスタンスのアタッチまたはデタッチ、インスタンスのスタンバイモードへの切り替え、または強制削除オプションを使用したグループの削除には適用されません。

ライフサイクルフックの考慮事項と制限

ライフサイクルフックを使用する場合は、以下の注意事項と制限事項に留意してください。

- Amazon EC2 Auto Scaling は、Auto Scaling グループの管理に役立つ独自のライフサイクルを提供します。このライフサイクルは、他の EC2 インスタンスのライフサイクルとは異なります。詳細については、「Amazon EC2 Auto Scaling インスタンスのライフサイクル」を参照してください。ウォームプール内のインスタンスには、ウォームプール内のインスタンスのライフサイクル状態の移行 で説明されている独自のライフサイクルもあります。
- スポットインスタンスによりライフサイクルフックを使用できますが、使用可能なキャパシティがなくなった場合、ライフサイクルフックはインスタンスの終了を防ぐことができません。これは、2 分間の中断通知により、いつでも起こり得ます。詳細については、『Amazon EC2 ユーザーガイド』の「スポットインスタンス中断」を参照してください。ただし、キャパシティの再調整を有効にして、Amazon EC2 スポットサービスから再調整のレコメンデーションを受け取ったスポットインスタンスを積極的に置換することができます。これは、スポットインスタンスの中断リスクが高まったときに送信される信号です。詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。
- インスタンスは一定期間、待機状態に維持できます。ライフサイクルフックのデフォルトタイムアウトは 1 時間です (ハートビートタイムアウト)。インスタンスを待機状態に保つことができる最大時間を指定するグローバルタイムアウトもあります。グローバルタイムアウトは、48 時間か、ハートビートタイムアウトの 100 倍の時間のどちらか短い方になります。
- ライフサイクルフックの結果は、中止または続行のどちらかになります。インスタンスが起動中の場合、「続行」はアクションが正常に実行され、Amazon EC2 Auto Scaling がインスタンスをサービス開始できることを示します。それ以外の場合、中止はカスタムアクションが失敗し、インスタンスを終了して置き換えることができることを示します。インスタンスが終了する場合、「中止」と「続行」の両方でインスタンスを終了できます。ただし、「中止」は他のライフサイクルフックなど残りのアクションをすべて停止し、「続行」は他のライフサイクルフックを完了することを許可します。
- ライフサイクルフックが繰り返し失敗する場合、Amazon EC2 Auto Scaling はインスタンスの起動を許可する頻度を制限するので、ライフサイクルアクションをテストして永続的なエラーを修正するようにしてください。
- AWS CLI、AWS CloudFormation、または SDK を使用したライフサイクルフックの作成と更新は、AWS Management Consoleからライフサイクルフックを作成するときには利用できないオプションを提供します。例えば、コンソールには SNS トピックや SQS キューの ARN を指定するフィールドが表示されません。これは、Amazon EC2 Auto Scaling がすでに Amazon EventBridge にイベントを送信しているからです。これらのイベントはフィルタリングして、必要に応じて Lambda、Amazon SNS、および Amazon SQS などの AWS のサービスにリダイレクトできます。

- AWS CLI、AWS CloudFormation、または SDK を使用して CreateAutoScalingGroup API を呼び出すことによって、Auto Scaling グループの作成中に複数のライフサイクルフックを追加できます。ただし、各フックの通知ターゲットと IAM ロール (指定されている場合) が同じである必要があります。異なる通知ターゲットと異なるロールでライフサイクルフックを作成するには、PutLifecycleHook API に対する別個の呼び出しで、ライフサイクルフックを 1 つずつ作成します。
- インスタンス起動用のライフサイクルフックを追加すると、ヘルスチェックの猶予期間は、インスタンスが `InService` 状態に到達するとすぐに開始されます。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

スケーリングに関する考慮事項

- 動的スケーリングポリシーは、複数のインスタンスから集計した、CPU やネットワーク I/O などの CloudWatch メトリックスデータに応じて、スケールインとスケールアウトを行います。スケールアウトしても、Auto Scaling グループで集計するインスタンスメトリックスに、Amazon EC2 Auto Scaling が新しいインスタンスを追加するわけではありません。インスタンスが `InService` 状態に到達し、ウォームアップが完了するまで待機します。詳細については、デフォルトインスタンスのウォームアップのトピックの「パフォーマンスのスケーリングに関する考慮事項」を参照してください。
- スケールインでは、終了するインスタンスの削除が集約インスタンスメトリックスに直ちに反映されない場合があります。Amazon EC2 Auto Scaling の終了ワークフローが開始すると、終了するインスタンスは、グループの集計インスタンスメトリクスへのカウントを停止します。
- ほとんどの場合、ライフサイクルフックが呼び出されると、簡易スケーリングポリシーに起因するスケーリングアクティビティは、ライフサイクルアクションが完了し、クールダウン期間が終了するまで一時停止されます。クールダウン期間に長い間隔を設定すると、スケーリングが再開されるまでに時間がかかることになります。詳細については、クールダウントピックの「追加の遅延を発生させる可能性のあるライフサイクルフック」を参照してください。一般的に、ステップスケーリングポリシーまたはターゲット追跡スケーリングポリシーを使用できる場合には、簡易スケーリングポリシーを使わないことをおすすめします。

関連リソース

紹介ビデオについては、[AWSre: Invent 2018: Amazon EC2 Auto Scaling でキャパシティ管理を容易にする] の [YouTube] をご覧ください。

AWS CloudFormation のスタックテンプレートでライフサイクルフックを宣言する方法を理解するために使用できる、JSON および YAML のテンプレートスニペットをいくつかご用意しています。詳細については、AWS CloudFormation ユーザーガイドの AWS::AutoScaling::LifecycleHook のリファレンスを参照してください。

ライフサイクルフックのサンプルテンプレートとユーザーデータスクリプトは、GitHub リポジトリでダウンロードできます。

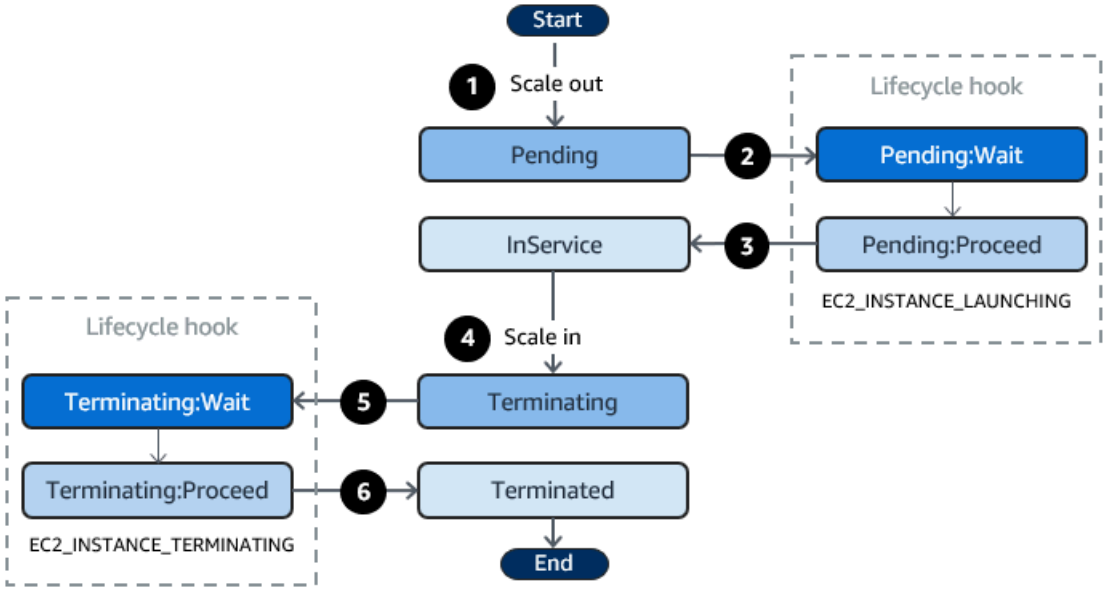
ライフサイクルフックの使用例については、次のブログ記事を参照してください。

- Lambda と Amazon EC2 Run コマンドを使用したスケーリングインスタンスのバックアップシステムの構築
- EC2 Auto Scaling インスタンスを終了する前にコードを実行します。

Auto Scaling グループでのライフサイクルフックの仕組み

Amazon EC2 インスタンスは、起動から終了まで、さまざまな状態に移行します。Auto Scaling グループ用にカスタムアクションを作成して、ライフサイクルフックによってインスタンスが待機状態へ移行しているときに、動作させることができます。

次の図は、スケールアウトとスケールインにライフサイクル フックを使用する場合の Auto Scaling インスタンスの状態の遷移を示しています。



(前の図には示されているように)。

1. Auto Scaling グループはスケールアウト イベントに応答し、インスタンスの起動を開始します。

2. ライフサイクルフックはインスタンスを待機状態 (Pending:Wait) にし、カスタムアクションを実行します。

   インスタンスは、ライフサイクルアクションが完了するまで、またはタイムアウト期間が終了するまで待機状態のままになります。デフォルトでは、インスタンスは 1 時間にわたり待機状態になった後、Auto Scaling グループは起動処理を継続します(Pending:Proceed)。さらに時間が必要な場合は、ハートビートを記録することにより、タイムアウト期間を再開できます。カスタムアクションは完了したが、タイムアウト期間はまだ終了していないという場合にライフサイクルアクションを完了すると、タイムアウト期間が終了し、Auto Scaling グループは起動プロセスを続行します。

3. インスタンスはInService状態になり、ヘルスチェックの猶予期間がスタートします。ただし、Auto Scaling グループが Elastic Load Balancing ロードバランサーに関連付けられている場合、インスタンスが InService 状態に到達する前にインスタンスはロードバランサーに登録され、ロードバランサーはそのヘルスチェックを開始します。ヘルスチェックの猶予期間が終了すると、Amazon EC2 Auto Scaling はインスタンスのヘルス状態のチェックを開始します。

4. Auto Scaling グループはスケールイン イベントに応答し、インスタンスの終了を開始します。Auto Scaling グループが Elastic Load Balancing で使用されている場合、まず、終了するインスタンスがロードバランサーから登録解除されます。ロードバランサーの Connection Draining が有効になっている場合、インスタンスは新しい接続の受け入れを停止し、既存の接続のドレインを待ってから登録解除プロセスを完了します。

5. ライフサイクルフックはインスタンスを待機状態 (Terminating:Wait) にし、カスタムアクションを実行します。

   インスタンスは、ライフサイクルアクションを完了するまで、またはタイムアウト期間が終了するまで (デフォルトでは 1 時間)、待機状態のままになります。ライフサイクルフックを完了するか、タイムアウト期間が終了すると、インスタンスは次の状態 (Terminating:Proceed)に移行します。

6. インスタンスが終了しました。

> **Important**
>
> ウォームプール内のインスタンスには、ウォームプール内のインスタンスのライフサイクル状態の移行 で説明されている、対応する待機状態を備えた独自のライフサイクルもあります。

ライフサイクルフックを Auto Scaling グループに追加するための準備

Auto Scaling グループにライフサイクルフックを追加する前に、ユーザーデータスクリプト、または通知ターゲットまたはが正しく設定されていることを確認するようにしてください。

- インスタンスの起動中にユーザーデータスクリプトを実行してインスタンスでカスタムアクションを使用するために、通知ターゲットを設定する必要はありません。ただし、ユーザーデータスクリプトを指定する起動テンプレートまたは起動設定を作成し、Auto Scaling グループに関連付けておく必要があります。ユーザーデータスクリプトの詳細については、「Amazon EC2 ユーザーガイド」の「起動時に Linux インスタンスでコマンドを実行する」を参照してください。
- ライフサイクルアクションの完了時に Amazon EC2 Auto Scaling に通知を行うには、スクリプトに CompleteLifecycleAction API コールを追加し、Auto Scaling インスタンスがこの API を呼び出すことを許可するポリシーを持つ IAM ロールを手動で作成する必要があります。起動テンプレートまたは起動設定では、起動時に Amazon EC2 インスタンスにアタッチされる IAM インスタンスプロファイルを使用して、このロールを指定する必要があります。詳細については、Auto Scaling グループでライフサイクルアクションを完了するおよびAmazon IAM インスタンスで実行されるアプリケーションの EC2 ロールを参照してください。
- Lambda などのサービスを使用してカスタムアクションを実行するには、すでに EventBridge ルールが作成されており、そのターゲットとして Lambda 関数が指定されている必要があります。詳細については、「ライフサイクル通知の通知ターゲットを設定する」を参照してください。
- ライフサイクルアクション完了時に Lambda に Amazon EC2 Auto Scaling への通知を許可するには、CompleteLifecycleAction API コールを関数コードに追加する必要があります。関数の実行ロールに、ライフサイクルフックを完了する許可を Lambda に付与する IAM ポリシーをアタッチする必要もあります。詳細については、「チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定」を参照してください。
- Amazon SNS や Amazon SQS などのサービスを使用してカスタムアクションを実行するには、SNS トピックまたは SQS キューを作成し、その Amazon リソースネーム (ARN) を準備しておく必要があります。また、SNS トピックまたは SQS ターゲットへの Amazon EC2 Auto Scaling アクセスを許可する IAM ロールを作成し、その ARN を準備しておく必要があります。詳細については、「ライフサイクル通知の通知ターゲットを設定する」を参照してください。

> **Note**
>
> デフォルトでは、コンソールでライフサイクルフックを追加すると、Amazon EC2 Auto Scaling は Amazon EventBridge にライフサイクルイベント通知を送信します。EventBridge またはユーザーデータスクリプトの使用は、推奨されるベストプラクティスです。Amazon SNS または Amazon SQS に直接通知を送信するライフサイクル

フックを作成するには、AWS CLI、AWS CloudFormation、または SDK を使用してライフサイクルフックを追加します。

ライフサイクル通知の通知ターゲットを設定する

ライフサイクルフックを Auto Scaling グループに追加して、インスタンスが待機状態になったときにカスタムアクションを実行できます。希望する開発アプローチに応じてターゲットサービスを選択し、これらのアクションを実行できます。

最初のアプローチでは、Amazon EventBridge を使用して、必要なアクションを実行する Lambda 関数を呼び出します。2 つ目のアプローチでは、通知が発行される Amazon Simple Notification Service (Amazon SNS) トピックを作成することです。クライアントは SNS トピックに登録し、サポートされているプロトコルを使用して公開されたメッセージを受信できます。最後のアプローチでは、Amazon Simple Queue Service (Amazon SQS) を使用します。これは、ポーリングモデルを介してメッセージを交換するために、分散アプリケーションで使用されるメッセージングシステムです。

ベストプラクティスとして、 EventBridge を使用することをお勧めします。Amazon SNS および Amazon SQS に送信される通知には、Amazon EC2 Auto Scaling が EventBridge に送信する通知と同じ情報が含まれています。EventBridge 以前は、SNS または SQS に通知を送信し、別のサービスを SNS または SQS に統合してプログラムによるアクションを実行するのがスタンダードな方法でした。今日、EventBridge では、対象となるサービスのオプションが増え、サーバーレス アーキテクチャを使用してイベントを処理しやすくなりました。

次の手順では、通知ターゲットの設定方法について説明します。

起動テンプレートまたは起動設定に、起動時にインスタンスを設定するユーザーデータスクリプトがある場合は、インスタンスでカスタムアクションを実行するための通知を受け取る必要はありません。

内容

> **Important**
>
> ライフサイクルフックで使用する EventBridge ルール、Lambda 関数、Amazon SNS トピック、および Amazon SQS キューは、常に Auto Scaling グループを作成したのと同じリージョンに存在する必要があります。

EventBridge を使用して通知を Lambda にルーティングする

EventBridge ルールを設定して、インスタンスが待機状態になったときに Lambda 関数を呼び出すことができます。Amazon EC2 Auto Scaling は、起動または終了するインスタンスとライフサイクルアクションを制御するために使用するトークンに関して、EventBridge にライフサイクルイベント通知を送信します。これらのイベントの例については、Amazon EC2 Auto Scaling イベントリファレンス を参照してください。

> **Note**
>
> AWS Management Consoleを使用してイベントルールを作成すると、EventBridge に Lambda 関数を呼び出す許可を付与するために必要な IAM アクセス許可がコンソールによって自動的に追加されます。AWS CLI を使用してイベントルールを作成する場合は、この権限を明示的に付与する必要があります。
> EventBridge コンソールでイベントルールを作成する方法の詳細については、「Amazon EventBridge ユーザーガイド」の「イベントに反応する Amazon EventBridge ルールの作成」を参照してください。
> - または -
> コンソールユーザー向けの初歩的なチュートリアルについては、「チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定」を参照してください。このチュートリアルは、起動イベントをリッスンし、CloudWatch Logs のログにそれらを書き込むシンプルな Lambda 関数を作成する方法を説明します。

Lambda 関数を呼び出す EventBridge ルールを作成するには

1. [Lambda コンソール] を使用して Lambda 関数を作成し、Amazon リソースネーム (ARN) を書き留めておきます。例えば、`arn:aws:lambda:region:123456789012:function:my-function` と指定します。EventBridge ターゲットを作成するには ARN が必要です。詳細については、[AWS Lambdaデベロッパーガイド] の [Lambda の開始方法] を参照してください。

2. インスタンス起動のイベントに一致するルールを作成するには、次の put-rule コマンドを使用します。

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state
 ENABLED
```

次の例は、インスタンス起動ライフサイクルアクションの pattern.json を示しています。*####*のテキストは、Auto Scaling グループの名前に置き換えてください。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "AutoScalingGroupName": [ "my-asg" ]
   }
}
```

コマンドが正常に実行された場合は、ルールの ARN を使用して EventBridge が応答します。この ARN をメモします。これは、ステップ 4 で入力する必要があります。

他のイベントに一致するルールを作成するには、イベントパターンを変更します。詳細については、「Auto Scaling イベントの処理に EventBridge を使用する」を参照してください。

3. ルールのターゲットとして使用する Lambda 関数を指定するには、次のput-argetコマンドを実行します。

```
aws events put-targets --rule my-rule --targets
 Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

上記のコマンドで、*my-rule* はステップ 2 でルールに指定した名前になり、Arn パラメータの値はステップ 1 で作成した関数の ARN になります。

4. ルールが Lambda 関数を呼び出せるようにする許可を追加するには、以下の Lambda add-permission コマンドを使用します。このコマンドは、EventBridge サービスのプリンシパル (events.amazonaws.com) 信頼し、指定のルールに許可を適用します。

```
aws lambda add-permission --function-name my-function --statement-id my-unique-id \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
 arn:aws:events:region:123456789012:rule/my-rule
```

上記のコマンドでは:

- *my-function*は、ルールがターゲットとして使用する Lambda 関数の名前です。
- *my-unique-id* は、Lambda 関数ポリシーのステートメントを記述するためにユーザーが定義する一意の識別子です。
- source-arnは EventBridge ルールの ARN です。

コマンドが正常に実行された場合は、次のような出力が表示されます。

```
{
  "Statement": "{\"Sid\":\"my-unique-id\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":
      {\"ArnLike\":
        {\"AWS:SourceArn\":
         \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"
}
```

Statement 値は、Lambda 関数ポリシーに追加されたステートメントの JSON 文字列バージョンです。

5. これらの指示に従った後、次のステップとして「Auto Scaling グループにライフサイクル フックを追加する」に進みます。

Amazon SNS を使用した通知の受信

Amazon SNS を使用して、ライフサイクルアクションが発生したときに、通知を受け取るよう、通知ターゲット (SNS トピック) をセットアップできます。次に、Amazon SNS は登録された受信者に通知を送信します。登録が確認されるまで、トピックに対して発行された通知は受信者に送信されません。

Amazon SNS を使用して通知をセットアップするには

1. Amazon SNS トピックを作成するには、[Amazon SNS コンソール] または次の [トピックの作成] のコマンドを使用します。このトピックが、使用している Auto Scaling グループと同じリー

ジョンにあることを確認します。詳細については、[Amazon Simple 通知サービスデベロッパーガイド] の [Amazon SNS の使用開始] を参照してください。

```
aws sns create-topic --name my-sns-topic
```

2. その Amazon リソースネーム (ARN) をメモします、例えば、`arn:aws:sns:region:123456789012:my-sns-topic`。ライフサイクルフックを作成するには、それが必要です。
3. IAM サービスロールを作成して、Amazon EC2 Auto Scaling に Amazon SNS 通知ターゲットへのアクセス権を付与します。

   SNS トピックへの Amazon EC2 Auto Scaling のアクセス権を付与するには

   a. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
   b. 左側のナビゲーションペインで、[Roles] を選択します。
   c. [ロールの作成] を選択します。
   d. [Select trusted entity] (信頼されたエンティティの選択) で、[AWS のサービス] を選択します。
   e. ユースケースでは、[他の AWS サービスのユースケース] で [EC2 Auto Scaling]、[EC2 Auto Scaling Notification Access] の順に選択します。
   f. [Next] (次へ) を 2 回選択して、[Name, review, and create] (名前、確認、および作成) ページに進みます。
   g. [Role name] (ロール名) にロールの名前 (**my-notification-role** など) を入力して、[Create role] (ロールを作成) を選択します。
   h. [Roles (ロール)] ページで作成したロールを選択して、[Summary (概要)] ページを開きます。ロールの [ARN] をメモします。例えば、`arn:aws:iam::123456789012:role/my-notification-role` と指定します。ライフサイクルフックを作成するには、それが必要です。
4. これらの指示に従った後、次のステップとして「ライフサイクルフックを追加する (AWS CLI)」に進みます。

Amazon SQS を使用した通知の受信

ライフサイクルアクションが発生したときにメッセージを受け取るよう、Amazon SQS を使用して通知ターゲットをセットアップできます。キューコンシューマーは、SQS キューをポーリングしてこれらの通知を処理する必要があります。

> **Important**
>
> FIFO キューはライフサイクルフックとの互換性がありません。

Amazon SQS を使用して通知をセットアップするには

1. [Amazon SQS コンソール] を使用して、Amazon SQS キューを作成します。キューが,使用している Auto Scaling グループと同じリージョンにあることを確認します。詳細については、[Amazon Simple キューサービス デベロッパーガイド] の [Amazon SQS の使用開始] を参照してください。
2. キューの ARN をメモします。例えば、`arn:aws:sqs:us-west-2:123456789012:my-sqs-queue`。ライフサイクルフックを作成するには、それが必要です。
3. IAM サービスロールを作成して、Amazon EC2 Auto Scaling に Amazon SQS 通知ターゲットへのアクセス権を付与します。

   SQS キューへ Amazon EC2 Auto Scaling のアクセス権を付与するには

   a. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
   b. 左側のナビゲーションペインで、[Roles] を選択します。
   c. [ロールの作成] を選択します。
   d. [Select trusted entity] (信頼されたエンティティの選択) で、[AWS のサービス] を選択します。
   e. ユースケースでは、[他の AWS サービスのユースケース] で [EC2 Auto Scaling]、[EC2 Auto Scaling Notification Access] の順に選択します。
   f. [Next] (次へ) を 2 回選択して、[Name, review, and create] (名前、確認、および作成) ページに進みます。
   g. [Role name] (ロール名) にロールの名前 (**`my-notification-role`** など) を入力して、[Create role] (ロールを作成) を選択します。
   h. [Roles (ロール)] ページで作成したロールを選択して、[Summary (概要)] ページを開きます。ロールの [ARN] をメモします。例えば、`arn:aws:iam::123456789012:role/my-notification-role` と指定します。ライフサイクルフックを作成するには、それが必要です。
4. これらの指示に従った後、次のステップとして「ライフサイクルフックを追加する (AWS CLI)」に進みます。

Amazon SNS と Amazon SQS の通知メッセージの例

インスタンスが待機状態のときに、Amazon SNSまたはAmazon SQSの通知ターゲットにメッセージが発行されます。メッセージには、次に示す情報が含まれます。

- `LifecycleActionToken` - ライフサイクル アクション トークン。
- `AccountId` — AWS アカウント ID。
- `AutoScalingGroupName` - Auto Scaling グループの名前。
- `LifecycleHookName` - ライフサイクルフックの名前。
- `EC2InstanceId` - EC2 インスタンスの ID。
- `LifecycleTransition` - ライフサイクルフックタイプ。
- `NotificationMetadata` — 通知メタデータ。

通知メッセージの例を次に示します。

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
AccountId: 123456789012
AutoScalingGroupName: my-asg
LifecycleHookName: my-hook
EC2InstanceId: i-0598c7d356eba48d7
LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata: hook message metadata
```

テスト通知メッセージの例

最初にライフサイクルフックを追加すると、テスト通知メッセージが通知ターゲットに発行されます。テスト通知メッセージの例を次に示します。

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
AccountId: 123456789012
AutoScalingGroupName: my-asg
```

```
AutoScalingGroupARN: arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

> **Note**
>
> Amazon EC2 Auto Scaling から EventBridge に配信されるイベントの例については、Amazon EC2 Auto Scaling イベントリファレンスを参照してください。

インスタンスメタデータを使用してターゲットライフサイクル状態を取得する

起動する各 Auto Scaling インスタンスは、いくつかのライフサイクル状態を経過します。特定のライフサイクル状態移行で実行されるようにインスタンス上のカスタムアクションをインスタンス内から呼び出すには、インスタンスメタデータを使用してターゲットライフサイクル状態を取得する必要があります。

例えば、インスタンスが終了する前にインスタンス上で何らかのコードを実行するために、インスタンス内部からのインスタンスの終了を検出するメカニズムが必要な場合があります。これには、インスタンスからインスタンスのライフサイクル状態を直接ポーリングするコードを書きます。次に、ライフサイクル フックを Auto Scaling グループに追加して、コードが complete-lifecycle-action コマンドを送信して続行するまでインスタンスを実行し続けることができます。

Auto Scaling インスタンスのライフサイクルには、2 つの主な定常状態 (`InService` および `Terminated`) と、2 つの副次的な定常状態 (`Detached` および `Standby`) があります。ウォームプールを使用する場合、ライフサイクルにはさらに 4 つの定常状態 (`Warmed:Hibernated`、`Warmed:Running`、`Warmed:Stopped`、および `Warmed:Terminated`) があります。

インスタンスが前述の定常状態のいずれかに移行する準備を行うと、Amazon EC2 Auto Scaling がインスタンスメタデータ項目の `autoscaling/target-lifecycle-state` の値を更新します。インスタンス内からターゲットライフサイクル状態を取得するには、インスタンスメタデータサービスを使用してインスタンスメタデータから状態を取得する必要があります。

> **Note**
>
> インスタンスメタデータは、アプリケーションがインスタンス情報をクエリするために使用できる、Amazon EC2 インスタンスに関するデータです。インスタンスメタデータサービス

は、ローカルコードがインスタンスメタデータにアクセスするために使用する、インスタンス上のコンポーネントです。ローカルコードには、インスタンスで実行されているユーザーデータスクリプトまたはアプリケーションなどがあります。

ローカルコードは、インスタンスメタデータサービスバージョン 1 (IMDSv1) とインスタンスメタデータサービスバージョン 2 (IMDSv2) の 2 つの方法のいずれかを使用して、実行中のインスタンスからのインスタンスメタデータにアクセスできます。IMDSv2 はセッション指向のリクエストを使用し、インスタンスメタデータへのアクセス試行に利用される可能性がある数種類の脆弱性を緩和します。これら 2 つの方法の詳細については、「Amazon EC2 ユーザーガイド」の「IMDSv2 の使用」を参照してください。

IMDSv2

```
[ec2-user ~]$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

IMDSv1

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

以下は出力例です。

```
InService
```

ターゲットライフサイクル状態とは、インスタンスの移行先状態のことです。現在のライフサイクル状態は、インスタンスの今の状態です。これらは、ライフサイクルアクションが完了し、インスタンスがターゲットライフサイクル状態への移行を完了した後で同じになる場合があります。インスタンスメタデータからインスタンスの現在のライフサイクル状態を取得することはできません。

Amazon EC2 Auto Scaling は、2022 年 3 月 10 日にターゲットライフサイクル状態の生成を開始しました。この日付以降にインスタンスがターゲットライフサイクル状態のいずれかに移行した場合は、インスタンスメタデータにターゲットライフサイクル状態項目が存在します。移行しなかった場合は項目が存在しないため、HTTP 404 エラーが発生します。

インスタンスメタデータの詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータの取得」を参照してください。

ターゲットライフサイクル状態を使用するユーザーデータスクリプトでカスタムアクションを伴うライフサイクルフックを作成する方法を説明するチュートリアルについては、「チュートリアル: データスクリプトとインスタンスメタデータを使用してライフサイクル状態を取得する」を参照してください。

> **Important**
>
> カスタムアクションをできるだけ早く呼び出しできるようにするには、ローカルコードで IMDS を頻繁にポーリングし、エラーがあれば再試行する必要があります。

Auto Scaling グループにライフサイクル フックを追加する

Auto Scaling インスタンスを待機状態にしてカスタムアクションを実行するために、Auto Scaling グループにライフサイクルフックを追加できます。カスタムアクションは、インスタンスの起動時または終了前に実行されます。インスタンスは、ライフサイクルアクションが完了するまで、またはタイムアウト期間が終了するまで待機状態のままになります。

AWS Management Consoleから Auto Scaling グループを作成したら、それに 1 つ、または複数のライフサイクルフック (合計で最大 50 個) を追加できます。AWS CLI、AWS CloudFormation、または SDK を使用して、Auto Scaling グループの作成中にライフサイクルフックを追加することもできます。

デフォルトでは、コンソールでライフサイクルフックを追加すると、Amazon EC2 Auto Scaling は Amazon EventBridge にライフサイクルイベント通知を送信します。EventBridge またはユーザーデータスクリプトの使用は、推奨されるベストプラクティスです。Amazon SNS または Amazon SQS に直接通知を送信するライフサイクルフックを作成するには、このトピックの例にあるように、put-lifecycle-hook コマンドを使用できます。

目次

ライフサイクルフックを追加する (コンソール)

Auto Scaling グループにライフサイクル フックを追加するには、次の手順に従います。スケールアウト (インスタンスの起動) とスケールイン (インスタンスの終了またはウォーム プールへの復帰) のためのライフサイクルフックを追加するには、2 つの個別のフックを作成する必要があります。

作業を開始する前に、「ライフサイクルフックを Auto Scaling グループに追加するための準備」で説明されているように必要に応じてカスタムアクションがセットアップされていることを確認してください。

スケールアウト用のライフサイクルフックを追加するには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。ページの下部にスプリットペインが開きます。
3. [Instance management (インスタンス管理)] タブの [Lifecycle hooks (ライフサイクルフック)] で、[Create lifecycle hook (ライフサイクルフックを作成)] を選択します。
4. スケールアウト (インスタンスが起動) のライフサイクルフックを定義するには、以下を実行してください。

   a. [Lifecycle hook name (ライフサイクルフック名)] で、ライフサイクルフックの名前を指定します。

   b. [Lifecycle transition (ライフサイクルの移行)] で、[Instance launch (インスタンスの起動)] を選択します。

   c. [ハートビートのタイムアウト] には、スケールアウト時にフックがタイムアウトするまで待機状態を維持する時間を秒単位で指定します。この時間の範囲は 30 ~ 7200 秒です。タイムアウト期間を長く設定すると、カスタムアクションが完了する時間が長くなります。その後、タイムアウト期間終了前に終了した場合は、complete-lifecycle-action コマンドを送信して、インスタンスが次の状態に進むことを許可します。

   d. [Default result] (デフォルトの結果) には、ライフサイクルフックのタイムアウト時間を過ぎた場合、または予期しない障害が発生した場合に実行するアクションを指定します。[続行] または 中止 のどちらかを選択できます。

      - [続行] を選択すると、Auto Scaling グループは他のライフサイクル フックを続行し、インスタンスをサービスに投入できます。

- [中止] を選択する場合、Auto Scaling グループは残りのアクションをすべて停止し、インスタンスをただちに終了します。

e. (オプション) [通知メタデータ] には、Amazon EC2 Auto Scaling が通知ターゲットにメッセージを送信するときに含めるその他の情報を指定します。

5. [Create] (作成) を選択します。

スケールイン用のライフサイクルフックを追加するには

1. スケールアウト用のライフサイクル フックを作成した後、[ライフサイクル フックの作成] を選択して、中断したところから続行します。
2. スケールイン (インスタンスを終了またはウォームプールに戻す) のライフサイクルフックを定義するには、以下を実行してください。

a. [Lifecycle hook name (ライフサイクルフック名)] で、ライフサイクルフックの名前を指定します。

b. [Lifecycle transition (ライフサイクルの移行)] で、[Instance terminate (インスタンスの終了)] を選択します。

c. [ハートビートのタイムアウト] には、スケールアウト時にフックがタイムアウトするまで待機状態を維持する時間を秒単位で指定します。CloudWatch から EC2 ログを取得するなどの最終タスクの実行に必要な時間に応じて、30 ~ 120 秒の短いタイムアウト期間で指定することをお勧めします。

d. [Default result] (デフォルトの結果) には、タイムアウト時間を超過した、または予期しない失敗が発生した場合に Auto Scaling グループが実行するアクションを指定します。[ABANDON] (中止) と [CONTINUE] (続行) は、どちらもインスタンスを終了します。

- [CONTINUE] (続行) を選択する場合、Auto Scaling グループは、終了前に他のライフサイクルフックなどの残りのアクションを続行できます。
- [中止] を選択すると、Auto Scaling グループはインスタンスをただちに終了します。

e. (オプション) [通知メタデータ] には、Amazon EC2 Auto Scaling が通知ターゲットにメッセージを送信するときに含めるその他の情報を指定します。

3. [Create] (作成) を選択します。

ライフサイクルフックを追加する (AWS CLI)

[put-lifecycle-hook] コマンドを使用して、ライフサイクルフックを作成および更新ができます。

スケールアウトでアクションを実行するには、以下のコマンドを使用します。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-launch-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

スケールインでアクションを実行するには、以下のコマンドを使用します。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

Amazon SNS または Amazon SQS を使用して通知を受信するには、`--notification-target-arn` および `--role-arn` オプションを追加します。

以下の例は、*my-sns-topic* という名前の SNS トピックを通知ターゲットとして指定するライフサイクルフックを作成します。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \
  --notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic \
  --role-arn arn:aws:iam::123456789012:role/my-notification-role
```

トピックは、以下のキーと値のペアが含まれたテスト通知を受け取ります。

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

デフォルトで、put-lifecycle-hook コマンドは、ハートビートタイムアウトが 3600 秒 (1 時間) のライフサイクルフックを作成します。

既存のライフサイクルフックのハートビートタイムアウトを変更するには、以下の例にあるように、`--heartbeat-timeout` オプションを追加します。

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \
  --auto-scaling-group-name my-asg --heartbeat-timeout 120
```

インスタンスがすでに待機状態になっている場合は、record-lifecycle-action-heartbeat CLI コマンドを使用して、ハートビートを記録することによってライフサイクルフックがタイムアウトしないよう

にすることができます。これにより、ライフサイクルフックを作成するときに指定するタイムアウト時間によってタイムアウト期間が延長されます。タイムアウト期間が終わる前に終了した場合は、インスタンスが次の状態に進むことを許可する complete-lifecycle-action CLI コマンドを送信することができます。詳細な説明と例については、「Auto Scaling グループでライフサイクルアクションを完了する」を参照してください。

Auto Scaling グループでライフサイクルアクションを完了する

Auto Scaling グループがライフサイクルイベントに応答する際、Auto Scaling グループはインスタンスを待機状態にしてイベント通知を送信します。インスタンスが待機状態にあるときに、カスタムアクションを実行できます。

ライフサイクルアクションをCONTINUE の結果で完了させることで、タイムアウト期間が経過する前に終了させることができます。ライフサイクル アクションを完了しない場合、タイムアウト期間が終了すると、ライフサイクル フックはデフォルトの結果コードに指定したステータスになります。

内容

ライフサイクルアクションを完了する (手動)

次の手順はコマンドライン インターフェイスに関するもので、コンソールではサポートされていません。インスタンス ID や Auto Scaling グループの名前など、置き換える必要がある情報は、斜体で表示されます。

ライフサイクルアクションを完了するには (AWS CLI)

1. カスタムアクション完了までにさらに時間が必要な場合は、record-lifecycle-action-heartbeat コマンドを使用してタイムアウト時間を再開し、インスタンスを待機状態に維持します。例えば、タイムアウト期間が 1 時間に設定されており、30 分後にこのコマンドを呼び出す場合、インスタンスはさらに 1 時間 (合計で 90 分) 待機状態のままになります。

   以下のコマンドに示すように、通知と共に受信したライフサイクルアクショントークンを指定できます。

   ```
   aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \
   ```

```
  --auto-scaling-group-name my-asg --lifecycle-action-
token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

または、以下のコマンドに示すように、通知とともに受信したインスタンスの ID を指定できます。

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-
hook \
  --auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
```

2. [complete-lifecycle-action] コマンドを使用してタイムアウト期間が終了する前にカスタムアクションを終了すると、Auto Scaling グループは起動を継続するか、インスタンスを終了することができます。以下のコマンドに示すように、ライフサイクルアクショントークンを指定できます。

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
  --lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

または、以下のコマンドに示すように、インスタンスの ID を指定できます。

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
  --instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook \
  --auto-scaling-group-name my-asg
```

ライフサイクルアクションを完了する (自動)

起動後にインスタンスを設定するユーザーデータスクリプトがある場合は、ライフサイクルアクションを手動で完了する必要はありません。complete-lifecycle-action コマンドをスクリプトに追加します。スクリプトは、インスタンスのメタデータからインスタンス ID を取得し、ブートストラップスクリプトが正常に完了したときに Amazon EC2 Auto Scaling に通知します。

まだそうしていない場合は、インスタンスメタデータからインスタンスのインスタンス ID を取得するためのスクリプトを更新します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータを取得する」を参照してください。

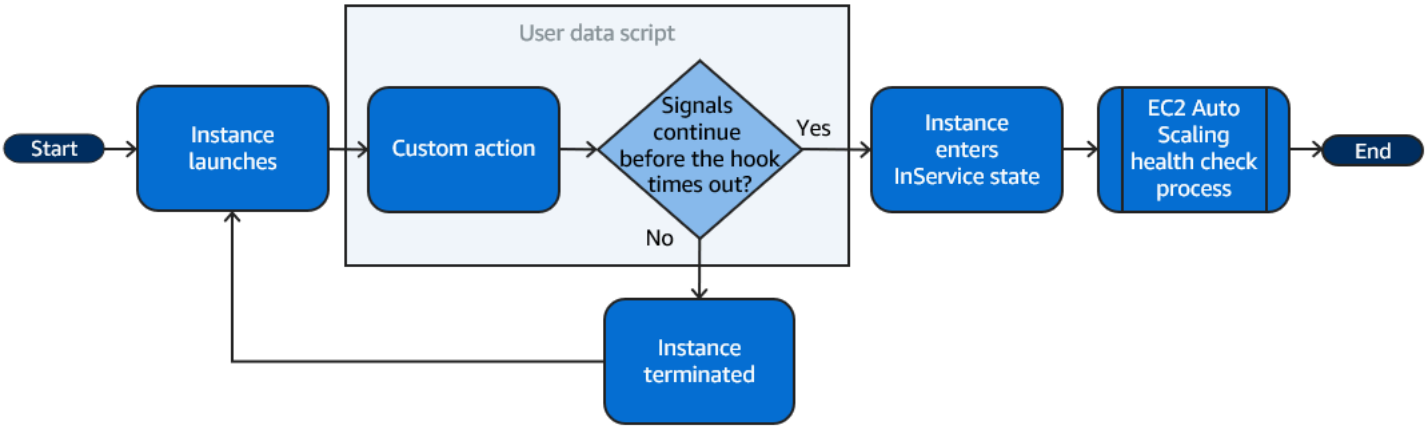
Lambda を使用する場合は、カスタムアクションが成功した際にインスタンスのライフサイクルを続行できるように、関数のコードにコールバックを設定することもできます。詳細については、「チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定」を参照してください。

チュートリアル: データスクリプトとインスタンスメタデータを使用してライフサイクル状態を取得する

ライフサイクルフックのカスタムアクションを作成する一般的な方法は、Amazon EC2 Auto Scaling が Amazon EventBridge などの他のサービスに送信する通知を使用することですが、その代わりに、インスタンスを設定してライフサイクルアクションを完了するコードをインスタンスそのものに移動させるユーザーデータスクリプトを使用することによって、追加のインフラストラクチャを作成する手間を省くことができます。

以下のチュートリアルは、ユーザーデータスクリプトとインスタンスメタデータを使用して開始する方法を説明します。グループ内のインスタンスのターゲットライフサイクル状態を読み取り、インスタンスのライフサイクルの特定のフェーズでコールバックアクションを実行して起動プロセスを続行するユーザーデータスクリプトを使用した、基本的な Auto Scaling グループ設定を作成します。

次の図は、ユーザーデータスクリプトを使用してカスタムアクションを実行する際のスケールアウトイベントのフローをまとめたものです。インスタンスが起動した後、タイムアウトまたは Amazon EC2 Auto Scaling が続行のシグナルを受信することによってライフサイクルフックが完了するまで、インスタンスのライフサイクルは一時停止されます。



内容

ステップ 1: ライフサイクルアクションを完了するための許可を持つ IAM ロールを作成する

ライフサイクルアクションを完了するためのコールバックの送信に AWS CLI または AWS SDK を使用するときは、ライフサイクルアクションを完了するための許可を持つ IAM ロールを使用する必要があります。

ポリシーを作成するには

1. IAM コンソールの [ポリシーページ] を開き、[ポリシーの作成] を選択します。
2. [JSON] タブを選択します。
3. [Policy Document] (ポリシードキュメント) ボックスで、以下のポリシードキュメントをコピーし、ボックスに貼り付けます。*sample text* を、お使いのアカウント番号と、作成する Auto Scaling グループの名前 (**TestAutoScalingEvent-group**) に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/
TestAutoScalingEvent-group"
    }
  ]
}
```

4. [Next] を選択します。
5. [ポリシー名] に「**TestAutoScalingEvent-policy**」と入力します。[Create policy] を選択します。

ポリシーの作成が完了したら、それを使用するロールを作成できます。

ロールを作成するには

1. 左側のナビゲーションペインで、[Roles] を選択します。

2. [ロールの作成] を選択します。
3. [Select trusted entity] (信頼されたエンティティの選択) で、[AWS のサービス] を選択します。
4. ユースケースに [EC2] を選択してから、[Next] (次へ) を選択します。
5. [Add permissions] (許可を追加) で、作成したポリシー ([TestAutoScalingEvent-policy]) を選択します。[次へ] を選択します。
6. [Name, review, and create] (名前、確認、および作成) ページで、[Role name] (ロール名) に **`TestAutoScalingEvent-role`** を入力し、[Create role] (ロールを作成) を選択します。

ステップ 2: 起動テンプレートを作成して IAM ロールとユーザーデータスクリプトを含める

Auto Scaling グループで使用する起動テンプレートを作成します。作成した IAM ロールと、提供されたサンプルユーザーデータスクリプトを含めます。

起動テンプレートを作成するには

1. Amazon EC2 コンソールの起動テンプレートページを開きます。
2. [起動テンプレートの作成] を選択します。
3. [起動テンプレート名] を使用する場合、**`TestAutoScalingEvent-template`** を入力します。
4. [Auto Scaling ガイダンス] で、チェックボックスを選択します。
5. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] で、[クイックスタート] から Amazon Linux 2 (HVM)、SSD Volume Type、64 ビット (x86) を選択します。
6. [Instance type] (インスタンスタイプ) には、Amazon EC2 インスタンスのタイプ (「t2.micro」など) を選択します。
7. [詳細設定] を使用する場合、セクションを展開してフィールドを表示します。
8. [IAM instance profile] (IAM インスタンスプロファイル) で、IAM ロールの IAM インスタンスプロファイル名を選択します (TestAutoScalingEvent-role)。インスタンスプロファイルは IAM ロールのコンテナであり、インスタンスの起動時に Amazon EC2 インスタンスに IAM ロール情報を渡すために使用できます。

   IAM コンソールを使用して IAM ロールを作成すると、コンソールが対応するロールと同じ名前を使用したインスタンスプロファイルを自動的に作成します。
9. [User data] (ユーザーデータ) では、以下のサンプルユーザーデータスクリプトをコピーして、フィールドに貼り付けます。`group_name` のサンプルテキストは、作成する Auto Scaling グ

ループの名前に置き換え、region は、Auto Scaling グループが使用する AWS リージョンに置き換えます。

```
#!/bin/bash

function get_target_state {
    echo $(curl -s http://169.254.169.254/latest/meta-data/autoscaling/target-
lifecycle-state)
}

function get_instance_id {
    echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
}

function complete_lifecycle_action {
    instance_id=$(get_instance_id)
    group_name='TestAutoScalingEvent-group'
    region='us-west-2'

    echo $instance_id
    echo $region
    echo $(aws autoscaling complete-lifecycle-action \
      --lifecycle-hook-name TestAutoScalingEvent-hook \
      --auto-scaling-group-name $group_name \
      --lifecycle-action-result CONTINUE \
      --instance-id $instance_id \
      --region $region)
}

function main {
    while true
    do
        target_state=$(get_target_state)
        if [ \"$target_state\" = \"InService\" ]; then
            # Change hostname
            export new_hostname="${group_name}-$instance_id"
            hostname $new_hostname
            # Send callback
            complete_lifecycle_action
            break
        fi
        echo $target_state
        sleep 5
```

```
    done
}

main
```

このシンプルなユーザーデータスクリプトは、以下を実行します。

- インスタンスメタデータを呼び出して、インスタンスメタデータからターゲットライフサイクル状態とインスタンス ID を取得する
- ターゲットライフサイクル状態が `InService` に変更されるまで、状態を繰り返し取得する
- ターゲットライフサイクル状態が `InService` の場合、インスタンスのホスト名を Auto Scaling グループの名前が先頭に付加されたインスタンス ID に変更する
- complete-lifecycle-action CLI コマンドを呼び出すことによってコールバックを送信し、EC2 起動プロセスを `CONTINUE` するように Amazon EC2 Auto Scaling に通知する

10. [起動テンプレートの作成] を選択します。
11. 確認ページで、[Auto Scaling グループの作成] を選択します。

> **Note**
>
> ユーザーデータスクリプトを開発する際の参考として使用できるその他の例については、Amazon EC2 Auto Scaling の GitHub リポジトリを参照してください。

ステップ 3: Auto Scaling グループを作成する

起動テンプレートを作成したら、Auto Scaling グループを作成します。

Auto Scaling グループを作成する

1. [Choose launch template or configuration] (起動テンプレートまたは起動設定を選択する) ページで、[Auto Scaling group name] (Auto Scaling グループ名) に Auto Scaling グループの名前 (**`TestAutoScalingEvent-group`**) を入力します。
2. [Next] (次へ) を選択して、[Choose instance launch options] (インスタンス起動オプションを選択) ページに進みます。
3. [Network] (ネットワーク) で VPC を選択します。

4. [Availability Zones and subnets] (アベイラビリティーゾーンとサブネット) で、1 つ、または複数のアベイラビリティーゾーンから 1 つ、または複数のサブネットを選択します。
5. [Instance type requirements] (インスタンスタイプの要件) セクションでは、このステップを簡略化するためにデフォルト設定を使用します。(起動テンプレートを上書きしないでください。) このチュートリアルでは、起動テンプレートで指定されたインスタンスタイプを使用して、オンデマンドインスタンスを 1 つだけ起動します。
6. 画面の最下部にある [Skip to review] (スキップして確認) を選択します。
7. [Review] (確認) ページで Auto Scaling グループの詳細を確認してから、[Create Auto Scaling group] (Auto Scaling グループを作成) を選択します。

ステップ 4: ライフサイクルフックを追加する

ライフサイクルアクションが完了するまでインスタンスを待機状態に維持するライフサイクルフックを追加します。

ライフサイクルフックを追加するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。ページの下部にスプリットペインが開きます。
3. 下部のペインで、[Instance management (インスタンス管理)] タブの [Lifecycle hooks (ライフサイクルフック)] で、[Create lifecycle hook (ライフサイクルフックを作成)] を選択します。
4. スケールアウト (インスタンスが起動) のライフサイクルフックを定義するには、以下を実行してください。

   a. [ライフサイクルフック名] で、**`TestAutoScalingEvent-hook`**を入力します。
   b. [Lifecycle transition (ライフサイクルの移行)] で、[Instance launch (インスタンスの起動)] を選択します。
   c. [Heartbeat timeout] (ハートビートタイムアウト) に、ユーザーデータスクリプトからのコールバックを待つ秒数として **`300`** を入力します。
   d. [デフォルトの結果] で、[中止] を選択します。フックがユーザーデータスクリプトからのコールバックを受け取ることなくタイムアウトすると、Auto Scaling グループは新しいインスタンスを終了します。
   e. (オプション) [Notification metadata] (通知メタデータ) を空のままにしておきます。
5. [Create] (作成) を選択します。

ステップ 5: 機能をテストして検証する

機能をテストするには、Auto Scaling グループの希望キャパシティを 1 つ増やすことによって Auto Scaling グループを更新します。インスタンスの起動直後にユーザーデータスクリプトが実行され、インスタンスのターゲットライフサイクル状態のチェックが開始されます。スクリプトは、ターゲットライフサイクル状態が `InService` になるとホスト名を変更し、コールバックアクションを送信します。これには通常、完了まで数秒しかかかりません。

Auto Scaling グループのサイズを増やすには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。上部ペインの最上部の行が見えている状態で、下部ペインの詳細を確認します。
3. 下部のペインの [詳細] タブで、[グループの詳細] 、[編集] を順に選択します。
4. [Desired capacity (希望するキャパシティ)] の場合は、現在の値を 1 ずつ増やします。
5. [Update] (更新) を選択します。インスタンスの起動中は、上部ペインの [Status (ステータス)] 列に [Updating capacity (キャパシティの更新)] というステータスが表示されます。

希望キャパシティを増やした後で、スケーリングアクティビティの説明からインスタンスが正常に起動され、終了されていないことを確認できます。

スケーリングを表示するには

1. [Auto Scaling グループ] ページに戻り、グループを選択します。
2. [アクティビティ] タブの [アクティビティ履歴] では、[ステータス] 列に、Auto Scaling グループがインスタンスを正常に起動したかどうかが表示されます。
3. ユーザーデータスクリプトが失敗した場合は、タイムアウト期間が過ぎたときに、ステータスが `Canceled` のスケーリングアクティビティと、`Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result` のステータスメッセージが表示されます。

ステップ 6: クリーンアップする

このチュートリアルのために作成したリソースでの作業が完了したら、次の手順を実行してそれらを削除してください。

ライフサイクルフックを削除するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [Instance management (インスタンス管理)] タブの [Lifecycle hooks (ライフサイクルフック)] で、[lifecycle hook (ライフサイクルフック)] を選択します。(`TestAutoScalingEvent-hook`)
4. [アクション]、[削除] の順に選択します。
5. 確認のために、もう一度 [削除] を選択します。

起動テンプレートを削除するには

1. Amazon EC2 コンソールの起動テンプレートページを開きます。
2. 起動テンプレート (`TestAutoScalingEvent-template`) を選択してから、[Actions] (アクション)、[Delete template] (テンプレートを削除) の順に選択します。
3. 確認を求められたら、**`Delete`** を入力して指定した起動テンプレートの削除を確認し、[Delete] (削除) を選択します。

サンプル Auto Scaling グループでの作業が完了したら、グループを削除してください。作成した IAM ロールと許可ポリシーも削除できます。

Auto Scaling グループを削除するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループ (`TestAutoScalingEvent-group`) の横にあるチェックボックスをオンにして、[Delete] (削除) を選択します。
3. 確認を求められたら、**`delete`** を入力して指定された Auto Scaling グループの削除を確認し、[Delete] (削除) を選択します。

   [Name (名前)] 列のロードアイコンに、Auto Scaling グループが削除されたことが示されます。インスタンスを終了してグループを削除するには数分かかります。

IAM ロールを削除するには

1. IAM コンソールの [Roles (ロール)] ページを開きます。
2. 関数のロール (`TestAutoScalingEvent-role`) を選択します。

3. [削除] を選択します。
4. 確認を求められたら、ロールの名前を入力し、[Delete] (削除) を選択します。

IAM ポリシーを削除するには

1. IAM コンソールのポリシーページを開きます。
2. 作成したポリシーを選択します。(`TestAutoScalingEvent-policy`)
3. [アクション]、[削除] の順に選択します。
4. 確認を求められたら、ポリシーの名前を入力し、[Delete] (削除) を選択します。

関連リソース

以下の関連トピックは、インスタンスメタデータで利用可能なデータに基づいてインスタンスに対してアクションを呼び出すコードを開発する際に役立ちます。

- インスタンスメタデータを使用してターゲットライフサイクル状態を取得する. このセクションでは、インスタンスの終了など、他のユースケースのライフサイクルステータスについて説明します。
- ライフサイクルフックを追加する (コンソール). この手順では、スケールアウト (インスタンスの起動) とスケールイン (インスタンスの終了またはウォームプールへの復帰) の両方にライフサイクルフックを追加する方法を示します。
- 「Amazon EC2 ユーザーガイド」の「インスタンスメタデータのカテゴリ」 このトピックでは、EC2 インスタンスでアクションを呼び出すために使用できるインスタンスメタデータのすべてのカテゴリを一覧表示します。

Amazon EventBridge を使用して Auto Scaling グループのインスタンスで発生するイベントに基づいて Lambda 関数を呼び出すルールを作成する方法を示すチュートリアルについては、「チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定」を参照してください。

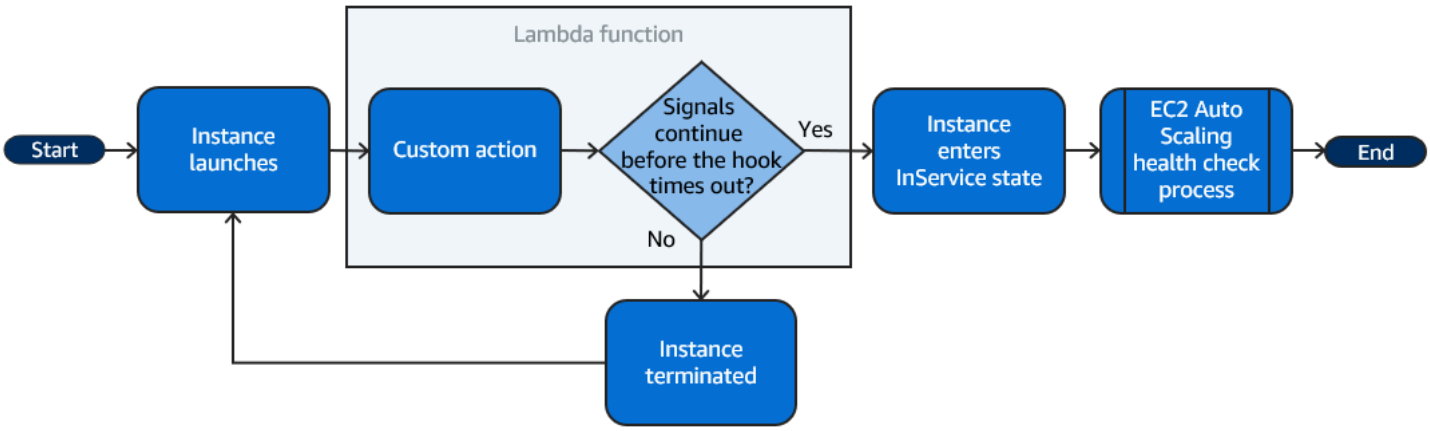
チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定

この演習では、フィルターパターンが含まれる Amazon EventBridge ルールを作成します。このルールに一致すると、AWS Lambda 関数がルールのターゲットとして呼び出されます。使用するフィルターパターンとサンプル関数コードを提供します。

すべてが正しく設定されると、このチュートリアルの最後に、インスタンスの起動時に Lambda 関数はカスタムアクションを実行します。カスタムアクションは、Lambda 関数に関連付けられている CloudWatch Logs ログ ストリーミングにイベントを口グするだけです。

Lambda 関数もコールバックを実行して、このアクションが成功するとインスタンスのライフサイクルを続行しますが、アクションが失敗するとインスタンスは起動を中止し、終了させます。

次の図は、Lambda 関数を使用してカスタムアクションを実行する場合のスケールアウトイベントのフローをまとめたものです。インスタンスが起動した後、タイムアウトまたは Amazon EC2 Auto Scaling が続行のシグナルを受信することによってライフサイクルフックが完了するまで、インスタンスのライフサイクルは一時停止されます。



内容

前提条件

このチュートリアルを開始する前に、Auto Scaling グループがまだない場合は、作成します。Auto Scaling グループを作成するには、Amazon EC2 コンソールで、Auto Scaling グループのページを開き、[Auto Scaling グループの作成] を選択します。

ステップ 1: ライフサイクルアクションを完了するための許可を持つ IAM ロールを作成する

Lambda 関数を作成する前に、実行ロールと許可ポリシーを作成して、Lambda がライフサイクルフックを完了できるようにする必要があります。

ポリシーを作成するには

1. IAM コンソールの [ポリシーページ] を開き、[ポリシーの作成] を選択します。
2. [JSON] タブを選択します。
3. [ポリシードキュメント] ボックスに、次のポリシードキュメントを貼り付け、[######] のテキストはアカウント番号と Auto Scaling グループの名前に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-
asg"
    }
  ]
}
```

4. [Next] を選択します。
5. [ポリシー名] に「**LogAutoScalingEvent-policy**」と入力します。[Create policy] を選択します。

ポリシーの作成が完了したら、それを使用するロールを作成できます。

ロールを作成するには

1. 左側のナビゲーションペインで、[Roles] を選択します。
2. [ロールの作成] を選択します。
3. [Select trusted entity] (信頼されたエンティティの選択) で、[AWS のサービス] を選択します。

4. ユースケースに [Lambda] を選択してから、[Next] (次へ) を選択します。
5. [Add permissions] (許可を追加) で、作成したポリシー ([LogAutoScalingEvent-policy]) と、[AWSLambdaBasicExecutionRole] という名前のポリシーを選択します。[次へ] を選択します。

> **Note**
>
> AWSLambdaBasicExecutionRole ポリシーには、ログを CloudWatch Logs に書き込むために関数が必要とするアクセス許可があります。

6. [Name, review, and create] (名前、確認、および作成) ページで、[Role name] (ロール名) に **`LogAutoScalingEvent-role`** を入力し、[Create role] (ロールを作成) を選択します。

ステップ 2: Lambda 関数を作成する

イベントの対象となる Lambda 関数を作成します。Node.js で記述したサンプルの Lambda 関数は、一致するイベントが Amazon EC2 Auto Scaling から出力されたときに EventBridge から呼び出されます。

Lambda 関数を作成するには

1. Lambda コンソールで [Functions (関数)] ページを開きます。
2. [関数の作成] を選択し、[一から作成] を選択します。
3. [基本的な情報] の [関数名] に「**`LogAutoScalingEvent`**」と入力します。
4. [ランタイム] で [Node.js 18.x] を選択します。
5. スクロールして [デフォルトの実行ロールの変更]を選択し、[実行ロール] で、[既存のロールを使用] を選択します。
6. 既存の[ロール] で、[ログ オート スケーリングイベント ロール] を選択します。
7. 他はデフォルト値のままにしておきます。
8. [機能の作成]を選択します。関数のコードと設定に戻ります。
9. コンソールで LogAutoScalingEvent 関数を開いたまま、エディタの [コードソース] で、index.mjs という名前のファイルに次のサンプルコードを貼り付けます。

```
import { AutoScalingClient, CompleteLifecycleActionCommand } from "@aws-sdk/client-
auto-scaling";
export const handler = async(event) => {
```

```
  console.log('LogAutoScalingEvent');
  console.log('Received event:', JSON.stringify(event, null, 2));
  var autoscaling = new AutoScalingClient({ region: event.region });
  var eventDetail = event.detail;
  var params = {
    AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
    LifecycleActionResult: 'CONTINUE', /* required */
    LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
    InstanceId: eventDetail['EC2InstanceId'],
    LifecycleActionToken: eventDetail['LifecycleActionToken']
  };
  var response;
  const command = new CompleteLifecycleActionCommand(params);
  try {
    var data = await autoscaling.send(command);
    console.log(data); // successful response
    response = {
      statusCode: 200,
      body: JSON.stringify('SUCCESS'),
    };
  } catch (err) {
    console.log(err, err.stack); // an error occurred
    response = {
      statusCode: 500,
      body: JSON.stringify('ERROR'),
    };
  }
  return response;
};
```

このコードは単にイベントをログに記録するので、このチュートリアルの最後に、この Lambda 関数に関連付けられている CloudWatch Logsのログ ストリーミングにイベントが表示されます。

10. [デプロイ] を選択します。

ステップ 3: EventBridge ルールを作成する

Lambda 関数を実行する EventBridge ルールの作成 EventBridge の使用に関する詳細については、「Auto Scaling イベントの処理に EventBridge を使用する」を参照してください。

コンソールを使用してルールを作成するには

1. [EventBridge コンソール] を開きます。
2. ナビゲーションペインで ルール] を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. [Define rule detail] (詳細の定義) で、次の操作を行います。
   a. [名前] に **`LogAutoScalingEvent-rule`** と入力します。
   b. [イベントバス] として、[デフォルト] を選択します。アカウント内の AWS のサービスがイベントを生成すると、イベントは常にアカウントのデフォルトイベントバスに送られます。
   c. ルールタイプ では、イベントパターンを持つルール] を選択します。
   d. [Next] を選択します。
5. [Build event pattern] (イベントパターンの作成) で、次の操作を行います。
   a. [Event source] (イベントソース) で、[AWS events or EventBridge partner events] (イベントまたは EventBridge パートナーイベント) を選択します。
   b. [イベントパターン] までスクロールして、次の操作を行います。
   c. i. イベントソース で AWS のサービス を選択します。
      ii. [AWS のサービス] には、[Auto Scaling] を選択します。
      iii. [イベントタイプ] で、[インスタンスの起動と終了] を選択します。
      iv. デフォルトで、ルールはすべてのスケールインイベントまたはスケールアウトイベントに一致します。スケールアウトイベントが発生し、ライフサイクルフックに基づいてインスタンスが待機状態になったときに通知するルールを作成するには、[Specific instance event(s)] (特定のインスタンスイベント) を選択してから、[EC2 Instance-launch Lifecycle Action] (EC2 インスタンス起動ライフサイクルアクション) を選択します。
      v. デフォルトでは、このルールはリージョン内のすべての Auto Scaling グループと一致します。ルールを特定の Auto Scaling グループに一致させるには、[特定のグループ名] を選択してグループを選択します。
      vi. [Next] を選択します。
6. [Select target(s)] (ターゲットを選択) で、以下の操作を行います。
   a. [Target types] (ターゲットタイプ) には、[AWS のサービス] を選択します。
   b. [Select a target] (ターゲットを選択) では、[Lambda function] (Lambda 関数) を選択します。

c. [Function] (機能) には、[LogAutoScalingEvent] を選択します。

d. [次へ] を 2 回選択します。

7. [Review and create] (確認して作成) ページで、[Create rule] (ルールの作成) を選択します。

ステップ 4: ライフサイクルフックを追加する

このセクションでは、Lambda が起動時にインスタンスで関数を実行できるように、ライフサイクルフックを追加します。

ライフサイクルフックを追加するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。ページの下部にスプリットペインが開きます。
3. 下部のペインで、[Instance management (インスタンス管理)] タブの [Lifecycle hooks (ライフサイクルフック)] で、[Create lifecycle hook (ライフサイクルフックを作成)] を選択します。
4. スケールアウト (インスタンスが起動) のライフサイクルフックを定義するには、以下を実行してください。

   a. [ライフサイクルフック名] で、**LogAutoScalingEvent-hook**を入力します。

   b. [Lifecycle transition (ライフサイクルの移行)] で、[Instance launch (インスタンスの起動)] を選択します。

   c. [ハートビートのタイムアウト] で、Lambda 関数からのコールバックを待機する秒数として**300**を入力します。

   d. [デフォルトの結果] で、[中止] を選択します。つまり、Lambda 関数からコールバックを受け取らずにフックがタイムアウトすると、Auto Scaling グループは新しいインスタンスを終了します。

   e. （オプション）[通知メタデータ] を空にします。EventBridge に渡すイベントデータには、Lambda 関数を呼び出すために必要な情報がすべて含まれています。

5. [Create] (作成) を選択します。

ステップ 5: イベントをテストし、検証する

イベントをテストするには、Auto Scaling グループで希望するキャパシティを 1 増やして Auto Scaling グループを更新します。Lambda 関数は、希望するキャパシティを増やしてから数秒以内に呼び出されます。

Auto Scaling グループのサイズを増やすには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択すると、下部のペインに詳細が表示され、上部のペインの一番上の行が表示されます。
3. 下部のペインの [詳細] タブで、[グループの詳細] 、[編集] を順に選択します。
4. [Desired capacity (希望するキャパシティ)] の場合は、現在の値を 1 ずつ増やします。
5. [Update] (更新) を選択します。インスタンスの起動中は、上部ペインの [Status (ステータス)] 列に [Updating capacity (キャパシティの更新)] というステータスが表示されます。

希望するキャパシティに増やしたら、Lambda 関数が呼び出されたことを確認できます。

Lambda 関数からの出力を表示するには

1. Amazon CloudWatch コンソールの [[Log groups (ロググループ)] ページ] を開きます。
2. Lambda 関数 (`/aws/lambda/LogAutoScalingEvent`) のロググループの名前を選択します。
3. ライフサイクルアクションの関数によって提供されるデータを表示するログのストリーミング名を選択します。

次に、スケーリング アクティビティの説明から、インスタンスが正常に起動したことを確認できます。

スケーリングを表示するには

1. [Auto Scaling グループ] ページに戻り、グループを選択します。
2. [アクティビティ] タブの [アクティビティ履歴] では、[ステータス] 列に、Auto Scaling グループがインスタンスを正常に起動したかどうかが表示されます。
   - アクションが成功した場合、スケーリング アクティビティのステータスは「成功」になります。

- 失敗した場合、数分待ってから、ステータスが「キャンセル済み」のスケーリング アクティビティが表示され、「インスタンスはユーザーのライフサイクルアクションを完了できませんでした:トークンE85EB647-4FE0-4909-B341-A6C42の例は中止されました:ライフサイクルアクションは中止された結果で完了しました」というステータスメッセージが表示されます。

Auto Scaling グループのサイズを縮小するには

このテスト用に起動した追加のインスタンスが必要なくなった場合は、[詳細] タブを開き、[Desired capacity (希望するキャパシティ)] を 1 減らすことができます。

ステップ 6: クリーンアップする

このチュートリアル専用に作成したリソースで作業が完了したら、次の手順に従ってリソースを削除します。

ライフサイクルフックを削除するには

1. Amazon EC2 コンソールで Auto Scaling グループのページを開きます。
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [Instance management (インスタンス管理)] タブの [Lifecycle hooks (ライフサイクルフック)] で、[lifecycle hook (ライフサイクルフック)] を選択します。(`LogAutoScalingEvent-hook`)
4. [アクション]、[削除] の順に選択します。
5. 確認のために、もう一度 [削除] を選択します。

Amazon EventBridge ルールを削除するには

1. Amazon EventBridge コンソールで [Rules] (ルール) ページを開きます。
2. [Event bus (イベントバス)] で、ルール (`Default`) に関連付けられているイベントバスを選択します。
3. `LogAutoScalingEvent-rule` ルールの横にあるチェックボックスをオンにします。
4. [削除] を選択します。
5. 確認を求められたら、ルールの名前を入力し、[Delete] (削除) を選択します。

サンプル関数の使用が終了したら、削除します。関数のログを保存するためのロググループや、作成した実行ロールや許可ポリシーも削除できます。

Lambda 関数を削除するには

1. Lambda コンソールで [Functions (関数)] ページを開きます。
2. 関数 (LogAutoScalingEvent) を選択します。
3. [アクション]、[削除] の順に選択します。
4. 確認を求められたら、**delete** を入力して指定した関数の削除を確認し、[Delete] (削除) を選択します。

ロググループを削除するには

1. Amazon CloudWatch コンソールの [Log groups (ロググループ)] ページを開きます。
2. 関数のロググループ (/aws/lambda/LogAutoScalingEvent) を選択します。
3. [アクション]、[ロググループの削除] の順にクリックします。
4. ロググループの削除ダイアログボックスで、[削除] をクリックします。

実行ロールを削除するには

1. IAM コンソールの [Roles (ロール)] ページを開きます。
2. 関数のロール (LogAutoScalingEvent-role) を選択します。
3. [削除] を選択します。
4. 確認を求められたら、ロールの名前を入力し、[Delete] (削除) を選択します。

IAM ポリシーを削除するには

1. IAM コンソールのポリシーページを開きます。
2. 作成したポリシーを選択します。(LogAutoScalingEvent-policy)
3. [アクション]、[削除] の順に選択します。
4. 確認を求められたら、ポリシーの名前を入力し、[Delete] (削除) を選択します。

関連リソース

以下の関連トピックは、Auto Scaling グループのインスタンスに発生するイベントに基づいて EventBridge ルールを作成する際に役立ちます。

- Auto Scaling イベントの処理に EventBridge を使用する. このセクションでは、スケールイン用のイベントなど、他のユースケースのイベントの例を示します。
- ライフサイクルフックを追加する (コンソール). この手順では、スケールアウト (インスタンスの起動) とスケールイン (インスタンスの終了またはウォームプールへの復帰) の両方にライフサイクルフックを追加する方法を示します。

インスタンスメタデータサービス (IMDS) を使用してインスタンス自体からアクションを呼び出す方法を示すチュートリアルについては、「チュートリアル: データスクリプトとインスタンスメタデータを使用してライフサイクル状態を取得する」を参照してください。

ウォームプールを使用してブート時間が長いアプリケーションのレイテンシーを低減する

ウォームプールを使用すると、インスタンスが大量のデータをディスクに書き込む必要があるなど、起動時間が非常に長いアプリケーションのレイテンシーを低減できます。ウォームプールの使用によって、アプリケーションのパフォーマンスを向上させるために、レイテンシーを管理するために Auto Scaling グループを過剰にプロビジョニングする必要がなくなりました。詳細については、次のブログ記事EC2「Auto Scaling ウォームプールによるAuto Scaling」を参照してください。

> **Important**
>
> 必要のないときにウォームプールを作成すると、不要なコストが発生する可能性があります。最初の起動時にアプリケーションに顕著なレイテンシーの問題が発生しない場合は、おそらくウォームプールを使用する必要はありません。

トピック

- を使用したウォームプールの作成と管理の例 AWS CLI

主要概念

開始する前に、以下の主要概念を理解してください。

ウォームプール

ウォームプールは、Auto Scaling グループの横にある、事前に初期化されたEC2インスタンスのプールです。アプリケーションがスケールアウトする必要があるときはいつでも、Auto Scaling グループはウォームプールに描画して、新しい希望する容量を満たすことができます。これにより、インスタンスがアプリケーショントラフィックを迅速化し、スケールアウトイベントへの応答を高速化できるようになります。インスタンスは、ウォームプールから離れたときに、グループの希望する容量にカウントされます。これは、ウォームスタートとして知られています。

インスタンスがウォームプールにある間、スケーリングポリシーは、`InService` 状態のインスタンスのメトリクス値がスケーリングポリシーのアラーム上限しきい値 (ターゲット追跡スケーリングポリシーのターゲット使用率と同じ値) を超える場合のみスケールアウトします。

ウォームプールのサイズ

デフォルトでは、ウォームプールのサイズは、Auto Scaling グループの最大容量と希望する容量の数値の差として計算されます。例えば、Auto Scaling グループの希望する容量が 6 で、最大容量が 10 の場合、ウォームプールを最初にセットアップし、プールが初期化されるときに、ウォームプールのサイズは 4 になります。

ウォームプールの最大容量を個別に指定するには、カスタム仕様 (`MaxGroupPreparedCapacity`) オプションを使用して、グループの現在の容量を超えるカスタム値を設定します。カスタム値を指定すると、ウォームプールのサイズは、グループのカスタム値と現在希望する容量の数値の差として計算されます。たとえば、Auto Scaling グループの希望容量が 6、最大容量が 20、カスタム値が 8 である場合、初めてウォームプールをセットアップしてプールが初期化されると、ウォームプールのサイズは 2 になります。

ウォームプールを使用するコストメリットを管理するために、大きな Auto Scaling グループで作業するときは、カスタム仕様 (`MaxGroupPreparedCapacity`) オプションのみを使用する場合があります。例えば、インスタンス 1,000 個、最大容量 1,500 個 (トラフィックの急増時に追加の容量を提供するため)、およびインスタンス 100 個のウォームプールがある Auto Scaling グループは、ウォームプール内に将来使用するインスタンスを 500 個予約しておくよりも、目標の達成に役立つ場合があります。

ウォームプールサイズの最小サイズ

最小サイズ設定 (MinSize) を使用して、ウォームプールで維持するインスタンスの最小数を静的に設定することを検討してください。デフォルトでは最小サイズは設定されていません。MinSize この設定はMaxGroupPreparedCapacity、Auto Scaling グループの希望する容量が よりも大きい場合でも、ウォームプールに最小数のインスタンスが維持されるように を指定する場合に便利ですMaxGroupPreparedCapacity。

ウォームプールインスタンスの状態

ウォームプール内のインスタンスは、次の 3 つの状態のいずれかで保持できます: Stopped、Running、Hibernated。インスタンスをStopped状態で保持することは、コストを最小限に抑えるための効果的な方法です。停止したインスタンスでは、使用したボリュームとインスタンスにアタッチされた Elastic IP アドレスの分だけ料金が発生します。

または、インスタンスを Hibernated状態のままにして、メモリコンテンツを削除せずにインスタンスを停止することもできます (RAM)。インスタンスが休止状態になると、オペレーティングシステムに の内容を Amazon EBSルートボリュームRAMに保存するよう通知します。インスタンスが再び起動されると、ルートボリュームは以前の状態に復元され、RAMコンテンツが再ロードされます。インスタンスが休止している間は、RAMコンテンツのストレージやインスタンスにアタッチされた Elastic IP アドレスなど、EBSボリュームに対してのみ料金が発生します。

ウォームプール内のインスタンスを Running 状態にしておくことも可能ですが、不必要な料金の発生避けるためにも、そうしておかないことを強くお勧めします。インスタンスを停止または休止状態にしておくと、インスタンス自体のコストが削減されます。インスタンスの料金は、インスタンスが実行されている場合にのみ発生します。

ライフサイクルフック

ライフサイクルフックを使用して、インスタンスに対してカスタムアクションを実行できるように、インスタンスを待機状態にします。カスタムアクションは、インスタンスの起動時または終了前に実行されます。

ウォームプール設定では、ライフサイクルフックは、初期化が完了するまで、スケールアウトイベント中にインスタンスが停止または休止されたり、稼働したりするのを遅延させます。ライフサイクルフックなしで Auto Scaling グループにウォームプールを追加すると、初期化の完了までに長い時間がかかるインスタンスは停止または休止状態になり、準備が整う前にスケールアウトイベントが開始する可能性があります。

インスタンスの再利用ポリシー

デフォルトでは、Amazon EC2 Auto Scaling は Auto Scaling グループのスケールイン時にインスタンスを終了します。次に、新しいインスタンスをウォームプールで起動し、終了したインスタンスを置換します。

置換する代わりに、インスタンスをウォームプールに戻す場合は、インスタンスの再利用ポリシーを指定します。これにより、アプリケーショントラフィックを処理するように設定されたインスタンスを再利用できます。ウォームプールが過剰にプロビジョニングされていないことを確認するために、Amazon EC2 Auto Scaling はウォームプール内のインスタンスを終了して、設定に基づいて必要以上に大きい場合にサイズを縮小できます。ウォームプール内のインスタンスを終了する際に、Amazon EC2 Auto Scaling はデフォルトの終了ポリシーを使用して、最初に終了するインスタンスを選択します。

> **Important**
>
> スケールイン時にインスタンスを休止し、Auto Scaling グループに既存のインスタンスがある場合は、インスタンスの休止要件を満たしている必要があります。要件を満たしていない場合、インスタンスがウォームプールに戻ると、休止状態ではなく停止状態にフォールバックします。

> **Note**
>
> 現在、インスタンス再利用ポリシーは、 AWS CLI または を使用してのみ指定できます SDK。この機能はコンソールからは利用できません。

前提条件

Auto Scaling グループのためにウォームプールを作成する前に、ライフサイクルフックを使用して新しいインスタンスを適切な初期状態で初期化する方法を決定します。

インスタンスがライフサイクルフックを理由として待機状態にあるときに、インスタンスに対してカスタムアクションを実行するには、次の 2 つのオプションがあります。

- 起動時にインスタンスでコマンドを実行する単純なシナリオでは、Auto Scaling グループの起動テンプレートまたは起動設定の作成時にユーザーデータスクリプトを含めることができます。ユー

ザーデータスクリプトは、インスタンスの起動時に cloud-init により実行される通常のシェルスクリプトまたは cloud-init ディレクティブです。このスクリプトは、実行されるインスタンスの ID を使用して、インスタンスが次の状態に移行するタイミングを制御することもできます。まだそうしていない場合は、インスタンスメタデータからインスタンスのインスタンス ID を取得するためのスクリプトを更新します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータの取得」を参照してください。

> **Tip**
>
> インスタンスの再起動時にユーザーデータスクリプトを実行するには、ユーザーデータがMIMEマルチパート形式であり、ユーザーデータの `#cloud-config`セクションで以下を指定する必要があります。
>
> ```
> #cloud-config
> cloud_final_modules:
>  - [scripts-user, always]
> ```

- インスタンスがウォームプールに出入りするときに何か AWS Lambda を行うなどの高度なシナリオでは、Auto Scaling グループのライフサイクルフックを作成し、ライフサイクル通知に基づいてカスタムアクションを実行するようにターゲットサービスを設定できます。詳細については、「サポートされている通知ターゲット」を参照してください。

インスタンス休止のための準備

`Hibernated` プールの状態を使用するように Auto Scaling インスタンスを準備するには、「Amazon EC2ユーザーガイド」の「休止の前提条件」トピックで説明されているように、インスタンスの休止をサポートするために正しく設定された新しい起動テンプレートまたは起動設定を作成します。次に、新しい起動テンプレートまたは起動設定を Auto Scaling グループに関連付けてインスタンスの更新を開始し、以前の起動テンプレートまたは起動設定に関連付けられているインスタンスを置換します。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。

ウォームプール内のインスタンスを更新する

ウォームプール内のインスタンスを更新するには、新しい起動テンプレートまたは起動設定を作成し、それを Auto Scaling グループに関連付けます。新しいインスタンスは、起動テンプレートまたは起動設定で指定された新しい更新AMIやその他の更新を使用して起動されますが、既存のインスタンスは影響を受けません。

新しい起動テンプレートまたは起動設定を使用する代替ウォームプールインスタンスを強制的に起動するには、インスタンスの更新を開始してグループのローリング更新を行うことができます。インスタンスの更新は、最初にInServiceインスタンスを置き換えます。その後、ウォームプール内のインスタンスが置き換えられます。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。

関連リソース

ウォームプールのライフサイクルフックの例GitHubについては、リポジトリを参照してください。

制限

- 混合インスタンスポリシー を持つ Auto Scaling グループにウォームプールを追加することはできません。また、起動テンプレートまたはスポットインスタンスをリクエストする起動設定を持つ Auto Scaling グループにウォームプールを追加することはできません。
- Amazon EC2 Auto Scaling は、ルートデバイスとして Amazon EBSボリュームがある場合にのみ、インスタンスを Stoppedまたは Hibernated状態にすることができます。ルートデバイスにインスタンスストアを使用するインスタンスは停止または休止できません。
- Amazon EC2 Auto Scaling は、Amazon EC2ユーザーガイドの休止の前提条件トピックに記載されているすべての要件を満たしている場合にのみ、インスタンスを Hibernated状態にすることができます。
- スケールアウトイベントがあるときにウォームプールが枯渇した場合、インスタンスは Auto Scaling グループ内に直接起動されます (コールドスタート)。また、アベイラビリティーゾーンが容量不足の場合にコールドスタートが発生する可能性があります。
- ウォームプール内のインスタンスが起動プロセス中に問題に遭遇し、InService 状態に到達できない場合、インスタンスは起動に失敗したと見なされ、終了します。これは、容量不足エラーやその他の要因など、根本的な原因に関係なく適用されます。
- Amazon Elastic Kubernetes Service (Amazon EKS) マネージド型ノードグループでウォームプールを使用しようとすると、まだ初期化中のインスタンスが Amazon EKSクラスターに登録される可能性があります。その結果、このクラスターは、インスタンスが停止または休止の準備を行っているときにインスタンスでジョブをスケジュールする場合があります。
- 同様に、Amazon ECSクラスターでウォームプールを使用しようとすると、インスタンスは初期化が完了する前にクラスターに登録される可能性があります。この問題を解決するには、ユーザーデータに特別なエージェント設定変数が含まれる起動テンプレートまたは起動設定を設定する必要があります。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Auto Scaling グループでウォームプールを使用する」を参照してください。

- ウォームプールの休止サポートは、Amazon EC2 Auto Scaling と休止 AWS リージョン が利用可能なすべての商用 で利用できます。ただし、以下を除きます。
  - アジアパシフィック (ハイデラバード)
  - アジアパシフィック (メルボルン)
  - カナダ西部 (カルガリー)
  - 中国 (北京) リージョン
  - 中国 (寧夏) リージョン
  - 欧州 (スペイン)
  - イスラエル (テルアビブ)

Auto Scaling グループのウォームプールでライフサイクルフックを使用する

ウォームプールのインスタンスは、移行ごとに適切なカスタムアクションを作成できるように、独自の独立したライフサイクルを維持します。このライフサイクルは、インスタンスがまだ初期化中、およびサービスを開始する前に、ターゲットサービス (Lambda 関数など) でアクションを呼び出すように設計されています。

> **Note**
>
> ライフサイクルフックの追加と管理、およびライフサイクルアクションの完了に使用するAPIオペレーションは変更されません。インスタンスのライフサイクルのみが変更されます。

ライフサイクルフック追加の詳細については、Auto Scaling グループにライフサイクル フックを追加する を参照してください。ライフサイクルアクション完了の詳細については、Auto Scaling グループでライフサイクルアクションを完了する を参照してください。

ウォームプールに入るインスタンスは、次のいずれかの理由でライフサイクルフックが必要になる場合があります。

- 初期化が完了するまでAMIに時間がかかる からEC2インスタンスを起動したい。
- ユーザーデータスクリプトを実行してEC2インスタンスをブートストラップしたい。

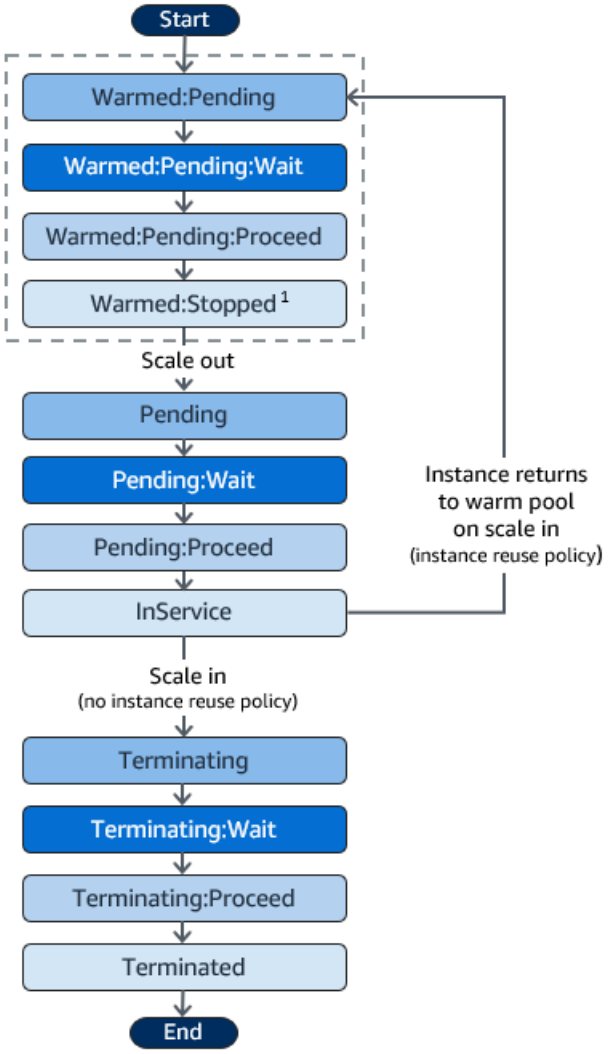
ウォームプールを離れるインスタンスは、次のいずれかの理由でライフサイクルフックが必要になる場合があります。

- 追加の時間を使用して、インスタンスを使用するEC2準備をすることができます。例えば、インスタンスの再起動時に、アプリケーションが正常に動作する前に、開始する必要があるサービスがあるとします。
- キャッシュデータを事前入力して、新しいサーバーが空のキャッシュで起動しないようにすることができます。
- 新しいインスタンスをマネージドインスタンスとして設定管理サービスに登録する場合。

ウォームプール内のインスタンスのライフサイクル状態の移行

オートスケーリングインスタンスは、ライフサイクルの一環として多くの状態に移行します。

次の図表は、ウォームプール使用時のオートスケーリング状態間の移行を示しています。



[1] この状態は、ウォームプールのプール状態設定によって異なります。プール状態が Running に設定されている場合、この状態は代わりに Warmed:Running になります。プール状態が Hibernated に設定されている場合、この状態は代わりに Warmed:Hibernated になります。

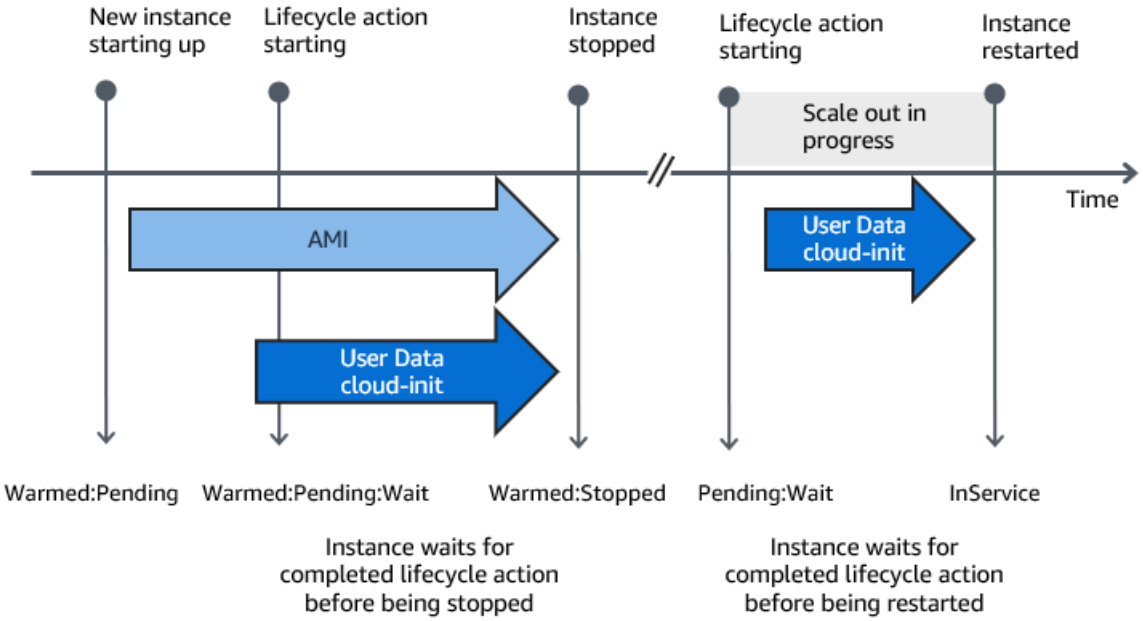
ライフサイクルフックを追加するときは、次の点を考慮してください。

- ライフサイクルフックが autoscaling:EC2_INSTANCE_LAUNCHING ライフサイクルアクションに対して設定されている場合、新しく起動したインスタンスは、Warmed:Pending:Wait 状態に達したときに一時停止してカスタムアクションを実行します。その後、インスタンスが再開して Pending:Wait 状態になったときに再度一時停止してカスタムアクションを実行します。
- ライフサイクルフックが EC2_INSTANCE_TERMINATING ライフサイクルアクションに対して設定されている場合、終了するインスタンスは、Terminating:Wait 状態に達したときに一時停止してカスタムアクションを実行します。ただし、スケールインでインスタンスをウォームプールに戻すインスタンスの再使用ポリシーを指定した場合、ウォームプールに戻るインスタンスは

`Warmed:Pending:Wait` 状態で一時停止して `EC2_INSTANCE_TERMINATING` ライフサイクルアクションにカスタムアクションを実行します。

- アプリケーションの需要がウォームプールを枯渇させる場合、Amazon EC2 Auto Scaling は、グループが最大容量に達していない限りAuto Scaling グループにインスタンスを直接起動できます。インスタンスがグループ内で直接起動する場合、インスタンスは `Pending:Wait` 状態でのみ一時停止してカスタムアクションを実行します。
- 次の状態に遷移するまでにインスタンスが待機状態を維持する時間を制御するには、complete-lifecycle-action コマンドを使用するようカスタムアクションを設定します。ライフサイクルフックでは、指定したライフサイクルアクションが完了したことを Amazon EC2 Auto Scaling に通知するか、タイムアウト期間が終了するまで (デフォルトでは 1 時間) 、インスタンスは待機状態のままになります。

スケールアウトイベントのフローの概要を次に示します。



インスタンスが待機状態になると、Amazon EC2 Auto Scaling は通知を送信します。これらの通知の例は、このガイドの EventBridge セクションで確認できます。詳細については、「ウォームプールのイベントとパターンの例」を参照してください。

サポートされている通知ターゲット

Amazon EC2 Auto Scaling は、ライフサイクル通知の通知ターゲットとして次のいずれかを定義するためのサポートを提供します。

- EventBridge ルール

- Amazon SNSトピック
- Amazon SQSキュー

> **Important**
>
> 起動時にインスタンスを設定するユーザーデータ (cloud-init) スクリプトが起動テンプレートまたは起動設定にある場合、起動または再起動されるインスタンスでカスタムアクションを実行するための通知を受け取る必要はありません。

次のセクションでは、通知ターゲットの設定方法について説明しているドキュメントへのリンクを示します。

EventBridge ルール: Amazon EC2 Auto Scaling がインスタンスを待機状態にするときにコードを実行するには、 EventBridge ルールを作成し、ターゲットとして Lambda 関数を指定します。異なるライフサイクル通知に基づいて異なる Lambda 関数を呼び出すには、複数のルールを作成し、各ルールを特定のイベントパターンおよび Lambda 関数に関連付けます。詳細については、「ウォームプールイベント向けの EventBridge ルールを作成する」を参照してください。

Amazon SNSトピック: インスタンスが待機状態になったときに通知を受信するには、Amazon SNSトピックを作成し、SNSメッセージ属性に基づいてライフサイクル通知を異なる方法で配信するように Amazon メッセージフィルタリングを設定します。詳細については、「Amazon SNS を使用した通知の受信」を参照してください。

Amazon SQSキュー: 関連するコンシューマーがそれらを選択して処理できるライフサイクル通知の配信ポイントを設定するには、Amazon SQSキューと、キューからのメッセージを処理するSQSキューコンシューマーを作成できます。キューコンシューマーに、メッセージ属性に基づいてライフサイクル通知を別々に処理させる場合は、特定の属性が目的の値と一致する際にメッセージを解析、処理するようにキューコンシューマーを設定する必要があります。詳細については、「Amazon SQS を使用した通知の受信」を参照してください。

Auto Scaling グループのためにウォームプールを作成する

このトピックでは、Auto Scaling グループのウォームプールを作成する方法について説明します。

> **Important**
>
> 続行する前に、ウォームプールを作成するための前提条件を満たし、Auto Scaling グループのためにライフサイクルフックが作成されていることを確認します。

ウォームプールを作成する

Auto Scaling グループのためにウォームプールを作成するには、次の手順を実行します。

ウォームプールを作成するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   ページの下部に分割されたペインが開きます。
3. [インスタンス管理] タブを選択します。
4. [ウォームプール] で、ウォームプールの作成を選択します。
5. ウォームプールを設定するには、次の手順を実行します。

   a. ウォームプールインスタンスの状態で、インスタンスがウォームプールに入ったときに、どの状態に移行するかを選択します。デフォルト: `Stopped`。

   b. 最小ウォームプールサイズに、ウォームプールに維持するインスタンスの最小数を入力します。

   c. インスタンスの再利用 の場合、[スケールインで再利用] チェックボックスをオンにすると、スケールインで Auto Scaling グループのインスタンスをウォームプールに戻すことができます。

   d. [ウォームプールサイズ] で、使用可能なオプションを 1 つ選択します。

   - デフォルトの仕様: ウォームプールのサイズは、Auto Scaling グループの最大容量と希望する容量の数値の差によって決まります。このオプションは、ウォームプール管理を合理化します。ウォームプールを作成すると、グループの最大容量を調整するだけで、ウォームプールのサイズを簡単に更新できます。
   - カスタム仕様: ウォームプールのサイズはAuto Scaling グループのカスタム値と希望する容量の数値の差によって決まります。このオプションを使用すると、グループの最大容量とは別に、ウォームプールのサイズを柔軟に管理できます。

6. [現在の設定に基づく推定ウォームプールサイズ] セクションを表示して、デフォルトまたはカスタム仕様がウォームプールのサイズにどのように適用されるかを確認します。ウォームプールのサイズは、Auto Scaling グループの希望する容量によって異なります。この容量は、グループがスケールすると変化します。
7. [Create] (作成) を選択します。

ウォームプールを削除する

ウォームプールが不要になった場合は、次の手順にしたがって削除します。

ウォームプールを削除するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   ページの下部に分割されたペインが開きます。
3. [インスタンス管理] タブを選択します。
4. [Warm pool] (ウォームプール) で、[Actions] (アクション)、[Delete] (削除) の順に選択します。
5. 確認を求めるメッセージが表示されたら、[削除] を選択します。

ヘルスチェックのステータスとヘルスチェックの失敗理由を表示する

ヘルスチェックにより、Amazon EC2 Auto Scaling はインスタンスが異常であり、いつ終了する必要があるかを判断できます。ウォームプールインスタンスが `Stopped`状態のままの場合、Amazon EBSが`Stopped`インスタンスの可用性に関する知識を利用して、異常なインスタンスを識別します。これは、 を呼び出し`DescribeVolumeStatus`APIて、インスタンスにアタッチされているEBSボリュームのステータスを決定することによって行われます。ウォームプールインスタンスが`Running`状態を維持している場合、EC2ステータスチェックに依存してインスタンスのヘルスを判断します。ウォームプールインスタンスのヘルスチェック猶予期間はありませんが、Amazon EC2 Auto Scaling はライフサイクルフックが終了するまでインスタンスのヘルスチェックを開始しません。

インスタンスに異常が見つかった場合、Amazon EC2 Auto Scaling は異常のあるインスタンスを自動的に削除し、新しいインスタンスを作成して置き換えます。インスタンスは、通常、ヘルスチェックに失敗してから数分以内に終了します。詳細については、「ヘルスチェック不合格の理由を表示する」を参照してください。

カスタムヘルスチェックもサポートされています。これは、インスタンスのヘルスを検出し、この情報を Amazon EC2 Auto Scaling に送信できる独自のヘルスチェックシステムがある場合に役立ちます。詳細については、「Auto Scaling グループに対してカスタムヘルスチェックを設定する」を参照してください。

Amazon EC2 Auto Scaling コンソールでは、ウォームプールインスタンスのステータス (正常または異常) を表示できます。 AWS CLI または のいずれかを使用して、ヘルスステータスを表示することもできますSDKs。

ウォームプールインスタンスのステータスを表示するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [Instance management (インスタンス管理)] タブにある、[Warm pool instances (ウォームプールインスタンス)] の [Lifecycle (ライフサイクル)] 列にインスタンスの状態が表示されます。

   ヘルスステータス列には、Amazon EC2 Auto Scaling がインスタンスのヘルスについて行った評価が表示されます。

   > **Note**
   >
   > 新しいインスタンスは正常に起動します。ライフサイクルフックが終了するまで、インスタンスの健全性はチェックされません。

ヘルスチェックの失敗の理由を表示するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [Activity (アクティビティ)] タブの [Activity history (アクティビティ履歴)] の下の [Status (ステータス)] 列に、Auto Scaling グループがインスタンスを正常に起動したか、終了したかが表示されます。

正常でないインスタンスを終了した場合、原因列には、終了の日時、およびヘルスチェックが失敗した理由が表示されます。例えば、2021-04-01T21:48:35Z に、EBSボリュームヘルスチェックの失敗に応じてインスタンスがサービス停止になりました」などです。

ウォームプールインスタンスのステータスを表示するには (AWS CLI)

次のdescribe-warm-poolコマンドを使用して、Auto Scaling グループのウォームプールを表示します。

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

出力例。

```
{
    "WarmPoolConfiguration": {
        "MinSize": 0,
        "PoolState": "Stopped"
    },
    "Instances": [
        {
            "InstanceId": "i-0b5e5e7521cfaa46c",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        },
        {
            "InstanceId": "i-0e21af9dcfb7aa6bf",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
```

```
            }
        }
    ]
}
```

ヘルスチェックの失敗理由を表示するには (AWS CLI)

次の describe-scaling-activities コマンドを使用します。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下に、応答の例を示します。Descriptionは、Auto Scaling グループがインスタンスを終了したことを示し、Causeは、ヘルスチェックが失敗した理由を示します。

スケーリングアクティビティは、開始時刻順に並べられます。まだ進行中のアクティビティを最初に説明します。

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
 response to EBS volume health check failure.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

を使用したウォームプールの作成と管理の例 AWS CLI

ウォームプールは、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または を使用して作成および管理できますSDKs。

次の例では、 AWS CLIを使用してウォームプールを作成、管理する方法を示します。

内容

例 1: インスタンスを **Stopped** 状態に保つ

次のput-warm-pool例では、インスタンスを Stopped状態に保つウォームプールを作成します。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped
```

例 2: インスタンスを **Running** 状態に保つ

次のput-warm-pool例では、インスタンスを Running状態ではなく Stopped状態に保つウォームプールを作成します。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Running
```

例 3: インスタンスを **Hibernated** 状態に保つ

次のput-warm-pool例では、インスタンスを Hibernated状態ではなく Stopped状態に保つウォームプールを作成します。これにより、メモリコンテンツを削除せずにインスタンスを停止できます (RAM)。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Hibernated
```

例 4: スケールイン時にインスタンスをウォームプールに戻す

次のput-warm-pool例では、インスタンスを Stopped状態に保ち、 --instance-reuse-policyオプションを含むウォームプールを作成します。インスタンス再利用ポリシー値は、EC2Auto Scalingグループがスケールインしたときにインスタンスをウォームプールに返すように Amazon Auto Scaling '{"ReuseOnScaleIn": true}'に指示します。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --instance-reuse-policy '{"ReuseOnScaleIn": true}'
```

例 5: ウォームプール内のインスタンスの最小数を指定する

次のput-warm-pool例では、少なくとも 4 つのインスタンスを維持するウォームプールを作成し、トラフィックの急増を処理するために少なくとも 4 つのインスタンスを使用できます。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 4
```

例 6: カスタム仕様を使用してウォームプールのサイズを定義する

デフォルトでは、Amazon EC2 Auto Scaling はAuto Scaling グループの最大容量と希望する容量の差としてウォームプールのサイズを管理します。ただし、--max-group-prepared-capacity オプションを使用して、グループの最大容量とは別に、ウォームプールのサイズを管理できます。

次のput-warm-pool例では、ウォームプールを作成し、ウォームプールと Auto Scaling グループの両方に同時に存在できるインスタンスの最大数を設定します。グループの希望容量が 800 の場合、このコマンドの実行後にウォームプールが初期化されると、最初のサイズは 100 になります。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900
```

ウォームプール内のインスタンスの最小数を維持するには、次のように、コマンドを使用して--min-sizeオプションを、含めます。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

例 7: 絶対的なウォームプールサイズを定義する

`--max-group-prepared-capacity` および `--min-size` オプションを同じ値に設定すると、ウォームプールは絶対サイズになります。次のput-warm-pool例では、10 インスタンスのウォームプールサイズを一定に維持するウォームプールを作成します。

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

例 8: ウォームプールを削除する

ウォームプールを削除するには、次のdelete-warm-poolコマンドを使用します。

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg
```

ウォームプールにインスタンスがある場合、またはスケーリングアクティビティが進行中の場合は、`--force-delete`オプションを指定して delete-warm-pool コマンドを使用します。このオプションは、Amazon EC2インスタンスと未処理のライフサイクルアクションも終了します。

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg --force-delete
```

Auto Scaling グループのゾーンシフト

ゾーンシフトは、Amazon Application Recovery Controller () の機能ですARC。ゾーンシフトを使用すると、1 回のアクションでアベイラビリティーゾーンのアプリケーション障害から迅速に回復できます。Auto Scaling グループのゾーンシフトを有効にすると、グループはARCゾーンシフトサービスに登録されます。その後、 AWS Management Console、、または を使用してゾーンシフトを開始できます。APIAuto Scaling グループは AWS CLI、アクティブなゾーンシフトでアベイラビリティーゾーンを障害ありとして扱います。

Auto Scaling グループのゾーンシフトの概念

先に進む前に、ARCゾーンシフトとの統合に関連する以下の主要概念を理解しておく必要があります。

ARC ゾーンシフト

Auto Scaling は、この機能を有効にするとAuto Scaling ARC グループをゾーンシフトに登録できます。登録後、 ARC `ListManagedResources` でリソースを表示できますAPI。詳細につい

ては、「Amazon Application Recovery Controller () デベロッパーガイド」の「 のゾーンシフト ARC」を参照してください。 ARC

アベイラビリティーゾーンの再調整

Auto Scaling は、各アベイラビリティーゾーンの容量のバランスを維持しようとします。アベイラビリティーゾーン間で不均衡が発生すると、Auto Scaling は自動的に不均衡の修正を試みます。詳細については、「インスタンスの分散」を参照してください。

動的なスケーリング

動的スケーリングは、スケーリングポリシーで選択したメトリクスに基づいて、Auto Scaling グループの希望する容量をスケーリングします。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。

ヘルスチェック

Auto Scaling は、Auto Scaling グループ内のすべてのインスタンスのヘルスステータスを定期的にチェックして、それらが実行中で良好な状態であることを確認します。異常なインスタンスが検出されると、Auto Scaling はそのインスタンスを置き換え対象としてマークします。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

インスタンスの更新

Auto Scaling グループのインスタンスを更新するには、インスタンスの更新を使用できます。インスタンスの更新が開始されると、Auto Scaling は Auto Scaling グループ内のすべてのインスタンスの置き換えを試みます。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。

プリスケーリング済み

アプリケーション用の残りのアベイラビリティーゾーンに十分な容量があるため、1 つのアベイラビリティーゾーンが失われるのを許容できます。

スケールアウト

Auto Scaling グループの希望するキャパシティを増やすと、Auto Scaling は新しい希望するキャパシティを満たすために追加のインスタンスの起動を試みます。デフォルトでは、Auto Scaling はバランスの取れた方法でインスタンスを起動し、Auto Scaling グループ内の有効な各アベイラビリティーゾーンで同じ容量を維持します。

Auto Scaling グループのゾーンシフトの仕組み

次のアベイラビリティーゾーンを持つ Auto Scaling グループがあるとします。

- us-east-1a
- us-east-1b
- us-east-1c

すべてのアベイラビリティーゾーンでゾーンシフトが有効になっていて、 で障害が発生したことに気付くus-east-1aため、ゾーンシフトをトリガーします。以下の動作は、 でゾーンシフトがトリガーされたときに発生しますus-east-1a。

- スケールアウト — Auto Scaling は、正常なアベイラビリティーゾーン (us-east-1b および) ですべての新しいキャパシティーリクエストを起動しますus-east-1c。
- 動的スケーリング – Auto Scaling は、スケーリングポリシーがすべてのアベイラビリティーゾーンで希望するキャパシティーを減らすのをブロックします。Auto Scaling は、スケーリングポリシーがすべてのアベイラビリティーゾーンで希望するキャパシティーを増やすのをブロックしません。
- インスタンスの更新 – Auto Scaling は、ゾーンシフトがアクティブの間に遅延したインスタンスの更新プロセスのタイムアウトを延長します。

次の表は、 でゾーンシフトがトリガーされたときの各オプションのヘルスチェックの動作を示していますus-east-1a。

アベイラビリティーゾーンのヘルスチェック動作選択の障害	ヘルスチェックの動作			
異常を置き換える	異常と思われるインスタンスは、すべてのアベイラビリティーゾーン (us-east-1a 、us-east-1b 、および) で置き換えられますus-east-1c 。			

アベイラビリティーゾーンのヘルスチェック動作選択の障害	ヘルスチェックの動作			
異常を無視する	異常と思われるインスタンスは、 us-east-1b および で置き換えられますus-east-1c 。アベイラビリティーゾーンのインスタンスは、アクティブなゾーンシフト () に置き換えられませんus-east-1a 。			

ゾーンシフトを使用するためのベストプラクティス

ゾーンシフトを使用するときにアプリケーションの高可用性を維持するには、次のベストプラクティスをお勧めします。

- EventBridge 通知をモニタリングして、進行中のアベイラビリティーゾーンの障害イベントがあるかどうかを確認します。詳細については、「Auto Scaling イベントの処理に EventBridge を使用する」を参照してください。
- 適切なしきい値を持つスケーリングポリシーを使用して、アベイラビリティーゾーンの損失を許容するのに十分な容量があることを確認します。
- インスタンスメンテナンスポリシーを最小正常率 100 に設定します。この設定では、Auto Scaling は、異常なインスタンスを終了する前に、新しいインスタンスを使用する準備ができるのを待ちます。

プリスケーリングされたお客様には、以下もお勧めします。

- 障害イベント中に異常なインスタンスを置き換える必要がないため、障害が発生したアベイラビリティーゾーンのヘルスチェック動作として異常を無視を選択します。
- Auto Scaling グループの でゾーンオートシフトARCを使用します。のゾーンオートシフト機能ARCを使用すると AWS 、 がアベイラビリティーゾーンの障害 AWS を検出したときに、リソースのトラフィックをアベイラビリティーゾーンから遠ざけることができます。詳細については、「Amazon Application Recovery Controller () デベロッパーガイド」の「 でのゾーンオートシフトARC」を参照してください。 ARC

クロスゾーン無効のロードバランサーをご利用のお客様には、以下もお勧めします。

- アベイラビリティーゾーンのディストリビューションにのみ、バランスの取れた を使用します。
- Auto Scaling グループとロードバランサーの両方でゾーンシフトを使用している場合は、まず Auto Scaling グループのゾーンシフトをキャンセルします。次に、ロードバランサーのゾーンシフトをキャンセルする前に、すべてのアベイラビリティーゾーン間で容量のバランスが取れるまで待ちます。
- ゾーンシフトを有効にしてクロスゾーン無効ロードバランサーを使用すると、容量が不均衡になる可能性があるため、Auto Scaling には追加の検証ステップが含まれます。ベストプラクティスに従っている場合は、 AWS Management Console チェックボックスを選択するか、`CreateAutoScalingGroup`、、または で `skip-zonal-shift-validation`フラグを使用することで`UpdateAutoScalingGroup`、この可能性を確認できます`AttachTrafficSources`。

Auto Scaling グループでのゾーンシフトの使用の詳細については、「Amazon Auto Scaling AWS でのゾーンシフトの使用」のブログを参照してください。 EC2 Auto Scaling

AWS Management Console または を使用してゾーンシフトを有効にする AWS CLI

ゾーンシフトを有効にするには、次のいずれかの方法を使用します。

Console

新しいグループでゾーンシフトを有効にするには (コンソール)

1. 起動テンプレートを使用して Auto Scaling グループを作成する 「」の手順に従い、ステップ 10 までの手順の各ステップを完了します。

2. 「他のサービスとの統合」ページの「Application Recovery Controller (ARC) ゾーンシフト」で、「チェックボックスをオンにしてゾーンシフトを有効にします。
3. ヘルスチェックの動作で、「異常を無視する」または「異常を置き換える」を選択します。詳細については、「Auto Scaling グループのゾーンシフトの仕組み」を参照してください。
4. 起動テンプレートを使用して Auto Scaling グループを作成する のステップを続行します。

AWS CLI

新しいグループでゾーンシフトを有効にするには (AWS CLI)

`--availability-zone-impairment-policy` パラメータを create-auto-scaling-group コマンドに追加します。

`--availability-zone-impairment-policy` パラメータには 2 つのオプションがあります。

- ZonalShiftEnabled – に設定すると`true`、Auto Scaling は Auto Scaling ARC グループをゾーンシフトに登録し、ARCコンソールでゾーンシフトを開始、更新、またはキャンセルできます。に設定すると`false`、Auto Scaling は Auto Scaling ARC グループをゾーンシフトから登録解除します。を に設定するには、ゾーンシフトが既に有効になっている必要がありますfalse。
- ImpairedZoneHealthCheckBehavior – に設定すると`replace-unhealthy`、異常なインスタンスはアベイラビリティーゾーンのアクティブなゾーンシフトに置き換えられます。に設定すると`ignore-unhealthy`、アベイラビリティーゾーン内の異常なインスタンスはアクティブなゾーンシフトに置き換えられません。詳細については、「Auto Scaling グループのゾーンシフトの仕組み」を参照してください。

次の例では、 という名前の新しい Auto Scaling グループでゾーンシフトを有効にします*my-asg*。

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --availability-zones us-east-1a us-east-1b us-east-1c \
  --availability-zone-impairment-policy '{
      "ZonalShiftEnabled": true,
      "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy
```

```
    }'
```

Console

既存のグループでゾーンシフトを有効にするには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. 統合タブの Application Recovery Controller (ARC) ゾーンシフトで、編集を選択します。
5. ゾーンシフトを有効にするには、チェックボックスをオンにします。
6. ヘルスチェックの動作で、「異常を無視する」または「異常を置き換える」を選択します。詳細については、「Auto Scaling グループのゾーンシフトの仕組み」を参照してください。
7. [Update] (更新) を選択します。

AWS CLI

既存のグループでゾーンシフトを有効にするには (AWS CLI)

`--availability-zone-impairment-policy` パラメータを update-auto-scaling-group コマンドに追加します。

`--availability-zone-impairment-policy` パラメータには 2 つのオプションがあります。

- ZonalShiftEnabled – に設定すると`true`、Auto Scaling は Auto Scaling ARC グループをゾーンシフトに登録し、ARCコンソールでゾーンシフトを開始、更新、またはキャンセルできます。に設定すると`false`、Auto Scaling は Auto Scaling ARC グループをゾーンシフトから登録解除します。を に設定するには、ゾーンシフトが既に有効になっている必要がありますfalse。
- ImpairedZoneHealthCheckBehavior – に設定すると`replace-unhealthy`、異常なインスタンスはアベイラビリティーゾーンのアクティブなゾーンシフトに置き換えられます。に設定すると`ignore-unhealthy`、アベイラビリティーゾーン内の異常なインスタンスはアクティブな

ゾーンシフトに置き換えられません。詳細については、「Auto Scaling グループのゾーンシフトの仕組み」を参照してください。

次の例では、指定した Auto Scaling グループでゾーンシフトを有効にします。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --availability-zone-impairment-policy '{
      "ZonalShiftEnabled": true,
      "ImpairedZoneHealthCheckBehavior": IgnoreUnhealthy
    }'
```

Auto Scaling グループのアベイラビリティーゾーンディストリビューション

以下の情報は、Auto Scaling グループのアベイラビリティーゾーン戦略について説明しています。

バランスの取れたベストエフォート

Auto Scaling は、有効なアベイラビリティーゾーン間で同じ数のインスタンスを維持します。アベイラビリティーゾーンで起動の試行が失敗した場合、Auto Scaling は別の正常なアベイラビリティーゾーンでインスタンスの起動を試みます。この戦略は、アベイラビリティーゾーンの冗長性を必要とし、不均衡なグループの影響を受けないアプリケーションにとって重要です。

残高のみ

Auto Scaling は、有効なアベイラビリティーゾーン間で同じ数のインスタンスを維持します。アベイラビリティーゾーンで起動の試行が失敗した場合、Auto Scaling はアベイラビリティーゾーンでインスタンスの起動を引き続き試みます。この戦略は、クォーラムベースのワークロードや、残りのアベイラビリティーゾーンに十分な容量があるために Auto Scaling グループがアベイラビリティーゾーンの損失を許容できるなどの特定の要件を満たすために重要です。

アベイラビリティーゾーンの分散戦略の選択は、 のネットワークセクションにあります。 AWS Management Console または、 update-auto-scaling-group コマンドcreate-auto-scaling-groupを使用できます。

詳細については、「起動テンプレートを使用して Auto Scaling グループを作成する」を参照してください。

Auto Scaling グループからインスタンスをデタッチまたはアタッチする

Auto Scaling グループからインスタンスをデタッチできます。インスタンスがデタッチされると、そのインスタンスは独立した存在になり、独自に管理することも、属していた元のグループとは別の異なる Auto Scaling グループにアタッチすることもできます。これは、例えば、既にアプリケーションを実行している既存のインスタンスを使用してテストを実行する場合などに便利です。

このトピックでは、インスタンスをデタッチおよびアタッチする方法について説明します。インスタンスをアタッチする場合、デタッチしたインスタンスではなく既存のインスタンスを使用することもできます。

インスタンスをデタッチして同じグループに再アタッチする代わりに、スタンバイ手順を使用して、グループからインスタンスを一時的に削除することをお勧めします。詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

内容

インスタンスのデタッチに関する考慮事項

インスタンスをデタッチするときは、次の点に注意してください。

- インスタンスをデタッチできるのは、インスタンスが `InService` 状態にある場合のみです。
- インスタンスをデタッチした後も、インスタンスは引き続き実行され、料金が発生します。不要な料金が発生しないように、デタッチされたインスタンスが不要になった場合は、必ず再アタッチまたは終了してください。
- 必要なキャパシティを、デタッチするインスタンスの数だけ減らすことを選択できます。容量を減らさない場合、Amazon EC2 Auto Scaling は新しいインスタンスを起動してデタッチされたインスタンスを置き換え、必要な容量を維持します。
- デタッチするインスタンスの数により Auto Scaling グループの最小キャパシティを下回る状況が発生する場合は、最小キャパシティを減らす必要があります。

- 必要なキャパシティを減らすことなく同じアベイラビリティーゾーンから複数のインスタンスをデタッチすると、AZRebalance プロセスを中断しない限り、グループ自体が再調整されます。詳細については、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。
- ロードバランサーターゲットグループまたは Classic Load Balancer にアタッチした Auto Scaling グループからインスタンスをデタッチすると、インスタンスはロードバランサーから登録解除されます。ロードバランサーで Connection Draining (登録解除の遅延) が有効になっている場合、Amazon EC2 Auto Scaling は処理中のリクエストが完了するまで待機します。

> **Note**
>
> Standby 状態にあるインスタンスをデタッチする場合は注意してください。Standby 状態にしたインスタンスをデタッチしようとすると、他のインスタンスが予期せず終了することがあります。

インスタンスのアタッチに関する考慮事項

インスタンスをアタッチするときは、次の点に注意してください。

- Amazon EC2 Auto Scaling は、アタッチされたインスタンスをグループ自体によって起動されたインスタンスと同じように扱います。つまり、アタッチしたインスタンスが選択された場合、そのインスタンスはスケールインイベント中に終了できます。によって付与されるアクセス許可 AWSServiceRoleForAutoScaling サービスにリンクされたロールにより、Amazon EC2 Auto Scaling はこれを行うことができます。
- インスタンスをアタッチすると、アタッチされるインスタンスの数によって、グループの必要なキャパシティーは増加します。新しいインスタンスを追加した後の必要なキャパシティがグループの最大サイズを超える場合、インスタンスを追加でアタッチするリクエストは失敗します。
- アベイラビリティーゾーン間で分散が不均等になるインスタンスをグループに追加すると、Amazon EC2 Auto Scaling はグループのバランスを再調整して、AZRebalanceプロセスを停止しない限り、均等分散を再確立します。詳細については、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。
- インスタンスをロードバランサーターゲットグループまたは Classic Load Balancer にアタッチした Auto Scaling グループにアタッチする場合、インスタンスはロードバランサーに登録されます。

アタッチするインスタンスについては、次の条件を満たす必要があります。

- インスタンスは Amazon で `running`状態ですEC2。
- インスタンスの起動AMIに使用される は、まだ存在している必要があります。
- インスタンスは他の Auto Scaling グループのメンバーではありません。
- インスタンスは、Auto Scaling グループで定義されているいずれかのアベイラビリティーゾーンに起動されます。
- Auto Scaling グループにロードバランサーターゲットグループまたは Classic Load Balancer がアタッチされている場合、インスタンスとロードバランサーの両方が同じ に存在する必要がありますVPC。

デタッチとアタッチを使用してインスタンスを別のグループに移行する

次のいずれかの手順を使用して、インスタンスを Auto Scaling グループからデタッチし、別の Auto Scaling グループにアタッチします。

デタッチしたインスタンスから新しい Auto Scaling グループを作成するには、「を使用して既存のインスタンスから Auto Scaling グループを作成する AWS CLI」を参照してください (起動設定を作成するため、非推奨)。

Console

Auto Scaling グループからインスタンスをデタッチする方法

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Instance management (インスタンス管理)] タブの [Instances (インスタンス)] でインスタンスを選択し、[Actions (アクション)]、[Detach (デタッチ)] の順に選択します。
4. [インスタンスをデタッチ] ダイアログボックスで、[インスタンスを置き換える] チェックボックスをオンのままにして、置換インスタンスを起動します。必要なキャパシティを減らすには、チェックボックスをオフにします。
5. 確認を求めるプロンプトが表示されたら、指定したインスタンスを Auto Scaling グループから削除することを確認するために **detach** と入力し、[インスタンスのデタッチ] を選択します。

インスタンスを別の Auto Scaling グループにアタッチできるようになりました。

Auto Scaling グループにインスタンスをアタッチする方法

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. (オプション) ナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。Auto Scaling グループを選択し、Auto Scaling グループの最大サイズが別のインスタンスを追加できる十分な大きさであることを確認します。大きさが十分でない場合は、[詳細] タブで最大キャパシティーを増やします。
3. ナビゲーションペインの [Instances] (インスタンス) で [Instances] (インスタンス) を選択してから、インスタンスを選択します。
4. [Actions]、[Instance Settings]、[Attach to Auto Scaling Group] の順に選択します。
5. [Attach to Auto Scaling Group (Auto Scaling Group にアタッチ)] ページで、[Auto Scaling group (Auto Scalingグループ)] を選択し、[Attach (アタッチ)] を選択します。
6. インスタンスがこの基準を満たさない場合、エラーメッセージとその詳細が表示されます。例えば、インスタンスが Auto Scaling グループと同じアベイラビリティーゾーンにない可能性があります。[閉じる] を選択して、この基準を満たす Auto Scaling グループでもう一度試してください。

AWS CLI

インスタンスをデタッチおよびアタッチするには、次のコマンド例を使用します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

Auto Scaling グループからインスタンスをデタッチする方法

1. 現在のインスタンスを記述するには、次のdescribe-auto-scaling-instancesコマンドを使用します。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

次の例は、このコマンドを実行したときに生成される出力を示しています。

グループから削除するインスタンスの ID を書き留めます。この ID は次のステップで必要になります。

```
{
```

```
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0c20ac468fa3049e8",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0787762faf1c28619",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
```

```
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0f280a4c58d319a8a",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        }
    ]
}
```

2. 必要なキャパシティを減らすことなくインスタンスをデタッチするには、次の detach-instances コマンドを使用します。

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg
```

インスタンスをデタッチし、必要なキャパシティを減らすには、--should-decrement-desired-capacity オプションを含めます。

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

インスタンスを別の Auto Scaling グループにアタッチできるようになりました。

Auto Scaling グループにインスタンスをアタッチする方法

1. インスタンスを別の Auto Scaling グループにアタッチするには、次の attach-instances コマンドを使用します

```
aws autoscaling attach-instances --instance-ids i-05b4f7d5be44822a6 --auto-
scaling-group-name my-asg-for-testing
```

2. インスタンスをアタッチした後に Auto Scaling グループのサイズを確認するには、次のdescribe-auto-scaling-groupsコマンドを使用します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg-for-testing
```

以下の応答例は、グループに実行中のインスタンスが 2 つあり、そのうちの 1 つはアタッチしたインスタンスであることを示しています。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg-for-testing",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "2",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 2,
            ...
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "2",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
```

```
                    "InstanceId": "i-00dcdfffdf5175890",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            ...
        }
    ]
}
```

Auto Scaling グループからインスタンスを一時的に削除する

インスタンスを InService 状態から Standby 状態に移行でき、インスタンスを更新またはトラブルシューティングして、インスタンスをサービスに返すことができます。スタンバイ状態のインスタンスはまだ Auto Scaling グループの一部ですが、ロードバランサートラフィックをアクティブに処理しません。

この機能は、Amazon EC2 Auto Scaling がヘルスチェックの一部として、またはスケールインイベント中にインスタンスを終了することを心配することなく、インスタンスを停止して起動したり、再起動したりするために役立ちます。

例えば、起動テンプレートまたは起動設定を変更することで、Auto Scaling グループの Amazon マシンイメージ (AMI) をいつでも変更できます。Auto Scaling グループが起動する後続のインスタンスは、この を使用しますAMI。ただし、Auto Scaling グループは現在稼働中のインスタンスを更新しません。これらのインスタンスを終了して Amazon EC2 Auto Scaling に置換させるか、インスタンスの更新機能を使用してインスタンスを終了して置き換えることができます。または、インスタンスをスタンバイ状態にしてソフトウェアを更新し、次にインスタンスをサービスに戻すことができます。

Auto Scaling グループからインスタンスをデタッチすることは、インスタンスをスタンバイ状態にすることと似ています。インスタンスを別のグループにアタッチしたり、スタンドアロンインスタンスのようにインスタンスを管理したりして終了させたりする場合は、EC2インスタンスをデタッチすると便利です。詳細については、「Auto Scaling グループからインスタンスをデタッチまたはアタッチする」を参照してください。

内容

- スタンバイ状態のインスタンスのヘルスステータス
- インスタンスをスタンバイに設定して一時的に削除する

スタンバイ状態の仕組み

Auto Scaling グループからインスタンスを一時的に削除できるように、スタンバイ状態は次のように機能します:

1. ユーザーはインスタンスをスタンバイ状態にします。スタンバイ状態を終了するまで、インスタンスはこの状態のままです。
2. Auto Scaling グループにアタッチされたロードバランサーターゲットグループまたは Classic Load Balancer がある場合、インスタンスはロードバランサーから登録解除されます。Connection Draining がロードバランサーに対して有効になっている場合、Elastic Load Balancingは登録解除プロセス完了前にデフォルトで 300 秒待ちます。これは、処理中のリクエストの完了に役立ちます。
3. インスタンスを更新またはトラブルシューティングできます。
4. スタンバイ状態を終了することにより、インスタンスを稼働状態に戻します。
5. Auto Scaling グループにアタッチされたロードバランサーターゲットグループまたは Classic Load Balancer がある場合、インスタンスはロードバランサーに登録されます。

Auto Scaling グループのインスタンスのライフサイクルの詳細については、「Amazon EC2 Auto Scaling インスタンスのライフサイクル」を参照してください。

考慮事項

インスタンスをスタンバイ状態に移行したり、スタンバイ状態から移行したりする際の考慮事項を次に示します。

- インスタンスをスタンバイ状態にするとき、このオペレーションを通じて必要なキャパシティを減らすことも、同じ値を維持することもできます。
  - Auto Scaling グループの希望する容量を減らしない場合、Amazon EC2 Auto Scaling はインスタンスを起動してスタンバイ状態のインスタンスを置き換えます。その目的は、1 つ以上のインスタンスがスタンバイ状態である間にアプリケーションのキャパシティーを維持できるようにすることです。
  - Auto Scaling グループの必要なキャパシティを減らすことを選択すると、スタンバイ状態のインスタンスを置き換えるためのインスタンスを起動できなくなります。

- インスタンスを稼働状態に戻すと、Auto Scaling グループ内のインスタンスの数を反映するために必要なキャパシティが増加します。
- 増加 (および減少) を行うには、新しい必要なキャパシティは、最小グループサイズと最大グループサイズの間にある必要があります。それ以外の場合は、このオペレーションは失敗します。
- インスタンスをスタンバイ状態にした後、またはスタンバイ状態を終了してインスタンスをサービスに戻した後、Auto Scaling グループがアベイラビリティーゾーン間でバランスが取れていないことがわかった場合、AZRebalanceAmazon EC2 Auto Scaling はプロセスを停止しない限り、アベイラビリティーゾーンのバランスを再調整して補正します。詳細については、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。
- スタンバイ状態のインスタンスに対して課金されます。

スタンバイ状態のインスタンスのヘルスステータス

Amazon EC2 Auto Scaling は、スタンバイ状態のインスタンスではヘルスチェックを実行しません。インスタンスがスタンバイ状態にあるとき、スタンバイ状態に移行する前の状態がヘルスステータスに反映されます。Amazon EC2 Auto Scaling は、インスタンスを稼働状態に戻すまで、インスタンスのヘルスチェックを実行しません。

例えば、正常なインスタンスをスタンバイ状態にしてから終了すると、Amazon EC2 Auto Scaling は引き続きインスタンスを正常と報告します。スタンバイ状態の終了したインスタンスを稼働状態に戻そうとすると、Amazon EC2 Auto Scaling はインスタンスのヘルスチェックを実行し、インスタンスが終了して異常であると判断して、代替インスタンスを起動します。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

インスタンスをスタンバイに設定して一時的に削除する

次のいずれかの手順を使用して、インスタンスをスタンバイ状態にして一時的に使用を中止します。

Console

インスタンスを一時的に削除するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。

3. [Instance management (インスタンス管理)] タブの [Instances (インスタンス)] で、インスタンスを選択します。
4. [Actions]、[Set to Standby] を選択します。
5. [スタンバイに設定] ダイアログボックスで、[インスタンスを置き換える] チェックボックスをオンのままにして、置換インスタンスを起動します。必要なキャパシティを減らすには、チェックボックスをオフにします。
6. 確認を求めるプロンプトが表示されたら、指定したインスタンスを Standby 状態にすることを確認するために **standby** と入力し、[スタンバイに設定] を選択します。
7. 必要に応じてインスタンスを更新、またはトラブルシューティングできます。終了したら、インスタンスを稼働状態に戻すために次のステップに進みます。
8. インスタンスを選択し、アクション、設定を選択します InService。Set to InService ダイアログボックスで、Set to InServiceを選択します。

AWS CLI

Auto Scaling グループからインスタンスを一時的に削除するには、次のコマンド例を使用します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

インスタンスを一時的に削除するには

1. 次の describe-auto-scaling-instances コマンドを実行して、更新するインスタンスを確認します。

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

次の例は、このコマンドを実行したときに生成される出力を示しています。

グループから削除するインスタンスの ID を書き留めます。この ID は次のステップで必要になります。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
```

```
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceId": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        ...
    ]
}
```

2. インスタンスを次の enter-standby コマンドを使用して、Standby 状態に移行させます。--should-decrement-desired-capacity オプションは、Auto Scaling グループで代わりのインスタンスが起動されないように希望するキャパシティーを減らします。

```
aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

以下に、応答の例を示します。

```
{
    "Activities": [
        {
            "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby:
 i-05b4f7d5be44822a6",
            "Cause": "At 2023-12-15T21:31:26Z instance i-05b4f7d5be44822a6 was
 moved to standby
              in response to a user request, shrinking the capacity from 4 to
 3.",
            "StartTime": "2023-12-15T21:31:26.150Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
        }
    ]
}
```

3. (オプション) 以下の describe-auto-scaling-instances コマンドを使用して、インスタンスが Standby であることを確認します。

```
aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6
```

以下に、応答の例を示します。インスタンスのステータスが Standby に設定されました。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "Standby"
        },
        ...
    ]
}
```

4. 必要に応じてインスタンスを更新、またはトラブルシューティングできます。終了したら、インスタンスを稼働状態に戻すために次のステップに進みます。

5. 次の exit-standby コマンドを使用してインスタンスをサービスに戻します。

```
aws autoscaling exit-standby --instance-ids i-05b4f7d5be44822a6 --auto-scaling-
group-name my-asg
```

以下に、応答の例を示します。

```
{
    "Activities": [
        {
            "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
```

```
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-05b4f7d5be44822a6",
            "Cause": "At 2023-12-15T21:46:14Z instance i-05b4f7d5be44822a6 was
 moved out of standby in
               response to a user request, increasing the capacity from 3 to
 4.",
            "StartTime": "2023-12-15T21:46:14.678Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
        }
    ]
}
```

6. (オプション) 以下の describe-auto-scaling-instances コマンドを使用して、インスタンスが稼働状態に戻っていることを確認します。

```
aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6
```

以下に、応答の例を示します。インスタンスのステータスが InService に設定されました。

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
       ...
```

```
    ]
}
```

Auto Scaling インフラストラクチャを削除する

スケーリングインフラストラクチャを完全に削除するには、次のタスクを実行します。

タスク

- Auto Scaling グループの削除
- (オプション) 起動設定の削除
- (オプション) 起動テンプレートの削除
- (オプション) ロードバランサーとターゲットグループの削除
- (オプション) CloudWatch アラームの削除

Auto Scaling グループの削除

Auto Scaling グループを削除すると、目的の値、最小値、および最大値は 0 に設定されます。その結果、インスタンスは削除されます。インスタンスを削除すると、関連するログまたはデータ、およびインスタンスのすべてのボリュームも削除します。1 つ以上のインスタンスを終了しない場合は、Auto Scaling グループを削除する前にこれらをデタッチすることができます。グループにスケーリングポリシーがある場合、グループを削除すると、ポリシー、基盤となるアラームアクション、および関連付けられたアクションがなくなったアラームが削除されます。

Auto Scaling グループを削除するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの隣にあるチェックボックスを選択し、[アクション]、[削除] を選択します。
3. 確認を求められたら、**delete** を入力して指定された Auto Scaling グループの削除を確認し、[Delete] (削除) を選択します。

   [Name (名前)] 列のロードアイコンに、Auto Scaling グループが削除されたことが示されます。[Desired] (希望する)、[Min] (最小)、[Max] (最大) 列には、Auto Scaling グループの `0` インスタンスが表示されます。インスタンスを終了し、グループを削除するには数分かかります。リストを更新して、現在の状態を確認します。

Auto Scaling グループを削除するには (AWS CLI)

次のdelete-auto-scaling-groupコマンドを使用して、Auto Scaling グループを削除します。グループにEC2インスタンスがある場合、このオペレーションは機能しません。インスタンスがゼロのグループの に対してのみ機能します。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

グループに進行中のインスタンスまたはスケーリングアクティビティがある場合は、 `--force-delete`オプションを指定して delete-auto-scaling-group コマンドを使用します。これにより、EC2インスタンスも終了します。Amazon Auto Scaling コンソールから EC2 Auto Scaling グループを削除すると、コンソールはこのオペレーションを使用してEC2インスタンスを終了し、同時にグループを削除します。

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg --force-
delete
```

(オプション) 起動設定の削除

今後使用できるように起動設定を保存するには、このステップをスキップします。

起動設定を削除するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. 左のナビゲーションペインの [Auto Scaling] で、[Auto Scaling グループ] を選択します。
3. ページの上部付近にある [起動設定] を選択します。確認を求めるプロンプトが表示されたら、[起動設定を表示] を選択して、[起動設定] ページを表示することを確認します。
4. 起動設定を選択し、[アクション]、[起動設定の削除] の順に選択します。
5. 確認を求めるメッセージが表示されたら、[削除] を選択します。

起動設定を削除するには (AWS CLI)

次の delete-launch-configuration コマンドを使用します。

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-launch-
config
```

(オプション) 起動テンプレートの削除

起動テンプレートを削除することも、1 つの起動テンプレートバージョンを削除することもできます。起動テンプレートを削除すると、そのすべてのバージョンが削除されます。

このステップをスキップして、後で使用するために起動テンプレートを維持することもできます。

起動テンプレートを削除するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインで、[インスタンス] の [テンプレートの起動] を選択します。
3. 起動テンプレートを選択し、次のいずれかの操作を行います。

   - [アクション]、[テンプレートの削除] の順に選択します。確認を求められたら、**Delete** を入力して指定した起動テンプレートの削除を確認し、[Delete] (削除) を選択します。
   - [アクション]、[Delete template version (テンプレートのバージョンの削除)] の順に選択します。削除するバージョンを選択し、[削除] を選択します。

起動テンプレートを削除するには (AWS CLI)

次の delete-launch-template コマンドを使用して、テンプレートとそのすべてのバージョンを削除します。

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

または、 delete-launch-template-versions コマンドを使用して特定の起動テンプレートのバージョンを削除することもできます。

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --
versions 1
```

(オプション) ロードバランサーとターゲットグループの削除

Auto Scaling グループが Elastic Load Balancing ロードバランサーに関連付けされていない場合、または今後使用できるようにロードバランサーを維持する場合、このステップをスキップします。

ロードバランサーを削除するには (コンソール)

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。

2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. ロードバランサーを選択してから、[Actions (アクション)]、[Delete (削除)] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

ターゲットグループを削除するには (コンソール)

1. ナビゲーションペインの [ロードバランシング] で [ターゲットグループ] を選択します。
2. ターゲットグループを選択し、[Actions (アクション)]、[Delete (削除)] を選択します。
3. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

Auto Scaling グループに関連付けられているロードバランサーを削除するには (AWS CLI)

Application Load Balancer と Network Load Balancer の場合は、次の delete-load-balancerおよび delete-target-group コマンドを使用します。

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

Classic Load Balancer の場合は、次のdelete-load-balancerコマンドを使用します。

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

(オプション) CloudWatch アラームの削除

Auto Scaling グループに関連付けられている CloudWatch アラームを削除するには、次の手順を実行します。例えば、ステップスケーリングまたはシンプルスケーリングポリシーに関連するアラームがあるかもしれません。

> **Note**
>
> Auto Scaling グループを削除すると、Amazon EC2 Auto Scaling がターゲット追跡スケーリングポリシーに対して管理する CloudWatch アラームが自動的に削除されます。

Auto Scaling グループが CloudWatch アラームに関連付けられていない場合、または今後使用するためにアラームを保持する場合は、このステップをスキップできます。

CloudWatch アラームを削除するには (コンソール)

1. で CloudWatch コンソールを開きますhttps://console.aws.amazon.com/cloudwatch/。
2. ナビゲーションペインで、[アラーム] を選択します。
3. アラームを選び、[Action (アクション)]、[Delete (削除)] を選択します。
4. 確認を求めるメッセージが表示されたら、[削除] を選択します。

CloudWatch アラームを削除するには (AWS CLI)

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-AddCapacity` アラームおよび `Step-Scaling-AlarmLow-RemoveCapacity` アラームを削除します。

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

Auto Scaling グループ内のインスタンスを置き換えます。

Amazon EC2 Auto Scaling には、新しい起動テンプレートを新しい Amazon マシンイメージ (EC2) で追加したり、新しいインスタンスタイプを追加したりするなど、更新を行った後に Auto Scaling グループ内の Amazon AMI インスタンスを置き換えることができる機能が用意されています。また、インスタンスを置き換えるのと同じオペレーションに更新を含めるオプションを提供することで、更新の効率化にも役立ちます。

このセクションには、以下の作業に役立つ情報が記載されています。

- インスタンスの更新をスタートして、Auto Scaling グループのインスタンスを置き換えます。
- 希望の設定を説明する特定の更新を宣言し、Auto Scaling グループを希望の設定に更新します。
- 更新済みのインスタンスの置き換えをスキップします。
- チェックポイントを使用してインスタンスを段階的に更新し、特定の時点でインスタンスの検証を実行します。
- ベイク時間を使用して、インスタンスの更新の終了時に一時停止し、インスタンスの状態を検証します。
- チェックポイントに到達すると、電子メールで通知を受信します。
- ロールバックを使用して Auto Scaling グループが以前使用していた設定に戻します。
- インスタンスの更新が何らかの理由で失敗した場合、または指定した Amazon CloudWatch アラームが ALARM状態になった場合、自動的にロールバックします。
- インスタンスの存続期間を制限し、Auto Scaling グループ全体でソフトウェアバージョンおよびインスタンス設定の一貫性を提供します。

内容

インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する

Auto Scaling グループのインスタンスを更新するには、インスタンスの更新を使用できます。設定の変更でインスタンスの置き換えが必要がある場合、Auto Scaling グループに多数のインスタンスが含まれているときには、この機能が効果的です。

インスタンスの更新が役立つ状況には、次のようなものがあります。

- Auto Scaling グループに新しい Amazon マシンイメージ (AMI) またはユーザーデータスクリプトをデプロイします。変更を含む新しい起動テンプレートを作成し、インスタンスの更新を使用して更新をすぐにロールアウトできます。
- インスタンスを新しいインスタンスタイプに移行して、最新の改善点と最適化を活用します。
- Auto Scaling グループの起動設定の使用から起動テンプレートの使用への切り替え。起動設定をコピーしてテンプレートを起動し、インスタンスの更新を使用してインスタンスを新しいテンプレートに更新できます。起動テンプレートの移行については、「Auto Scaling グループを起動テンプレートに移行する」を参照してください。

内容

Auto Scaling グループでのインスタンスの更新の仕組み

このトピックでは、インスタンスの更新がどのように機能するかを説明し、インスタンスを効果的に活用するための基礎知識について紹介します。

内容

仕組み

Auto Scaling グループのインスタンスを更新するには、アプリケーションの最新バージョンと、実行するその他の更新を含む新しい設定を定義できます。次に、インスタンスの更新を開始し、その設定に基づいて既存のインスタンスを新しいインスタンスに置き換えます。

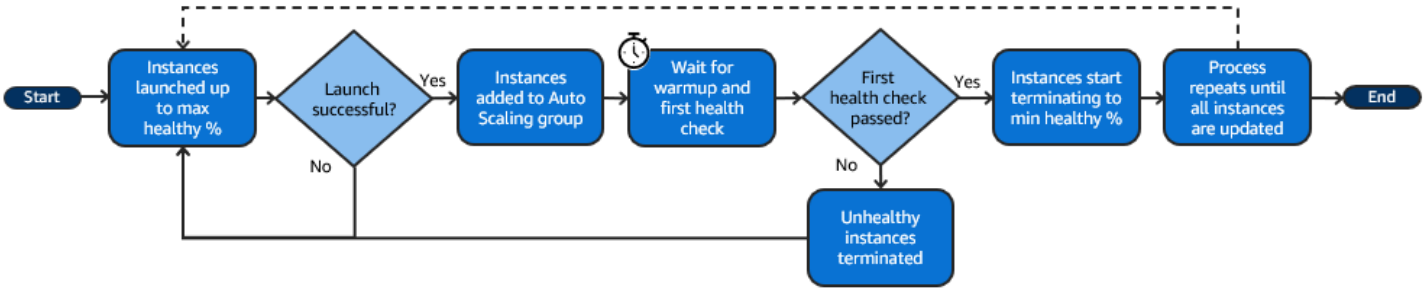
インスタンスの更新を実行するには：

1. 新しい起動テンプレートを作成するか、新しい Amazon マシンイメージ (AMI) などの必要な設定変更で既存のテンプレートを更新します。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。
2. Amazon EC2 Auto Scaling コンソール AWS CLI、または SDK を使用してインスタンスの更新を開始します。
   - 作成した新しい起動テンプレートまたは起動テンプレートのバージョンを指定します。これは、新しいインスタンスを起動するために使用されます。
   - 適切な最小正常率と最大正常率を設定します。この設定により、同時に置き換えられるインスタンスの数と、古いインスタンスを終了する前に新しいインスタンスを起動するかどうかが制御されます。
   - 必要に応じて、以下の設定を行います。
     - チェックポイント – 一定割合のインスタンスが置き換えられた後、インスタンスの更新を一時停止して、進行状況を確認します。
     - ベイク時間 — インスタンスの更新が完了と見なされる前に、インスタンスの更新の終了時に一時停止してインスタンスの状態を検証します。
     - スキップマッチング – 古いインスタンスを新しい設定と比較し、一致しないインスタンスのみを置き換えます。コンソールからインスタンスの更新を開始すると、スキップマッチングはデフォルトで有効になります。
     - 複数のインスタンスタイプ – 希望する設定の一部として、新規または更新された混合インスタンスポリシーを適用します。

インスタンスの更新が開始されると、Amazon EC2 Auto Scaling は次の処理を行います。

- 最小正常率および最大正常率に基づいて、バッチ内のインスタンスを置き換えます。
- 最小正常率が 100% に設定されている場合、古いインスタンスを終了する前に新しいインスタンスを起動します。これにより、希望するキャパシティが常に維持されます。
- インスタンスのヘルスステータスを確認し、ウォームアップ時間を確保してから、残りのインスタンスを置き換えます。
- 異常が見つかったインスタンスを終了して置き換えます。
- インスタンスの更新が成功すると、新しい設定変更で Auto Scaling グループ設定が自動的に更新されます。
- ウォーム プール内のインスタンスの前に、`InService` インスタンスを置き換えます。

次のフロー図は、最小正常率を 100% に設定した場合の、終了前の起動動作を示しています。



> **Note**
>
> インスタンス更新の最小正常率と最大正常パーセンテージは、インスタンスメンテナンスポリシーを設定していない場合、または既存のポリシーを上書きする必要がある場合のみ指定する必要があります。詳細については、「インスタンスメンテナンスポリシー」を参照してください。
>
> 同様に、インスタンス更新のインスタンスウォームアップ期間を指定する必要があるのは、デフォルトのウォームアップを有効にしていない場合、またはデフォルトを上書きする必要がある場合のみです。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

重要な概念

開始する前に、以下の インスタンス更新の主要概念を理解してください。

最小正常率

最小正常率は、更新を続行できるように、インスタンスの更新中に稼働状態を維持し、正常な状態で使用できる状態に維持する必要があるキャパシティの割合です。例えば、最小正常率が 90% で、最大正常率が 100% の場合、キャパシティの 10% が一度に置き換えられます。新しいインスタンスがヘルスチェックに合格しない場合、Amazon EC2 Auto Scaling はそれらを終了して置き換えます。インスタンスの更新で正常なインスタンスを起動できない場合、最終的に失敗し、グループの残りの 90% はそのままになります。新しいインスタンスが正常で、ウォームアップ期間が終了した場合、Amazon EC2 Auto Scaling は他のインスタンスを引き続き置き換えることができます。

インスタンスの更新では、インスタンスを 1 つずつ、複数ずつ、またはすべてを一度に置き換えることができます。一度に 1 つのインスタンスを置き換えるには、最小正常率と最大正常率の両方を 100% に設定します。この変更により、終了する前にインスタンスの更新の動作が起動するように変更され、グループのキャパシティが希望するキャパシティを下回ることが防止されます。すべてのインスタンスを一度に置き換えるには、最小正常率を 0% に設定します。

最大正常率

最大正常率は、インスタンスを置き換えるときに Auto Scaling グループが増加できる、適切なキャパシティの割合です。最小と最大の差分は 100 を超えることはできません。範囲を大きくすると、同時に置き換えることができるインスタンスの数が増えます。

インスタンスのウォームアップ

インスタンスのウォームアップとは、新しいインスタンスの状態が `InService` に変更されてから、初期化が完了したとみなされるまでの期間です。インスタンスの更新中にインスタンスがヘルスチェックに合格した場合、Amazon EC2 Auto Scaling は、新しく起動されたインスタンスが正常であると判断した後、すぐに次のインスタンスの置き換えに移行しません。ウォームアップ期間を待機してから、次のインスタンスの置き換えに進みます。これは、アプリケーションがリクエストに応答するまでにまだ初期化時間が必要な場合に役立ちます。

インスタンスのウォームアップは、デフォルトのインスタンスのウォームアップと同じように機能します。したがって、同じスケーリングに関する考慮事項が適用されます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

必要な設定

必要な設定は、Amazon EC2 Auto Scaling が Auto Scaling グループ全体にデプロイする新しい設定です。例えば、インスタンスに新しい起動テンプレートおよび新しいインスタンスタイ

プを指定できます。インスタンスの更新中、Amazon EC2 Auto Scaling は Auto Scaling グループを希望の設定に更新します。インスタンスの更新中にスケールアウトイベントが発生した場合、Amazon EC2 Auto Scaling は、グループの現在の設定ではなく、必要な設定で新しいインスタンスを起動します。インスタンスの更新が成功すると、Amazon EC2 Auto Scaling は Auto Scaling グループ設定を更新して、インスタンスの更新の一部として指定した新しい必要な設定を反映します。

スキップマッチング

スキップマッチングは、Amazon EC2 Auto Scaling に、すでに最新の更新があるインスタンスを無視するように指示します。これにより、必要以上のインスタンスを置き換えることはありません。これは、Auto Scaling グループが特定バージョンの起動テンプレートを使用し、異なるバージョンを使用するインスタンスのみを置き換えることを確認したいときに役立ちます。

チェックポイント

チェックポイントとは、インスタンスの更新が指定した時間のみ一時停止する時点のことです。インスタンスの更新には、複数のチェックポイントを含めることができます。Amazon EC2 Auto Scaling は、チェックポイントごとにイベントを発行します。したがって、チェックポイントに達したときに通知されるように、Amazon EventBridge などのターゲットにイベントを送信する SNS ルールを追加できます。チェックポイントに到達した後で、デプロイを検証する機会があります。問題が特定された場合、インスタンスの更新をキャンセルまたはロールバックできます。更新プログラムを段階的にデプロイできることは、チェックポイントのキーの利点です。チェックポイントを使用しない場合、ローリング置換は継続的に実行されます。

インスタンスの更新を開始するときに設定できるすべてのデフォルト設定の詳細については、「インスタンスの更新のデフォルト値について説明する」を参照してください。

ヘルスチェックの猶予期間

Amazon EC2 Auto Scaling は、Auto Scaling グループが使用するヘルスチェックのステータスに基づいて、インスタンスが正常かどうかを決定します。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

これらのヘルスチェックをできるだけ早く開始するには、グループのヘルスチェック猶予期間をあまり長く設定しないでください。ただし、Elastic Load Balancing ヘルスチェックがターゲットがリクエストを処理できるかどうかを判断するのに十分な長さに設定します。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

インスタンスタイプの互換性

インスタンスタイプを変更する前に、起動テンプレートで動作することを確認することをお勧めします。これにより、指定した AMI との互換性が確認されます。例えば、準仮想化 (PV) AMI から元のインスタンスを起動したが、ハードウェア仮想マシン (HVM) AMI でのみサポートされている現行世代のインスタンスタイプに変更するとします。この場合、起動テンプレートで AMIHVM を使用する必要があります。

インスタンスを起動せずにインスタンスタイプの互換性を確認するには、次の例で示すように、「run-instances」コマンドを `--dry-run` オプションとともに使用します。

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --
dry-run
```

互換性の決定方法については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプを変更するための互換性」を参照してください。

制限

- 合計持続時間: インスタンスの更新がインスタンスをアクティブに置き換えることができる最大時間は、14 日間です。
- 加重グループ固有の動作の違い: 混合インスタンスグループが、グループの希望する容量以上のインスタンスの重みで設定されている場合、Amazon EC2 Auto Scaling はすべての`InService`インスタンスを一度に置き換えることがあります。このような状況を回避するには、インスタンスの重みを使用するように Auto Scaling グループを設定する トピックの推奨事項に従ってください。Auto Scaling グループで分量を使用するとき、最大の分量よりも大きい必要キャパシティを指定します。
- 1 時間のタイムアウト: スタンバイ状態またはスケールインから保護されているインスタンスを置き換えるのを待っているためにインスタンスの更新が引き続き置換できない場合、または新しいインスタンスがヘルスチェックに合格しない場合、Amazon EC2 Auto Scaling は 1 時間再試行し続けます。加えて、問題の解決に役立つステータスメッセージが表示されます。1 時間経っても問題が解決しない場合は、オペレーションは失敗となります。1 時的な問題が発生した場合に、回復する時間を与えることを意図しています。
- ユーザーデータによるコードのデプロイ: スキップマッチングは、ユーザーデータスクリプトからデプロイされたコードの変更をチェックしません。ユーザーデータを使用して新しいコードを取得し、これらの更新を新しいインスタンスにインストールする場合は、起動テンプレートのバージョン更新がない場合でも、すべてのインスタンスが最新のコードを受信できるように、スキップマッチングをオフにすることをお勧めします。

- 更新制限：目的の設定でインスタンスの更新が実行中に Auto Scaling グループの起動テンプレート、起動設定、または混合インスタンスポリシーを更新しようとすると、これらのリクエストは失敗し、次の検証エラーが発生します。`An active instance refresh with a desired configuration exists. All configuration options derived from the desired configuration are not available for update while the instance refresh is active.`

インスタンスの更新のデフォルト値について説明する

インスタンスの更新を開始する前に、インスタンスの更新に影響するさまざまな設定をカスタマイズできます。一部の設定のデフォルトは、コンソールとコマンドライン (AWS CLI または AWS SDK) のどちらを使用するかによって異なります。

次の表には、インスタンスの更新設定のデフォルト値が一覧表示されています。

設定	AWS CLI または AWS SDK	Amazon EC2 Auto Scaling コンソール
CloudWatch アラーム	無効 (null)	無効
[自動ロールバック]	無効 (`false`)	無効
ベーク時間	ゼロ	ゼロ
チェックポイント	無効 (`false`)	無効
[チェックポイントの遅延]	1 時間 (`3600` 秒)	1 時間
インスタンスのウォームアップ	定義されている場合は デフォルトのインスタンスのウォームアップ。定義されていない場合はヘルスチェックの猶予期間。	定義されている場合は デフォルトのインスタンスのウォームアップ。定義されていない場合はヘルスチェックの猶予期間。
最大正常率	インスタンスのメンテナンスポリシーによって異なります。インスタンスメンテナンスポリシーがない場合、デ	インスタンスのメンテナンスポリシーによって異なります。インスタンスメンテナンスポリシーがない場合、デ

設定	AWS CLI または AWS SDK	Amazon EC2 Auto Scaling コンソール
	フォルトで 100% (null) になります。	フォルトで 100% (null) になります。
最小正常率	インスタンスのメンテナンスポリシーによって異なります。インスタンスメンテナンスポリシーがない場合、デフォルトで 90% になります。	インスタンスのメンテナンスポリシーによって異なります。インスタンスメンテナンスポリシーがない場合、デフォルトで 90% になります。
[スケールインで保護されたインスタンス]	待機	Ignore (無視)
スキップマッチング	無効 (`false`)	有効
[スタンバイインスタンス]	待機	Ignore (無視)

各設定の説明は次のとおりです。

CloudWatch アラーム (**`AlarmSpecification`**)

CloudWatch アラーム仕様. CloudWatch アラームは、問題を特定し、アラームが `ALARM`状態になった場合にオペレーションを失敗させるために使用できます。詳細については、「自動ロールバックでインスタンスの更新を開始」を参照してください。

自動ロールバック (**`AutoRollback`**)

インスタンスの更新が失敗した場合に、Amazon EC2 Auto Scaling が Auto Scaling グループを以前の設定にロールバックするかどうかを制御します。詳細については、「手動または自動ロールバックを使用して変更を元に戻す」を参照してください。

ベーク時間 (**`BakeTime`**)

インスタンスの更新が完了と見なされるまで、インスタンスの更新の終了時に待機する時間。

チェックポイント (**`CheckpointPercentages`**)

Amazon EC2 Auto Scaling が段階的にインスタンスを置き換えるかどうかを制御します。これは、すべてのインスタンスを置き換える前にインスタンスの検証を実行する必要がある場合に便

利です。詳細については、「インスタンスの更新にチェックポイントを追加する」を参照してください。

チェックポイントの遅延 (**CheckpointDelay**)

チェックポイントの後に続行する前に待機する時間 (秒単位)。詳細については、「インスタンスの更新にチェックポイントを追加する」を参照してください。

インスタンスのウォームアップ (**InstanceWarmup**)

Amazon EC2 Auto Scaling が、新しいインスタンスの初期化が完了すると見なされるまで待機してから、次のインスタンスの置き換えに進む秒単位の時間。Auto Scaling グループのデフォルトのインスタンスウォームアップを既に正しく定義している場合、インスタンスのウォームアップ (デフォルトをオーバーライドする場合を除く) を変更する必要はありません。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

最大正常率 (**MaxHealthyPercentage**)

インスタンスを置き換える際、Auto Scaling グループの希望キャパシティを最大で何パーセントまで増やせるかを示す割合。

最小正常率 (**MinHealthyPercentage**)

Auto Scaling グループの希望キャパシティのうち、オペレーションを続行するために必要な、稼働中で正常な、使用準備が整っているキャパシティの割合。

スケールインで保護されたインスタンス (**ScaleInProtectedInstances**)

スケールインから保護されているインスタンスが見つかった場合の Amazon EC2 Auto Scaling の動作を制御します。これらのインスタンスの詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

Amazon EC2 Auto Scaling には以下のオプションがあります。

- 置き換え (`Refresh`) – スケールインから保護されているインスタンスを置き換えます。
- 無視 (`Ignore`) – スケールインから保護されているインスタンスを無視し、保護されていないインスタンスの置き換えを続行します。
- 待機 (`Wait`) – スケールイン保護を解除するまで 1 時間待機します。そうしない場合、インスタンスの更新が失敗します。

スキップマッチング (**SkipMatching**)

Amazon EC2 Auto Scaling が目的の設定に一致するインスタンスの置き換えをスキップするかどうかを制御します。必要な設定が指定されていない場合、インスタンスの更新が開始される前に

Auto Scaling グループが使用していた同じ起動テンプレートおよびインスタンスタイプを持つインスタンスの置き換えはスキップされます。詳細については、「スキップマッチングでのインスタンスの更新の使用」を参照してください。

スタンバイインスタンス (**StandbyInstances**)

インスタンスが Standby状態で見つかった場合の Amazon EC2 Auto Scaling の動作を制御します。これらのインスタンスの詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

Amazon EC2 Auto Scaling には以下のオプションがあります。

- 終了 (Terminate) – Standby にあるインスタンスを終了します。
- 無視 (Ignore) – Standby 状態にあるインスタンスを無視し、InService 状態にあるインスタンスの置き換えを続行します。
- 待機 (Wait) – インスタンスをサービスに戻すまで 1 時間待機します。そうしない場合、インスタンスの更新が失敗します。

AWS Management Console または を使用してインスタンスの更新を開始する AWS CLI

> **Important**
>
> 進行中のインスタンスの更新をロールバックし、すべての変更を元に戻すことができます。これが実行されるには、インスタンスの更新を開始する前に、Auto Scaling グループがロールバックを使用するための前提条件を満たす必要があります。詳細については、「手動または自動ロールバックを使用して変更を元に戻す」を参照してください。

次の手順は、 AWS Management Console または を使用してインスタンスの更新を開始するのに役立ちます AWS CLI。

インスタンスの更新の開始 (コンソール)

初めてインスタンスの更新を開始する場合は、コンソールにより、利用できる機能とオプションを理解することができます。

コンソールでインスタンスの更新を開始する (基本的な手順)

Auto Scaling グループの混合インスタンスポリシーを事前定義していない場合は、次の手順に従います。混合インスタンスポリシーを事前定義している場合は、コンソールでインスタンスの更新を開始する (混合インスタンスグループ) を参照してインスタンスの更新を開始します。

インスタンスの更新をスタートするには

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   Auto Scaling グループページの下部に分割ペインが開きます。
3. [Instance refresh] (インスタンスの更新) タブの [Active instance refresh] (アクティブインスタンスの更新) で、[Start instance refresh] (インスタンスの更新を開始する) を選択します。
4. [可用性の設定] では、次の操作を行います。

   a. インスタンス置換方法：

      - Auto Scaling グループにインスタンスメンテナンスポリシーを設定していない場合、インスタンス置換方法のデフォルト設定は終了と起動です。これは、インスタンス更新の従来のデフォルト動作です。
      - Auto Scaling グループにインスタンスメンテナンスポリシーを設定すると、インスタンス置換方法のデフォルト値が提供されます。インスタンスメンテナンスポリシーを上書きするには、[オーバーライド] を選択します。オーバーライドは、現在のインスタンスの更新にのみ適用されます。次回インスタンスの更新を開始すると、これらの値はインスタンスメンテナンスポリシーのデフォルト値にリセットされます。

      次の手順では、インスタンス置換方法を更新する方法について説明します。

      i. 次のいずれかのインスタンス置換方法を選択します。

         - 終了前の起動: 既存のインスタンスを終了する前に、新しいインスタンスをプロビジョニングする必要があります。これは、コスト削減よりも可用性を重視するアプリケーションに適しています。
         - 終了と起動: 新しいインスタンスは、既存のインスタンスが終了すると同時にプロビジョニングされます。これは、可用性よりもコスト削減を優先するアプリケーション

に適しています。これは、現在利用可能なキャパシティを超えて処理能力を増やす必要のないアプリケーションにも適しています。

- カスタム動作 : このオプションを使用すると、インスタンスを置き換えるときに使用可能なキャパシティの最小範囲と最大範囲をカスタム設定できます。これにより、コストと可用性の適切なバランスを実現できます。

ii. [正常率を設定]には、次のフィールドの 1 つまたは両方に値を入力します。有効化フィールドは、インスタンス置換方法で選択したオプションによって異なります。

- 最小 : インスタンスの更新を進めるために必要な最小正常率を設定します。
- 最大 : インスタンスの更新中に可能な最大正常率を設定します。

iii. 現在のグループサイズに基づいて置換中の推定一時容量を表示セクションを展開し、最小と最大の値がどのようにグループに適用されるかを確認します。使用される正確な値は、希望するキャパシティ値によって異なります。これは、グループがスケールすると変化します。

iv. 無効な置換サイズに対するフォールバック動作の設定セクションを展開し、可用性を優先するために最大正常率に違反するか、最小正常率に違反するかを選択します。

デフォルトの最小正常率オプションを維持することは、非常に小さなグループにはお勧めしません。Auto Scaling グループにインスタンスが 1 つしかない場合、インスタンスの更新を開始すると停止する場合があります。

このステップでは、インスタンスメンテナンスポリシーがまだない Auto Scaling グループを使用している場合のフォールバック動作を設定します。グループにインスタンス メンテナンス ポリシーがある場合、このオプションは使用できず、表示されません。このオプションは、終了と起動の置き換え方法でのみ使用できます。他の置き換え方法は、可用性を優先するために最大正常率に違反します。

b. [インスタンスのウォームアップ] には、新しいインスタンスの状態が `InService` に変更されてから初期化が完了するまでの秒数を入力します。Amazon EC2 Auto Scaling は、次のインスタンスの置き換えに進む前に、この時間待機します。

ウォームアップ中、新しく起動されたインスタンスは Auto Scaling グループの (`CPUUtilization`、`NetworkIn`、`NetworkOut` など) 集計インスタンスのメトリクスにも計上されません。Auto Scaling グループにスケーリング ポリシーを追加した場合、スケーリングアクティビティは並行して実行されます。インスタンス更新のウォームアップ期間に長い間隔を設定する場合、新しく起動されたインスタンスがメトリクスに表示されるまでに時間がかかります。したがって、適切なウォームアップ期間により、Amazon EC2 Auto Scaling が古いメトリクスデータでスケーリングされるのを防ぐことができます。

Auto Scaling グループにデフォルトのインスタンスのウォームアップを既に正しく定義している場合、インスタンスのウォームアップを変更する必要はありません。ただし、デフォルトを上書きする場合は、このオプションの値を設定できます。デフォルトのインスタンスのウォームアップにおける設定の詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

5. [設定を更新] で、以下を実行します。

   a. (オプション) [Checkpoints] (チェックポイント) で [Enable checkpoints] (チェックポイントを有効にする) を選択し、インスタンスの更新に増分または段階的なアプローチを使用するインスタンスを置換します。これにより、置換セット間の検証にさらに時間がかかります。チェックポイントの有効化を選択しない場合、インスタンスはほぼ連続した 1 回のオペレーションで置換されます。

      チェックポイントを有効にする場合は、追加ステップ チェックポイントを有効にする (コンソール) を参照してください。

   b. (オプション) ベイク時間には、インスタンスの更新が完了と見なされるまで、インスタンスの更新の終了時に待機する時間を指定します。

   c. スキップマッチングを有効または無効にする:

      - 起動テンプレートに既に一致しているインスタンスの置き換えをスキップするには、[スキップマッチングを有効にする] のチェックボックスをオンのままにします。
      - このチェックボックスをオフにしてスキップマッチングを無効にすると、すべてのインスタンスを置換することができます。

      スキップマッチングを有効にすると、既存の起動テンプレートを使用する代わりに、新しい起動テンプレートまたは起動テンプレートの新しいバージョンを設定できます。これは、[インスタンスの更新を開始] ページの [必要な設定] セクションで行います。

      > **Note**
      >
      > スキップマッチング機能を使用して、現在起動設定を使用している Auto Scaling グループを更新するには、希望する設定で起動テンプレートを選択する必要があります。起動設定でのスキップマッチングはサポートされていません。

d. [スタンバイインスタンス] の場合、[無視]、[終了]、[待機] のいずれかを選択します。これにより、インスタンスが Standby 状態で見つかった場合の処理が決まります。詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

   [待機] を選択する場合、これらのインスタンスをサービスに戻すために追加のステップを実行する必要があります。そうしない場合、インスタンスの更新がすべての InService インスタンスを置き換えて 1 時間待機します。次に、Standby インスタンスが残っていると、インスタンスの更新は失敗します。この状況を防ぐには、代わりにこれらのインスタンスに対して [無視] または [終了] を選択してください。

e. [スケールインで保護されたインスタンス] の場合、[無視]、[置換]、[待機] のいずれかを選択します。これにより、スケールインで保護されたインスタンスが見つかった場合の処理が決まります。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

   [待機] を選択する場合、これらのインスタンスからスケールイン保護を解除するために追加のステップを実行する必要があります。そうしない場合、インスタンスの更新は保護されていないすべてのインスタンスを置き換えて 1 時間待機します。次に、スケールインで保護されたインスタンスが残っている場合、インスタンスの更新が失敗します。この状況を防ぐには、代わりにこれらのインスタンスに対して [無視] または [置き換え] を選択してください。

6. (オプション) CloudWatch アラームで、Enable CloudWatch アラームを選択し、1 つ以上のアラームを選択します。 CloudWatch アラームを使用して問題を特定し、アラームが ALARM状態になった場合にオペレーションを失敗させることができます。詳細については、「自動ロールバックでインスタンスの更新を開始」を参照してください。

7. (オプション) [必要な設定] セクションを拡張し、Auto Scaling グループに実行する更新を指定します。

   このステップでは、コンソールインターフェイスで選択を行う代わりに、JSON または YAML 構文を使用してパラメータ値を編集できます。このためには、[Use console interface] (コンソールインターフェイスを使用する) の代わりに[Use code editor] (コードエディタを使用する) を選択します。以下の手順では、コンソールインターフェイスを使用して選択する方法について説明します。

   a. 起動テンプレートを更新する場合:

      - Auto Scaling グループの新しい起動テンプレートまたは新しい起動テンプレートバージョンを作成していない場合、このチェックボックスをオンにしないでください。

- 新しい起動テンプレートまたは新しい起動テンプレートバージョンを作成した場合は、このチェックボックスをオンにします。このオプションを選択すると、Amazon EC2 Auto Scaling に現在の起動テンプレートと現在の起動テンプレートバージョンが表示されます。他の利用可能なバージョンもすべて確認できます。起動テンプレートを選択し、バージョンを選択します。

  バージョンを選択すると、バージョン情報が表示されます。これは、インスタンスの更新の一部としてインスタンスを置換するときに使用される起動テンプレートのバージョンです。インスタンスの更新に成功すると、グループのスケール時など、新しいインスタンスが起動するたびに起動テンプレートのこのバージョンが使用されます。

b. インスタンスタイプと購入オプションのセットを選択して、起動テンプレートのインスタンスタイプを上書きする場合:

- 起動テンプレートで指定したインスタンスタイプおよび購入オプションを使用する場合、このチェックボックスをオンにしないでください。
- 起動テンプレートのインスタンスタイプをオーバーライドまたはスポットインスタンスを実行する場合、このチェックボックスをオンにします。各インスタンスタイプを手動で追加するか、プライマリインスタンスタイプおよび一致する追加のインスタンスタイプを取得する推奨オプションを選択できます。スポットインスタンスを起動する予定がある場合は、いくつかの異なるインスタンスタイプを追加することをお勧めします。これにより、選択したアベイラビリティーゾーンに十分なインスタンス容量がない場合に、Amazon EC2 Auto Scaling は別のインスタンスタイプを起動できます。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。

  > **Warning**
  >
  > スポットインスタンスの中断を処理できないアプリケーションにはスポットインスタンスを使用しないでください。Amazon EC2 スポットサービスが容量を再利用する必要がある場合、中断が発生する可能性があります。

このチェックボックスをオンにした場合は、起動テンプレートがまだスポットインスタンスをリクエストしていないことを確認してください。複数のインスタンスタイプを使用し、スポットインスタンスとオンデマンドインスタンスを起動する Auto Scaling グループを作成するため、スポットインスタンスをリクエストする起動テンプレートを使用することはできません。

> **Note**
>
> 現在起動設定を使用している Auto Scaling グループでこれらのオプションを設定するには、起動テンプレートの更新で起動テンプレートを選択する必要があります。起動設定のインスタンスタイプの上書きはサポートされていません。

8. (オプション) [ロールバック設定] で [自動ロールバックを有効にする] を選択すると、インスタンスの更新が失敗した場合に自動的にロールバックされます。

   この設定は、Auto Scaling グループがロールバックを使用するための前提条件を満たしている場合にのみ有効にできます。

   詳細については、「手動または自動ロールバックを使用して変更を元に戻す」を参照してください。
9. すべての選択内容を見直し、正しく設定されていることを確認します。

   現在の設定と提案された変更の違いが、想定外または望ましくない形でアプリケーションに影響を及ぼさないよう、この時点で確認することをお勧めします。インスタンスタイプが起動テンプレートと互換性があることを確認するには、「インスタンスタイプの互換性」を参照してください。
10. インスタンス更新の選択内容が正しい場合は、[インスタンスの更新の開始] を選択します。

コンソールでインスタンスの更新を開始する (混合インスタンスグループ)

混合インスタンスポリシーで Auto Scaling グループを作成した場合は、次の手順に従います。グループに混合インスタンスポリシーをまだ定義していない場合は、コンソールでインスタンスの更新を開始する (基本的な手順) を参照してインスタンスの更新を開始します。

インスタンスの更新をスタートするには

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   Auto Scaling グループページの下部に分割ペインが開きます。
3. [Instance refresh] (インスタンスの更新) タブの [Active instance refresh] (アクティブインスタンスの更新) で、[Start instance refresh] (インスタンスの更新を開始する) を選択します。

4. [可用性の設定] では、次の操作を行います。

   a. インスタンス置換方法：

      - Auto Scaling グループにインスタンスメンテナンスポリシーを設定していない場合、インスタンス置換方法のデフォルト設定は終了と起動です。これは、インスタンス更新の従来のデフォルト動作です。
      - Auto Scaling グループにインスタンスメンテナンスポリシーを設定すると、インスタンス置換方法のデフォルト値が提供されます。インスタンスメンテナンスポリシーを上書きするには、[オーバーライド] を選択します。オーバーライドは、現在のインスタンスの更新にのみ適用されます。次回インスタンスの更新を開始すると、これらの値はインスタンスメンテナンスポリシーのデフォルト値にリセットされます。

      次の手順では、インスタンス置換方法を更新する方法について説明します。

      i. 次のいずれかのインスタンス置換方法を選択します。

         - 終了前の起動: 既存のインスタンスを終了する前に、新しいインスタンスをプロビジョニングする必要があります。これは、コスト削減よりも可用性を重視するアプリケーションに適しています。
         - 終了と起動: 新しいインスタンスは、既存のインスタンスが終了すると同時にプロビジョニングされます。これは、可用性よりもコスト削減を優先するアプリケーションに適しています。これは、現在利用可能なキャパシティを超えて処理能力を増やす必要のないアプリケーションにも適しています。
         - カスタム動作 : このオプションを使用すると、インスタンスを置き換えるときに使用可能なキャパシティの最小範囲と最大範囲をカスタム設定できます。これにより、コストと可用性の適切なバランスを実現できます。

      ii. [正常率を設定]には、次のフィールドの 1 つまたは両方に値を入力します。有効化フィールドは、インスタンス置換方法で選択したオプションによって異なります。

         - 最小 : インスタンスの更新を進めるために必要な最小正常率を設定します。
         - 最大 : インスタンスの更新中に可能な最大正常率を設定します。

      iii. 現在のグループサイズに基づいて置換中の推定一時容量を表示セクションを展開し、最小と最大の値がどのようにグループに適用されるかを確認します。使用される正確な値は、希望するキャパシティ値によって異なります。これは、グループがスケールすると変化します。

iv. 無効な置換サイズに対するフォールバック動作の設定セクションを展開し、可用性を優先するために最大正常率に違反するか、最小正常率に違反するかを選択します。

デフォルトの最小正常率オプションを維持することは、非常に小さなグループにはお勧めしません。Auto Scaling グループにインスタンスが 1 つしかない場合、インスタンスの更新を開始すると停止する場合があります。

このステップでは、インスタンスメンテナンスポリシーがまだない Auto Scaling グループを使用している場合のフォールバック動作を設定します。グループにインスタンス メンテナンス ポリシーがある場合、このオプションは使用できず、表示されません。このオプションは、終了と起動の置き換え方法でのみ使用できます。他の置き換え方法は、可用性を優先するために最大正常率に違反します。

b. [インスタンスのウォームアップ] には、新しいインスタンスの状態が `InService` に変更されてから初期化が完了するまでの秒数を入力します。Amazon EC2 Auto Scaling は、次のインスタンスの置き換えに進む前に、この時間待機します。

ウォームアップ中、新しく起動されたインスタンスは Auto Scaling グループの (`CPUUtilization`、`NetworkIn`、`NetworkOut` など) 集計インスタンスのメトリクスにも計上されません。Auto Scaling グループにスケーリング ポリシーを追加した場合、スケーリングアクティビティは並行して実行されます。インスタンス更新のウォームアップ期間に長い間隔を設定する場合、新しく起動されたインスタンスがメトリクスに表示されるまでに時間がかかります。したがって、適切なウォームアップ期間により、Amazon EC2 Auto Scaling が古いメトリクスデータでスケーリングされるのを防ぐことができます。

Auto Scaling グループにデフォルトのインスタンスのウォームアップを既に正しく定義している場合、インスタンスのウォームアップを変更する必要はありません。ただし、デフォルトを上書きする場合は、このオプションの値を設定できます。デフォルトのインスタンスのウォームアップにおける設定の詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

5. [設定を更新] で、以下を実行します。

a. (オプション) [Checkpoints] (チェックポイント) で [Enable checkpoints] (チェックポイントを有効にする) を選択し、インスタンスの更新に増分または段階的なアプローチを使用するインスタンスを置換します。これにより、置換セット間の検証にさらに時間がかかります。チェックポイントの有効化を選択しない場合、インスタンスはほぼ連続した 1 回のオペレーションで置換されます。

チェックポイントを有効にする場合は、追加ステップ チェックポイントを有効にする (コンソール) を参照してください。

b. スキップマッチングを有効または無効にする:

- 起動テンプレートと既に一致しているインスタンスの置き換えおよびすべてのインスタンスタイプのオーバーライドをスキップするには、[スキップマッチングを有効にする] チェックボックスをオンのままにします。
- このチェックボックスをオフにしてスキップマッチングを無効にすると、すべてのインスタンスを置換することができます。

スキップマッチングを有効にすると、既存の起動テンプレートを使用する代わりに、新しい起動テンプレートまたは起動テンプレートの新しいバージョンを設定できます。これは、[インスタンスの更新を開始] ページの [必要な設定] セクションで行います。[Desired configuration] (希望する設定) でインスタンスタイプの上書きを更新することもできます。

c. [スタンバイインスタンス] の場合、[無視]、[終了]、[待機] のいずれかを選択します。これにより、インスタンスが Standby 状態で見つかった場合の処理が決まります。詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

[待機] を選択する場合、これらのインスタンスをサービスに戻すために追加のステップを実行する必要があります。そうしない場合、インスタンスの更新はすべての InService インスタンスを置き換えて 1 時間待機します。次に、Standby インスタンスが残っていると、インスタンスの更新は失敗します。この状況を防ぐには、代わりにこれらのインスタンスに対して [無視] または [終了] を選択してください。

d. [スケールインで保護されたインスタンス] の場合、[無視]、[置換]、[待機] のいずれかを選択します。これにより、スケールインで保護されたインスタンスが見つかった場合の処理が決まります。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

[待機] を選択する場合、これらのインスタンスからスケールイン保護を解除するために追加のステップを実行する必要があります。そうしない場合、インスタンスの更新は保護されていないすべてのインスタンスを置き換えて 1 時間待機します。次に、スケールインで保護されたインスタンスが残っている場合、インスタンスの更新が失敗します。この状況を防ぐには、代わりにこれらのインスタンスに対して [無視] または [置き換え] を選択してください。

6. （オプション) CloudWatch アラームで、Enable CloudWatch アラームを選択し、1 つ以上のアラームを選択します。 CloudWatch アラームを使用して問題を特定し、アラームが ALARM状態になった場合にオペレーションを失敗させることができます。詳細については、「自動ロールバックでインスタンスの更新を開始」を参照してください。
7. [Desired configuration] (希望する設定) セクションで以下を実行します。

   このステップでは、コンソールインターフェイスで選択を行う代わりに、JSON または YAML 構文を使用してパラメータ値を編集できます。このためには、[Use console interface] (コンソールインターフェイスを使用する) の代わりに[Use code editor] (コードエディタを使用する) を選択します。以下の手順では、コンソールインターフェイスを使用して選択する方法について説明します。

   a. 起動テンプレートを更新する場合:

      - Auto Scaling グループの新しい起動テンプレートまたは新しい起動テンプレートバージョンを作成していない場合、このチェックボックスをオンにしないでください。
      - 新しい起動テンプレートまたは新しい起動テンプレートバージョンを作成した場合は、このチェックボックスをオンにします。このオプションを選択すると、Amazon EC2 Auto Scaling に現在の起動テンプレートと現在の起動テンプレートバージョンが表示されます。他の利用可能なバージョンもすべて確認できます。起動テンプレートを選択し、バージョンを選択します。

        バージョンを選択すると、バージョン情報が表示されます。これは、インスタンスの更新の一部としてインスタンスを置換するときに使用される起動テンプレートのバージョンです。インスタンスの更新に成功すると、グループのスケール時など、新しいインスタンスが起動するたびに起動テンプレートのこのバージョンが使用されます。

   b. これらの設定を使用して、起動テンプレートで定義されているインスタンスタイプと購入オプションを上書きする場合:

      デフォルトでは、このチェックボックスはオンになっています。Amazon EC2 Auto Scaling は、Auto Scaling グループの混合インスタンスポリシーで現在設定されている値を各パラメータに入力します。変更するパラメータの値のみを更新します。これらの設定に関するガイダンスについては、複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ を参照してください。

> **Warning**
>
> このチェックボックスはオフにしないことをお勧めします。混合インスタンスポリシーの使用を停止する場合にのみオフにします。インスタンスの更新が成功すると、Amazon EC2 Auto Scaling はグループを更新して、必要な設定と一致させます。混合インスタンスポリシーが含まれなくなった場合、Amazon EC2 Auto Scaling は現在実行中のスポットインスタンスを徐々に終了し、オンデマンドインスタンスに置き換えます。または、起動テンプレートがスポットインスタンスをリクエストした場合、Amazon EC2 Auto Scaling は現在実行中のオンデマンドインスタンスを徐々に終了し、スポットインスタンスに置き換えます。

8. (オプション) [ロールバック設定] で [自動ロールバックを有効にする] を選択すると、何らかの理由でインスタンスの更新が失敗した場合に自動的にロールバックされます。

   この設定は、Auto Scaling グループがロールバックを使用するための前提条件を満たしている場合にのみ有効にできます。

   詳細については、「手動または自動ロールバックを使用して変更を元に戻す」を参照してください。
9. すべての選択内容を見直し、正しく設定されていることを確認します。

   現在の設定と提案された変更の違いが、想定外または望ましくない形でアプリケーションに影響を及ぼさないよう、この時点で確認することをお勧めします。インスタンスタイプが起動テンプレートと互換性があることを確認するには、「インスタンスタイプの互換性」を参照してください。

   インスタンス更新の選択内容が正しい場合は、[インスタンスの更新の開始] を選択します。

インスタンスの更新 (AWS CLI) の開始

インスタンスの更新をスタートするには

次の start-instance-refresh コマンドを使用して、 からインスタンスの更新を開始します AWS CLI。JSON 設定ファイルで変更する設定を指定できます。設定ファイルを参照するとき、次の例に示すように、ファイルパスおよび名前を指定します。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 50,
      "AutoRollback": true,
      "ScaleInProtectedInstances": Ignore,
      "StandbyInstances": Terminate
    }
}
```

設定が指定されない場合、デフォルト値が使用されます。詳細については、「インスタンスの更新のデフォルト値について説明する」を参照してください。

出力例:

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

AWS Management Console または を使用してインスタンスの更新をモニタリングする AWS CLI

AWS Management Console または を使用して、進行中のインスタンスの更新をモニタリングしたり、過去 6 週間の過去のインスタンスの更新のステータスを検索したりできます AWS CLI。

インスタンスの更新ステータスをモニタリングおよびチェックする

インスタンスの更新ステータスをモニタリングおよびチェックするには、以下のいずれかの方法を使用します。

Console

> **Tip**
>
> この手順では、名前付き列がすでに表示されている必要があります。非表示の列を表示、あるいは表示される行の数を変更するには、セクションの右上隅にある歯車アイコンを選択し、設定モーダルを開きます。必要に応じて設定を更新し、[確認] を選択します。

インスタンスの更新のステータスをモニタリングおよび確認する (コンソール)

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部に分割されたペインが開きます。
3. [Instance refresh history] (インスタンスの更新履歴) の [Instance refresh] (インスタンスの更新) タブで、[Status] (ステータス) 列を確認し、リクエストのステータスを決定できます。初期化中は `Pending` のステータスになります。その後、ステータスはすぐに `InProgress` に変わります。すべてのインスタンスが更新されると、ステータスが `Successful` に変わります。
4. グループのスケーリングアクティビティを表示することで、進行中のアクティビティの成功または失敗をさらにモニタリングできます。[Activity (アクティビティ)] タブの [Activity history (アクティビティ履歴)] では、インスタンスの更新が開始されると、インスタンスの終了時、インスタンスの起動時に、それぞれ別のエントリが表示されます。多数のスケーリングアクティビティが存在する場合、アクティビティ履歴の上部にある [>] アイコンを選択すると、さらに多くのアクティビティを表示できます。アクティビティの失敗の原因となりうる問題のトラブルシューティングについては、「Amazon EC2 Auto Scaling の問題のトラブルシューティング」を参照してください。
5. (オプション) [インスタンス管理] タブの[インスタンス] で、必要に応じて特定のインスタンスの進行状況を確認できます。

AWS CLI

インスタンスの更新ステータスを確認するには (AWS CLI)

次の describe-instance-refreshes コマンドを使用します。

```
aws autoscaling describe-instance-refreshes --auto-scaling-group-name my-asg
```

以下は出力例です。

インスタンスの更新は、開始時刻順に並べられます。まだ進行中のインスタンスの更新については、最初に説明します。

```
{
  "InstanceRefreshes":[
```

```
    {
      "InstanceRefreshId":"08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",
      "AutoScalingGroupName":"my-asg",
      "Status":"InProgress",
      "StatusReason":"Waiting for instances to warm up before continuing. For
 example: i-0645704820a8e83ff is warming up.",
      "StartTime":"2023-11-24T16:46:52+00:00",
      "PercentageComplete":50,
      "InstancesToUpdate":0,
      "Preferences":{
        "MaxHealthyPercentage":120,
        "MinHealthyPercentage":90,
        "InstanceWarmup":60,
        "SkipMatching":false,
        "AutoRollback":true,
        "ScaleInProtectedInstances":"Ignore",
        "StandbyInstances":"Ignore"
      }
    },
    {
      "InstanceRefreshId":"0e151305-1e57-4a32-a256-1fd14157c5ec",
      "AutoScalingGroupName":"my-asg",
      "Status":"Successful",
      "StartTime":"2023-11-22T13:53:37+00:00",
      "EndTime":"2023-11-22T13:59:45+00:00",
      "PercentageComplete":100,
      "InstancesToUpdate":0,
      "Preferences":{
        "MaxHealthyPercentage":120,
        "MinHealthyPercentage":90,
        "InstanceWarmup":60,
        "SkipMatching":false,
        "AutoRollback":true,
        "ScaleInProtectedInstances":"Ignore",
        "StandbyInstances":"Ignore"
      }
    }
  ]
}
```


グループのスケーリングアクティビティを表示することで、進行中のアクティビティの成功または失敗をさらにモニタリングできます。スケーリングアクティビティは、インスタンスの更新に関する問題のトラブルシューティングにおいて、より詳細な情報を得るために役立ちます。詳細

については、「Amazon EC2 Auto Scaling の問題のトラブルシューティング」を参照してください。

インスタンスの更新ステータス

インスタンスの更新を開始するとき、[ペンディング] ステータスになります。成功、失敗、キャンセル、InProgress、またはRollbackSuccessfulに達するまで、保留中からRollbackFailedに渡されます。

インスタンスの更新には、次のステータスがあります。

ステータス	説明
[保留中]	リクエストは作成されましたが、インスタンスの更新が開始されていません。
InProgress	インスタンスの更新が進行中です。
[成功]	インスタンスの更新が正常に完了しました。
Failed (失敗)	インスタンスの更新を完了できませんでした。ステータスの理由とスケーリングアクティビティを使用してトラブルシューティングを行うことができます。
[キャンセル中]	進行中のインスタンスの更新をキャンセルしています。
[キャンセル済]	インスタンスの更新はキャンセルされました。
RollbackInProgress	インスタンスの更新がロールバックされています。
RollbackFailed	ロールバックを完了できませんでした。ステータスの理由とスケーリングアクティビティを使用してトラブルシューティングを行うことができます。
RollbackSuccessful	ロールバックが正常に完了しました。
ベーキング	インスタンスの更新が完了した後、指定されたベイク時間を待機します。

AWS Management Console または を使用してインスタンスの更新をキャンセルする AWS CLI

まだ進行中のインスタンスの更新はキャンセルできますが、完了した後にキャンセルすることはできません。

インスタンスの更新をキャンセルしても、既に置き換えられたインスタンスはロールバックされません。インスタンスの変更をロールバックするには、代わりにロールバックを実行してください。詳細については、「手動または自動ロールバックを使用して変更を元に戻す」を参照してください。

トピック

- インスタンスの更新のキャンセル (コンソール)
- インスタンスの更新のキャンセル (AWS CLI)

インスタンスの更新のキャンセル (コンソール)

インスタンスの更新をキャンセルするには

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [アクティブインスタンスの更新] の [インスタンスの更新] タブで、[アクション] および [キャンセル] を選択します。
4. 確認を求められたら、[確認] を選択します。

インスタンス更新のステータスは [キャンセル中] に設定されます。キャンセルが完了した後、インスタンスの更新のステータスは [キャンセル済み] に設定されます。

インスタンスの更新のキャンセル (AWS CLI)

インスタンスの更新をキャンセルするには

から cancel-instance-refresh コマンドを使用し AWS CLI 、Auto Scaling グループ名を指定します。

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

出力例:

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

手動または自動ロールバックを使用して変更を元に戻す

まだ進行中のインスタンスの更新をロールバックできます。終了後はロールバックできません。ただし、新しいインスタンスの更新を開始することによって Auto Scaling グループを再度更新できます。

ロールバックすると、Amazon EC2 Auto Scaling はこれまでにデプロイされたインスタンスを置き換えます。新しいインスタンスは、インスタンスの更新を開始する前に Auto Scaling グループに最後に保存した設定と一致します。

Amazon EC2 Auto Scaling では、次の方法でロールバックできます。

- 手動ロールバック: ロールバックを手動で開始して、デプロイされた内容をロールバックポイントまで戻します。
- 自動ロールバック: Amazon EC2 Auto Scaling は、何らかの理由でインスタンスの更新が失敗した場合、または指定したいずれかの CloudWatch アラームが ALARM状態になった場合に、デプロイされた内容を自動的に元に戻します。

内容

考慮事項

以下の考慮事項は、ロールバックを使用する場合に適用されます。

- ロールバックオプションは、インスタンスの更新の開始の一部として必要な設定を指定した場合にのみ使用できます。
- 以前のバージョンの起動テンプレートにロールバックできるのは、バージョンが特定の番号の付いたバージョンである場合のみです。Auto Scaling グループが $Latest または $Default の起動テンプレートバージョンを使用するように設定されている場合、ロールバックオプションは使用できません。

- また、 AWS Systems Manager Parameter Store から AMI エイリアスを使用するように設定された起動テンプレートにロールバックすることもできません。
- Auto Scaling グループに最後に保存した設定は、安定した状態である必要があります。安定していない状態でも、ロールバックのワークフローは実行されますが、最終的には失敗します。問題を解決するまで、Auto Scaling グループは失敗状態になり、インスタンスを正常に起動できなくなる可能性があります。これはサービスまたはアプリケーションの可用性に影響する可能性があります。

ロールバックを手動で開始する

Console

インスタンスの更新のロールバックを手動で開始するには (コンソール)

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [インスタンスの更新] タブの [アクティブインスタンスの更新] で、[アクション] および [ロールバック開始] を選択します。
4. 確認を求められたら、[確認] を選択します。

AWS CLI

インスタンスの更新のロールバックを手動で開始するには (AWS CLI)

から AWS CLI rollback-instance-refresh コマンドを使用し、Auto Scaling グループ名を指定します。

```
aws autoscaling rollback-instance-refresh --auto-scaling-group-name my-asg
```

出力例:

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

> **Tip**
>
> このコマンドでエラーが発生した場合は、 を AWS CLI ローカルで最新バージョンに更新していることを確認してください。

自動ロールバックでインスタンスの更新を開始

自動ロールバック機能を使用すると、エラーが発生したときや、指定された Amazon CloudWatch アラームが ALARM状態になったときなど、インスタンスの更新が失敗したときに自動的にロールバックできます。

自動ロールバックを有効にしていて、インスタンスの置き換え中にエラーが発生した場合、インスタンスの更新は失敗してロールバックされるまで 1 時間かけてすべての置換を完了しようとします。これらのエラーは通常、EC2 起動の失敗、ヘルスチェックの設定ミス、 Standby状態またはスケールインから保護されているインスタンスの無視や終了の許可などが原因で発生します。

CloudWatch アラームの指定はオプションです。アラームを指定するには、まずアラームを作成する必要があります。メトリクスアラームと複合アラームを作成できます。アラームの作成については、「Amazon CloudWatch ユーザーガイド」を参照してください。Elastic Load Balancing メトリクスを例にとると、Application Load Balancer を使用する場合は HTTPCode_ELB_5XX_Count メトリクスと HTTPCode_ELB_4XX_Count メトリクスを使用できます。

考慮事項

- CloudWatch アラームを指定しても自動ロールバックを有効にせず、アラームの状態が になるとALARM、インスタンスの更新はロールバックなしで失敗します。
- インスタンスの更新を開始するときに、最大 10 個のアラームを選択できます。
- CloudWatch アラームを選択する場合、アラームは互換性のある状態である必要があります。アラームの状態が INSUFFICIENT_DATA または ALARM の場合、インスタンスの更新を開始しようとするとエラーが発生します。
- Amazon EC2 Auto Scaling が使用するアラームを作成する場合、アラームには欠落しているデータポイントの処理方法を含める必要があります。メトリクスのデータポイントが頻繁に欠落する仕様の場合は、これらの期間中、アラームの状態が INSUFFICIENT_DATA になります。この場合、Amazon EC2 Auto Scaling は新しいデータポイントが見つかるまでインスタンスを置き換えることができません。アラームに以前の ALARM または OK 状態を強制的に維持するには、代わりに欠落データを無視することを選択できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「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。

Console

自動ロールバックを使用してインスタンスの更新を開始するには (コンソール)

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。
3. [Instance refresh] (インスタンスの更新) タブの [Active instance refresh] (アクティブインスタンスの更新) で、[Start instance refresh] (インスタンスの更新を開始する) を選択します。
4. インスタンスの更新の開始 (コンソール) の手順に従い、必要に応じてインスタンスの更新設定を行います。
5. (オプション) 更新設定の CloudWatch アラーム で CloudWatch アラームを有効にする を選択し、1 つ以上のアラームを選択して問題を特定し、アラームが ALARM状態になった場合は操作を失敗させます。
6. [ロールバックの設定] で、[自動ロールバックを有効にする] を選択して、失敗したインスタンスの更新を、インスタンスの更新を開始する前に最後に Auto Scaling グループに保存した設定に自動的にロールバックします。
7. 選択内容を確認して、[インスタンスの更新を開始する]を選択します。

AWS CLI

自動ロールバック (AWS CLI) でインスタンスの更新を開始するには

start-instance-refresh コマンドを使用して、 の true AutoRollbackオプションに を指定しますPreferences。

次の例は、何かが失敗した場合に自動的にロールバックするインスタンスの更新を開始する方法を示しています。を置き換える ## 独自の パラメータ値。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
```

```
            "LaunchTemplateName": "my-launch-template",
            "Version": "1"
        }
    },
    "Preferences": {
      "AutoRollback": true
    }
}
```

または、インスタンスの更新が失敗したとき、または指定された CloudWatch アラームが ALARM状態にあるときに自動的にロールバックするには、次の例のように、 で AlarmSpecificationオプションを指定Preferencesし、アラーム名を指定します。を置き換える *##* 独自の パラメータ値。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1"
        }
    },
    "Preferences": {
      "AutoRollback": true,
      "AlarmSpecification": { "Alarms": [ "my-alarm" ] }
    }
}
```

成功すると、コマンドは以下のような出力を返します。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

> **Tip**
>
> このコマンドでエラーが発生した場合は、 を AWS CLI ローカルで最新バージョンに更新していることを確認してください。

スキップマッチングでのインスタンスの更新の使用

スキップマッチングは、Amazon EC2 Auto Scaling に、すでに最新の更新があるインスタンスを無視するように指示します。これにより、必要以上のインスタンスを置き換えることはありません。これは Auto Scaling グループが特定のバージョンの起動テンプレートを使用していることを確認し、異なるバージョンを使用するインスタンスのみを置き換えたいときに役立ちます。

スキップマッチングを使用する際は、以下を考慮してください。

- スキップマッチングと必要な設定の両方を使用してインスタンスの更新を開始すると、Amazon EC2 Auto Scaling は目的の設定に一致するインスタンスがあるかどうかを確認します。次に、必要な設定に一致しないインスタンスのみを置き換えます。インスタンスの更新が成功すると、Amazon EC2 Auto Scaling はグループを更新して必要な設定を反映します。
- スキップマッチングでインスタンスの更新を開始したが、必要な設定を指定しない場合、Amazon EC2 Auto Scaling はAuto Scaling グループに最後に保存した設定と一致するインスタンスがあるかどうかを確認します。次に、最後に設定した構成と一致しないインスタンスのみが置き換えられます。
- スキップマッチングは、新しい起動テンプレート、起動テンプレートの新しいバージョン、一連のインスタンスタイプにで使用できます。スキップマッチングを有効にしますが、これらのうちどれも変更されていない場合は、インスタンスの更新はインスタンスを置き換えることなく直ちに成功します。必要な設定に他の変更 (スポット配分戦略の変更など) を行った場合、Amazon EC2 Auto Scaling はインスタンスの更新が成功するのを待ちます。次に、新しい必要な設定を反映するように Auto Scaling グループ設定を更新します。
- 新しい起動設定では、スキップマッチングを使用することはできません。
- インスタンスの更新を開始し、必要な設定を指定すると、Amazon EC2 Auto Scaling はすべてのインスタンスが目的の設定を使用するようにします。したがって、起動テンプレートに必要なバージョンとして `$Default` または `$Latest` を指定し、インスタンスの更新中に新しいバージョンの起動テンプレートを作成すると、既に置き換えられているインスタンスは再び置き換えられます。
- スキップマッチングでは、起動テンプレートのユーザーデータスクリプトが更新されたコードを取得して新しいインスタンスにインストールするかどうかを判断できません。その結果、マッチングをスキップすると、古いコードがインストールされているインスタンスの置換がスキップされる可能性があります。この場合、起動テンプレートのバージョンを更新しなくても、スキップマッチングをオフにして、すべてのインスタンスが最新のコードを確実に受信できるようにする必要があります。

このセクションでは、スキップマッチングを有効にしてインスタンスの更新を開始する AWS CLI 手順について説明します。コンソールを使用する手順については、「インスタンスの更新の開始 (コンソール)」を参照してください。

スキップマッチング (基本手順)

このセクションのステップに従って、 AWS CLI を使用して以下を実行します。

- インスタンスに適用する起動テンプレートを作成します。
- インスタンスの更新を開始して起動テンプレートを Auto Scaling グループに適用します。スキップマッチングを有効にしない場合、すべてのインスタンスが置き換えられます。これは、インスタンスのプロビジョニングに使用した起動テンプレートが、必要な設定に指定した起動テンプレートと同じであっても当てはまります。

新しい起動テンプレートでスキップマッチングを使用するには

1. create-launch-template コマンドを使用して、Auto Scaling グループの新しい起動テンプレートを作成します。Auto Scaling グループ用に作成されたインスタンスの詳細を定義する `--launch-template-data`オプションと JSON 入力を含めます。

   例えば、次のコマンドを使用して、AMI ID で基本的な起動テンプレートを作成します。*ami-0123456789abcdef0* および*t2.micro*インスタンスタイプ。

   ```
   aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
    --version-description version1 \
     --launch-template-data
    '{"ImageId":"ami-0123456789abcdef0","InstanceType":"t2.micro"}'
   ```

   成功すると、コマンドは以下のような出力を返します。

   ```
   {
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b729example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2023-01-30T18:16:06.000Z",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
     }
   }
   ```

詳細については、「を使用した起動テンプレートの作成と管理の例 AWS CLI」を参照してください。

2. start-instance-refresh コマンドを使用してインスタンス置換ワークフローを開始し、新しい起動テンプレートを ID で適用する *lt-068f72b729example*。 起動テンプレートは新しいため、バージョンは 1 つだけです。つまり、起動テンプレートの 1 バージョンはこのインスタンスの更新の対象となります。インスタンスの更新中にスケールアウトイベントが発生し、Amazon EC2 Auto Scaling がこの起動テンプレート1のバージョンを使用して新しいインスタンスをプロビジョニングした場合、それらは置き換えられません。操作が正常に完了すると、新しい起動テンプレートが Auto Scaling グループに正常に適用されます。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
          "LaunchTemplateId": "lt-068f72b729example",
          "Version": "$Default"
       }
    },
    "Preferences": {
      "SkipMatching": true
    }
}
```

成功すると、コマンドは以下のような出力を返します。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

スキップマッチング (混合インスタンスグループ)

混合インスタンスポリシーを持つ Auto Scaling グループがある場合は、このセクションのステップに従って、 AWS CLI を使用してスキップマッチングでインスタンスの更新を開始します。次のオプションがあります。

- ポリシーで指定されたすべてのインスタンスタイプに適用する新しい起動テンプレートを指定します。
- ポリシーの起動テンプレートを変更するかどうかを問わず、更新された一連のインスタンスタイプを指定します。例えば、不要なインスタンスタイプから移行できます。置き換えるインスタンスのAMI、セキュリティグループ、またはその他の詳細を変更せずに、起動テンプレートをそのまま使用します。

ニーズに合ったオプションに応じて、次のいずれかのセクションにある手順に従ってください。

新しい起動テンプレートでスキップマッチングを使用するには

1. create-launch-template コマンドを使用して、Auto Scaling グループの新しい起動テンプレートを作成します。Auto Scaling グループ用に作成されたインスタンスの詳細を定義する `--launch-template-data`オプションと JSON 入力を含めます。

   例えば、次のコマンドを使用して、AMI ID で起動テンプレートを作成します。*ami-0123456789abcdef0*.

   ```
   aws ec2 create-launch-template --launch-template-name my-new-template --version-
   description version1 \
     --launch-template-data '{"ImageId":"ami-0123456789abcdef0"}'
   ```

   成功すると、コマンドは以下のような出力を返します。

   ```
   {
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-04d5cc9b88example",
        "LaunchTemplateName": "my-new-template",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2023-01-31T15:56:02.000Z",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
     }
   }
   ```

   詳細については、「を使用した起動テンプレートの作成と管理の例 AWS CLI」を参照してください。

2. Auto Scaling グループの既存の混合インスタンスポリシーを表示するには、describe-auto-scaling-groups コマンドを実行します。この情報は、インスタンスの更新を開始する次のステップで必要になります。

   次のコマンド例では、 という名前の Auto Scaling グループに設定された混合インスタンスポリシーを返します。*my-asg*.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

   成功すると、コマンドは以下のような出力を返します。

   ```
   {
     "AutoScalingGroups":[
       {
         "AutoScalingGroupName":"my-asg",
         "AutoScalingGroupARN":"arn",
         "MixedInstancesPolicy":{
           "LaunchTemplate":{
             "LaunchTemplateSpecification":{
               "LaunchTemplateId":"lt-073693ed27example",
               "LaunchTemplateName":"my-old-template",
               "Version":"$Default"
             },
             "Overrides":[
               {
                 "InstanceType":"c5.large"
               },
               {
                 "InstanceType":"c5a.large"
               },
               {
                 "InstanceType":"m5.large"
               },
               {
                 "InstanceType":"m5a.large"
               }
             ]
           },
           "InstancesDistribution":{
             "OnDemandAllocationStrategy":"prioritized",
             "OnDemandBaseCapacity":1,
             "OnDemandPercentageAboveBaseCapacity":50,
   ```

```
            "SpotAllocationStrategy":"price-capacity-optimized"
          }
        },
        "MinSize":1,
        "MaxSize":5,
        "DesiredCapacity":4,
        ...
      }
    ]
}
```

3. start-instance-refresh コマンドを使用してインスタンス置換ワークフローを開始し、新しい起動テンプレートを ID で適用します。*lt-04d5cc9b88example*。 起動テンプレートは新しいため、バージョンは 1 つだけです。つまり、起動テンプレートの 1 バージョンはこのインスタンスの更新の対象となります。インスタンスの更新中にスケールアウトイベントが発生し、Amazon EC2 Auto Scaling がこの起動テンプレート1のバージョンを使用して新しいインスタンスをプロビジョニングした場合、それらは置き換えられません。操作が正常に完了すると、更新された混合インスタンスポリシーが Auto Scaling グループに正常に適用されます。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容。

```
{
  "AutoScalingGroupName":"my-asg",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "LaunchTemplateSpecification":{
          "LaunchTemplateId":"lt-04d5cc9b88example",
          "Version":"$Default"
        },
        "Overrides":[
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          },
          {
            "InstanceType":"m5.large"
          },
```

```
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
          }
        }
      }
    },
    "Preferences":{
      "SkipMatching":true
    }
}
```

成功すると、コマンドは以下のような出力を返します。

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

次の手順では、起動テンプレートを変更せずに更新された一連のインスタンスタイプを指定します。

更新された一連のインスタンスタイプでスキップマッチングを使用するには

1. Auto Scaling グループの既存の混合インスタンスポリシーを表示するには、describe-auto-scaling-groups コマンドを実行します。この情報は、インスタンスの更新を開始する次のステップで必要になります。

   次のコマンド例では、 という名前の Auto Scaling グループに設定された混合インスタンスポリシーを返します。*my-asg*.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

   成功すると、コマンドは以下のような出力を返します。

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg",
      "AutoScalingGroupARN":"arn",
      "MixedInstancesPolicy":{
        "LaunchTemplate":{
          "LaunchTemplateSpecification":{
            "LaunchTemplateId":"lt-073693ed27example",
            "LaunchTemplateName":"my-template-for-auto-scaling",
            "Version":"$Default"
          },
          "Overrides":[
            {
              "InstanceType":"c5.large"
            },
            {
              "InstanceType":"c5a.large"
            },
            {
              "InstanceType":"m5.large"
            },
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
        }
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":4,
      ...
    }
  ]
}
```

2. start-instance-refresh コマンドを使用してインスタンス置換ワークフローを開始し、更新を適用します。特定のインスタンスタイプを使用するインスタンスを置き換える場合、必要な設定で希望するインスタンスタイプのみを含む混合インスタンスポリシーを指定する必要があります。代わりに新しいインスタンスタイプを追加するかどうかを選択できます。

   次のコマンド例では、不要なインスタンスタイプなしでインスタンスの更新を開始します。*m5a.large*。 グループ内のインスタンスタイプが残りの 3 つのインスタンスタイプのいずれとも一致しない場合、インスタンスは置き換えられます。(インスタンスの更新は、新しいインスタンスをプロビジョンするインスタンスタイプを選択しませんのでご注意ください。代わりに割り当て戦略によって選択されます) 操作が正常に完了すると、更新された混合インスタンスポリシーが Auto Scaling グループに正常に適用されます。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

   config.json の内容

```
{
  "AutoScalingGroupName":"my-asg",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "LaunchTemplateSpecification":{
          "LaunchTemplateId":"lt-073693ed27example",
          "Version":"$Default"
        },
        "Overrides":[
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          },
          {
            "InstanceType":"m5.large"
          }
        ]
      },
      "InstancesDistribution":{
        "OnDemandAllocationStrategy":"prioritized",
        "OnDemandBaseCapacity":1,
        "OnDemandPercentageAboveBaseCapacity":50,
```

```
        "SpotAllocationStrategy":"price-capacity-optimized"
        }
      }
    }
  },
  "Preferences":{
    "SkipMatching":true
  }
}
```

インスタンスの更新にチェックポイントを追加する

インスタンスの更新を使用するとき、段階的にインスタンスを置き換え、進行しながらインスタンスに検証を実行できます。段階的な置き換えを行うには、チェックポイントを追加します。チェックポイントは、インスタンスの更新が一時停止する時点です。チェックポイントを使用すると、Auto Scaling グループの更新の選択方法をより詳細に管理できます。これにより、アプリケーションが確実かつ予測可能な方法で機能することを確認できます。

内容

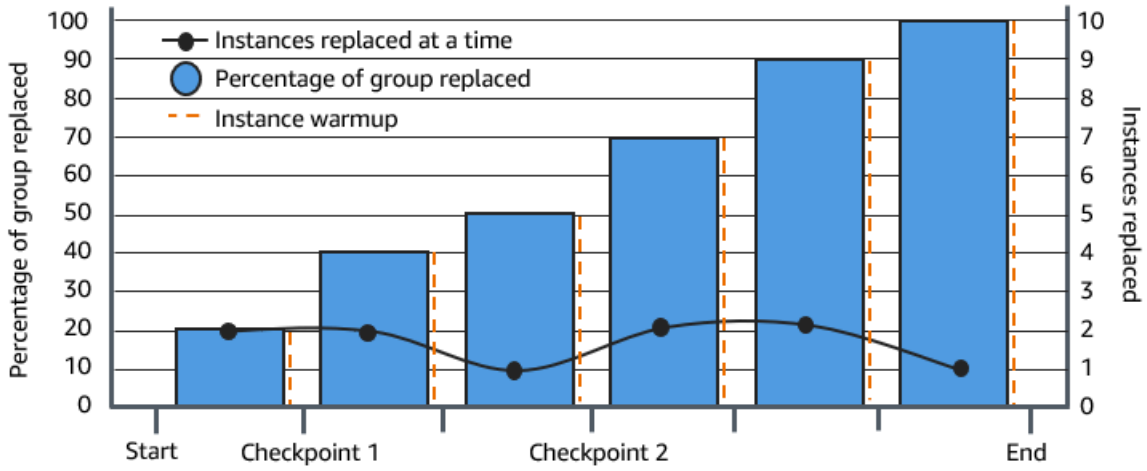
- 仕組み
- 考慮事項
- AWS Management Console または を使用して を使用してチェックポイントを有効にする AWS CLI

仕組み

インスタンスの更新を開始するときは、Auto Scaling グループ内のインスタンスの合計数のパーセンテージとしてチェックポイントを指定します。これらのチェックポイントは、チェックポイントに到達する前に、Auto Scaling グループ内の新しいインスタンスでなければならないインスタンスの最小割合を示します。例えば、チェックポイントが `[20, 50, 100]` の場合、インスタンスの 20% が新しいときに最初のチェックポイントに到達し、50% が新しいときに 2 番目のチェックポイントに到達し、すべてのインスタンスが新しいときに最後のチェックポイントに到達します。

Amazon EC2 Auto Scaling は、グループの最小正常率を維持しながら、指定されたチェックポイントの割合を尊重してインスタンスの置換をペース調整します。チェックポイントの割合に達するために、Amazon EC2 Auto Scaling は、最小の正常な割合が許容する数よりも少ないが、それ以上は置き換えないことがあります。

10 個のインスタンスがある次の Auto Scaling グループについて考えてみましょう。チェックポイントのパーセンテージは [20,50,100] で、最小正常率は 80 %、最大正常率は 100% です。最小の正常なパーセンテージを維持するために、一度に置き換えることができるインスタンスは 2 つだけです。次の図は、チェックポイントに達する前にインスタンスを置き換えるプロセスの概要を示しています。



上記の例では、開始される新しいインスタンスごとにインスタンスのウォームアップ期間が設けられます。インスタンスを待機状態にして、起動または終了時にカスタムアクションを実行するライフサイクルフックを使用することもできます。

Amazon EC2 Auto Scaling は、100% 完全なチェックポイントを除き、チェックポイントごとにイベントを発行します。Amazon EventBridge などのターゲットにイベントを送信する SNS ルールを追加できます。これにより、必要な検証を実行できるタイミングが通知されます。詳細については、「インスタンスの更新イベント用の EventBridge ルールを作成する」を参照してください。

考慮事項

チェックポイントを使用する際は、次の考慮事項に注意してください。

- チェックポイントは割合に基づいているため、置換されるインスタンスの数はグループサイズに応じて変化します。スケールアウトアクティビティが発生し、グループサイズが大きくなると、進行中のオペレーションがチェックポイントに再び到達する可能性があります。この場合、Amazon EC2 Auto Scaling は別の通知を送信し、チェックポイント間の待機時間を繰り返してから続行します。
- 特定の状況下では、チェックポイントをスキップすることができます。例えば、Auto Scaling グループに 2 個のインスタンスがあり、チェックポイントの割合が [10,40,100] だとします。最初のインスタンスが置き換えられると、Amazon EC2 Auto Scaling はグループの 50% が置き換え

られたと計算します。50% は最初の 2 つのチェックポイントよりも高いため、最初のチェックポイント (10) をスキップし、2 番目のチェックポイント (40) の通知を送信します。

- オペレーションをキャンセルすると、それ以降の置換は行われません。オペレーションをキャンセルするか、最後のチェックポイントに到達する前に失敗した場合、すでに置き換えられたインスタンスは前の設定にロールバックされません。
- 部分的な更新の場合、オペレーションを再実行すると、Amazon EC2 Auto Scaling は最後のチェックポイントの時点から再起動せず、以前のインスタンスのみが置き換えられたときにも停止しません。ただし、新しいインスタンスをターゲットにする前に、まず古いインスタンスを置き換え対象とします。
- グループ内のインスタンスの数に比べてチェックポイントのパーセンテージが低すぎる場合、実際の完了パーセンテージがそのチェックポイントのパーセンテージより高くなることもあります。例えば、チェックポイントのパーセンテージが 20 % で、グループに 4 つのインスタンスがあるとします。Amazon EC2 Auto Scaling が 4 つのインスタンスのいずれかを置き換える場合、実際の置き換え率 (25%) はチェックポイントの割合 (20%) よりも高くなります。
- チェックポイントに到達すると、表示される全体的な完了パーセンテージは、インスタンスのウォームアップが完了するまで更新されません。例えば、チェックポイントの割合は `[20,50]` で、チェックポイントの遅延は 15 分、最小正常率は 80% です。Auto Scaling グループには 10 個のインスタンスがあり、次の置換を行います。
  - 0:00: 2 個の古いインスタンスが新しいインスタンスに置き換えられます。
  - 0:10: 2 個の新しいインスタンスがウォームアップを完了します。
  - 0:25: 2 個の古いインスタンスが新しいインスタンスに置き換えられます。(最小正常率を維持するために、2 個のインスタンスのみが置換されます)。
  - 0:35: 2 個の新しいインスタンスがウォームアップを完了します。
  - 0:35: 1 個の古いインスタンスが新しいインスタンスに置き換えられます。
  - 0:45: 1 個の新しいインスタンスがウォームアップを完了します。

  0:35 で、オペレーションは新しいインスタンスの起動を停止します。新しいインスタンスがウォームアップされていないため、完了率は、完了した置換の数 (50%) を正確に反映していません。新しいインスタンスがウォームアップ期間を 0:45 に完了すると、完了率は 50% と表示されます。

AWS Management Console または を使用して を使用してチェックポイントを有効にする AWS CLI

AWS Management Console または を使用してチェックポイント AWS CLI を有効にできます。

チェックポイントを有効にする (コンソール)

インスタンスの更新を開始する前にチェックポイントを有効にして、増分または段階的なアプローチを使用し、インスタンスを置換することができます。これにより、検証にさらに時間がかかります。

チェックポイントを使用するインスタンスの更新をスタートするには

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   Auto Scaling グループページの下部に分割ペインが開きます。
3. [Instance refresh] (インスタンスの更新) タブの [Active instance refresh] (アクティブインスタンスの更新) で、[Start instance refresh] (インスタンスの更新を開始する) を選択します。
4. [Start instance refresh] (インスタンスの更新をスタートする) ページで、[Minimum healthy percentage] (最小正常率) および [Instance warmup] (インスタンスのウォームアップ) に値を入力します。
5. [Enable checkpoints] (チェックポイントを有効にする) チェックボックスをオンにします。

   これにより、最初のチェックポイントのしきい値をパーセンテージで定義できるボックスが表示されます。
6. [Proceed until ____ % of the group is refreshed] (グループの ____% が更新されるまで続行する) に数値 (1 ~ 100) を入力します。これにより、最初のチェックポイントの割合が設定されます。
7. 別のチェックポイントを追加するには、[Add checkpoint (チェックポイントの追加)]を選択し、次のチェックポイントの割合を定義します。
8. チェックポイントに到達した後の Amazon EC2 Auto Scaling の待機時間を指定するには、チェックポイント**1hour**間の待機のフィールドを更新します。時間単位は、時、分、秒のいずれかです。
9. インスタンスの更新の選択が終了したら、[インスタンスの更新を開始する] を選択します。

チェックポイントを有効にする (AWS CLI)

を使用してチェックポイントを有効にしてインスタンスの更新を開始するには AWS CLI、次のパラメータを定義する設定ファイルが必要です。

- `CheckpointPercentages`: 置き換えるインスタンスの割合のしきい値を指定します。これらのしきい値は、チェックポイントを提供します。置換およびウォームアップされたインスタンスの割

合が指定されたしきい値の 1 つに達すると、オペレーションは指定された期間待機します。待機時間をCheckpointDelay秒単位で指定します。指定した期間が経過すると、インスタンスの更新は次のチェックポイント (該当する場合) に到達するまで続行されます。

- CheckpointDelay: チェックポイントに到達してから続行するまでに待機する時間 (秒) を規定します。検証の実行に十分な時間を選択します。

CheckpointPercentages 配列に表示される最後の値は、正常に置換する必要がある Auto Scaling グループの割合を示します。このパーセンテージが正常に置き換えられ、各インスタンスの初期化が完了したとみなされると、オペレーションは Successful になります。

複数のチェックポイントを作成するには

複数のチェックポイントを作成するには、次の start-instance-refresh コマンドの例を使用します。この例では、最初に Auto Scaling グループの 1% を更新する、インスタンスの更新を設定します。10 分の待機後、次の 19% が更新され、さらに 10 分待機します。最後に、オペレーション終了前にグループの残りの部分が更新されます。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20,100],
      "CheckpointDelay": 600
    }
}
```

単一のチェックポイントを作成するには

1 つのチェックポイントを作成するには、次の start-instance-refresh コマンドの例を使用します。この例では、最初に Auto Scaling グループの 20% を更新する、インスタンスの更新を設定します。10 分の待機後、オペレーション終了前にグループの残りの部分が更新されます。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [20,100],
      "CheckpointDelay": 600
    }
}
```

Auto Scaling グループを部分的に更新するには

Auto Scaling グループの一部のみを置き換えてから完全に停止するには、次の start-instance-refresh コマンドの例を使用します。この例では、最初に Auto Scaling グループの 1% を更新する、インスタンスの更新を設定します。10 分の待機後、オペレーション終了前に次の 19% が更新されます。

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

config.json の内容:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20],
      "CheckpointDelay": 600
    }
}
```

インスタンスの最大存続期間に基づいて Auto Scaling インスタンスを置き換える

インスタンスの最大有効期間は、インスタンスが終了し置き換えられるまでに稼働できる最大時間 (秒単位) を指定します。一般的なユースケースでは、内部のセキュリティポリシーや外部のコンプライアンスコントロールにより、スケジュールどおりにインスタンスを置換する要件がある場合があります。

86,400 秒 (1 日) 以上の値を指定する必要があります。以前に設定した値をクリアするには、新しい値 0 を指定します。この設定は、Auto Scaling グループの現在および今後のすべてのインスタンスに適用されます。

内容

考慮事項

この機能を使用する際の考慮事項を次に示します。

- 古いインスタンスが置き換えられて新しいインスタンスが起動するたび、新しいインスタンスは Auto Scaling グループに現在関連付けられている起動テンプレートまたは起動設定を使用します。起動テンプレートまたは起動設定でアプリケーションの別のバージョンの Amazon マシンイメージ (AMI) ID が指定されている場合、このバージョンのアプリケーションは自動的にデプロイされます。
- インスタンスの最大有効期間の設定が低すぎると、インスタンスが想定よりも早く置き換えられる可能性があります。Amazon EC2 Auto Scaling は通常、インスタンスを一度に 1 つずつ置き換え、置き換えの間に一時停止します。ただし、指定された最大インスタンスライフタイムが各インスタンスを個別に置き換えるのに十分な時間を提供しない場合、Amazon EC2 Auto Scaling は一度に複数のインスタンスを置き換える必要があります。Auto Scaling グループの現在のキャパシティの最大 10% まで、複数のインスタンスが置換される場合があります。一度に多くのインスタンスが置換されないようにするには、インスタンスの最大有効期間を長く設定するか、インスタンススケールイン保護を使用して、個々のインスタンスが一時的に終了しないようにします。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。
- デフォルトでは、Amazon EC2 Auto Scaling はインスタンスを終了するための新しいスケーリングアクティビティを作成し、終了します。インスタンスの終了中、別のスケーリングアクティビティが新しいインスタンスを起動します。インスタンス メンテナンス ポリシーを使用して、終了前に起動するようにこの動作を変更できます。詳細については、「インスタンスメンテナンスポリシー」を参照してください。

最大インスタンスライフタイムを設定する

コンソールで Auto Scaling グループを作成する場合、インスタンスの最大ライフタイムを設定することはできません。ただし、グループ作成後に、グループを編集してインスタンスの最大ライフタイムを設定できます。

グループの最大インスタンスライフタイムを設定するには (コンソール)

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部に分割ペインが開き、選択したグループに関する情報が表示されます。
3. [詳細] タブで、[高度な設定]、[編集] の順に選択します。
4. [Maximum instance lifetime] (最大インスタンス有効期間) に、インスタンスが稼働できる最大秒数を入力します。
5. [Update] (更新) を選択します。

[Activity history] (アクティビティ履歴) の [Activity] (アクティビティ) タブでは、履歴全体にグループのインスタンスの置換を表示できます。

グループの最大インスタンスライフタイムを設定するには (AWS CLI)

を使用して、新規または既存の Auto Scaling グループのインスタンスの最大有効期間 AWS CLI を設定することもできます。

新しい Auto Scaling グループの場合は、create-auto-scaling-group コマンドを使用します。

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

次の例は、インスタンスの最大有効期間を 2592000 秒 (30 日) で示す config.json ファイルです。

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
```

```
        "Version": "$Default"
    },
    "MinSize": 1,
    "MaxSize": 5,
    "MaxInstanceLifetime": 2592000,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

既存の Auto Scaling グループの場合は、update-auto-scaling-group コマンドを使用します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --
max-instance-lifetime 2592000
```

Auto Scaling グループの最大インスタンス有効期間を確認するには

describe-auto-scaling-groups コマンドを使用します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

制限

- すべてのインスタンスで最大ライフタイムが正確であるとは限りません: インスタンスが置き換えられるのは、最大期間が終了したときだけとは限りません。状況によっては、Amazon EC2 Auto Scaling が最大インスタンス有効期間パラメータを更新した直後にインスタンスの置き換えを開始する必要がある場合があります。この動作の理由は、すべてのインスタンスを同時に置き換えることを避けることです。
- インスタンスのスケールイン保護が優先: Amazon EC2 Auto Scaling は、インスタンスのスケールイン保護を提供し、終了できるインスタンスの制御を支援します。インスタンスでこの保護が有効になっている場合、Amazon EC2 Auto Scaling は、インスタンスの最大有効期間に達した場合でもインスタンスを終了しません。
- 起動前に終了したインスタンス: Auto Scaling グループにインスタンスが 1 つしかない場合、Amazon EC2 Auto Scaling はインスタンスを終了してからデフォルトで新しいインスタンスを起動するため、インスタンスの最大有効期間機能によって停止する可能性があります。この動作を終了前に起動するように変更するには、「インスタンスメンテナンスポリシー」を参照してください。

スケーリングでアプリケーションのコンピューティングキャパシティを増減する

スケーリングは、アプリケーションのコンピューティングキャパシティーを増減する機能です。スケーリングは、Amazon EC2インスタンスを起動または終了するように Auto Scaling グループに指示するイベントまたはスケーリングアクションから始まります。

Amazon EC2 Auto Scaling には、アプリケーションのニーズに合わせてスケーリングを調整するさまざまな方法が用意されています。そのため、アプリケーションを十分に理解していることが重要です。次の考慮事項に注意が必要です。

- Amazon EC2 Auto Scaling はアプリケーションのアーキテクチャでどのような役割を果たしますか？ 自動スケーリングはキャパシティーの増減手段として考えるのが一般的ですが、一定数のサーバーを維持する場合にも便利です。
- どのようなコストの制約がお客様にとって重要か。Amazon EC2 Auto Scaling はEC2インスタンスを使用するため、使用したリソースに対してのみ料金が発生します。コストの制約を知ることは、アプリケーションをスケーリングするタイミングと量を決定するときに役立ちます。
- アプリケーションにとって重要なメトリクスは何ですか？ Amazon CloudWatch は、Auto Scaling グループで使用できるさまざまなメトリクスをサポートしています。

内容

スケーリング方法を選択する

Amazon EC2 Auto Scaling には、Auto Scaling グループをスケーリングするためのいくつかの方法が用意されています。

固定数のインスタンスを維持する

Auto Scaling グループのデフォルトは、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がアタッチされていないことです。これにより、固定サイズのままになります。Auto Scaling グループの作成後、希望するキャパシティを満たすのに十分なインスタンスを起動すると、グループが起動します。グループにスケーリング条件がアタッチされていない場合、インスタンスが異常になっても、希望するキャパシティは維持されます。Amazon EC2 Auto Scaling は、Auto Scaling グループ内の各インスタンスの状態をモニタリングします。インスタンスが異常になったことがわかった場合、新しいインスタンスに置き換えられます。このプロセスの詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

手動でスケールする

手動スケーリングは、Auto Scaling グループをスケールする最も基本的な方法です。Auto Scaling グループの希望するキャパシティを更新するか、Auto Scaling グループのインスタンスを終了できます。詳細については、「Amazon EC2 Auto Scaling の手動スケーリング」を参照してください。

スケジュールに基づくスケーリング

スケジュールに基づくスケーリングとは、日付と時刻の関数として自動的に実行されるスケーリングアクションのことです。グループのインスタンスの数を増減しなければならない状況が予測可能なスケジュールで発生するため、いつその数を増減すべきかが正確にわかっている場合に、このスケーリング方法は便利です。詳細については、「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。

需要に基づいて動的にスケールする

動的なスケーリングを使用して、リソースをスケーリングする高度な方法では、需要の変化に応じて Auto Scaling グループを動的にサイズ変更するスケーリングポリシーを定義できます。例えば、現在 2 つのインスタンスで実行されているウェブ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 Auto Scaling グループのCPU使用率を約 50% に維持したいとします。この方法は、トラフィックがいつ変化するかわからない場合に、トラフィックの変化が発生したときにスケールするのに役立ちます。自動的に反応するようにスケーリングポリシーを設定できます。トラフィックの変化に応じてスケールするために使用できる複数のポリシータイプ (またはその組み合わせ) があります。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。

プロアクティブにスケールする

また、予測スケーリングと動的スケーリング (それぞれ、事前アプローチと事後アプローチ) を組み合わせて、EC2キャパシティをより迅速にスケーリングすることもできます。予測スケーリングを使用して、トラフィックフローの日次および週次のパターンの前に Auto Scaling グループのEC2インスタンス数を増やします。詳細については、「Amazon EC2 Auto Scaling の予測スケーリング」を参照してください。

Auto Scaling グループのスケーリング制限を設定する

スケーリング制限は、Auto Scaling グループで希望する最小グループサイズと最大グループサイズを表します。最小サイズと最大サイズに制限を個別に設定します。

グループで希望するキャパシティは、最小および最大サイズの制限内の数値でサイズ変更できます。希望するキャパシティは、グループの最小サイズ以上、最大サイズ以下である必要があります。

- 希望するキャパシティ: Auto Scaling グループ作成時の初期キャパシティを表します。Auto Scaling グループは希望するキャパシティを維持しようとします。まず、希望するキャパシティに指定された数のインスタンスを起動します。Auto Scaling グループに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がアタッチされていない限り、このインスタンス数が維持されます。
- 最小キャパシティ: 最小グループサイズを表します。スケーリングポリシーが設定されている場合、グループの希望するキャパシティを最小キャパシティよりも小さくすることはできません。
- 最大キャパシティ: 最大グループサイズを表します。スケーリングポリシーが設定されている場合、グループの希望するキャパシティを最大キャパシティよりも大きくすることはできません。

最小および最大サイズの制限は、次のシナリオにも適用されます。

- 希望するキャパシティを更新してAuto Scaling グループを手動でスケールする場合。
- 希望するキャパシティを更新する、スケジュールされたアクションが実行される場合。グループに新しい最小および最大サイズ制限を指定せずにスケジュールされたアクションを実行すると、グループの現在の最小サイズ制限と最大サイズ制限が適用されます。

Auto Scaling グループは常に、希望するキャパシティを維持しようとします。インスタンスが予期せず終了した場合 (スポットインスタンスの中断、ヘルスチェックの失敗、人為的なアクションなど)、グループは自動的に新しいインスタンスを起動して、希望するキャパシティを維持します。

コンソールでこれらの設定を変更するには

1. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。
2. ナビゲーションペインの 自動スケーリング で、[Auto Scaling Groups] (Auto Scaling グループ) を選択します。
3. Auto Scaling グループページで、Auto Scaling グループの横にあるチェックボックスをオンにします。

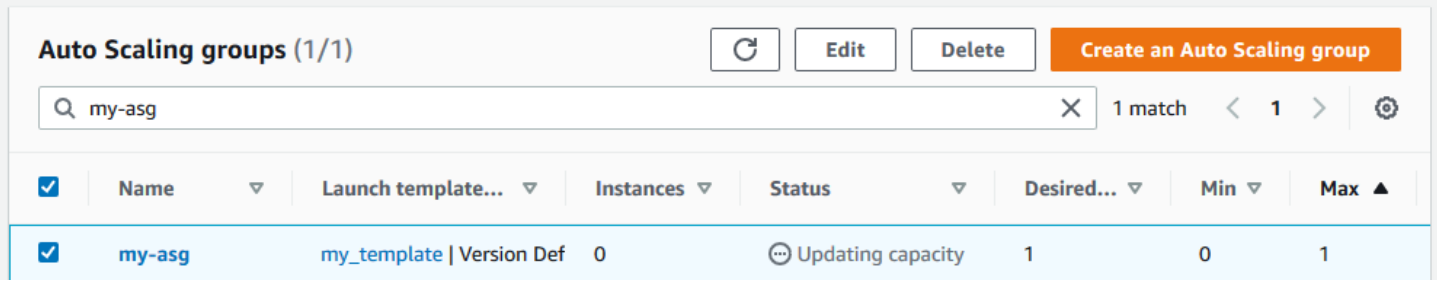
   ページの下部にスプリットペインが開きます。
4. 下側のペインでは、[詳細] タブで、グループの最小キャパシティ、最大キャパシティ、希望するキャパシティの現在の設定を表示または変更します。詳細については、「既存の Auto Scaling グループの希望するキャパシティを変更する」を参照してください。

[詳細] ペインより上方には、Auto Scaling グループの現在のインスタンス数、希望するキャパシティ、最小キャパシティ、最大キャパシティ、ステータス列などの情報が表示されます。Auto Scaling グループがインスタンスの重み付けを使用する場合、希望するキャパシティに寄与するキャパシティユニット数も表示されます。

一覧から列を追加または削除するには、ページ上部にある設定アイコンを選択します。次に、[Auto Scaling groups attributes] (Auto Scaling グループの属性) で各列のオンとオフを指定して、[Confirm] (確認) を選択します。

変更後に Auto Scaling グループのサイズを確認するには

[Instances (インスタンス)] 列には、現在実行中のインスタンスの数が表示されます。インスタンスの起動または終了中は、[Status (ステータス)] 列に次のイメージで示すように「Updating capacity (キャパシティーの更新)」というステータスが表示されます。



数分待ってから、ビューを更新して最新のステータスを確認します。スケーリングアクティビティが完了すると、[Instances] (インスタンス) 列に更新された値が表示されます。

[Instances] (インスタンス) の [Instance management] (インスタンス管理) タブから、インスタンス数、および現在実行されているインスタンスのステータスを表示できます。

Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する

CloudWatch は、Auto Scaling インスタンス全体で、 CPUやネットワーク I/O などの使用状況データを収集して集計します。これらのメトリクスを使用して、選択したメトリクスの値の増減に応じて Auto Scaling グループ内にあるインスタンスの数を調整するスケーリングポリシーを作成します。

インスタンスが `InService` 状態になってから、集約されたメトリクスに使用状況データが反映されるまでの待機時間を指定できます。この指定された時間は、デフォルトのインスタンスウォームアップと呼ばれます。これは、アプリケーショントラフィックの処理をまだ行っておらず、コンピューティングリソースの使用率が一時的に高くなっている可能性がある個々のインスタンスのメトリクスが、動的スケーリングに影響を与えないようにします。

ターゲット追跡ポリシーとステップスケーリングポリシーのパフォーマンスを最適化するには、デフォルトのインスタンスウォームアップを有効にして設定することを強くお勧めします。デフォルトでは有効ではなく設定されてもいません。

デフォルトのインスタンスウォームアップを有効にするときは、Auto Scaling グループがインスタンスメンテナンスポリシーを使用するように設定されている場合、またはインスタンスの更新を使用してインスタンスを置き換える場合、インスタンスが初期化を完了する前に最小の正常な割合にカウントされないようにできることに注意してください。

内容

パフォーマンスのスケーリングに関する考慮事項

ほとんどのアプリケーションでは、機能ごとに異なるウォームアップ時間を設けるよりも、1 つのデフォルトのインスタンスウォームアップ時間をすべての機能に適用するほうが便利です。例えば、デフォルトのインスタンスウォームアップを設定しない場合、インスタンスの更新機能はデフォルトのウォームアップ時間としてヘルスチェックの猶予期間を使用します。ターゲット追跡ポリシーとステップスケーリングポリシーがある場合は、デフォルトのクールダウンに設定された値をデフォルトのウォームアップ時間として使用します。予測スケーリングポリシーがある場合、デフォルトのウォームアップ時間はありません。

インスタンスがウォームアップしている間、動的スケーリングポリシーは、ウォームアップしていないインスタンスのメトリクス値がポリシーのアラーム上限しきい値 (またはターゲット追跡スケーリングポリシーのターゲット利用率) より大きい場合にのみスケールアウトします。需要が減少すると、動的スケーリングは控えめになり、アプリケーションの可用性を保護します。これにより、新しいインスタンスがウォームアップを完了するまで、動的スケーリングのスケールインアクティブティがブロックされます。

スケールアウト中、Amazon EC2 Auto Scaling は、グループに追加するインスタンスの数を決定する際に、ウォームアップ中のインスタンスをグループのキャパシティーの一部として考慮します。したがって、同程度の量のキャパシティの追加を必要とする複数のアラーム違反は、単一のスケーリングアクティビティになります。その目的は、スケールアウトを継続的に (ただし過剰になることなく) 行うことです。

デフォルトのインスタンスウォームアップが有効になっていない場合、 にメトリクスを送信 CloudWatch して現在の容量にカウントするまでインスタンスが待機する時間は、インスタンスによって異なります。そのため、実際に発生しているワークロードと比較して、予期せずスケーリングポリシーが実行される可能性があります。

例えば、反復的な on-and-offワークロードパターンを持つアプリケーションを考えてみましょう。予測スケーリングポリシーを使用して、インスタンス数を増やすかどうかを繰り返し決定します。予測スケーリングポリシーにはデフォルトのウォームアップ時間がないため、インスタンスは集約されたメトリクスにすぐに反映されます。これらのインスタンスの起動時のリソース使用量が多い場合、インスタンスを追加すると、集約されたメトリックスが急増する可能性があります。使用量が安定するまでにかかる時間によっては、これらの指標を使用する動的スケーリングポリシーに影響する可能性があります。動的スケーリングポリシーのアラーム上限しきい値を超えると、グループのサイズは再び大きくなります。新しいインスタンスがウォームアップしている間、スケールインアクティビティはブロックされます。

デフォルトのインスタンスウォームアップ時間を選択する

デフォルトのインスタンスのウォームアップを設定する上で重要なのは、インスタンスが InService の状態に達した後、初期化を終了し、リソースの消費が安定するまでに必要な時間を決定することです。インスタンスのウォームアップ時間を選択するときは、正当なトラフィックの使用状況データを収集しつつ、スタートアップ時の一時的な使用量の急増に関連するデータ収集を最小限に抑えるという最適なバランスを保つようにします。

Auto Scaling グループが Elastic Load Balancing ロードバランサーにアタッチされているとします。新しいインスタンスが起動を完了すると、InService 状態に入る前にロードバランサーに登録されます。インスタンスが InService 状態になった後も、リソースの消費は引き続き一時的に急増する場合があり、安定化する時間が必要です。例えば、大量のアセットをダウンロードしてキャッシュする必要があるアプリケーションサーバーのリソース消費が安定するまでにかかる時間は、ダウンロードする大量のアセットがない軽量のウェブサーバーよりも長くなります。インスタンスのウォームアップは、リソース消費の安定化に必要な遅延時間を提供します。

> **Important**
>
> ウォームアップ時間に必要な時間がわからない場合は、300 秒から開始できます。次に、アプリケーションに最適なスケーリングパフォーマンスが得られるまで、徐々に減らすか、増やします。最適にするには、これを数回実行する必要がある場合があります。あるいは、独自のウォームアップ時間 (EstimatedInstanceWarmup) が設定されているスケーリングポリシーがある場合は、この値を使用して開始することもできます。詳細については、「事前に設定されたインスタンスのウォームアップ時間を含むスケーリングポリシーを検索する」を参照してください。

スタートアップ時に実行される設定タスクやスクリプトを伴うユースケースにライフサイクルフックの使用を検討してください。ライフサイクルフックは、インスタンスが初期化を終了するまで、新しいインスタンスがサービスに使用されるのを遅らせることができます。ライフサイクルフックは特に、完了に時間がかかるブートストラップスクリプトがある場合に便利です。ライフサイクルフックを追加すると、デフォルトのインスタンスウォームアップの値を減らすことができます。ライフサイクルフック使用の詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

グループに対するインスタンスのデフォルトウォームアップを有効にする

インスタンスのデフォルトウォームアップは、Auto Scaling グループの作成時に有効化できます。既存のグループに対して有効化することも可能です。

デフォルトのインスタンスウォームアップ機能を有効にすると、以下の機能のウォームアップパラメータに値を指定する必要がなくなります

- インスタンスの更新
- ターゲットトラッキングスケーリング
- ステップスケーリング

Console

新しいグループに対してインスタンスのデフォルトウォームアップを有効にする (コンソール)

Auto Scaling グループを作成するときに、[Configure advanced options] (詳細オプションを設定) ページの [Additional settings] (追加設定)で、[Enable default instance warmup] (インスタンスのデフォルトウォームアップを有効にする) オプションを選択します。アプリケーションに必要なウォームアップ時間を選択します。

AWS CLI

新しいグループに対してインスタンスのデフォルトウォームアップを有効にする (AWS CLI)

Auto Scaling グループに対してインスタンスのデフォルトウォームアップを有効にするには、`--default-instance-warmup` オプションを追加して、0 から 3600 までの値 (秒単位) を指定します。`-1` の値は、有効化後にこの設定をオフにします。

次のcreate-auto-scaling-groupコマンドは、 という名前の Auto Scaling グループを作成し*my-asg*、 *120* 秒の値でデフォルトのインスタンスウォームアップを有効にします。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120 ...
```

> **Tip**
>
> 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。

Console

既存のグループに対してインスタンスのデフォルトウォームアップを有効にする (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. [詳細] タブで、[高度な設定]、[編集] の順に選択します。
5. [デフォルトのインスタンスのウォームアップ] で、アプリケーションに必要なウォームアップ時間を選択します。
6. [Update] (更新) を選択します。

AWS CLI

既存のグループに対してインスタンスのデフォルトウォームアップを有効にする (AWS CLI)

次の例では、 update-auto-scaling-group コマンドを使用して、 という名前の既存の Auto Scaling グループに対して、 *120* 秒の値でデフォルトのインスタンスウォームアップを有効にします*my-asg*。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120
```

> **Tip**
>
> 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。

グループに対するデフォルトのインスタンスウォームアップ時間を検証する

AWS CLIを使用して Auto Scaling グループのデフォルトのインスタンスウォームアップ時間を検証するには、次の手順に従います。

Auto Scaling グループに対するインスタンスのデフォルトウォームアップ時間を検証するには

次の describe-auto-scaling-groups コマンドを使用します。を Auto Scaling グループの名前*my-asg*に置き換えます。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下に、応答の例を示します。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "DefaultInstanceWarmup": 120
        }
    ]
}
```

事前に設定されたインスタンスのウォームアップ時間を含むスケーリングポリシーを検索する

`EstimatedInstanceWarmup` に対して独自のウォームアップ時間を持つポリシーがあるかどうかを確認するには、 AWS CLIを使用して次の describe-policies コマンドを実行します。を Auto Scaling グループの名前*my-asg*に置き換えます。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?EstimatedInstanceWarmup!=`null`]'
```

以下は出力例です。

```
[
  {
```

```
    "AutoScalingGroupName":"my-asg",
    "PolicyName":"cpu50-target-tracking-scaling-policy",
    "PolicyARN":"arn",
    "PolicyType":"TargetTrackingScaling",
    "StepAdjustments":[],
    "EstimatedInstanceWarmup":120,
    "Alarms":[{
        "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-
asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
      },
      {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }],
    "TargetTrackingConfiguration":{
      "PredefinedMetricSpecification":{
        "PredefinedMetricType":"ASGAverageCPUUtilization"
      },
      "TargetValue":50.0,
      "DisableScaleIn":false
    },
    "Enabled":true
  },

    ... additional policies ...

]
```

以前に設定したスケーリングポリシーのインスタンスウォームアップをクリアする

デフォルトのインスタンスウォームアップを有効にしたら、まだウォームアップ時間が残っているスケーリングポリシーを更新して、以前に設定した値をクリアします。そうしないと、デフォルトのインスタンスウォームアップがオーバーライドされます。

スケーリングポリシーは、 コンソール、 AWS CLI、または を使用して更新できます AWS SDKs。このセクションでは、コンソールの手順について説明します。 AWS CLI または を使用する場合は

AWS SDKs、既存のポリシー設定を保持し、 `EstimatedInstanceWarmup`プロパティを削除してください。既存のスケーリングポリシーを更新すると、ポリシーはプログラムで を呼び出すときに指定したポリシーに置き換えられますPutScalingPolicy。元の値は保持されません。

以前に設定したスケーリングポリシーのインスタンスウォームアップをクリアする

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [自動スケーリング] タブの [動的スケーリングポリシー] で、目的のポリシーを選択し、[アクション]、[編集] を選択します。
4. [インスタンスのウォームアップ] で、インスタンスのウォームアップ値をクリアして、代わりにデフォルトのインスタンスウォームアップ値を使用します。
5. [Update] (更新) を選択します。

Amazon EC2 Auto Scaling の手動スケーリング

Auto Scaling グループのEC2インスタンス数はいつでも手動で調整できます。インスタンス数を手動で変更するこのプロセスは、手動スケーリングと呼ばれます。手動スケーリングは、特に 1 回限りのキャパシティ変更を行う場合、自動スケーリングの代替手段です。

グループを手動でスケーリングすると、Amazon EC2 Auto Scaling は、定義したスケーリングポリシーとスケジュールされたアクションに基づいて、通常の自動スケーリングアクティビティを再開します。デフォルトのインスタンスウォームアップが有効になっているグループの場合、新しいインスタンスは、ウォームアップ期間が経過してから自動スケーリングに使用されるメトリクスに反映されます。このウォームアップ期間は、新しいキャパシティでグループを安定化するのに役立ちます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

グループを手動でスケールする前に、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一時的に無効にした方がよい場合があります。これにより、手動スケーリングアクションと自動スケーリングアクティビティの間に競合が発生するのを防ぎます。詳細については、「スケーリングアクティビティをオフにする」を参照してください。

内容

- 既存の Auto Scaling グループの希望するキャパシティを変更する
- Auto Scaling グループのインスタンスを終了する (AWS CLI)

既存の Auto Scaling グループの希望するキャパシティを変更する

Auto Scaling グループの希望するキャパシティを変更すると、Amazon EC2 Auto Scaling は、新しい希望するサイズに達するようにインスタンスを起動および終了するプロセスを管理します。

Console

Auto Scaling グループのサイズを変更するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部に分割されたペインが開きます。
3. [詳細] タブで、[グループの詳細]、[編集] の順に選択します。
4. [希望する容量] で、希望するキャパシティを増加または減少させます。例えば、グループのサイズを 1 つ増やすには、現在の値が 1 の場合、2 と入力します。

   [希望する容量] の新しい値が、[最小の希望する容量] と [最大の希望する容量] より大きい場合、[最大の希望する容量] は自動的に新しい希望するキャパシティの値に引き上げられます。
5. 完了したら、[更新] を選択します。

指定したグループサイズと同じ量のインスタンスが起動されたことを検証します。例えば、グループのサイズを 1 つ増やした場合は、Auto Scaling グループが追加のインスタンスを 1 つ起動していることを検証します。

Auto Scaling グループのサイズが変更されたことを確認するには

1. [アクティビティ] タブの [アクティビティ履歴] で、Auto Scaling グループに関連付けられているアクティビティの進行状況を表示できます。[ステータス] 列には、インスタンスの現在のステータスが表示されます。インスタンスが起動している間、ステータス列には [Not yet in service] と表示されます。ステータスは、インスタンスが起動されると Successful に変わります。更新アイコンを使用して、インスタンスの現在のステータス

を表示することもできます。詳細については、「Auto Scaling グループのスケーリングアクティビティを検証する」を参照してください。

2. [インスタンス管理] タブの [インスタンス] で、インスタンスのステータスを表示できます。インスタンスの起動には短時間かかります。

   - [ライフサイクル] 列には、インスタンスの状態が表示されます。最初、インスタンスの状態は Pending です。インスタンスがトラフィックを受信できるようになったら、そのステータスは InService です。
   - ヘルスステータス列には、インスタンスの Amazon EC2 Auto Scaling ヘルスチェックの結果が表示されます。

AWS CLI

以下の例では、最小サイズが 1 で、最大サイズが 5 である Auto Scaling グループを作成したことを前提としています。したがって、このグループでは現在インスタンスを実行中です。

Auto Scaling グループのサイズを変更するには

次の例に示すように、 set-desired-capacity コマンドを使用して Auto Scaling グループのサイズを変更します。

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2
```

Auto Scaling グループのデフォルトのクールダウン期間を受け入れることを選択した場合は、以下の例に示しているように --honor-cooldown オプションを指定する必要があります。詳細については、「Amazon EC2 Auto Scaling のスケーリングのクールダウン」を参照してください。

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2 --honor-cooldown
```

Auto Scaling グループのサイズを確認するには

次の例のように、 describe-auto-scaling-groups コマンドを使用して Auto Scaling グループのサイズが変更されていることを確認します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

グループおよび起動されたインスタンスの詳細を示す出力例を次に示します。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 2,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 300,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
```

```
                    },
                    "InstanceId": "i-0c20ac468fa3049e8",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            "CreatedTime": "2019-03-18T23:30:42.611Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-c87f2be0",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN": "arn",
            "TrafficSources": []
        }
    ]
}
```

`DesiredCapacity` が新しい値を示していることに注意してください。また、Auto Scaling グループによって追加のインスタンスが起動されています。

Auto Scaling グループのインスタンスを終了する (AWS CLI)

Auto Scaling グループを手動でスケールインしたいが、特定のインスタンスを終了したい場合があります。Auto Scaling グループを手動でスケールインするには、terminate-instance-in-auto-scaling-group コマンドを使用し、次の例に示すように、終了するインスタンスの ID と `--should-decrement-desired-capacity`オプションを指定します。

```
aws autoscaling terminate-instance-in-auto-scaling-group \
  --instance-id i-026e4c9f62c3e448c --should-decrement-desired-capacity
```

スケーリングアクティビティの詳細を示す出力例を次に示します。

```
{
    "Activities": [
        {
            "ActivityId": "b8d62b03-10d8-9df4-7377-e464ab6bd0cb",
```

```
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-026e4c9f62c3e448c",
            "Cause": "At 2023-09-23T06:39:59Z instance i-026e4c9f62c3e448c was taken
 out of service in response to a user request, shrinking the capacity from 1 to 0.",
            "StartTime": "2023-09-23T06:39:59.015000+00:00",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":\"us-
west-2c\"}"
        }
    ]
}
```


このオプションはコンソールでは利用できません。ただし、Amazon コンソールのインスタンスページを使用して、Auto Scaling グループのインスタンスを終了できます。 EC2これを行うと、Amazon EC2 Auto Scaling はインスタンスが実行されていないことを検出し、ヘルスチェックプロセスの一環として自動的に置き換えます。新しいインスタンスが起動されるまでに、インスタンスを終了してから 1～2 分かかります。インスタンスを終了する方法については、「Amazon EC2ユーザーガイド」の「インスタンスの終了」を参照してください。

グループ内のインスタンスを終了し、アベイラビリティーゾーン間で分散が不均等になる場合、AZRebalanceAmazon EC2 Auto Scaling はプロセスを停止しない限り、グループのバランスを再調整して均等分散を再確立します。詳細については、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。

Amazon EC2 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、予測可能な負荷の変化に基づいてアプリケーションに自動スケーリングを設定できます。特定の時間にグループの希望するキャパシティを増減するスケジュールされたアクションを作成します。

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す。Amazon EC2 Auto Scaling では、このパターンに沿ったスケーリングスケジュールを設定できます。

- 水曜日の朝、Auto Scaling グループの前もって設定した希望するキャパシティを増やすというスケジュールされた 1 つのアクションでキャパシティが増加します。
- 金曜日の夜、Auto Scaling グループの前もって設定した希望するキャパシティを減らすという別のスケジュールされたアクションでキャパシティが減少します。

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キャパシティがありますが、それ以外の時間帯に不要なキャパシティ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すると、スケーリングに対して両方のアプローチの利点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

スケジュールされたスケーリングのしくみ

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。これにより、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に Amazon EC2 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、Auto Scaling グループ、スケーリングアクティビティを実行するタイミング、新しい希望するキャパシティを指定し、必要に応じて、新しい最小キャパシティと新しい最大キャパシティも指定します。スケジュールされたアクションは、一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。

指定された時点で、Amazon EC2 Auto Scaling は、現在の容量を指定された希望する容量と比較することで、新しい容量値に基づいてスケーリングします。

- 現在の容量が指定された希望する容量を下回る場合、Amazon EC2 Auto Scaling は指定された希望する容量にインスタンスをスケールアウトまたは追加します。
- 現在の容量が指定された希望する容量より大きい場合、Amazon EC2 Auto Scaling は指定された希望する容量にインスタンスをスケールインまたは削除します。

スケジュールされたアクションは、指定された日時に、グループの希望するキャパシティ、最小キャパシティ、最大キャパシティを設定します。一度に作成できるスケジュールされたアクションは、これらのキャパシティのいずれか (例えば、希望するキャパシティ) に対してのみです。ただし、最小キャパシティと最大キャパシティを含めてアクションで指定した希望するキャパシティがこれらの制限を超えないようにすることが必要な場合があります。

繰り返しのスケジュール

AWS CLI または を使用して定期的なスケジュールを作成するにはSDK、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かを記述します。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。

を使用して定期的なスケジュールを作成するには AWS Management Console、スケジュールされたアクションの繰り返しパターン、タイムゾーン、開始時刻、およびオプションの終了時刻を指定します。すべての反復パターンオプションは、cron 式に基づいています。または、独自のカスタム cron 式を記述することもできます。

このサポートされた cron 式は、スペースで区切られた 5 つのフィールド ([分] [時間] [日] [月] [曜日]) で構成されます。例えば、cron 式 `30 6 * * 2` は毎週火曜日の午前 6:30 に繰り返される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。cron 式のその他の例については、「」を参照してくださいhttps://crontab.guru/examples.html。この形式で独自の cron 式を記述する方法については、クロンタブを参照してください。

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、Amazon EC2 Auto Scaling はこの時点でアクションを実行し、指定された繰り返し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。スケジュールされたアクションは、終了時刻に達するとアカウントに残りません。
- 繰り返し時間が終了時刻と完全に一致する場合、Amazon EC2 Auto Scaling は終了時刻にスケジュールされたアクションを実行しません。

- AWS CLI または UTCを使用する場合は、開始時刻と終了時刻を で設定する必要がありますSDK。

[Time zone] (タイムゾーン)

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 () ですUTC。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 (DST) を遵守するタイムゾーンを指定すると、アクションは に合わせて自動的に調整されますDST。

有効な値は、Internet Assigned Numbers Authority (IANA) Time Zone データベースのタイムゾーンの正規名です。例えば、米国東部時間を識別する正規名は `America/New_York` です。詳細については、https://www.iana.org/time-zones を参照してください。

などの場所ベースのタイムゾーンは、 に合わせて`America/New_York`自動的に調整されますDST。ただし、 などの UTCベースのタイムゾーン`Etc/UTC`は絶対時間であり、 に合わせて調整されませんDST。

例えば、タイムゾーンが `America/New_York` である繰り返しのスケジュールがります。最初のスケーリングアクションは、 がDST開始する前に`America/New_York`タイムゾーンで行われます。次のスケーリングアクションは、 のDST起動後に`America/New_York`タイムゾーンで実行されます。最初のアクションは現地時間の午前 8 時UTC～5 時に開始され、2 回目のアクションは現地時間の午前 8 時UTC～4 時に開始されます。

を使用してスケジュールされたアクションを作成し、 を観察するタイムゾーンを指定する AWS Management Console とDST、定期的なスケジュールと開始時刻と終了時刻の両方が に合わせて自動的に調整されますDST。

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の実行順序は、それらのアクションが同じグループ内で実行される場合は維持されますが、複数のグループ間で実行される場合は必ずしも維持されません。
- 通常、スケジュールされたアクションは数秒以内に実行されます。ただし、アクションは、スケジュールされた開始時間から最大 2 分遅れる場合があります。Auto Scaling スケジュールされたグループ内のアクションは指定された順序で実行されるため、開始時間が互いに近い、スケジュールされたアクションの実行には時間がかかる可能性があります。

- Auto Scaling グループのスケジュールされたスケーリングを一時的にオフにするには、`ScheduledActions` プロセスを一時的停止します。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、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。
- スケジュールされたアクションを作成した後、名前以外のすべての設定を更新できます。

制限

- スケジュールされたアクションの名前は、Auto Scaling グループごとに一意である必要があります。
- スケジュールされたアクションには、一意の時間値が必要です。別のスケーリングアクティビティがすでにスケジュールされているときにアクティビティをスケジュールしようとすると、呼び出しは拒否され、このスケジュールされた開始時間を持つスケジュールされたアクションがすでに存在していることを示すエラーが返されます。
- 1 つの Auto Scaling グループあたり最大 125 のスケジュールされたアクションを作成できます。

スケジュールされたアクションの作成

Auto Scaling グループにスケジュールされたアクションを作成するには、次のいずれかの方法を使用します。

Console

スケジュールされたアクションを作成するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Automatic scaling (自動スケーリング)] タブの [Scheduled actions (スケジュールされたアクション)] で、[Create scheduled action (スケジュールされたアクションの作成)] を選択します。
4. スケジュールされたアクションに [Name (名前)] を入力します。

5. [希望する容量]、[最小]、[最大] で、グループの新しい希望するキャパシティと、新しい最小キャパシティと最大キャパシティを選択します。希望するキャパシティは、グループの最小サイズ以上、最大サイズ以下である必要があります。
6. [Recurrence (反復)] で、使用可能なオプションの 1 つを選択します。
   - 定期的なスケジュールでスケーリングする場合は、Amazon EC2 Auto Scaling がスケジュールされたアクションを実行する頻度を選択します。
     - [Every (毎)] で始まるオプションを選択した場合、cron 式が作成されます。
     - [Cron] を選択した場合は、いつアクションを実行するかを Cron 式を入力します。
   - スケーリングを 1 回だけ行う場合は、[Once (一度)] を選択します。
7. [Time zone (タイムゾーン)] でタイムゾーンを選択。デフォルト: `Etc/UTC`。

   リストされているタイムゾーンはすべて、IANAタイムゾーンデータベースからのものです。詳細については、https://en.wikipedia.org/wiki/List_of_tz_database_time_zones を参照してください。
8. 特定の開始時間には、以下の日付と時刻を定義します。
   - 定期的なスケジュールを選択した場合、開始時間によって、定期的なシリーズの最初のスケジュールされたアクションが実行されるタイミングが定義されます。
   - [Once (一度)] を反復として使用する場合、開始時刻は、スケジュールアクションを実行する日付と時刻を定義します。
9. (オプション) 定期的なスケジュールの場合は、[Set End Time (終了時刻の設定)] を選択して終了時間を特定し、[End By (までに終了)] に日付と時刻を選択します。
10. [Create] (作成) を選択します。コンソールに Auto Scaling グループのスケジュールされたアクションが表示されます。

AWS CLI

スケジュールされたアクションを作成するには、次のいずれかのコマンド例を使用します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

例: 1 回のみスケールするには

`--start-time "YYYY-MM-DDThh:mm:ssZ"` および `--desired-capacity`オプションを指定して、次の put-scheduled-update-group-action コマンドを使用します。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
  --auto-scaling-group-name my-asg --start-time "2021-03-31T08:00:00Z" --desired-capacity 3
```

例: 定期的なスケーリングをスケジュールするには

`--recurrence "cron expression"` および `--desired-capacity`オプションを指定して、次の put-scheduled-update-group-action コマンドを使用します。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
  --auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

デフォルトでは、Amazon EC2 Auto Scaling はUTCタイムゾーンに基づいて指定された繰り返しスケジュールを実行します。別のタイムゾーンを指定するには、次の例のように、 `--time-zone`オプションとIANAタイムゾーンの名前を含めます。

```
--time-zone "America/New_York"
```

詳細については、https://en.wikipedia.org/wiki/List_of_tz_database_time_zones を参照してください。

スケジュールされたアクションの詳細を表示する

Auto Scaling グループの今後のスケジュールされたアクションの詳細を表示するには、次のいずれかの方法を使用します。

Console

スケジュールされたアクションの詳細を表示するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling スケーリンググループを選択します。
3. [自動スケーリング] タブの [予定されたアクション] セクションで、今後のスケジュールされたアクションを表示できます。

コンソールには、ローカル時間の開始時刻と終了時刻の値が表示され、UTCオフセットは指定された日時で有効であることに注意してください。UTC オフセットは、現地時間から までの時間および分単位の差ですUTC。タイムゾーンの値は、例えば America/New_York のようにリクエストされたタイムゾーンが表示されます。

AWS CLI

次の describe-scheduled-actions コマンドを使用します。

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

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。

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2020-12-01T00:30:00Z",
      "Time": "2020-12-01T00:30:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4
    }
  ]
}
```

スケーリングアクティビティを検証する

スケジュールされたスケーリングに関連するスケーリングアクティビティを検証するには、「Auto Scaling グループのスケーリングアクティビティを検証する」を参照してください。

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を削除するには、次のいずれかの方法を使用します。

Console

スケジュールされたアクションを削除するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling スケーリンググループを選択します。
3. [Automatic scaling (自動スケーリング)] タブの [Scheduled actions (スケジュールされたアクション)] で、スケジュールされたアクションを選択します。
4. [Actions] で、[Delete] を選択します。
5. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

AWS CLI

次の delete-scheduled-action コマンドを使用します。

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \
  --scheduled-action-name my-recurring-action
```

Amazon EC2 Auto Scaling の動的スケーリング

動的スケーリングは、トラフィックの変化に応じて Auto Scaling グループのキャパシティをスケールします。

Amazon EC2 Auto Scaling は、次のタイプの動的スケーリングポリシーをサポートしています。

- ターゲット追跡スケーリング - Amazon CloudWatch メトリクスとターゲット値に基づいて、 グループの現在の容量を増減します。これはサーモスタットで家の温度を管理する方法と似ています (温度を選択すれば、後はサーモスタットがすべてを実行します)。
- ステップスケーリング - アラーム違反の規模に応じて変動する一連のスケーリング調整値 (ステップ調整値) に基づいて、グループの現在のキャパシティを増減させます。
- シンプルなスケーリング - 1 つのスケーリング調整値に基づいて、各スケーリングアクティビティ間のクールダウン期間により、グループの現在のキャパシティを増減させます。

ターゲット追跡スケーリングポリシーを使用し、Auto Scaling グループのキャパシティの変化に反比例して変化するメトリクスを選択することを強くお勧めします。つまり、Auto Scaling グループのサ

イズを 2 倍にすると、メトリクスが 50% 減少するようにします。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。ターゲットあたりの平均CPU使用率や平均リクエスト数などのメトリクスが含まれます。

ターゲット追跡により、Auto Scaling グループは、アプリケーションの実際の負荷に正比例してスケールします。つまり、ターゲット追跡ポリシーは、負荷の変化に対応するキャパシティの緊急のニーズを満たすだけでなく、季節変動などにより経時的に発生する負荷の変化にも対応できます。

ターゲット追跡ポリシーにより、 CloudWatch アラームとスケーリング調整を手動で定義する必要もなくなります。Amazon EC2 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。

内容

動的スケーリングポリシーが機能する方法

動的スケーリングポリシーは、特定の CloudWatch メトリクスを追跡するように Amazon EC2 Auto Scaling に指示し、関連付けられた CloudWatch アラームが にあるときに実行するアクションを定義しますALARM。アラームの状態の呼び出しに使用されるメトリクスは、Auto Scaling グループのすべてのインスタンスから取得されるメトリクスの集約です。 (例えば、2 つのインスタンスを持つ Auto Scaling グループがあるとCPUします。1 つのインスタンスは 60%、もう 1 つのインスタンスは 40% ですCPU。 平均すると、50% です) CPU。ポリシーが有効な場合、Amazon EC2 Auto Scaling はアラームのしきい値を超えたときにグループの希望する容量を調整します。

動的スケーリングポリシーが呼び出されると、容量計算によってグループの最小サイズと最大サイズ範囲外の数値が生成された場合、Amazon EC2 Auto Scaling は新しい容量が最小サイズと最大サイ

ズの制限を超えないようにします。キャパシティは、次の 2 つの方法のいずれかで測定されます。1 つは、インスタンスに関して希望するキャパシティを設定するときに選択したものと同じユニットを使用する方法で、もう 1 つはキャパシティユニット (インスタンスの重み付けが適用されている場合) を使用する方法です。

- 例 1: Auto Scaling グループの最大キャパシティーは 3、現在のキャパシティーは 2、および 3 つのインスタンスを追加する動的スケーリングポリシーがあります。このポリシーを呼び出すと、Amazon EC2 Auto Scaling はグループに 1 つのインスタンスのみを追加して、グループが最大サイズを超えないようにします。
- 例 2: Auto Scaling グループの最小キャパシティーは 2、現在のキャパシティーは 3、および 2 つのインスタンスを削除する動的スケーリングポリシーがあります。このポリシーを呼び出すと、Amazon EC2 Auto Scaling はグループから 1 つのインスタンスのみを削除し、グループが最小サイズを下回ることを防ぎます。

希望するキャパシティーが最大サイズ制限に達すると、スケールアウトは停止します。需要が減り、容量が減少した場合、Amazon EC2 Auto Scaling は再びスケールアウトできます。

インスタンスの重み付けを使用する場合は例外です。この場合、Amazon EC2 Auto Scaling は最大サイズ制限を超えてスケールアウトできますが、インスタンスの最大重みまでスケールアウトできます。その意図は、できるだけ新しい希望するキャパシティーに近づけることですが、それでもグループのために指定されている割り当て戦略を遵守することです。割り当て戦略によって、起動するインスタンスタイプを決定します。重みは、インスタンスタイプに基づいて、各インスタンスがグループの希望するキャパシティーに影響するキャパシティーユニット数を決定します。

- 例 3: Auto Scaling グループの最大キャパシティーは 12、現在のキャパシティーは 10、および 5 つのキャパシティーユニットを追加する動的スケーリングポリシーがあります。インスタンスタイプには、1、4、6 の 3 つの重みのいずれかが割り当てられます。ポリシーを呼び出すとき、Amazon EC2 Auto Scaling は、配分戦略に基づいて重みが 6 のインスタンスタイプを起動することを選択します。このスケールアウトイベントの結果は、希望するキャパシティーが 12 で、現在のキャパシティーが 16 のグループになります。

複数の動的スケーリングポリシー

ほとんどの場合、ターゲット追跡スケーリングポリシーは、自動的にスケールアウトとスケールインするように Auto Scaling グループからインスタンスをデタッチするには、以下の手順を使用します。 グループを設定するのに十分です。ターゲット追跡スケーリングポリシーでは、希望する結果

を選択したら、その結果を達成するために、必要に応じて Auto Scaling グループに対してインスタンスを追加および削除できます。

高度なスケーリング設定では、Auto Scaling グループに複数のスケーリングポリシーを設定できます。例えば、1 つ以上のターゲット追跡スケーリングポリシー、1 つ以上のステップスケーリングポリシー、またはその両方を定義できます。これにより、複数のシナリオに対応できる柔軟性が高まります。

複数の動的スケーリングポリシーがどのように連携するかを説明するために、Auto Scaling グループと Amazon SQSキューを使用して単一のEC2インスタンスにリクエストを送信するアプリケーションを検討してください。アプリケーションのパフォーマンスを最適なレベルに維持するために、Auto Scaling グループをスケールアウトするタイミングをコントロールする 2 つのポリシーがあるとします。1 つは、カスタムメトリクスを使用してキュー内のSQSメッセージ数に基づいて容量を追加および削除するターゲット追跡ポリシーです。もう 1 つのステップスケーリングポリシーは、Amazon CloudWatch `CPUUtilization` メトリクスを使用して、インスタンスが指定された期間に 90% の使用率を超えたときに容量を追加するものです。

同時に有効なポリシーが複数ある場合は、各ポリシーで、Auto Scaling グループが同時にスケールアウト (またはスケールイン) するように指定している可能性があります。例えば、SQSカスタム`CPUUtilization`メトリクスがスパイクして CloudWatch アラームのしきい値を超過すると同時に、カスタムメトリクスアラームのしきい値をスパイクして超過する可能性があります。

このような状況が発生すると、Amazon EC2 Auto Scaling はスケールアウトとスケールインの両方に最大の容量を提供するポリシーを選択します。たとえば、 のポリシーが 1 つのインスタンス`CPUUtilization`を起動し、SQSキューのポリシーが 2 つのインスタンスを起動するとします。両方のポリシーのスケールアウト条件が同時に満たされた場合、Amazon EC2 Auto Scaling はSQSキューポリシーに優先します。これにより、Auto Scaling グループは 2 つのインスタンスを起動することになります。

ポリシーがスケールインに異なる基準を使用する場合でも、最大キャパシティーを提供するポリシーを優先するというアプローチが適用されます。例えば、1 つのポリシーが 3 つのインスタンスを終了し、別のポリシーがインスタンス数を 25% 減らし、スケールイン時にグループに 8 つのインスタンスがある場合、Amazon EC2 Auto Scaling はグループに対してインスタンスの最大数を提供するポリシーを優先します。その結果、Auto Scaling グループは 2 つのインスタンスを終了します (8 の 25% = 2)。目的は、Amazon EC2 Auto Scaling が多数のインスタンスを削除できないようにすることです。

ただし、ターゲット追跡スケーリングポリシーをステップスケーリングポリシーとともに使用する場合、ポリシー間の競合によって望ましくない動作が生じる可能性があるため、注意することをお勧め

します。例えば、ターゲット追跡ポリシーがスケールインする準備が整う前に、ステップスケーリングポリシーがスケールインアクティビティを開始した場合、スケールインアクティビティはブロックされません。スケールインアクティビティが完了した後で、ターゲット追跡ポリシーにより、グループに再びスケールアウトするよう指示できます。

Amazon EC2 Auto Scaling のターゲット追跡スケーリングポリシー

ターゲット追跡スケーリングポリシーは、ターゲットメトリクス値に基づいて Auto Scaling グループのキャパシティを自動的にスケールします。個々のアプリケーションの一意の使用パターンに自動的に適応します。これにより、アプリケーションはEC2インスタンスの最適なパフォーマンスと高い使用率を維持し、手動による介入なしでコスト効率を向上させることができます。

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。Amazon EC2 Auto Scaling は、メトリクスがターゲットから逸脱したときにスケーリングイベントを呼び出す CloudWatch アラームを作成および管理します。例として、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 2 つのインスタンス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 Auto Scaling グループのCPU使用率を約 50% に維持したいとします。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、平均CPU使用率 50% をターゲットとするターゲット追跡スケーリングポリシーを作成します。その後、 が 50% CPUを超えると、Auto Scaling グループはスケールアウトするか、容量を増やして、増加した負荷を処理します。CPU が 50% を下回ると、使用率が低い期間にコストを最適化するために容量をスケールインまたは縮小します。

トピック

複数のターゲット追跡スケーリングポリシー

スケーリングパフォーマンスを最適化するために複数のターゲット追跡スケーリングポリシーを同時に使用できますが、それぞれが異なるメトリクスを使用する必要があります。例えば、使用率とスループットは互いに影響し合う可能性があります。これは、これらのメトリクスのいずれかが変更されるたびに、通常、他のメトリクスも影響を受けることを意味します。このため、複数のメトリクスを使用することで、Auto Scaling グループの負荷に関する追加情報が提供されます。これにより、Amazon EC2 Auto Scaling は、グループに追加する容量を決定する際に、より多くの情報に基づいた意思決定を行うことができます。

Amazon EC2 Auto Scaling の目的は、常に可用性を優先することです。ターゲット追跡ポリシーのいずれかでスケールアウトが必要な場合、Auto Scaling グループがスケールアウトされます。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な状態) でスケールインできる場合にのみスケールインされます。

メトリクスを選択する

事前定義されたメトリクスまたはカスタムメトリクスのいずれかを使用して、ターゲット追跡スケーリングポリシーを作成できます。事前定義されたメトリクスを使用すると、スケーリングに最もよく使用されるメトリクスに簡単にアクセスできます。カスタムメトリクスを使用すると、数秒ほどでより細かい間隔で公開される高解像度 CloudWatch メトリクスなど、利用可能な他のメトリクスをスケールできます。独自の高解像度メトリクスや、他の AWS のサービスが公開するメトリクスを公開できます。

高解像度メトリクスを使用したターゲット追跡ポリシーの作成の詳細については、「」を参照してください高解像度メトリクスを使用してターゲット追跡ポリシーを作成し、応答を高速化する。

ターゲット追跡は、以下の事前定義されたメトリクスをサポートしています。

- `ASGAverageCPUUtilization`— Auto Scaling グループの平均CPU使用率。
- `ASGAverageNetworkIn` - すべてのネットワークインターフェイスで Auto Scaling グループが受信した平均バイト数。
- `ASGAverageNetworkOut` - すべてのネットワークインターフェイスで Auto Scaling グループが送信した平均バイト数。
- `ALBRequestCountPerTarget` — Auto Scaling グループのターゲットあたりの Application Load Balancer リクエストの平均。

> **Important**
>
> ターゲットあたりのCPU使用率、ネットワーク I/O、Application Load Balancer リクエスト数のメトリクスに関するその他の貴重な情報は、Amazon EC2ユーザーガイドのインスタンスで利用可能な CloudWatch メトリクスのリストトピックと、CloudWatch Application Load Balancer ユーザーガイドの Application Load Balancer トピックのメトリクスにあります。

カスタム CloudWatch メトリクスを指定 CloudWatch することで、 で他の使用可能なメトリクスまたは独自のメトリクスを選択できます。を使用してターゲット追跡スケーリングポリシーのカスタマイズされたメトリクス仕様を指定する例については AWS CLI、「」を参照してくださいAWS CLIのスケーリングポリシーの例。

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。

- 使用率の変化に応じて迅速にスケーリングできるように、1 分以下の間隔で利用可能なメトリクスのみを使用することをお勧めします。低い間隔で公開されるメトリクスにより、ターゲット追跡ポリシーは Auto Scaling グループの使用の変化をより迅速に検出して対応できます。
- CPU 使用率などEC2、Amazon によって公開される事前定義されたメトリクスを選択する場合は、詳細モニタリングを有効にすることをお勧めします。デフォルトでは、すべての Amazon EC2メトリクスは 5 分間隔で発行されますが、詳細モニタリングを有効にすることで 1 分間隔で設定できます。詳細モニタリングを有効にする方法については、「」を参照してくださいAuto Scaling インスタンスのモニタリングを設定する。
- 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、インスタンスの使用頻度を示す必要があります。メトリクス値は Auto Scaling グループのインスタンス数に比例して増減する必要があります。それにより、メトリクスデータを使用して比例的にインスタンス数をスケールアウトまたはスケールインできます。例えば、Auto Scaling グループの負荷がインスタンス全体に分散されている場合、Auto Scaling グループのCPU使用率 (メトリクスディメンション EC2`CPUUtilization`を持つ Amazon メトリクス`AutoScalingGroupName`) は機能します。
- 以下のメトリクスはターゲットの追跡では機能しません。
  - Auto Scaling グループの前にあるロードバランサーが受信したリクエスト数 (つまり、Elastic Load Balancing メトリクス`RequestCount`)。ロードバランサーによって受信されたリクエストの数は、Auto Scaling グループの使用率に基づいて変化しません。

  - ロードバランサーのリクエストのレイテンシー (Elastic Load Balancing メトリクス`Latency`)。リクエストのレイテンシーは、使用率の増加により増える場合がありますが、必ずしも比例して変化するわけではありません。
  - CloudWatch Amazon SQSキューメトリクス `ApproximateNumberOfMessagesVisible`。キュー内のメッセージ数は、キューからのメッセージを処理する Auto Scaling グループのサイズに比例して変わらない可能性があるためです。ただし、Auto Scaling グループのEC2インスタンスごとにキュー内のメッセージ数を測定するカスタムメトリクスは機能します。詳細については、「Amazon に基づくスケーリングポリシー SQS」を参照してください。
- `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ロードバランサーターゲットグループを識別する必要があります。を使用してターゲット追跡スケーリングポリシーの `ResourceLabel`パラメータを指定する例については AWS CLI、「」を参照してくださいAWS CLIのスケーリングポリシーの例。
- メトリクスが実際の 0 の値を に出力する場合 CloudWatch （ 例: `ALBRequestCountPerTarget` ）、Auto Scaling グループは、アプリケーションへのトラフィックが長時間続かない場合に 0 にスケールインできます。Auto Scaling グループにリクエストがルーティングされないときに Auto Scaling グループを 0 にスケールインするには、グループの最小キャパシティを 0 に設定する必要があります。
- スケーリングポリシーで使用する新しいメトリクスを公開する代わりに、メトリクス数式を使用して既存のメトリクスを組み合わせることができます。詳細については、「Metric Math を使用してターゲット追跡スケーリングポリシーを作成する」を参照してください。

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、Auto Scaling グループ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、ターゲット値は単一のインスタンスからの最適な 1 分間の平均スループットを表します。

インスタンスウォームアップ時間を定義する

オプションで、新しく起動されたインスタンスのウォームアップ時間を秒単位で指定できます。指定されたウォームアップ時間が終了するまで、インスタンスは Auto Scaling グループの集計EC2インスタンスメトリクスにカウントされません。

インスタンスのウォームアップ期間中、スケーリングポリシーは、ウォームアップしていないインスタンスからのメトリクス値がポリシーのターゲット使用率を超える場合にのみスケールアウトします。

グループが再度スケールアウトする場合、まだウォームアップ中のインスタンスは、次のスケールアウトアクティビティの希望キャパシティーの一部として計上されます。その目的は、スケールアウトを継続的に (ただし過剰になることなく) 行うことです。

スケールアウトアクティビティの進行中は、インスタンスがウォームアップを終了するまで、スケーリングポリシーによって開始されるすべてのスケールインアクティビティがブロックされます。インスタンスのウォームアップが完了し、スケールインイベントが発生した場合、現在終了処理中のインスタンスは、新しい希望するキャパシティを計算する際にグループの現在のキャパシティにカウントされます。したがって、Auto Scaling グループから必要以上の数のインスタンスが削除されることがなくなります。

デフォルト値

値が設定されていない場合、スケーリングポリシーは、グループに定義されているデフォルトのインスタンスウォームアップの値であるデフォルト値を使用します。インスタンスのデフォルトウォームアップが null の場合、デフォルトクールダウンの値にフォールバックします。ウォームアップ時間が変更されたときにすべてのスケーリングポリシーを簡単に更新できるように、デフォルトのインスタンスウォームアップを使用することをお勧めします。

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。Amazon EC2 Auto Scaling は、 CloudWatchターゲット追跡スケーリングポリシーに関連付けられたアラームを作成および管理し、必要に応じてアラームを編集、置換、または削除して、アプリケーションのスケーリングエクスペリエンスと、その変化する利用パターンをカスタマイズできます。
- ターゲット追跡スケーリングポリシーは、トラフィック減少時に徐々にスケールインすることにより、トラフィックレベルが変動する期間中の可用性を優先します。より詳細な制御が必要な場合

は、ステップスケーリングポリシーが適しています。ターゲット追跡ポリシーのスケールイン部分を一時的に無効にできます。これにより、デプロイを成功させるためのインスタンスの最小数を維持できます。

- 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 に変わりますINSUFFICIENT_DATA。この場合、Amazon EC2 Auto Scaling は新しいデータポイントが見つかるまでグループをスケーリングできません。
- 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、追加または削除するインスタンス数を決定するときに、その数を切り上げまたは切り捨てて常に控えめに動作するためです。これにより、不十分な数のインスタンスを追加したり、インスタンスを過剰に削除したりすることがなくなります。ただし、インスタンス数が少ない、より小さな Auto Scaling グループでは、グループの使用率はターゲット値からかなり離れているように見える場合があります。例えば、CPU使用率にターゲット値を 50% に設定し、Auto Scaling グループがターゲットを超えたとします。1.5 インスタンスを追加すると、CPU使用率が 50% に近づくと判断される場合があります。1.5 インスタンスを追加することはできないので、これを切り上げて、2 インスタンスを追加します。これにより、CPU使用率が 50% を下回る場合がありますが、アプリケーションがそれをサポートするのに十分なリソースを確保できます。同様に、1.5 インスタンスを削除するとCPU使用率が 50% を超えると判断した場合、1 つのインスタンスだけが削除されます。

  より多くのインスタンスのある大規模な Auto Scaling グループの場合、使用率はより多くのインスタンス間で分散されます。その場合、インスタンスの追加または削除により、ターゲット値と実際のメトリクスデータポイントとの差が小さくなります。
- 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、Auto Scaling グループをスケールアウトする必要があるとみなされます。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、ターゲット追跡スケーリングポリシーを使用して Auto Scaling グループをスケールアウトすることはできません。

ターゲット追跡スケーリングポリシーを作成する

Auto Scaling グループ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、次のいずれかの方法を使用します。

開始する前に、希望するメトリクスが 1 分間隔で使用可能であることを確認します (Amazon EC2メトリクスのデフォルトの 5 分間隔と比較して)。

Console

新しい Auto Scaling グループにターゲット追跡スケーリングポリシーを作成するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. [Auto Scaling グループの作成] を選択します。
3. ステップ 1、2、および 3 で、必要に応じてオプションを選択し、「ステップ 4: グループサイズとスケーリングポリシーを設定する」に進みます。
4. [スケーリング] で、[最小の希望する容量] と [最大の希望する容量] を更新してスケールする範囲を指定します。これら 2 つの設定により、Auto Scaling グループは動的にスケーリングできます。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。
5. [自動スケーリング] で、[ターゲット追跡スケーリングポリシー] を選択します。
6. ポリシーを定義するには、以下の作業を行います。

   a. ポリシーの名前を指定します。

   b. [Metric type (メトリクスタイプ)] でメトリクスを選択します。

      [Application Load Balancer request count per target (ターゲットあたりのApplication Load Balancer リクエスト数)] を選択し、[Target group (ターゲットグループ)] のターゲットグループを選択します。

   c. メトリクスの [Target value] を指定します。

   d. (オプション) [インスタンスのウォームアップ] で、必要に応じてインスタンスのウォームアップ値を更新します。

   e. (オプション) [Disable scale in to create only a scale-out policy (スケールインを無効にしてスケールアウトポリシーのみを作成する)] を選択します。これにより、必要に応じて異なるタイプの別のスケールインポリシーを作成できます。

7. Auto Scaling グループの作成に進みます。スケーリングポリシーは、Auto Scaling グループの作成後に作成されます。

既存の Auto Scaling グループにターゲット追跡スケーリングポリシーを作成するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. スケーリング制限が適切に設定されていることを検証します。例えば、グループの希望するキャパシティが既に最大に達している場合は、スケールアウトするために新しい最大を指定する必要があります。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。
4. [Automatic scaling] (オートスケーリング) タブの [Dynamic scaling policies] (動的スケーリングポリシー) で、[Create dynamic scaling] (動的スケーリングポリシーの作成) を選択します。
5. ポリシーを定義するには、以下の作業を行います。

   a. [ポリシータイプ] で、[ターゲット追跡スケーリング] のデフォルト設定を維持します。

   b. ポリシーの名前を指定します。

   c. [Metric type (メトリクスタイプ)] でメトリクスを選択します。選択できるメトリクスタイプは 1 つだけです。複数のメトリクスを使用するには、複数のポリシーを作成します。

      [Application Load Balancer request count per target (ターゲットあたりのApplication Load Balancer リクエスト数)] を選択し、[Target group (ターゲットグループ)] のターゲットグループを選択します。

   d. メトリクスの [Target value] を指定します。

   e. (オプション) [インスタンスのウォームアップ] で、必要に応じてインスタンスのウォームアップ値を更新します。

   f. (オプション) [Disable scale in to create only a scale-out policy (スケールインを無効にしてスケールアウトポリシーのみを作成する)] を選択します。これにより、必要に応じて異なるタイプの別のスケールインポリシーを作成できます。
6. [Create] (作成) を選択します。

AWS CLI

ターゲット追跡スケーリングポリシーを作成するには、次の例を使用して開始できます。*user input placeholder* を、ユーザー自身の情報に置き換えます。

> **Note**
>
> その他の例については、「AWS CLIのスケーリングポリシーの例」を参照してください。

ターゲット追跡スケーリングポリシーを作成するには (AWS CLI)

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の という名前config.json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": "ASGAverageCPUUtilization"
    }
}
```

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

2. コマンドput-scaling-policyと前のステップで作成した config.json ファイルを使用して、スケーリングポリシーを作成します。

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-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 の名前を返します。

```
{
```

```
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu50-target-tracking-
scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

高解像度メトリクスを使用してターゲット追跡ポリシーを作成し、応答を高速化する

ターゲット追跡は、1 分未満の間隔で公開される秒レベルのデータポイントを含む高解像度 CloudWatch メトリクスをサポートします。ターゲット追跡ポリシーを設定して、クライアントサービス、ライブストリーミングサービス、e コマースウェブサイト、オンデマンドデータ処理など APIs、需要パターンが変動するアプリケーションの高解像度 CloudWatch メトリクスを通じて使用率をモニタリングします。キャパシティと需要の一致精度を高めるために、ターゲット追跡では、このきめ細かなモニタリングを使用して、EC2インスタンスの需要と使用率の変化をより迅速に検出して対応します。

高解像度でメトリクスを発行する方法の詳細については、「Amazon CloudWatch ユーザーガイド」の「カスタムメトリクスの発行」を参照してください。高解像度でのCPU使用率などのEC2メトリクスにアクセスして公開するには、 CloudWatch エージェントを使用することをお勧めします。

AWS リージョン

高解像度メトリクスを使用したターゲット追跡は、 AWS リージョン を除くすべての で使用できます AWS GovCloud (US) Regions。

高解像度メトリクスを使用したターゲット追跡ポリシーの仕組み

追跡するメトリクスと、メトリクスに維持するターゲット値を定義して、ターゲット追跡ポリシーを作成します。高解像度メトリクスでスケールするには、メトリクスの名前を指定し、ターゲット追跡がこのメトリクスを監視するメトリクス期間を 60 秒未満の値に設定します。現在サポートされている最小間隔は 10 秒です。これより低い間隔でメトリクスを発行できます。

> **Note**
>
> 60 を超えるメトリクス期間はサポートされていません。

1 つの CloudWatch メトリクスでターゲット追跡を設定したり、複数の CloudWatch メトリクスをクエリしたり、数式を使用してこれらのメトリクスに基づいて新しい 1 つの時系列を作成したりできます。どちらのオプションでも、メトリクス期間を定義できます。

例

例 1

次の例では、高解像度 CloudWatch メトリクスに基づいてターゲット追跡ポリシーを作成します。メトリクスは 10 秒の解像度で発行されます。期間を定義することで、ターゲット追跡を有効にして、このメトリクスを 10 秒の精度でモニタリングできます。*user input placeholder* を、ユーザー自身の情報に置き換えます。

```
$ cat ~/config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
      "MetricName": "MyHighResolutionMetric",
      "Namespace": "MyNamespace",
      "Dimensions": [
        {
          "Name": "MyOptionalDimensionName",
          "Value": "MyOptionalMetricDimensionValue"
        }
      ],
      "Statistic": "Average",
      "Unit": "None"
      "Period": "10
  }
```

```
}
```

例 2

Metric Math 式を使用すると、複数のメトリクスを 1 つの時系列に結合してスケーリングできます。Metric Math は、既存のメトリクスをインスタンスごとの平均に変換するために特に便利です。ターゲット追跡では、メトリクスが Auto Scaling グループの容量に反比例していることを前提としているため、メトリクスの変換は不可欠です。したがって、容量が増加すると、メトリクスはほぼ同じ比率で減少します。

例えば、アプリケーションで処理される保留中のジョブを表すメトリクスがあるとします。Metric Math を使用して、保留中のジョブを Auto Scaling グループの実行中の容量で割ります。Auto Scaling は容量メトリクスを 1 分単位で発行するため、このメトリクスには分未満の間隔の値はありません。スケーリングにより高い解像度を使用する場合、容量と保留中のジョブメトリクスの期間が一致しなくなる可能性があります。この不一致を回避するには、 FILL式を使用して、欠落した値を前の 1 分間のタイムスタンプに記録された容量番号で埋めることをお勧めします。

次の例では、Metric Math を使用して、保留中のジョブメトリクスを容量で割ります。期間では、両方のメトリクスを 10 秒に設定します。メトリクスは 1 分間隔で発行されるため、キャパシティメトリクスに対して FILLオペレーションを使用しています。

Metric Math を使用して複数のメトリクスを変更するには

```
{
    "CustomizedMetricSpecification": {
        "Metrics": [
            {
                "Label": "Pending jobs to be processed",
                "Id": "m1",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "MyPendingJobsMetric",
                        "Namespace": "Custom",
                    },
                    "Stat": "Sum"
                    "Period": 10
                },
                "ReturnData": false
            },
            {
                "Label": "Get the running instance capacity (matching the period to
 that of the m1)",
```

```
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "GroupInService",
                        "Namespace": "AWS/AutoScaling",
                        "Dimensions": [
                            {
                                "Name": "AutoScalingGroupName",
                                "Value": "my-asg"
                            }
                        ]
                    },
                    "Stat": "Average"
                    "Period": 10
                },
                "ReturnData": false
            },
            {
                "Label": "Calculate the pending job per capacity (note the use of the
 FILL expression)",
                "Id": "e1",
                "Expression": "m1 / FILL(m2,REPEAT)",
                "ReturnData": true
            }
        ]
    },
    "TargetValue": 100
}
```

考慮事項

ターゲット追跡と高解像度メトリクスを使用する場合は、次の点を考慮してください。

- 望ましくない自動スケーリング結果につながる可能性のあるデータポイントが欠落していないことを確認するには、指定した期間と同じかそれ以上の解像度で CloudWatch メトリクスを発行する必要があります。
- ターゲット値を Auto Scaling グループで維持するメトリクス値として per-instance-per-minute定義します。メトリクスの期間に基づいて値が乗算できるメトリクスを使用する場合は、適切なターゲット値を設定することが重要です。例えば、 SUM 統計を使用するリクエスト数や保留中のジョブなどのカウントベースのメトリクスは、選択した期間に応じて異なるメトリクス値を持ちます。それでも、1 分あたりの平均に対してターゲットを設定していることを前提とする必要があります。

- Amazon EC2 Auto Scaling の使用には追加料金はかかりませんが、Amazon EC2インスタンス、CloudWatch メトリクス、 CloudWatch アラームなどのリソースに対して料金を支払う必要があります。前の例で作成した高解像度アラームの料金は、標準 CloudWatch アラームとは異なります。CloudWatch 料金の詳細については、「Amazon CloudWatch の料金」を参照してください。
- ターゲット追跡では、メトリクスがインスタンスのインスタンスごとの平均使用率を表す必要がありますEC2。これを実現するには、ターゲット追跡ポリシー設定の一部として Metric Math オペレーションを使用できます。メトリクスを Auto Scaling グループの実行中の容量で割ります。単一の時系列の作成に使用するメトリクスごとに、同じメトリクス期間が定義されていることを確認してください。これらのメトリクスが異なる間隔で発行される場合は、間隔が大きいメトリクスのFILLオペレーションを使用して、欠落しているデータポイントを埋めます。

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

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 CloudWatch ユーザーガイド」の「Metric Math の使用」を参照してください。

Metric Math の数式には、次の考慮事項が適用されます。

- 使用可能な CloudWatch メトリクスをクエリできます。各メトリクスは、メトリクス名、名前空間、0 以上のディメンションの一意の組み合わせです。
- 算術演算子 (+ - * / ^)、統計関数 (AVGや などSUM) 、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 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API。

例: インスタンスあたりの Amazon SQSキューバックログ

インスタンスごとに Amazon SQSキューのバックログを計算するには、キューから取得できるメッセージのおおよその数を取得し、その数を Auto Scaling グループの実行中の容量で割ります。これは、 `InService`状態のインスタンスの数です。詳細については、「Amazon に基づくスケーリングポリシー SQS」を参照してください。

この数式の口ジックは次のとおりです。

```
sum of (number of messages in the queue)/(number of InService instances)
```

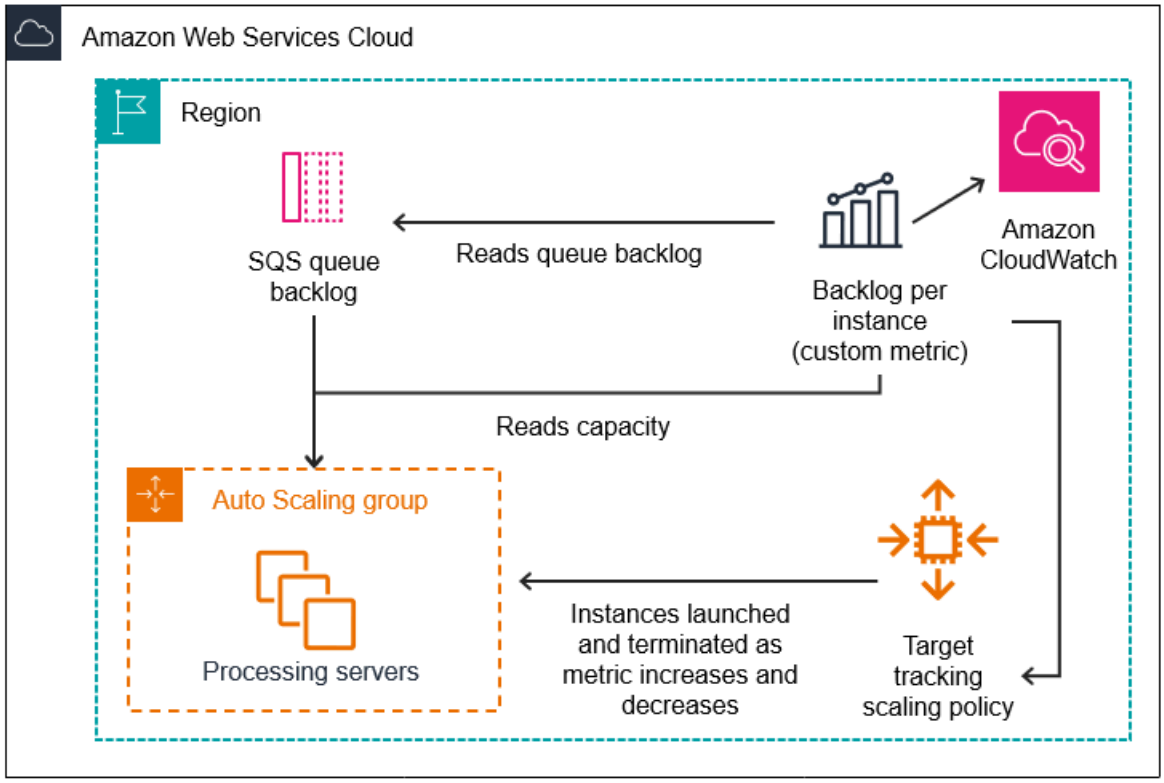
次に、 CloudWatch メトリクス情報は次のとおりです。

ID	CloudWatch メトリクス	統計	間隔
m1	ApproximateNumberOfMessagesVisible	合計	1 分
m2	GroupInServiceInstances	[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)`

このメトリクスのアーキテクチャを以下に図で示します。



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

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

   次の例を参考にして開始してください。*user input placeholder* を、ユーザー自身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 group size (the number of InService instances)",
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "GroupInServiceInstances",
                        "Namespace": "AWS/AutoScaling",
                        "Dimensions": [
                            {
                                "Name": "AutoScalingGroupName",
                                "Value": "my-asg"
                            }
                        ]
                    },
                    "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、「Amazon EC2 Auto Scaling APIリファレンスTargetTrackingConfiguration」の「」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 CloudWatch情報の検索に役立つ追加のリソースです。
>
> - サービスで AWS 使用可能なメトリクスの詳細については、「Amazon CloudWatch ユーザーガイド」のAWS 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。
> - で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、次の例に示すように、 JSON ファイルを入力として使用して put-scaling-policy コマンドを実行します。

```
aws autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-
scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-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 つの CloudWatch アラームARNsの を返します。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/sqs-backlog-target-
tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
```

```
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

> **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。

Amazon EC2 Auto Scaling のステップスケーリングポリシーと簡易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簡易スケーリングポリシーは、 CloudWatch アラームに基づいて Auto Scaling グループの容量を事前定義された増分でスケーリングします。アラームのしきい値を超えると、スケールアウト (容量の増加) とスケールイン (キャパシティの減少) を処理するスケーリングポリシーを個別に定義できます。

ステップスケーリングと簡易スケーリングでは、スケーリングプロセスを呼び出す CloudWatch アラームを作成および管理します。アラームに違反すると、Amazon EC2 Auto Scaling はそのアラームに関連付けられたスケーリングポリシーを開始します。

ターゲット追跡スケーリングポリシーを使用して、ターゲットあたりの平均CPU使用率や平均リクエスト数などのメトリクスをスケールすることを強くお勧めします。キャパシティーが増加すると減少し、キャパシティーが減少すると増加するメトリクスを使用すると、ターゲット追跡を使用してインスタンス数を比例的にスケールアウトしたり、インスタンス数を増やすことができます。これにより、Amazon EC2 Auto Scaling がアプリケーションの需要曲線に厳密に従うようになります。詳細については、「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。

内容

- スケールアウト用のステップスケーリングポリシーを作成する
- スケールイン用のステップスケーリングポリシーを作成する
- 簡易スケーリングポリシー

ステップスケーリングポリシーのしくみ

ステップスケーリングを使用するには、まず Auto Scaling グループ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。次に、アラームしきい値を超えたときにグループをスケールする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ます。

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

- アラームメトリクスが 60% に達したら、10 インスタンスの増分で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 インスタンスの増分で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 インスタンスの増分でスケールアウトする

アラームのしきい値が指定された評価期間数を超えると、Amazon EC2 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。

各インスタンスにはウォームアップ期間があり、スケーリングアクティビティが短期間に発生する変更に対して過剰に反応しないようにします。オプションでスケーリングポリシーのウォームアップ期間を設定できます。ただし、ウォームアップ時間が変更されたときにすべてのスケーリングポリシーを簡単に更新できるように、デフォルトのインスタンスウォームアップを使用することをお勧めします。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

シンプルスケーリングポリシーは、ステップスケーリングポリシーに似ています。ただし、単一のスケーリング調整に基づいており、各スケーリングアクティビティ間にクールダウン期間があります。詳細については、「簡易スケーリングポリシー」を参照してください。

ステップスケーリングのステップ調整

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インスタンス数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ている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。Amazon EC2 Auto Scaling は、 CloudWatch (raw 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集約タイプを適用します。ステップ調整によって定義された上限と下限に対して、この集約メトリクス値を比較することにより、実行するステップ調整が決定されます。

違反しきい値に比例して上限と下限を指定します。例えば、メトリクスが 50% を超えたときに 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。各ポリシーに対して、調整タイプ `PercentChangeInCapacity` (またはコンソールの [グループの割合]) の一連のステップ調整を次のように行いました。

例: スケールアウトポリシーのステップ調整値

下限	上限	調整
0	10	0
10	20	10
20	null	30

例: スケールインポリシーのステップ調整値

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

これにより、次のスケーリング設定が作成されます。

```
Metric value

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

ここで、現在のキャパシティと希望するキャパシティがともに 10 の Auto Scaling グループ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、グループの希望キャパシティーと現在のキャパシティー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、現在のキャパシティーと必要なキャパシティーが維持されます。
- メトリクス値が 60 になった場合、スケールアウトのポリシーの 2 番目のステップ調整値 (10 インスタンスの 10% を追加) に基づいて、グループの必要なキャパシティーを 1 インスタンスだけ増やして、11 インスタンスにします。新しいインスタンスが実行され、その指定されたウォームアップ時間が終了した後、グループの現在のキャパシティは 11 インスタンスに増加します。このキャパシティーの増加後も、メトリクス値が 70 に上昇すると、グループの希望するキャパシティーはさらに 3 インスタンス増加し、14 インスタンスになります。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に基づきます (11 インスタンスの 30% である 3.3 インスタンスを 3 インスタンスに切り捨て、追加します)。
- メトリクス値が 40 になった場合、スケールインポリシーの 2 番目のステップ調整値 (14 インスタンスの 10%、つまり 1.4 インスタンスを丸めた 1 インスタンスを削除) に基づいて、グループの必要なキャパシティーを 1 インスタンスだけ減らして、13 インスタンスにします。このキャパシティーが減少した後も、メトリクス値が 30 に下がると、グループの希望するキャパシティーはさらに 3 インスタンス減少し、10 インスタンスになります。これは、スケールインポリシーの 3 番目のステップ調整に基づきます (13 インスタンスの 30% である 3.9 インスタンスを 3 インスタンスに切り捨て、減算します)。

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。

- を使用する場合は AWS Management Console、上限と下限を絶対値として指定します。 AWS CLI または を使用する場合はSDK、違反しきい値に対する上限と下限を指定します。
- 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、Auto Scaling グループの現在のキャパシティーに対する割合、またはキャパシティーユニットで指定できます。通常、キャパシティユニットは、インスタンスの重み付け機能を使用していない限り、1 つのインスタンスを意味します。

Amazon EC2 Auto Scaling は、ステップスケーリングと簡易スケーリングに対して次の調整タイプをサポートしています。

- `ChangeInCapacity` - 指定した値だけ現在のキャパシティーを増減させます。正の調整値は現在のキャパシティーを増やし、負の調整値は現在のキャパシティーを減らします。例えば、グループの現在のキャパシティーが 3 で、調整値が 5 の場合、このポリシーが実行されると、キャパシティーユニットが 5 個キャパシティーに追加され、合計 8 個のキャパシティーユニットになります。
- `ExactCapacity` - グループの現在のキャパシティーを指定された値に変更します。この調整タイプには負ではない値を指定します。例えば、グループの現在のキャパシティーが 3 で、調整値が 5 の場合、このポリシーが実行されると、キャパシティーは 5 キャパシティーユニットに変更されます。
- `PercentChangeInCapacity` - 指定したパーセンテージだけ現在のキャパシティーを増減させます。正の値はキャパシティーを増やし、負の値はキャパシティーを減らします。例えば、現在のキャパシティーが 10 で、調整値が 10% の場合、このポリシーが実行されると、キャパシティーユニットが 1 個追加され、合計 11 個のキャパシティーユニットになります。

> **Note**
>
> 結果の値が整数でない場合、その値を以下のように四捨五入します。
>
> - 1 より大きい値は小数点以下が切り捨てられます。例えば、`12.7` は `12` に丸められます。
> - 0 と 1 の間の値は 1 に丸められます。例えば、`.67` は `1` に丸められます。

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

PercentChangeInCapacity では、MinAdjustmentMagnitude パラメータを使用して、スケールするインスタンスの最小数を指定することもできます。例えば、25 パーセント追加するポリシーを作成し、インスタンスの最小増減値を 2 に指定するとします。4 つのインスタンスを持つ Auto Scaling グループがあり、スケーリングポリシーを実行する場合、25 パーセントは 1 インスタンスです。ただし、最小増減値が 2 に指定されているため、2 つのインスタンスが追加されます。

インスタンスの重み付けを使用すると、MinAdjustmentMagnitude パラメータをゼロ以外の値に設定した効果が変化します。値はキャパシティーユニットで指定します。スケールするインスタンスの最小数を設定するには、このパラメータを、最大インスタンスの重みと同じ以上の値に設定します。

インスタンスの重み付けを使用している場合、Auto Scaling グループの現在のキャパシティは、必要に応じて希望するキャパシティを超える場合があることに注意してください。デクリメントする絶対数、またはデクリメントする割合が現在のキャパシティーと希望するキャパシティーの差よりも小さい場合、スケーリングアクションは実行されません。しきい値アラームに違反している場合にスケーリングポリシーの結果を確認する際は、このような動作を考慮に入れる必要があります。例えば、希望するキャパシティーが 30 で、現在のキャパシティーが 32 であるとします。アラームに違反している場合、スケーリングポリシーで希望するキャパシティーを 1 ずつ減らすと、スケーリングアクションは実行されません。

インスタンスのウォームアップ

ステップスケーリングの場合、オプションで、新しく起動されたインスタンスのウォームアップ時間を秒単位で指定できます。指定されたウォームアップ時間が終了するまで、インスタンスは Auto Scaling グループの集計EC2インスタンスメトリクスにカウントされません。

インスタンスのウォームアップ期間中、スケーリングポリシーは、ウォームアップしていないインスタンスからのメトリクス値がポリシーのアラーム上限しきい値を超える場合にのみスケールアウトします。

グループが再度スケールアウトする場合、まだウォームアップ中のインスタンスは、次のスケールアウトアクティビティの希望キャパシティーの一部として計上されます。したがって、複数のアラーム超過によってもステップ調整値の範囲は変わらず、1 つのスケーリングアクティビティという結果になります。その目的は、スケールアウトを継続的に (ただし過剰になることなく) 行うことです。

例えば、2 つのステップからなるポリシーを作成するとします。最初のステップでは、メトリクスが 60 になったら 10% を追加し、2 番目のステップではメトリクスが 70 になったら 30% を追加します。Auto Scaling グループの目的キャパシティーは 10 で、現在のキャパシティは 10 です。集約されたメトリクス値が 60 未満の間は、目的キャパシティーと現在のキャパシティーは変わりません。メトリクスが 60 になり、1 つのインスタンス (10 インスタンスの 10 パーセント) が追加されたとします。その後、新しいインスタンスがまだウォームアップしている間に、メトリクスは 62 になります。スケーリングポリシーは、現在のキャパシティー (まだ 10) に基づいて新しく必要なキャパシティーを計算します。ただし、グループの必要キャパシティーはすでに 11 インスタンスに増えているため、スケーリングポリシーはこれ以上必要キャパシティーは増やしません。新しいインスタンスがまだウォームアップ中にメトリックスが 70 になった場合は、3 インスタンス (10 インスタンスの 30%) を追加する必要があります。ただし、グループの必要なキャパシティーはすでに 11 であるため、2 つのインスタンスのみを追加し、新しい必要なキャパシティーは 13 インスタンスになります。

スケールアウトアクティビティの進行中は、インスタンスがウォームアップを終了するまで、スケーリングポリシーによって開始されたすべてのスケールインアクティビティがブロックされます。インスタンスのウォームアップが完了し、スケールインイベントが発生した場合、現在終了中のインスタンスは、新しい必要キャパシティーを計算する際にグループの現在のキャパシティーにカウントされます。したがって、Auto Scaling グループから必要以上の数のインスタンスが削除されることがなくなります。例えば、インスタンスがすでに終了しているときに、必要なキャパシティーを 1 つデクリメントした同じステップ調整の範囲内でアラームに違反した場合、スケーリングアクションは実行されません。

デフォルト値

値が設定されていない場合、スケーリングポリシーは、グループに定義されているデフォルトのインスタンスウォームアップの値であるデフォルト値を使用します。インスタンスのデフォルトウォームアップが null の場合、デフォルトクールダウンの値にフォールバックします。

考慮事項

ステップおよびシンプルな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。

スケールアウト用のステップスケーリングポリシーを作成する

Auto Scaling グループのスケールアウト用のステップスケーリングポリシーを作成するには、次のいずれかの方法を使用します。

Console

ステップ 1: メトリクスの上限しきい値の CloudWatch アラームを作成する

1. で CloudWatch コンソールを開きますhttps://console.aws.amazon.com/cloudwatch/。
2. 必要に応じて、リージョンを変更します。ナビゲーションバーから、Auto Scaling グループがあるリージョンを選択します。
3. ナビゲーションペインで、[Alarms]、[All alarms] (アラーム、すべてのアラーム) の順に選択してから、[Create alarm] (アラームの作成) を選択します。
4. [メトリクスの選択] を選択します。
5. すべてのメトリクスタブで、EC2、Auto Scaling グループ別 を選択し、検索フィールドに Auto Scaling グループの名前を入力します。次に、`CPUUtilization` を選択し、[メトリクスの選択] を選択します。[Specify metric and conditions] ページに、メトリクスに関するグラフや他の情報が表示されます。
6. [期間] でアラームの評価期間 (1 分など) を選択します。アラームを評価する場合、各期間は 1 つのデータポイントに集約されます。

   > **Note**
   >
   > 期間が短いほど、作成されるアラームの感度が高くなります。

7. [条件] で、次の操作を行います。

   - [Threshold type] で [静的] を選択します。
   - [**`CPUUtilization`** が次の場合] で、アラーム違反のしきい値に対するメトリクスの値を [より高い] または [以上] のいずれかに指定します。次に、[than] (次よりも:) にアラーム違反のしきい値を入力します。

8. [追加設定] で、次の操作を行います。

   - [Datapoints to alarm] (アラームへのデータポイント)に、メトリクス値がしきい値を満たす必要があるデータポイント数 (評価期間) を入力します。例えば、5 分の期間が 2 回連続すると、アラームの状態が呼び出されるまで 10 分かかります。
   - [Missing data treatment (欠落データの処理)] では、[Treat missing data as bad (breaching threshold) (欠落データを問題として処理する (しきい値を超過))] を選択します。詳細については、「Amazon CloudWatch ユーザーガイド」の CloudWatch 「アラームが欠落データを処理する方法の設定」を参照してください。

9. [Next (次へ)] を選択します。

   [Configure actions] (アクションの設定) ページが表示されます。

10. 通知 で、アラームが `ALARM`状態、 `OK` 状態、または `INSUFFICIENT_DATA`状態になったときに通知する Amazon SNSトピックを選択します。

    同じアラーム状態または複数の異なるアラーム状態について複数の通知を送信するには、[Add notification (通知の追加)] を選択します。

    アラームの通知を送信しない場合は、 [削除] を選択します。

11. Configure actions (アクションの設定) ページの他のセクションは空にしたままにすることができます。他のセクションを空のままにすると、スケーリングポリシーへの関連付けなしでアラームが作成されます。その後、Amazon EC2 Auto Scaling コンソールからアラームをスケーリングポリシーに関連付けることができます。
12. [Next (次へ)] を選択します。
13. 名前 (`Step-Scaling-AlarmHigh-AddCapacity` など) を入力し、必要に応じてアラームの説明を入力して [次へ] を選択します。
14. [アラームの作成] を選択します。

CloudWatch アラームの作成後に中断した場所を続行するには、次の手順に従います。

ステップ 2: スケールアウト用のステップスケーリングポリシーを作成する

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。

3. スケーリング制限が適切に設定されていることを検証します。例えば、グループの希望するキャパシティが既に最大に達している場合は、スケールアウトするために新しい最大を指定する必要があります。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。
4. [Automatic scaling] (オートスケーリング) タブの [Dynamic scaling policies] (動的スケーリングポリシー) で、[Create dynamic scaling] (動的スケーリングポリシーの作成) を選択します。
5. [ポリシータイプ] では、[ステップスケーリング] を選択し、ポリシーの名前を指定します。
6. CloudWatch アラームでは、アラームを選択します。アラームをまだ作成していない場合は、 CloudWatch アラームの作成を選択し、前の手順のステップ 4 ~ 14 を完了してアラームを作成します。
7. [Take the action (このアクションを実行)] を使用して、このポリシーの実行時に行う現在のグループサイズの変更を指定します。特定の数のインスタンスまたは既存のグループサイズに対する割合を追加したり、グループを正確なサイズに設定したりできます。

   例えば、グループのキャパシティを 30% 増やすスケールアウトポリシーを作成するには、[Add] を選択し、次のフィールドに「30」を入力後 [percent of group] を選択します。デフォルトでは、このステップ調整値の下限はアラームしきい値であり、上限は正 (+) の無限大です。
8. 別のステップを追加するには、[Add step (ステップの追加)] を選択し、スケールする量と、アラームしきい値に対するステップの下限と上限を定義します。
9. 最小数のインスタンスを設定してスケールするには、[Add capacity units in increments of at least (最小数の増分でキャパシティーユニットを追加する)] 1 [capacity units (キャパシティーユニット)] の数値フィールドをアップデートします。
10. (オプション) [インスタンスのウォームアップ] で、必要に応じてインスタンスのウォームアップ値を更新します。
11. [Create] (作成) を選択します。

AWS CLI

スケールアウト (キャパシティを増やす) 用のステップスケーリングポリシーを作成するには、次のコマンド例を使用します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

を使用するときは AWS CLI、まず、メトリクスの値が増えたときにスケールアウトする方法を Amazon EC2 Auto Scaling に指示するステップスケーリングポリシーを作成します。次に、監

視するメトリクスを特定し、アラームのメトリクスの上限しきい値とその他の詳細を定義し、アラームをスケーリングポリシーに関連付けることでアラームを作成します。

ステップ 1: スケールアウト用のポリシーを作成する

次のput-scaling-policyコマンドを使用して、 という名前のステップスケーリングポリシーを作成します。調整タイプPercentChangeInCapacityは でmy-step-scale-out-policy、以下のステップ調整値 (CloudWatch アラームしきい値を 60% と仮定) に基づいて グループの容量を増やします。

- メトリクスの値が 60% 以上、75% 未満の場合は、インスタンス数を 10% 増やします。
- メトリクスの値が 75% 以上、85% 未満の場合は、インスタンス数を 20% 増やします。
- メトリクスの値が 85% 以上の場合は、インスタンス数を 30% 増やします。

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=0.0,MetricIntervalUpperBound=15.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=15.0,MetricIntervalUpperBound=25.0,ScalingAdjustment=20 \
                     MetricIntervalLowerBound=25.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

ポリシーの Amazon リソースネーム () を記録しますARN。ポリシーの CloudWatch アラームを作成するのに必要です。

```
{
    "PolicyARN":
 "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-
aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy
}
```

ステップ 2: メトリクスの上限しきい値の CloudWatch アラームを作成する

次の CloudWatch put-metric-alarmコマンドを使用して、2 分間の連続した 2 つの評価期間の平均CPUしきい値 60% に基づいて Auto Scaling グループのサイズを増やすアラームを作成します。

独自のカスタムメトリクスを使用するには、名前を --metric-name で指定し、その名前空間を --namespace で指定します。

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-AddCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 60 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

スケールイン用のステップスケーリングポリシーを作成する

Auto Scaling グループのスケールイン用のステップスケーリングポリシーを作成するには、次のいずれかの方法を使用します。

Console

ステップ 1: メトリクスのしきい値が低い場合の CloudWatch アラームを作成する

1. で CloudWatch コンソールを開きますhttps://console.aws.amazon.com/cloudwatch/。
2. 必要に応じて、リージョンを変更します。ナビゲーションバーから、Auto Scaling グループがあるリージョンを選択します。
3. ナビゲーションペインで、[Alarms]、[All alarms] (アラーム、すべてのアラーム) の順に選択してから、[Create alarm] (アラームの作成) を選択します。
4. [メトリクスの選択] を選択します。
5. すべてのメトリクスタブで、EC2、Auto Scaling グループ別 を選択し、検索フィールドに Auto Scaling グループの名前を入力します。次に、CPUUtilization を選択し、[メトリクスの選択] を選択します。[Specify metric and conditions] ページに、メトリクスに関するグラフや他の情報が表示されます。
6. [期間] でアラームの評価期間 (1 分など) を選択します。アラームを評価する場合、各期間は 1 つのデータポイントに集約されます。

   > **Note**
   >
   > 期間が短いほど、作成されるアラームの感度が高くなります。

7. [条件] で、次の操作を行います。

- [Threshold type] で [静的] を選択します。
- [**CPUUtilization** が次の場合] で、アラーム違反のしきい値に対するメトリクスの値を [より低い] または [以下] のいずれかに指定します。次に、[than] (次よりも:) にアラーム違反のしきい値を入力します。

> **Important**
>
> スケールインポリシー (メトリクス低) で使用するアラームについては、しきい値に対して [より高い] や [以上] を選択しないようにします。

8. [追加設定] で、次の操作を行います。

   - [Datapoints to alarm] (アラームへのデータポイント)に、メトリクス値がしきい値を満たす必要があるデータポイント数 (評価期間) を入力します。例えば、5 分の期間が 2 回連続すると、アラームの状態が呼び出されるまで 10 分かかります。
   - [Missing data treatment (欠落データの処理)] では、[Treat missing data as bad (breaching threshold) (欠落データを問題として処理する (しきい値を超過))] を選択します。詳細については、「Amazon CloudWatch ユーザーガイド」の CloudWatch 「アラームが欠落データを処理する方法の設定」を参照してください。

9. [Next (次へ)] を選択します。

   [Configure actions] (アクションの設定) ページが表示されます。

10. 通知 で、アラームが `ALARM`状態、 `OK` 状態、または `INSUFFICIENT_DATA`状態になったときに通知する Amazon SNSトピックを選択します。

    同じアラーム状態または複数の異なるアラーム状態について複数の通知を送信するには、[Add notification (通知の追加)] を選択します。

    アラームの通知を送信しない場合は、 [削除] を選択します。

11. Configure actions (アクションの設定) ページの他のセクションは空にしたままにすることができます。他のセクションを空のままにすると、スケーリングポリシーへの関連付けなしでアラームが作成されます。その後、Amazon EC2 Auto Scaling コンソールからアラームをスケーリングポリシーに関連付けることができます。
12. [Next (次へ)] を選択します。
13. 名前 (`Step-Scaling-AlarmLow-RemoveCapacity` など) を入力し、必要に応じてアラームの説明を入力して [次へ] を選択します。

14. [アラームの作成] を選択します。

CloudWatch アラームの作成後に中断した場所を続行するには、次の手順に従います。

ステップ 2: スケールイン用のステップスケーリングポリシーを作成する

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. スケーリング制限が適切に設定されていることを検証します。例えば、グループの希望するキャパシティが既に最小に達している場合は、スケールインするために新しい最小を指定する必要があります。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。
4. [Automatic scaling] (オートスケーリング) タブの [Dynamic scaling policies] (動的スケーリングポリシー) で、[Create dynamic scaling] (動的スケーリングポリシーの作成) を選択します。
5. [ポリシータイプ] では、[ステップスケーリング] を選択し、ポリシーの名前を指定します。
6. CloudWatch アラームでは、アラームを選択します。アラームをまだ作成していない場合は、 CloudWatch アラームの作成を選択し、前の手順のステップ 4 ~ 14 を完了してアラームを作成します。
7. [Take the action (このアクションを実行)] を使用して、このポリシーの実行時に行う現在のグループサイズの変更を指定します。特定の数のインスタンスまたは既存のグループサイズに対する割合を削除したり、グループを正確なサイズに設定したりできます。

   例えば、グループのキャパシティを 2 インスタンス減らすスケールインポリシーを作成するには、[Remove] を選択し、次のフィールドに「2」を入力後 [capacity units] を選択します。デフォルトでは、このステップ調整値の上限はアラームしきい値であり、下限は負 (-) の無限大です。
8. 別のステップを追加するには、[Add step (ステップの追加)] を選択し、スケールする量と、アラームしきい値に対するステップの下限と上限を定義します。
9. [Create] (作成) を選択します。

AWS CLI

スケールイン (キャパシティを減らす) 用のステップスケーリングポリシーを作成するには、次のコマンド例を使用します。*user input placeholder* を、ユーザー自身の情報に置き換えます。

を使用するときは AWS CLI、まず、メトリクスの値が減少しているときにスケールインする方法を Amazon EC2 Auto Scaling に指示するステップスケーリングポリシーを作成します。次に、監視するメトリクスを特定し、アラームのメトリクスの下限しきい値とその他の詳細を定義し、アラームをスケーリングポリシーに関連付けることでアラームを作成します。

ステップ 1: スケールイン用のポリシーを作成する

次のput-scaling-policyコマンドを使用して、 という名前のステップスケーリングポリシーを作成します。このポリシーの調整タイプはmy-step-scale-in-policy、関連付けられた CloudWatch アラームChangeInCapacityがメトリクスのしきい値を下回った場合に、グループの容量を 2 インスタンス減らします。

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

ポリシーの Amazon リソースネーム () を記録しますARN。ポリシーの CloudWatch アラームを作成するのに必要です。

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy
}
```

ステップ 2: メトリクスのしきい値が低い場合の CloudWatch アラームを作成する

次の CloudWatch put-metric-alarmコマンドを使用して、2 分間の連続した 2 つの評価期間の平均 CPUしきい値 40% に基づいて Auto Scaling グループのサイズを小さくするアラームを作成します。独自のカスタムメトリクスを使用するには、名前を `--metric-name` で指定し、その名前空間を `--namespace` で指定します。

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmLow-RemoveCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 40 \
  --comparison-operator LessThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

簡易スケーリングポリシー

次の例は、 CLI コマンドを使用してシンプルなスケーリングポリシーを作成する方法を示しています。これらの例は、お客様が使用を希望される場合のリファレンスとしてこのドキュメントに残してありますが、シンプルスケーリングポリシーではなくターゲット追跡ポリシーまたはステップスケーリングポリシーを使用することをお勧めします。

ステップスケーリングポリシーと同様に、簡易スケーリングポリシーでは、スケーリングポリシーの CloudWatch アラームを作成する必要があります。作成するポリシーでは、インスタンスを追加するか削除するかとその個数を定義するか、グループのサイズそのものを設定する必要もあります。

ステップスケーリングポリシーとシンプルスケーリングポリシーの主な違いの 1 つは、ステップスケーリングポリシーにはステップ調整があることです。ステップスケーリングを使用すると、指定したステップ調整に基づいて、グループのサイズの変化を大きくすることも小さくすることもできます。

シンプルスケーリングポリシーでは、進行中のスケーリングアクティビティまたはヘルスチェックの置き換えが完了し、クールダウン期間が終わるのを待ってから、追加のアラームに応答する必要があります。これとは対照的に、ステップスケーリングでは、ポリシーは、スケーリングアクティビティまたはヘルスチェックの置換が進行中であっても、引き続き別のアラームに応答します。これは、Amazon EC2 Auto Scaling がアラームメッセージを受信すると、すべてのアラーム違反を評価することを意味します。このため、スケーリング調整が 1 つだけであっても、シンプルスケーリングではなくステップスケーリングポリシーを使用することをお勧めします。

Amazon EC2 Auto Scaling は当初、簡易スケーリングポリシーのみをサポートしていました。ターゲット追跡ポリシーおよびステップスケーリングポリシーが導入されるより前にスケーリングポリシーを作成していた場合、そのポリシーはシンプルスケーリングポリシーとして扱われます。

スケールアウト用のシンプルスケーリングポリシーを作成するには

次のput-scaling-policyコマンドを使用して、 という名前の簡易スケーリングポリシーを作成します。このポリシーはmy-simple-scale-out-policy、関連付けられた CloudWatch アラー

ムPercentChangeInCapacityがメトリクスのしきい値を超過すると、調整タイプ でグループの容量を 30% 増やします。

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity
```

ポリシーの Amazon リソースネーム () を記録しますARN。ポリシーの CloudWatch アラームを作成するのに必要です。

スケールイン用のシンプルスケーリングポリシーを作成するには

次のput-scaling-policyコマンドを使用して、 という名前の簡易スケーリングポリシーを作成します。このポリシーの調整タイプはmy-simple-scale-in-policy、関連付けられた CloudWatch アラームChangeInCapacityがメトリクスのしきい値を下回った場合に、グループの容量を 1 インスタンス減らします。

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

ポリシーの Amazon リソースネーム () を記録しますARN。ポリシーの CloudWatch アラームを作成するのに必要です。

Amazon EC2 Auto Scaling のスケーリングのクールダウン

> **Important**
>
> ベストプラクティスとして、シンプルスケーリングポリシーとスケーリングクールダウンを使用しないことをお勧めします。スケーリングパフォーマンスにより適しているのは、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です。スケーリングメトリクスの値が減少または増加するにつれて Auto Scaling グループのサイズを比例して変更するスケーリングポリシーの場合は、簡易スケーリングまたはステップスケーリングのいずれかでターゲットを追跡することをお勧めします。

Auto Scaling グループにシンプルスケーリングポリシーを作成するときは、スケーリングのクールダウンを同時に設定することをお勧めします。

Auto Scaling グループは、インスタンスの起動または終了後、クールダウン期間が終了するのを待ってから、シンプルスケーリングポリシーによって開始される追加のスケーリングアクティビティを開始します。クールダウン期間の目的は、Auto Scaling グループを安定させ、1 つ前のスケーリングアクティビティの効果を確認できるようになる前に追加のインスタンスを起動または終了しないようにすることです。

例えば、CPU使用率の簡易スケーリングポリシーで 2 つのインスタンスの起動が推奨されているとします。Amazon EC2 Auto Scaling は 2 つのインスタンスを起動し、クールダウン期間が終了するまでスケーリングアクティビティを一時停止します。シンプルスケーリングポリシーによって開始されたスケーリングアクティビティを再開できるのは、このクールダウン期間の終了後になります。CPU 使用率がアラームの上限しきい値を再度超えると、Auto Scaling グループは再びスケールアウトし、クールダウン期間が有効になります。ただし、メトリクス値を低減させるためには 2 個のインスタンスで十分であった場合、グループは現行のサイズのままになります。

内容

考慮事項

以下の考慮事項は、シンプルスケーリングポリシーとスケーリングクールダウンの使用時に適用されます。

- ターゲット追跡ポリシーとステップスケーリングポリシーは、クールダウン期間の終了を待つことなく、ただちにスケールアウトアクティビティを開始できます。その代わり、Auto Scaling グループがインスタンスを起動するときは、常に個々のインスタンスにウォームアップ期間が設定されます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。
- 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刻に開始されるときも、クールダウン期間の終了を待つことなく、ただちにスケーリングアクティビティを開始できます。
- インスタンスが異常になると、Amazon EC2 Auto Scaling はクールダウン期間が終了するのを待ってから、異常のあるインスタンスを置き換えます。

- 複数のインスタンスが起動または終了される場合は、最後のインスタンスの起動または終了が完了した時点でクールダウン期間 (デフォルトのクールダウンまたはスケーリングポリシー固有のクールダウン) が実施されます。
- Auto Scaling グループを手動でスケールするときは、クールダウンの終了を待たないことがデフォルトになります。ただし、 AWS CLI または を使用して手動でスケーリングする場合、この動作を上書きSDKしてデフォルトのクールダウンに従うことができます。
- デフォルトで、Elastic Load Balancing は登録解除 (Connection Draining) プロセスが完了するまで 300 秒間待機します。グループが Elastic Load Balancing ロードバランサーの背後にある場合は、クールダウン期間を開始する前に、終了されるインスタンスが登録解除されるのを待ちます。

追加の遅延を発生させる可能性のあるライフサイクルフック

ライフサイクルフックが呼び出される場合、クールダウン期間はライフサイクルアクションの完了後、またはタイムアウト期間の終了後に開始されます。例えば、インスタンス起動用のライフサイクルフックがある Auto Scaling グループについて考えてみましょう。アプリケーションで需要の増加が生じると、グループはキャパシティーを追加するためにインスタンスを起動します。ライフサイクルフックが存在することから、インスタンスは待機状態になり、シンプルスケーリングポリシーによるスケーリングアクティビティは一時停止されます。インスタンスが `InService` 状態になると、クールダウン期間が開始します。クールダウン期間が終了すると、シンプルなスケーリングポリシーのアクティビティが再開されます。

Elastic Load Balancing が有効化されているときは、スケールインにおいて、終了対象に選択されているインスタンスが Connection Draining (登録解除遅延) を開始するときにクールダウン期間が開始されます。クールダウン期間は、Connection Draining が終了するのを待つことも、ライフサイクルフックがそのアクションを完了するのを待つこともありません。つまり、シンプルスケーリングポリシーによるスケーリングアクティビティは、スケールインイベントの結果がグループのキャパシティに反映され次第再開できることになります。これ以外の場合は、3 つのアクティビティ (Connection Draining、ライフサイクルフック、およびクールダウン期間) のすべてが完了するまで待機することになり、Auto Scaling グループがスケーリングを一時停止しなければならない時間が大幅に長くなります。

デフォルトのクールダウン期間を変更する

Amazon Auto Scaling EC2Auto Scaling コンソールで Auto Scaling グループを最初に作成するときに、デフォルトのクールダウンを設定することはできません。デフォルトで、このクールダウン期間は 300 秒 (5 分) に設定されています。これは、グループの作成後に必要に応じて更新できます。

デフォルトのクールダウン期間を変更する (コンソール)

Auto Scaling グループの作成後、[Details] (詳細) タブの [Advanced configurations] (高度な設定) で [Edit] (編集) を選択します。[Default cooldown] (デフォルトのクールダウン) で、インスタンスの起動時間やその他のアプリケーションニーズに基づいて、希望する時間を選択します。

デフォルトのクールダウン期間を変更する (AWS CLI)

以下のコマンドを使用して、新しい、または既存の Auto Scaling グループのデフォルトのクールダウンを変更します。デフォルトのクールダウンが定義されていない場合は、デフォルト値の 300 秒が使用されます。

- create-auto-scaling-group
- update-auto-scaling-group

デフォルトのクールダウンの値を確認するには、 describe-auto-scaling-groups コマンドを使用します。

特定のシンプルスケーリングポリシーにクールダウン期間を設定する

デフォルトで、すべてのシンプルスケーリングポリシーは Auto Scaling グループに定義されているデフォルトのクールダウン期間を使用します。特定のシンプルスケーリングポリシーにクールダウン期間を設定するには、ポリシーを作成または更新するときに、オプションのクールダウンパラメータを使用します。ポリシーにクールダウン期間を指定すると、デフォルトのクールダウンが上書きされます。

スケーリングポリシー固有のクールダウン期間の一般的な用途の 1 つに、スケールインポリシーとの使用があります。このポリシーはインスタンスを終了するため、Amazon EC2 Auto Scaling は追加のインスタンスを終了するかどうかを判断するのに必要な時間を短縮します。インスタンスを終了するオペレーションは、インスタンスを起動するよりもはるかに高速です。したがって、デフォルトのクールダウン期間である 300 秒は長すぎます。この場合、スケールインポリシーの値をより短くしたスケーリングポリシー固有のクールダウン期間は、グループがより迅速にスケールインできるようにすることで、コストの削減に役立ちます。

コンソールでシンプルスケーリングポリシーを作成または更新するには、グループの作成後に [Automatic scaling] (自動スケーリング) タブを選択します。を使用して簡易スケーリングポリシーを作成または更新するには AWS CLI、 put-scaling-policy コマンドを使用します。詳細については、「ステップスケーリングポリシーおよび簡易スケーリングポリシー」を参照してください。

Amazon に基づくスケーリングポリシー SQS

> **Important**
>
> 以下の情報とステップは、カスタムメトリクスとして発行する前に、SQSキュー属性を使用してインスタンスごとに Amazon `ApproximateNumberOfMessages`キューバックログを計算する方法を示しています CloudWatch。ただし、Metric Math を使用することで、独自のメトリクスを公開するコストと労力を節約できるようになりました。詳細については、「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」を参照してください。

このセクションでは、Amazon Simple Queue Service (Amazon SQS) キューのシステム負荷の変化に応じて Auto Scaling グループをスケールする方法を示します。Amazon の使用方法の詳細についてはSQS、Amazon Simple Queue Service デベロッパーガイドを参照してください。

Amazon SQSキューのアクティビティに応じてスケーリングすることを検討するシナリオがいくつかあります。例えば、ユーザーがイメージをアップロードしてオンラインで使用できるウェブアプリがあるとします。このシナリオでは、各イメージはサイズ変更されエンコードされてから公開されます。アプリケーションは Auto Scaling グループのEC2インスタンスで実行され、一般的なアップロードレートを処理するように設定されます。異常なインスタンスは終了され置き換えられて、常に現在のインスタンスレベルが維持されます。このアプリは処理のためイメージの raw ビットマップデータを SQS キューに配置します。イメージが処理され、処理されたイメージはユーザーから確認できる場所に発行されます。このシナリオのアーキテクチャは、イメージのアップロード数が時間の経過とともに変化しない場合に有効です。ただし、アップロード数が時間の経過とともに変化する場合は、動的スケーリングを使用して Auto Scaling グループのキャパシティーを拡張することを検討してください。

内容

適切なメトリクスを用いたターゲット追跡を使用する

カスタム Amazon SQSキューメトリクスに基づくターゲット追跡スケーリングポリシーを使用すると、動的スケーリングはアプリケーションの需要曲線により効果的に調整できます。ターゲット追跡のメトリクスの選択の詳細については、「メトリクスを選択する」を参照してください。

ターゲット追跡`ApproximateNumberOfMessagesVisible`に などの CloudWatch Amazon SQSメトリクスを使用する場合の問題は、キュー内のメッセージ数が、キューからのメッセージを処理する Auto Scaling グループのサイズに比例して変化しない可能性があることです。これは、SQSキュー内のメッセージ数は、必要なインスタンスの数だけを定義するわけではないためです。Auto Scaling グループ内のインスタンスの数は複数の要因によって決まります。例えば、メッセージを処理するためにかかる時間や、許容されるレイテンシー (キュー遅延) の量などです。

解決策は、インスタンスあたりのバックログのメトリクスを使用して、ターゲット値を維持するインスタンスあたりの適正バックログにすることです。これらの数は以下のように計算できます。

- インスタンスあたりのバックログ: インスタンスあたりのバックログを計算するには、`ApproximateNumberOfMessages`キュー属性から始めてキューの長さ (SQSキューから取得できるメッセージの数) を決定します。その数をフリートの実行キャパシティーで割ります。Auto Scaling グループの場合、これは `InService` 状態にあるインスタンスの数です。これでインスタンスあたりのバックログを得ることができます。
- インスタンスあたりの適正バックログ: ターゲット値を計算するには、まずレイテンシーの点でアプリケーションが受け付けることができる数を決定します。次に、許容されるレイテンシー値を取得し、EC2インスタンスがメッセージを処理するのにかかる平均時間で割ります。

例として、現在、10 個のインスタンスを持つ Auto Scaling グループがあり、キュー (`ApproximateNumberOfMessages`) にある可視メッセージの数が 1,500 であるとします。メッセージあたりの平均処理時間が 0.1 秒で、最長許容レイテンシーが 10 秒の場合、インスタンスあたりの許容バックログは 10/0.1、つまり 100 メッセージとなります。つまり、100 がターゲット追跡ポリシーのターゲット値です。インスタンスあたりのバックログがターゲット値に達すると、スケールアウトイベントが発生します。インスタンスあたりのバックログが既に 150 メッセージ (1,500 メッセージ/10 インスタンス) あるため、グループは、ターゲット値との比例を維持するために 5 インスタンス分スケールアウトします。

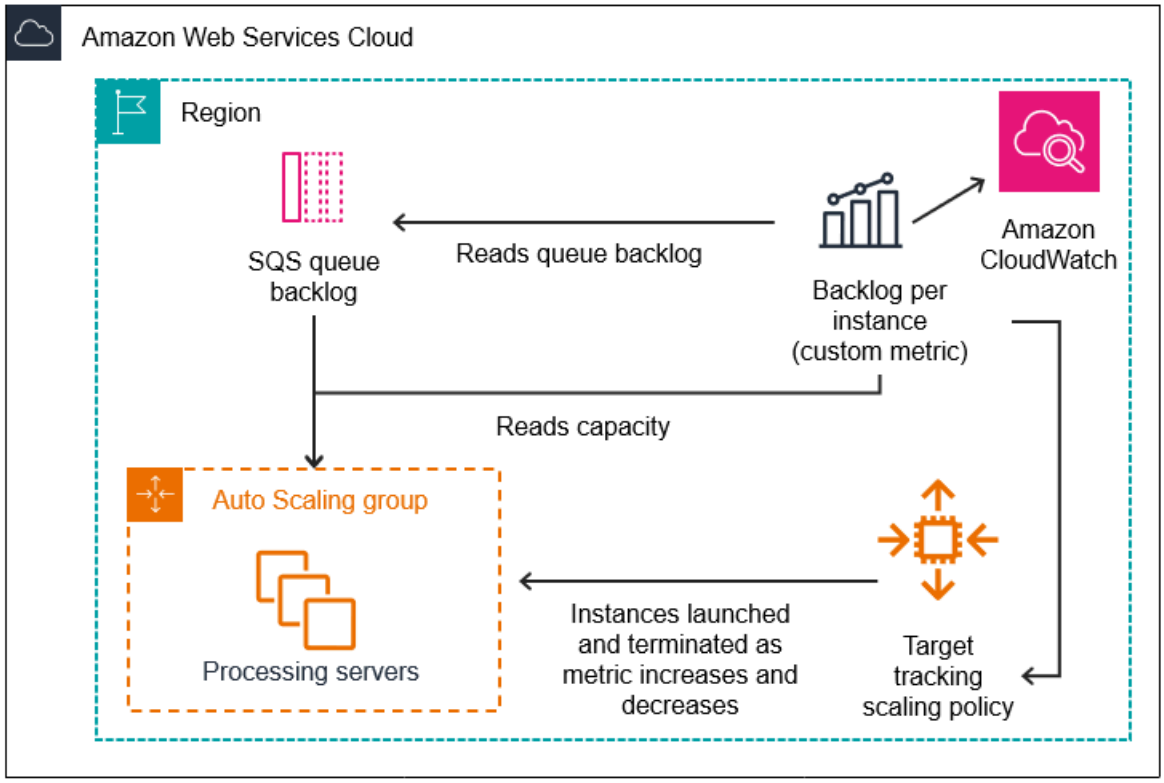
次の手順は、カスタムメトリクスを公開し、これらの計算に基づいてスケーリングするように Auto Scaling グループを設定するターゲット追跡スケーリングポリシーを作成する方法を示しています。

> **Important**
>
> コストを削減するには、必ず代わりに Metric Math を使用してください。詳細については、「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」を参照してください。

この設定は 3 つの主要な部分で構成されます。

- SQS キューからのメッセージを処理する目的でEC2インスタンスを管理する Auto Scaling グループ。
- Auto Scaling グループのEC2インスタンスごとにキュー内のメッセージ数 CloudWatch を測定する Amazon に送信するカスタムメトリクス。
- カスタムメトリクスと設定されたターゲット値に基づいてスケーリングするように Auto Scaling グループを設定するターゲット追跡ポリシー。 CloudWatch alarms はスケーリングポリシーを呼び出します。

次の図は、この設定のアーキテクチャを示しています。



制限事項と前提条件

この設定を使用するには、次の制限事項に注意する必要があります。

- カスタムメトリクスを に発行SDKするには、 AWS CLI または を使用する必要がありますCloudWatch。その後、 を使用してメトリクスをモニタリングできます AWS Management Console。
- Amazon EC2 Auto Scaling コンソールは、カスタムメトリクスを使用するターゲット追跡スケーリングポリシーをサポートしていません。スケーリングポリシーのカスタムメトリクスを指定するSDKには、 AWS CLI または を使用する必要があります。

以下のセクションでは、実行する必要があるタスク AWS CLI に を使用する方法について説明します。例えば、キューの現在の使用状況を反映するメトリクスデータを取得するには、 SQS get-queue-attributes コマンドを使用します。CLI がインストールされ、設定されていることを確認します。

開始する前に、使用する Amazon SQSキューが必要です。以下のセクションでは、キュー (標準または FIFO) 、Auto Scaling グループ、およびキューを使用するアプリケーションを実行しているEC2インスタンスが既にあることを前提としています。Amazon の詳細についてはSQS、Amazon Simple Queue Service デベロッパーガイドを参照してください。

Amazon に基づいてスケーリングを設定する SQS

このセクションでは、Amazon に基づいてスケーリングを設定する方法について説明しますSQS。

タスク

- ステップ 1: CloudWatch カスタムメトリクスを作成する
- ステップ 2: ターゲット追跡スケーリングポリシーを作成する
- ステップ 3: スケーリングポリシーをテストする

ステップ 1: CloudWatch カスタムメトリクスを作成する

カスタムメトリックスは、選択したメトリックス名と名前空間を使用して定義されます。カスタムメトリクスの名前空間を `AWS/` で始めることはできません。カスタムメトリクスの発行の詳細については、「Amazon CloudWatch ユーザーガイド」の「カスタムメトリクスの発行」トピックを参照してください。

この手順に従って、まず AWS アカウントから情報を読み取ってカスタムメトリクスを作成します。次に、前のセクションで推奨されたようにインスタンスメトリクスごとにバックログを計算します。最後に、この番号を 1 分単位で CloudWatch に発行します。可能な限り、システム負荷の変化に迅速に対応できるように、メトリクスを 1 分単位でスケーリングすることを強くお勧めします。

CloudWatch カスタムメトリクスを作成するには (AWS CLI)

1. SQS get-queue-attributes コマンドを使用して、キューで待機しているメッセージの数を取得します (ApproximateNumberOfMessages)。

```
aws sqs get-queue-attributes --queue-url https://
sqs.region.amazonaws.com/123456789/MyQueue \
  --attribute-names ApproximateNumberOfMessages
```

2. describe-auto-scaling-groups コマンドを使用して、グループの実行キャパシティーを取得します。これは InService ライフサイクル状態にあるインスタンスの数です。このコマンドは、Auto Scaling グループのインスタンスとそのライフサイクル状態を返します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

3. キューからの取得に使用できるメッセージの概数をグループの実行キャパシティで除算して、インスタンスあたりのバックログを算出します。
4. インスタンス値ごとにバックログを取得して CloudWatch カスタムメトリクスに発行するために 1 分ごとに実行されるスクリプトを作成します。カスタムメトリクスを公開する際は、そのメトリクスの名前、名前空間、単位、値、および 0 以上のディメンションを指定します。ディメンションには、そのディメンションの名前と値を含みます。

   カスタムメトリクスを発行するには、 *italics*のプレースホルダー値を任意のメトリクス名、メトリクスの値、名前空間 (AWS「」で始まらない場合)、ディメンション (オプション) に置き換え、次のput-metric-dataコマンドを実行します。

```
aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance --
namespace MyNamespace \
  --unit None --value 20 --
dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
```

アプリケーションが目的のメトリクスを出力すると、データが CloudWatch に送信されます。メトリクスは CloudWatch コンソールに表示されます。アクセスするには、 にログイン AWS Management

Console し、 CloudWatch ページに移動します。その後、メトリクスを表示するには、メトリクスページに移動するか、検索ボックスを使用してメトリクスを検索します。メトリクスの表示については、「Amazon CloudWatch ユーザーガイド」の「使用可能なメトリクスの表示」を参照してください。

ステップ 2: ターゲット追跡スケーリングポリシーを作成する

この時点で、作成したメトリクスをターゲット追跡スケーリングポリシーに追加できるようになっています。

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. 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カスタマイズされたメトリクス仕様をホームディレクトリの という名前config.jsonのJSONファイルに保存します。*user input placeholder* を、ユーザー自身の情報に置き換えます。TargetValue には、インスタンスあたりの適正バックログメトリックスを計算して、それを入力します。この数を計算するには、上記のセクションで説明しているとおり、標準のレイテンシー値を決定し、その値をメッセージの処理にかかる平均時間で割ります。

   ステップ 1 で作成したメトリクスにディメンションを指定しなかった場合は、カスタムメトリクスの仕様にディメンションを含めないでください。

   ```
   $ cat ~/config.json
   {
      "TargetValue":100,
      "CustomizedMetricSpecification":{
         "MetricName":"MyBacklogPerInstance",
         "Namespace":"MyNamespace",
         "Dimensions":[
            {
               "Name":"MyOptionalMetricDimensionName",
               "Value":"MyOptionalMetricDimensionValue"
            }
         ],
         "Statistic":"Average",
         "Unit":"None"
      }
   }
   ```

2. 前のステップで作成した config.json ファイルとともに put-scaling-policy コマンドを使用して、スケーリングポリシーを作成します。

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-
policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://~/config.json
```

これにより、2 つのアラーム (スケールアウトとスケールイン) が作成されます。また、に登録されているポリシーの Amazon リソースネーム (ARN) も返します。このポリシーは CloudWatch、メトリクスのしきい値に違反するたびにスケーリングを呼び出すために CloudWatch を使用します。

ステップ 3: スケーリングポリシーをテストする

設定が完了したら、スケーリングポリシーが機能していることを確認します。SQS キュー内のメッセージ数を増やし、Auto Scaling グループが追加のEC2インスタンスを起動したことを確認することでテストできます。SQS キュー内のメッセージ数を減らし、Auto Scaling グループがEC2インスタンスを終了したことを確認することでテストすることもできます。

スケールアウト機能をテストするには

1. 「Amazon SQS標準キューの作成とメッセージの送信」または「Amazon SQSFIFOキューの作成とメッセージの送信」の手順に従って、キューにメッセージを追加します。インスタンスあたりのバックログメトリックスがターゲット値を超えるようにキュー内のメッセージ数を増やしたことを確認します。

   この変更により、アラームの呼び出しに数分かかる場合があります。
2. describe-auto-scaling-groups コマンドを使用して、グループがインスタンスを起動したことを確認します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

スケールイン機能をテストするには

1. メッセージの受信と削除 (コンソール) に関するページの手順に従い、キューからメッセージを削除します。インスタンスあたりのバックログメトリックスがターゲット値を下回るようにキュー内のメッセージ数を減らしたことを確認します。

   この変更により、アラームの呼び出しに数分かかる場合があります。

2. describe-auto-scaling-groups コマンドを使用して、グループがインスタンスを終了したことを確認します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Amazon SQSとインスタンスのスケールイン保護

インスタンスの終了時に処理されていないメッセージはSQSキューに返され、実行中の別のインスタンスで処理できます。長時間実行されるタスクが実行されるアプリケーションでは、オプションでインスタンススケールイン保護を使用して、Auto Scaling グループのスケールイン時に終了するキューワーカーを制御できます。

次の擬似コードは、長時間実行されるキュー駆動のワーカープロセスをスケールイン終了から保護する方法の 1 つを示しています。

```
while (true)
{
  SetInstanceProtection(False);
  Work = GetNextWorkUnit();
  SetInstanceProtection(True);
  ProcessWorkUnit(Work);
  SetInstanceProtection(False);
}
```

詳細については、「インスタンスの終了を的確に処理するようにアプリケーションを設計する」を参照してください。

Auto Scaling グループのスケーリングアクティビティを検証する

Amazon EC2コンソールの Amazon EC2 Auto Scaling セクションで、Auto Scaling グループのアクティビティ履歴を使用して、現在進行中のスケーリングアクティビティの現在のステータスを表示できます。スケーリングが終了すると、それが成功したか失敗したかを確認できます。これは、Auto Scaling グループを作成している場合や、既存のグループにスケーリング条件を追加している場合に特に便利です。

ターゲット追跡、ステップ、または簡易スケーリングポリシーを Auto Scaling グループに追加すると、Amazon EC2 Auto Scaling は直ちにメトリクスに対するポリシーの評価を開始します。メトリクスアラームは、メトリクスが指定された評価期間数のしきい値を超えると、 ALARM 状態になりま

す。これは、スケーリングポリシーは、それが作成された直後から、スケーリング処理に反映されることを意味します。Amazon EC2 Auto Scaling がスケーリングポリシーに応じて希望する容量を調整すると、アカウントのスケーリングアクティビティを確認できます。Amazon EC2 Auto Scaling からスケーリングアクティビティに関する E メール通知を受信する場合は、「」の手順に従いますAmazon EC2 Auto Scaling の Amazon SNS 通知オプション。

> **Tip**
>
> 以下の手順では、Auto Scaling グループについて [Activity history] (アクティビティ履歴) と [Instances] (インスタンス) の各セクションを調べます。どちらのセクションにも、名前付きの列がすでに表示されているはずです。非表示の列を表示する、または表示される行の数を変更するには、各セクションの右上隅にある歯車アイコンを選択して設定モーダルを開き、必要に応じて設定を更新してから、[Confirm] (確認) を選択します。

Auto Scaling グループのスケーリングアクティビティを表示するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループが配置されているリージョンを選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. [Activity] (アクティビティ) タブで、[Activity history] (アクティビティ履歴) の下の [Status] (ステータス) 列に、Auto Scaling グループがインスタンスを正常に起動あるいは終了したか、または、依然としてスケーリングが進行中なのかが表示されます。
5. (オプション) スケーリングアクティビティが多数存在する場合は、アクティビティ履歴の上部端にある [>] アイコンを選択して、スケーリングアクティビティの次ページを表示します。
6. [Instance management (インスタンス管理)] タブにある、[Instances (インスタンス)] の [Lifecycle (ライフサイクル)] 列にインスタンスの状態が含まれます。インスタンスが起動し、ライフサイクルフックが終了すると、そのライフサイクル状態は `InService` に変わります。ヘルスステータス列には、EC2インスタンスのインスタンスヘルスチェックの結果が表示されます。

Auto Scaling グループのスケーリングアクティビティを表示するには (AWS CLI)

次の describe-scaling-activities コマンドを使用します。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下は出力例です。

スケーリングアクティビティは、開始時刻順に並べられます。まだ進行中のアクティビティを最初に説明します。

```
{
  "Activities": [
    {
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-
b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-
tracking-policy changing the desired capacity from 3 to 2.  At 2020-02-11T18:34:31Z
 an instance was taken out of service in response to a difference between desired and
 actual capacity, shrinking the capacity from 3 to 2.  At 2020-02-11T18:34:31Z instance
 i-06c4794c2499af1df was selected for termination.",
      "StartTime": "2020-02-11T18:34:31.268Z",
      "EndTime": "2020-02-11T18:34:53Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn"
    },
...
  ]
}
```

出力のフィールドの説明については、「Amazon EC2 Auto Scaling APIリファレンス」の「アクティビティ」を参照してください。

削除されたグループのスケーリングアクティビティの取得方法と、発生する可能性のあるエラーの種類と処理方法の詳細については、「Amazon EC2 Auto Scaling の問題のトラブルシューティング」を参照してください。

Auto Scaling グループのスケーリングポリシーを無効化する

このトピックでは、Auto Scaling グループに含まれるインスタンス数の変更を開始しないように、スケーリングポリシーを一時的に無効にする方法について説明します。スケーリングポリシーを無効に

すると、設定の詳細が保持されるため、ポリシーをすばやく再度有効にできます。これは、不要なポリシーを一時的に削除し、後で再作成するよりも簡単です。

スケーリングポリシーが無効になっている場合、スケーリングポリシーが無効になっている間は、違反したメトリクスアラームに対して Auto Scaling グループはスケールアウトまたはスケールインしません。ただし、進行中のスケーリングアクティビティは停止しません。

無効にしたスケーリングポリシーは、Auto Scaling グループに追加できるスケーリングポリシーの数のクォータにカウントされることに注意してください。

スケーリングポリシーを無効にするには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Automatic scaling] (オートスケーリング) タブの、[Dynamic scaling policies] (動的スケーリングポリシー) で、目的のスケーリングポリシーの右上隅にあるチェックボックスをオンにします。
4. 画面の上部までスクロールし、[Dynamic scaling policies] (動的スケーリングポリシー) セクションを選択した後、[Actions] (アクション)、[Disable] (無効) の順にクリックします。

スケーリングポリシーを再度有効にする準備ができたら、これらのステップを繰り返し、[Actions (アクション)]、[Enable (有効にする)] の順に選択します。スケーリングポリシーを再度有効にすると、現在ALARM状態にあるアラームがある場合、Auto Scaling グループはすぐにスケーリングアクションを開始できます。

スケーリングポリシーを無効にするには (AWS CLI)

次のように、put-scaling-policy コマンドを --no-enabled オプションと共に使用します。ポリシーの作成時に指定するオプションと同様に、コマンド内のすべてのオプションを指定します。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
   --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
   --estimated-instance-warmup 360 \
   --target-tracking-configuration '{ "TargetValue": 70,
 "PredefinedMetricSpecification": { "PredefinedMetricType":
 "ASGAverageCPUUtilization" } }' \
   --no-enabled
```

スケーリングポリシーを再度有効にするには (AWS CLI)

次のように、put-scaling-policy コマンドを --enabled オプションと共に使用します。ポリシーの作成時に指定するオプションと同様に、コマンド内のすべてのオプションを指定します。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
   --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
   --estimated-instance-warmup 360 \
   --target-tracking-configuration '{ "TargetValue": 70,
 "PredefinedMetricSpecification": { "PredefinedMetricType":
 "ASGAverageCPUUtilization" } }' \
   --enabled
```

スケーリングポリシーを説明するには (AWS CLI)

describe-policies コマンドを使用して、スケーリングポリシーの有効なステータスを確認します。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg \
   --policy-names my-scaling-policy
```

以下は出力例です。

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "my-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-9ca53fdd-7cf5-4223-938a-ae1199204502",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-
ae1199204502"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c",
```

```
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 70.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
}
```

Auto Scaling グループのスケーリングポリシーを削除する

不要になった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のタイプによっては、CloudWatch アラームを削除する必要がある場合もあります。ターゲット追跡スケーリングポリシーを削除すると、関連する CloudWatch アラームもすべて削除されます。ステップスケーリングポリシーまたは簡易スケーリングポリシーを削除すると、基盤となるアラームアクションは削除されますが、関連付けられたアクションがなくなった場合でも CloudWatch アラームは削除されません。

スケーリングポリシーを削除するには (コンソール)

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Automatic scaling] (オートスケーリング) タブの、[Dynamic scaling policies] (動的スケーリングポリシー) で、目的のスケーリングポリシーの右上隅にあるチェックボックスをオンにします。
4. 画面の上部までスクロールし、[Dynamic scaling policies] (動的スケーリングポリシー) セクションを選択した後、[Actions] (アクション)、[Delete] (削除) の順にクリックします。
5. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

6. （オプション) ステップスケーリングポリシーまたは簡易スケーリングポリシーを削除した場合は、次の手順を実行して、ポリシーに関連付けられた CloudWatch アラームを削除します。今後使用できるように、このサブステップをスキップしてアラームを保持することもできます。

   a. で CloudWatch コンソールを開きますhttps://console.aws.amazon.com/cloudwatch/。
   b. ナビゲーションペインで、[アラーム] を選択します。
   c. アラームを選び (`Step-Scaling-AlarmHigh-AddCapacity` など)、[Action (アクション)]、[Delete (削除)] を選択します。
   d. 確認を求めるメッセージが表示されたら、[削除] を選択します。

Auto Scaling グループのスケーリングポリシーを取得するには (AWS CLI)

スケーリングポリシーを削除する前に、次の describe-policies を使用して、Auto Scaling グループに対して作成されたスケーリングポリシーを確認します。この出力は、ポリシーおよび CloudWatch アラームを削除するときに使用できます。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

`--query` パラメータを使用して、スケーリングポリシーのタイプで結果をフィルタリングできます。この `query` の構文は Linux または macOS で動作します。Windows では、一重引用符を二重引用符に変更します。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

以下は出力例です。

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "PolicyName": "cpu50-target-tracking-scaling-policy",
        "PolicyARN": "PolicyARN",
        "PolicyType": "TargetTrackingScaling",
        "StepAdjustments": [],
        "Alarms": [
            {
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
```

```
                "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
            },
            {
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
                "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
            }
        ],
        "TargetTrackingConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ASGAverageCPUUtilization"
            },
            "TargetValue": 50.0,
            "DisableScaleIn": false
        },
        "Enabled": true
    }
]
```

スケーリングポリシーを削除するには (AWS CLI)

次の [delete-policy] コマンドを使用します。

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \
  --policy-name cpu50-target-tracking-scaling-policy
```

CloudWatch アラームを削除するには (AWS CLI)

ステップおよび簡易スケーリングポリシーの場合は、delete-alarms コマンドを使用して、ポリシーに関連付けられた CloudWatch アラームを削除します。今後使用できるように、このステップをスキップしてアラームを保持することもできます。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-AddCapacity アラームおよび Step-Scaling-AlarmLow-RemoveCapacity アラームを削除します。

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-
Scaling-AlarmLow-RemoveCapacity
```

AWS CLIのスケーリングポリシーの例

Amazon EC2 Auto Scaling のスケーリングポリシーは、、 AWS Command Line Interface （ AWS CLI ）AWS Management Console、または を使用して作成できますSDKs。

次の例は、 コマンドを使用して AWS CLI put-scaling-policy Amazon EC2 Auto Scaling のスケーリングポリシーを作成する方法を示しています。*user input placeholder* を、ユーザー自身の情報に置き換えます。

を使用してスケーリングポリシーの作成を開始するには AWS CLI、ターゲット追跡スケーリングポリシー「」および「」の入門演習を参照してくださいステップスケーリングポリシーおよび簡易スケーリングポリシー。

例 1: 事前定義されたメトリクス指定を使用してターゲット追跡スケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ASGAverageCPUUtilization"
  }
}
```

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

> **Note**
>
> ファイルが現在のディレクトリにない場合は、ファイルへのフルパスを入力します。ファイルからの AWS CLI パラメータ値の読み取りの詳細については、「 AWS Command Line Interface ユーザーガイド」の「ファイルからのパラメータのロード AWS CLI」を参照してください。

例 2: カスタマイズされたメトリクス仕様を使用してターゲット追跡スケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy
 \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [{
      "Name": "MyOptionalMetricDimensionName",
      "Value": "MyOptionalMetricDimensionValue"
    }],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

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

例 3: スケールアウトにのみターゲット追跡スケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy
 \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
  },
  "DisableScaleIn": true
}
```

例 4: スケールアウトにステップスケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
```

```
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
                     MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

ポリシーの Amazon リソースネーム () を記録しますARN。 CloudWatch アラームを作成するARNときは、 が必要です。

例 5: スケールインにステップスケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

ポリシーの Amazon リソースネーム () を記録しますARN。 CloudWatch アラームを作成するARNときは、 が必要です。

例 6: スケールアウトに単純なスケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity --min-adjustment-magnitude 2
```

ポリシーの Amazon リソースネーム () を記録しますARN。 CloudWatch アラームを作成するARNときは、 が必要です。

例 7: スケールに簡易スケーリングポリシーを適用するには

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

ポリシーの Amazon リソースネーム () を記録しますARN。 CloudWatch アラームを作成するARNときは、 が必要です。

Amazon EC2 Auto Scaling の予測スケーリング

予測スケーリングは、負荷の履歴データを分析して、トラフィックフローの日ごとまたは週ごとのパターンを検出することで機能します。この情報を使用して将来のキャパシティのニーズを予測し、Amazon EC2 Auto Scaling が予想される負荷に合わせて Auto Scaling グループのキャパシティをプロアクティブに増やすことができるようにします。

予測スケーリングは、次のような状況に適しています。

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

一般に、トラフィックが増加する規則的なパターンがあり、アプリケーションの初期化に長い時間がかかる場合は、予測スケーリングの使用を検討する必要があります。反応的な性質を持つ動的スケーリングのみを使用する場合と比較して、予測スケーリングを使用すると、予測される負荷に先立ってキャパシティーを起動することで、より迅速にスケーリングできます。予測スケーリングでは、キャパシティを過剰にプロビジョニングする必要がないので、EC2 課金の費用を節約できる可能性もあります。

例えば、営業時間中の使用率が高く、夜間の使用量が少ないアプリケーションを考えてみましょう。各営業日の開始時に、予測スケーリングにより、トラフィックが最初に流入する前にキャパシティーを追加できます。これにより、使用率の低い期間から高い使用率の期間に移行するときに、アプリケーションの高可用性とパフォーマンスを維持するのに役立ちます。トラフィックの変化に動的スケーリングが反応するのを待つ必要はありません。また、アプリケーションの負荷パターンを確認し、スケジュールされたスケーリングを使用して適切なキャパシティーをスケジュールしようと時間を費やす必要はありません。

トピック

予測スケーリングの仕組み

このトピックでは、予測スケーリングの仕組みと、予測スケーリングポリシーを作成するときに考慮すべき点について説明します。

トピック

仕組み

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

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

最初に予測スケーリングを有効にしたときは、予測専用モードで実行されます。このモードでは、キャパシティ予測が生成されますが、実際には、それらの予測に基づいて Auto Scaling グループをスケールすることはありません。これにより、予測の正確性と適合性を評価できます。`GetPredictiveScalingForecast` API オペレーションまたは AWS Management Console を使用して、予測データを表示できます。

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

- 予測で負荷の増加が予想される場合、Amazon EC2 Auto Scaling はスケールアウトによってキャパシティを増やします。
- 予測で負荷の減少が予想される場合、キャパシティを削除するためのスケールインは行いません。不要になったキャパシティを削除する場合は、動的スケーリングポリシーを作成する必要があります。

デフォルトでは、Amazon EC2 Auto Scaling は、1 時間ごとの予測に基づいて、その時間の開始時に Auto Scaling グループをスケールします。必要に応じて、PutScalingPolicy API オペレーションの SchedulingBufferTime プロパティ、または AWS Management Console の [起動前のインスタンス] 設定を使用して、より早い開始時刻を指定できます。これにより、Amazon EC2 Auto Scaling は予測された需要よりも先に新しいインスタンスを起動し、インスタンスが立ち上がってトラフィックを処理する準備が整うまでの時間を確保します。

予測された需要より前に新しいインスタンスを起動できるように、Auto Scaling グループのデフォルトのインスタンスウォームアップを有効にすることを強くお勧めします。これは、スケールアウトアクティビティの後で、動的スケーリングポリシーがキャパシティを減らす必要があることを示している場合でも、Amazon EC2 Auto Scaling がスケールインしない期間を指定します。これにより、新しく起動したインスタンスが、スケールインオペレーションの対象となる前に、増加したトラフィックの処理を開始するのに十分な時間を確保できます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

最大キャパシティの制限

Auto Scaling グループには、グループに対して起動できる EC2 インスタンスの最大数を制限する最大キャパシティ設定があります。デフォルトでは、スケーリングポリシーが設定されている場合、最大キャパシティを超えてキャパシティを増やすことはできません。

それ以外の方法として、予測キャパシティが Auto Scaling グループの最大キャパシティに近づいた場合や、それを超えた場合に、グループの最大キャパシティを自動的に増やすことができます。この動作を有効にするには、PutScalingPolicy API オペレーションの MaxCapacityBreachBehavior および MaxCapacityBuffer プロパティ、または AWS Management Console の [最大キャパシティーの動作] 設定を使用します。

> **Warning**
>
> 最大キャパシティを自動的に増やす場合は注意してください。これにより、増加した最大キャパシティがモニタリングおよび管理されていない場合、意図したよりも多くのインスタンスが起動される可能性があります。その後、増加した最大キャパシティは、手動で更新するまで Auto Scaling グループの新しい通常の最大キャパシティになります。最大キャパシティは、自動的に元の最大キャパシティまで減少しません。

考慮事項

- 予測スケーリングがワークロードに適しているかどうかを確認します。ワークロードは、曜日または時刻に固有の定期的な負荷パターンを示す場合に、予測スケーリングに適しています。これを確認するには、[予測のみ] モードで予測スケーリングポリシーを設定し、コンソールの推奨事項を参照してください。Amazon EC2 Auto Scaling は、潜在的なポリシーのパフォーマンスに関する観察内容に基づいた推奨事項を提供します。予測スケーリングがアプリケーションをアクティブにスケーリングできるようにする前に、予測および推奨事項を評価します。
- 予測スケーリングでは、予測を開始するには 24 時間以上の履歴データが必要です。ただし、履歴データが 2 週間あれば予測がより効果的です。新しい Auto Scaling グループを作成し、古いグループを削除してアプリケーションを更新する場合、予測スケーリングが再び予測の生成を開始するには、新しい Auto Scaling グループに 24 時間の履歴負荷データが必要です。カスタムメトリクスを使用して新旧の Auto Scaling グループ全体のメトリクスを集計できます。そうでない場合、より正確な予測を得るために数日かかる場合があります。
- アプリケーションのすべての負荷を正確に表し、スケーリングが最も重要なアプリケーションの側面である負荷メトリクスを選択します。
- 動的スケーリングを予測スケーリングと組み合わせて使用すると、アプリケーションの需要曲線に忠実に従って、トラフィックが少ない時間帯にスケールインし、トラフィックが予想を上回る場合はスケールアウトできます。複数のスケーリングポリシーがアクティブな場合、各ポリシーによって希望するキャパシティーが個別に決定され、希望するキャパシティーはそれらの最大値に設定されます。例えば、ターゲット追跡スケーリングポリシーでターゲット使用率を維持するために 10 インスタンスが必要で、予測スケーリングポリシーでターゲット使用率を維持するために 8 つのインスタンスが必要である場合、グループの希望するキャパシティーは 10 に設定されます。動的スケーリングを初めて使用する場合は、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。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。
- 予測スケーリングの中核的な前提は、Auto Scaling グループが同種構成であり、すべてのインスタンスの容量が同じであるということです。これがグループに当てはまらない場合、予測された容量が正確ではない可能性が生じます。このため、混合型のインスタンスグループ向けに予測スケーリングポリシーを作成するときは注意が必要です。キャパシティが同じではない異なるタイプのインスタンスがプロビジョニングされる可能性があるからです。以下は、予測された容量が不正確になる場合の例です。
  - 予測スケーリングポリシーが CPU 使用率に基づいているのに、各 Auto Scaling インスタンスの vCPU の数がインスタンスタイプに応じて異なる。

- 予測スケーリングポリシーがネットワークインまたはネットワークアウトに基づいているのに、各 Auto Scaling インスタンスのネットワーク帯域幅のスループットがインスタンスタイプに応じて異なる。例えば、M5 と M5n インスタンスタイプは類似していますが、M5n インスタンスタイプの方が大幅に高いネットワークスループットを提供します。

サポートされるリージョン

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- ヨーロッパ (パリ)
- ヨーロッパ (ストックホルム)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)

- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWS GovCloud (米国西部)

Auto Scaling グループの予測スケーリングポリシーを作成する

次の手順は、AWS Management Console または AWS CLI を使用して予測スケーリングポリシーを作成するのに役立ちます。

Auto Scaling グループが新しい場合に Amazon EC2 Auto Scaling がグループの予測を生成するには、そのグループが少なくとも 24 時間分のデータを提供する必要があります。

内容

予測スケーリングポリシーを作成する (コンソール)

予測スケーリングポリシーを初めて作成する場合は、コンソールを使用して予測専用モードで複数の予測スケーリングポリシーを作成することをお勧めします。これにより、異なるメトリクスおよび目標値の潜在的な影響をテストできます。Auto Scaling グループごとに複数の予測スケーリングポリシーを作成できますが、アクティブなスケーリングに使用できるポリシーは 1 つだけです。

コンソールで予測スケーリングポリシーを作成する (事前定義されたメトリクス)

事前定義されたメトリクス (CPU、ネットワーク I/O、またはターゲットあたりの Application Load Balancer リクエスト数) を使用して予測スケーリングポリシーを作成するには、以下の手順を実行します。予測スケーリングポリシーを作成する最も簡単な方法は、事前定義されたメトリクスを使用することです。その代わりにカスタムメトリクスを使用する場合は、「コンソールで予測スケーリングポリシーを作成する (カスタムメトリクス)」を参照してください。

予測スケーリングポリシーを作成する

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部に分割されたペインが開きます。

3. [Automatic scaling] (自動スケーリング) タブの [Scaling policies] (スケーリングポリシー) で、[Create predictive scaling policy] (予測スケーリングポリシーの作成) を選択します。
4. ポリシーの名前を入力します。
5. Amazon EC2 Auto Scaling にすぐにスケーリングを開始させるには、[Scale based on forecast] (予測に基づくスケーリング) をオンにします。

   ポリシーを予測のみモードにしておくには、[Scale based on forecast] (予測に基づくスケーリング) をオフのままにします。
6. [Metrics] (メトリクス) で、オプションのリストからメトリクスを選択します。オプションには、[CPU]、[Network In] (ネットワーク入力)、[Network Out] (ネットワーク出力)、[Application Load Balancer request count] (Application Load Balancer リクエスト数)、および [Custom metric pair] (カスタムメトリクスペア) が含まれます。

   [Application Load Balancer request count per target] (ターゲットあたりの Application Load Balancer リクエスト数) を選択した場合、[Target group] (ターゲットグループ) のターゲットグループを選択します。[Application Load Balancer request count per target] (ターゲットあたりの Application Load Balancer リクエスト数) は、Application Load Balancer ターゲットグループを Auto Scaling グループにアタッチしている場合にのみサポートされます。

   [Custom metric pair] (カスタムメトリクスペア) を選択した場合、[Load metric] (負荷のメトリクス) と [Scaling metric] (スケーリングのメトリクス) のドロップダウンリストから個々のメトリクスを選択します。
7. [Target utilization] (ターゲット使用率) に、Amazon EC2 Auto Scaling が維持する必要があるターゲット値を入力します。Amazon EC2 Auto Scaling は、平均使用率がターゲット使用率になるまで、または指定したインスタンスの最大数に達するまで、キャパシティーをスケールアウトします。

スケーリングメトリクスが以下である場合..。	ターゲット使用率は以下の内容になります。
CPU	各インスタンスが使用する CPU の理想的な割合。
ネットワーク入力	各インスタンスが受信する理想的な 1 分あたりの平均バイト数。
ネットワーク出力	各インスタンスが送信する理想的な 1 分あたりの平均バイト数。

スケーリングメトリクスが以下である場合..。	ターゲット使用率は以下の内容になります。
ターゲットあたりの Application Load Balancer リクエスト数	各インスタンスが受信する理想的な 1 分あたりの平均リクエスト数。

8. (オプション) [起動前のインスタンス] で、予測によって負荷の増加が必要とされる際、どれくらい前にインスタンスを起動しておくかを選択します。
9. (オプション) [Max capacity behavior] (最大容量の動作) で、予測されたキャパシティーが定義された最大値を超える場合に、Amazon EC2 Auto Scaling がグループの最大容量を超えてスケールアウトできるようにするかどうかを選択します。この設定をオンにすると、トラフィックが最高になると予測される期間中にスケールアウトが実行されます。
10. (オプション) [Buffer maximum capacity above the forecasted capacity] (予測キャパシティーを超える最大キャパシティーのバッファ) で、予測キャパシティーが最大キャパシティーに近づいたか、それを超えたときに使用する追加キャパシティーを選択します。この値は予測キャパシティーに対する割合 (%) として指定されます。たとえば、バッファが 10 の場合、バッファは 10% です。従って、予測キャパシティーが 50 で最大キャパシティーが 40 の場合、有効な最大キャパシティーは 55 です。

    これを 0 に設定すると、Amazon EC2 Auto Scaling は最大キャパシティーを超えてスケールすることはできても、予測キャパシティーまでとなり、それを超えることはできません。
11. [Create predictive scaling policy] (予測スケーリングポリシーを作成) を選択します。

コンソールで予測スケーリングポリシーを作成する (カスタムメトリクス)

カスタムメトリクスを使用して予測スケーリングポリシーを作成するには、以下の手順を実行します。カスタムメトリクスには、CloudWatch が提供するその他メトリクス、またはユーザーが CloudWatch に発行するメトリクスを含めることができます。CPU、ネットワーク I/O、またはターゲットあたりの Application Load Balancer リクエスト数を使用するには、「コンソールで予測スケーリングポリシーを作成する (事前定義されたメトリクス)」を参照してください。

カスタムメトリクスを使用して予測スケーリングポリシーを作成するには、以下を実行する必要があります。

- Amazon EC2 Auto Scaling が CloudWatch のメトリクスとやり取りできるようにする未処理のクエリを提供する必要があります。詳細については、「カスタムメトリクスを使用した高度な予測ス

ケーリングポリシー」を参照してください。Amazon EC2 Auto Scaling が CloudWatch からメトリクスデータを抽出できることを確実にするため、各クエリがデータポイントを返していることを確認します。これは、CloudWatch コンソール、または CloudWatch GetMetricData API 操作を使用して確認します。

> **Note**
>
> Amazon EC2 Auto Scaling コンソールの JSON エディタには、サンプル JSON ペイロードが提供されています。これらの例では、AWS が提供するその他の CloudWatch メトリクス、またはユーザーが以前に CloudWatch に発行したメトリクスを追加するために必要なキーと値のペアを参照できます。これらを開始点として使用してから、必要に応じてカスタマイズすることができます。

- メトリクス計算を使用する場合は、独自のシナリオに適した JSON を手動で構築する必要があります。詳細については、「Metric Math 式を使用する」を参照してください。ポリシーでメトリクス計算を使用する前に、メトリクス数式に基づくメトリクスクエリが有効であり、単一の時系列を返すことを確認してください。これは、CloudWatch コンソール、または CloudWatch GetMetricData API 操作を使用して確認します。

誤ったデータ (間違った Auto Scaling グループ名など) を提供することによってクエリでエラーが発生した場合、予測にはデータがありません。カスタムメトリクス問題のトラブルシューティングについては、「予測スケーリングポリシーでのカスタムメトリクスに関する考慮事項」を参照してください。

予測スケーリングポリシーを作成する

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部に分割されたペインが開きます。
3. [Automatic scaling] (自動スケーリング) タブの [Scaling policies] (スケーリングポリシー) で、[Create predictive scaling policy] (予測スケーリングポリシーの作成) を選択します。
4. ポリシーの名前を入力します。
5. Amazon EC2 Auto Scaling にすぐにスケーリングを開始させるには、[Scale based on forecast] (予測に基づくスケーリング) をオンにします。

ポリシーを予測のみモードにしておくには、[Scale based on forecast] (予測に基づくスケーリング) をオフのままにします。

6. [Metrics] (メトリクス) では、[Custom metric pair] (カスタムメトリクスのペア) を選択します。

   a. [Load metric] (ロードメトリクス) では、[Custom CloudWatch metric] (カスタム CloudWatch メトリクス) を選択してカスタムメトリクスを使用します。ポリシーのロードメトリクス定義が含まれる JSON ペイロードを構築し、それを JSON エディタボックスに貼り付けて、ボックス内にすでに入力されているペイロードを置き換えます。

   b. [Scaling metric] (スケーリングメトリクス) では、[Custom CloudWatch metric] (カスタム CloudWatch メトリクス) を選択してカスタムメトリクスを使用します。ポリシーのスケーリングメトリクス定義が含まれる JSON ペイロードを構築し、それを JSON エディタボックスに貼り付けて、ボックス内にすでに入力されているペイロードを置き換えます。

   c. (オプション) カスタム容量メトリクスを追加するには、[Add custom capacity metric] (カスタム容量メトリクスを追加する) のチェックボックスをオンにします。ポリシーの容量メトリクス定義が含まれる JSON ペイロードを構築し、それを JSON エディタボックスに貼り付けて、ボックス内にすでに入力されているペイロードを置き換えます。

      このオプションを有効にする必要があるのは、容量メトリクスデータが複数の Auto Scaling グループにまたがる場合に容量の新しい時系列を作成するためのみです。この場合は、メトリクス計算を使用してデータを単一の時系列に集計する必要があります。

7. [Target utilization] (ターゲット使用率) に、Amazon EC2 Auto Scaling が維持する必要があるターゲット値を入力します。Amazon EC2 Auto Scaling は、平均使用率がターゲット使用率になるまで、または指定したインスタンスの最大数に達するまで、キャパシティーをスケールアウトします。

8. (オプション) [起動前のインスタンス] で、予測によって負荷の増加が必要とされる際、どれくらい前にインスタンスを起動しておくかを選択します。

9. (オプション) [Max capacity behavior] (最大容量の動作) で、予測されたキャパシティーが定義された最大値を超える場合に、Amazon EC2 Auto Scaling がグループの最大容量を超えてスケールアウトできるようにするかどうかを選択します。この設定をオンにすると、トラフィックが最高になると予測される期間中にスケールアウトが実行されます。

10. (オプション) [Buffer maximum capacity above the forecasted capacity] (予測キャパシティーを超える最大キャパシティーのバッファ) で、予測キャパシティーが最大キャパシティーに近づいたか、それを超えたときに使用する追加キャパシティーを選択します。この値は予測キャパシティーに対する割合 (%) として指定されます。たとえば、バッファが 10 の場合、バッファは

10% です。従って、予測キャパシティーが 50 で最大キャパシティーが 40 の場合、有効な最大キャパシティーは 55 です。

これを 0 に設定すると、Amazon EC2 Auto Scaling は最大キャパシティーを超えてスケールすることはできても、予測キャパシティーまでとなり、それを超えることはできません。

11. [Create predictive scaling policy] (予測スケーリングポリシーを作成) を選択します。

予測スケーリングポリシーの作成 (AWS CLI)

次のように AWS CLI を使用して Auto Scaling グループの予測スケーリングポリシーを設定します。各##############を独自の情報に置き換えます。

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

例 1: 予測を作成するが、スケーリングしない予測スケーリングポリシー

次のポリシー例では、予測スケーリングに CPU 使用率メトリクスを使用し、ターゲット使用率が 40 である完全なポリシー設定を示しています。使用するモードを明示的に指定しない限り、デフォルトで ForecastOnly モードが使用されます。この設定を config.json という名前のファイルに保存してください。

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ]
}
```

コマンドラインからこのポリシーを作成するには、以下の例にあるように、設定ファイルが指定された put-scaling-policy コマンドを実行します。

```
aws autoscaling put-scaling-policy --policy-name cpu40-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

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

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/cpu40-predictive-scaling-
policy",
  "Alarms": []
}
```

例 2: 予測とスケーリングを行う予測スケーリングポリシー

Amazon EC2 Auto Scaling の予測およびスケーリングを許可するポリシーには、Mode プロパティを ForecastAndScale の値で追加します。次の例は、Application Load Balancer リクエスト数メトリクスを使用するポリシー設定を示しています。ターゲット使用率は 1000 で、予測スケーリングは ForecastAndScale モードに設定されています。

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 1000,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ALBRequestCount",
                "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
            }
        }
    ],
    "Mode": "ForecastAndScale"
}
```

このポリシーを作成するには、次の例に示すように、設定ファイルを指定して put-scaling-policy コマンドを実行します。

```
aws autoscaling put-scaling-policy --policy-name alb1000-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

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

```
{
```

```
  "PolicyARN": "arn:aws:autoscaling:region:account-
id:scalingPolicy:19556d63-7914-4997-8c81-d27ca5241386:autoScalingGroupName/my-
asg:policyName/alb1000-predictive-scaling-policy",
  "Alarms": []
}
```

例 3: 最大キャパシティーを超えてスケーリングできる予測スケーリングポリシー

次の例は、通常よりも高い負荷を処理する必要がある場合に、グループの最大サイズ制限を超えてスケーリングできるポリシーを作成する方法を示しています。デフォルトでは、Amazon EC2 Auto Scaling は、定義した最大キャパシティーを超えて EC2 のキャパシティーをスケーリングしません。しかし、パフォーマンスや可用性の問題を回避するために、キャパシティーを少し増やしてスケーリングすると便利な場合があります。

キャパシティーがグループの最大サイズに達している、または非常に近いと予測されるときに、Amazon EC2 Auto Scaling が追加のキャパシティーをプロビジョニングする余地を提供するには、次の例に示すように、`MaxCapacityBreachBehavior` および `MaxCapacityBuffer` プロパティを指定します。`MaxCapacityBreachBehavior` に値 `IncreaseMaxCapacity` を指定する必要があります。グループに含めることができるインスタンスの最大数は、`MaxCapacityBuffer` の値によって異なります。

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 70,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ],
    "MaxCapacityBreachBehavior": "IncreaseMaxCapacity",
    "MaxCapacityBuffer": 10
}
```

この例では、10% のバッファ (`"MaxCapacityBuffer": 10`) を使用するようにポリシーが設定されています。したがって、予測キャパシティーが 50、最大キャパシティーが 40 の場合、有効な最大キャパシティーは 55 です。キャパシティーを最大キャパシティーを超えてスケーリングし、予測キャパシティーに等しくするが、予測キャパシティーを超えないようにするポリシーでは、バッファは 0 です (`"MaxCapacityBuffer": 0`)。

このポリシーを作成するには、次の例に示すように、設定ファイルを指定して put-scaling-policy コマンドを実行します。

```
aws autoscaling put-scaling-policy --policy-name cpu70-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

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

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:d02ef525-8651-4314-
bf14-888331ebd04f:autoScalingGroupName/my-asg:policyName/cpu70-predictive-scaling-
policy",
  "Alarms": []
}
```

予測スケーリングポリシーの評価

予測スケーリングポリシーを使用して Auto Scaling グループをスケーリングする前に、Amazon EC2 Auto Scaling コンソールでポリシーの推奨事項やその他のデータを確認します。予測が正確であることがわかるまで、予測スケーリングポリシーが実際のキャパシティをスケーリングすることが好ましくないため、これは重要です。

Auto Scaling グループが新しい場合、Amazon EC2 Auto Scaling が最初の予測を作成するために 24 時間かかります。

Amazon EC2 Auto Scaling が予測を作成するとき、履歴データを使用します。Auto Scaling グループにまだ最新の履歴データがない場合、Amazon EC2 Auto Scaling は現在利用可能な履歴集計から作成された集計で予測を一時的にバックフィルすることがあります。予測は、ポリシーの作成日より最大 2 週間前にバックフィルされます。

内容

- 予測スケーリングの推奨事項の表示
- 予測スケーリングのモニタリンググラフの確認
- CloudWatch による予測スケーリングメトリクスのモニタリング

予測スケーリングの推奨事項の表示

分析を効果的に行うには、Amazon EC2 Auto Scaling に比較対象となる予測スケーリングポリシーが少なくとも 2 つ必要です。(ただし、単一ポリシーの結果を確認することはできます) 複数のポリシーを作成するとき、1 つのメトリクスを使用するポリシーを異なるメトリクスを使用するポリシーと比較して評価できます。異なる目標値およびメトリクスの組み合わせによる影響を評価することもできます。予測スケーリングポリシーが作成された後、Amazon EC2 Auto Scaling は、どのポリシーがグループのスケーリングに適しているかについて、すぐに評価を開始します。

Amazon EC2 Auto Scaling コンソールで推奨事項を表示するには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [予測スケーリングポリシー] の [Auto Scaling] タブで、ポリシーの詳細および推奨事項を表示できます。推奨事項は、予測スケーリングポリシーを使用しない場合よりも優れた性能を発揮するかどうかについて説明します。

   予測スケーリングポリシーがグループに適切かどうかが不明な場合、[可用性への影響] 列および [コストへの影響] 列を確認し、適切なポリシーを選択してください。各列の情報は、ポリシーの影響について説明します。

   - [可用性への影響]: ポリシーを使用しない場合と比較し、ワークロードを処理するために十分なインスタンスをプロビジョニングすることにより、ポリシーが可用性への悪影響を回避できるかどうかについて説明します。
   - [コストへの影響]: ポリシーを使用しない場合と比較し、インスタンスをオーバープロビジョニングしないことにより、ポリシーがコストへの悪影響を回避できるかどうかについて説明します。過度にオーバープロビジョニングすることにより、インスタンスが十分に活用されない、あるいはアイドル状態になり、コストへの影響が増す一方です。

   複数のポリシーがある場合、より低コストで最も可用性のメリットが高いポリシーの名前の横に [最良予測] タグが表示されます。可用性への影響がより重視されます。
4. (オプション) 推奨結果の必要な期間を選択するには、[評価期間] のドロップダウンから [2 日]、[1 週間]、[2 週間]、[4 週間]、[6 週間]、[8 週間] のいずれから希望する値を選択します。デフォルトでは、評価期間は過去の 2 週間です。評価期間が長いほど、推奨結果のデータポイン

トが増えます。ただし、需要が非常に高い時期の後など、負荷パターンが変化した場合、データポイントを追加しても結果が改善されない可能性があります。この場合、最新のデータを見ることでより焦点を絞った推奨事項を得ることができます。

> **Note**
>
> 推奨事項は [予測のみ] モードのポリシーに対してのみ生成されます。推奨機能は、評価期間中にポリシーが [予測のみ] モードのときにより効果的に機能します。ポリシーを [予測とスケーリング] モードで開始し、後で [予測のみ] モードに切り替える場合、そのポリシーの結果に偏りが生じる可能性があります。これは、ポリシーが既に実際のキャパシティに関与しているためです。

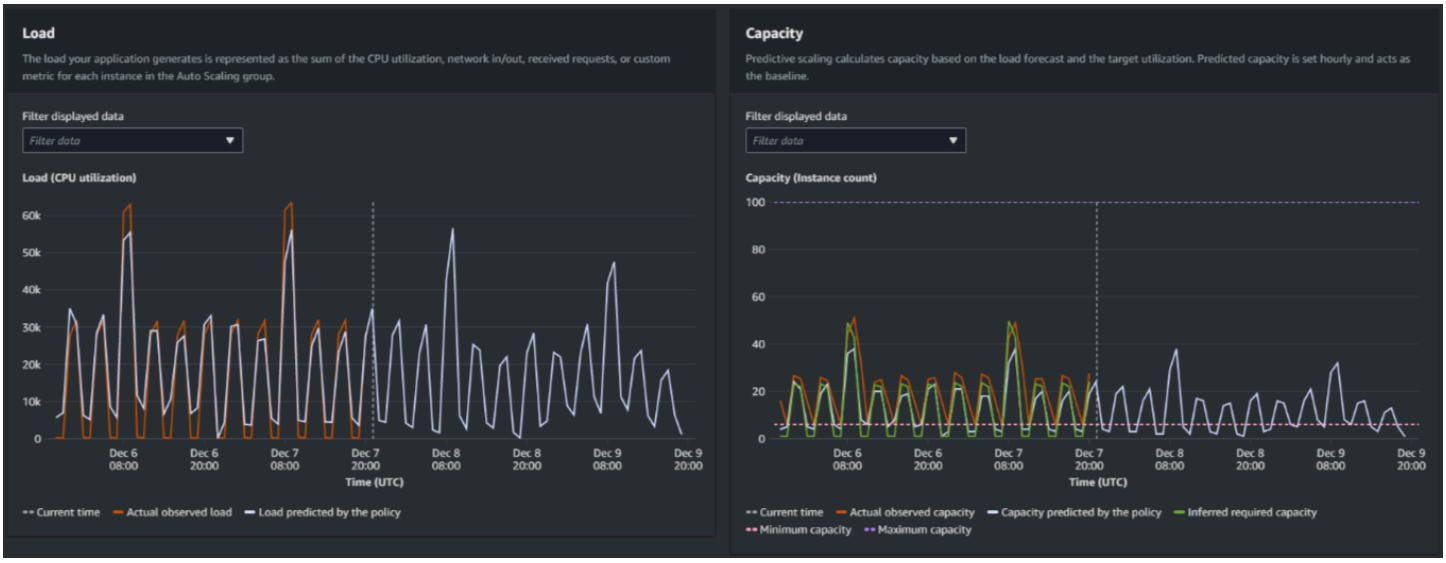
予測スケーリングのモニタリンググラフの確認

Amazon EC2 Auto Scaling コンソールでは、以前の日、週間、月の予測を確認し、時間の経過とともにポリシーがどの程度機能しているか視覚化できます。ポリシーが実際のキャパシティをスケーリングするかどうかを決定するとき、この情報を使用して予測の精度を評価できます。

Amazon EC2 Auto Scaling コンソールで予測スケーリングのモニタリンググラフを表示するには

1. [予測スケーリングポリシー] リストからポリシーを選択します。
2. [モニタリング] セクションでは、ポリシーの負荷およびキャパシティに関する過去および今後の予測を実際の値と比較できます。[負荷] グラフには、選択した負荷メトリクスの負荷予測および実際の値が表示されます。[キャパシティ] グラフには、ポリシーによって予測されたインスタンスの数が表示されます。実際に起動されたインスタンスの数も含まれます。縦線は履歴の値と今後の予測を区切っています。これらのグラフは、ポリシーの作成後にすぐ利用できます。
3. (オプション) グラフに表示される履歴データの量を変更するには、ページ上部の [評価期間] のドロップダウンから希望する値を選択します。評価期間はこのページのデータをはいかなる方法で変換することはありません。表示される履歴データの量のみを変更します。

次の画像は、予測が複数回適用された場合の [負荷] グラフおよび [キャパシティ] グラフを表示しています。予測スケーリングは、履歴の負荷データに基づいて負荷を予測します。アプリケーションが生成する負荷は、Auto Scaling グループの各インスタンスの CPU 使用率、ネットワークの入出力、受信したリクエスト、カスタムメトリクスのいずれかの合計として表されます。予測スケーリングは、負荷予測およびスケーリングメトリクスで達成したい目標の使用率に基き、今後のキャパシティのニーズを計算します。



[負荷] グラフのデータを比較

各水平線は、1 時間間隔で報告される異なる一連のデータポイントを表しています。

1. [実際に観測された負荷] は、選択した負荷メトリクスの SUM 統計を使用し、過去の 1 時間ごとの合計負荷を表示します。
2. [ポリシーによって予測される負荷] は、1 時間ごとの負荷予測を表示します。この予測は過去 2 週間分の実際の負荷観測に基づいています。

[キャパシティ] グラフのデータの比較

各水平線は、1 時間間隔で報告される異なる一連のデータポイントを表しています。

1. [実際に観測されたキャパシティ] には、Auto Scaling グループの過去の実際のキャパシティが表示されます。これは、他のスケーリングポリシーや、選択した期間に有効な最小グループサイズによって異なります。
2. [ポリシーによって予測されるキャパシティ] には、ポリシーが [予測とスケーリング] モードになっているときに各時間の開始時に予想されるベースラインキャパシティが表示されます。
3. [推定必要容量は] には、スケーリング指標を選択した目標値に維持するための理想的な容量を表示します。
4. [最小容量] には、Auto Scaling グループの最小容量を表示します。
5. [最大容量] には、Auto Scaling グループの最大容量を表示します。

推定される必要キャパシティを計算するため、最初は各インスタンスが指定された目標値で均等に使用されていると仮定します。実際には、インスタンスは均等に使用されません。ただし、使用率がインスタンス間で均等に分散されていると仮定することにより、必要なキャパシティの量を推定できます。次に、キャパシティ要件は、予測スケーリングポリシーに使用したスケーリングメトリクスに反比例するように計算されます。つまり、キャパシティが増加するにつれ、スケーリングメトリクスは同じ割合で減少します。例えば、キャパシティが 2 倍になった場合、スケーリングメトリクスは半減します。

推定された必要キャパシティの計算式は、次のとおりです。

```
sum of (actualCapacityUnits*scalingMetricValue)/(targetUtilization)
```

例えば、特定の時間の `actualCapacityUnits` (`10`) および `scalingMetricValue` (`30`) を算出します。その後、予測スケーリングポリシー (`60`) で指定した `targetUtilization` を使用し、同じ時間に推定される必要キャパシティを計算します。これは `5` の値を返します。これは、スケーリングメトリクスの目標値とは正反比例し、キャパシティを維持するために必要なキャパシティの推定量は 5 であることを意味します。

> **Note**
>
> コスト削減およびアプリケーションの可用性を調整および改善するため、さまざまな手段が用意されています。
>
> - ベースラインキャパシティに予測スケーリングを使用し、追加のキャパシティに動的スケーリングを使用します。動的スケーリングは予測スケーリングとは独立して動作し、現在の使用率に基づいてスケールインおよびスケールアウトを行います。まず、Amazon EC2 Auto Scaling は、動的スケーリングポリシーごとに推奨されるインスタンスの数を計算します。次に、最も多くのインスタンスを提供するポリシーに基づいてスケーリングします。
> - 負荷が減少したときにスケールインを実行できるようにするには、Auto Scaling グループには、スケールイン部分を有効にした動的スケーリングポリシーが常に少なくとも 1 つ必要です。
> - 最小キャパシティおよび最大キャパシティを制限しすぎないようにすることにより、スケーリングパフォーマンスを向上させることができます。推奨されるインスタンスの数が最小キャパシティおよび最大キャパシティの範囲に収まらないポリシーは、スケールインおよびスケールアウトができなくなります。

CloudWatch による予測スケーリングメトリクスのモニタリング

必要に応じて、Amazon EC2 Auto Scaling コンソールではなく Amazon CloudWatch から予測スケーリングのモニタリングデータにアクセスすることもできます。予測スケーリングポリシーを作成すると、ポリシーは、今後の負荷とキャパシティを予測するために使用するデータを収集します。このデータは収集後、定期的かつ自動的に CloudWatch に保存されます。その後、CloudWatch を使用して、経時的なポリシーのパフォーマンスを視覚化できます。また、CloudWatch アラームを作成して、パフォーマンス指標が CloudWatch で定義した制限を超えて変化したときに通知させることもできます。

トピック

履歴予測データの視覚化

CloudWatch では、予測スケーリングポリシーの負荷とキャパシティ予測データを表示できます。これは、他の CloudWatch メトリクスに対する予測を 1 つのグラフで視覚化する場合に便利です。また、経時的な傾向を確認するために、より長い期間を表示することもできます。最大 15 か月間の履歴メトリクスにアクセスして、ポリシーの動作をより的確に把握できます。

詳細については、「予測スケーリングのメトリクスとディメンション」を参照してください。

CloudWatch コンソールを使用して履歴予測データを表示する方法

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[Metrics] (メトリクス)、[All metrics] (すべてのメトリクス) の順に選択します。
3. [Auto Scaling] メトリクス名前空間を選択します。
4. 以下のオプションのいずれかを選択して、負荷予測またはキャパシティ予測メトリクスのいずれかを表示します。
   - 予測スケーリングの負荷予測
   - 予測スケーリングのキャパシティ予測
5. 検索フィールドに、予測スケーリングポリシー名または Auto Scaling グループ名を入力し、Enter キーを押して結果をフィルタリングします。

6. メトリクスをグラフ表示するには、メトリクスの横にあるチェックボックスを選択します。グラフの名前を変更するには、鉛筆アイコンを選択します。時間範囲を変更するには、事前定義済みの値を選択するか、[custom] を選択します。詳細については、「Amazon CloudWatch ユーザーガイド」の「メトリクスのグラフ化」を参照してください。
7. 統計を変更するには、[Graphed metrics] タブを選択します。列見出しまたは個々の値を選択し、続いて各種統計を選択します。各メトリクスの任意の統計を選択できますが、すべての統計が PredictiveScalingLoadForecast および PredictiveScalingCapacityForecast メトリクスに有用なわけではありません。例えば、平均、最小、最大統計は有用ですが、合計統計は有用ではありません。
8. グラフに別のメトリクスを追加するには、[Browse] (参照) で [All] (すべて) を選択し、追加したいメトリクスを見つけて、その横にあるチェックボックスをオンにします。最大 10 個のメトリクスを追加できます。

   例えば、CPU 使用率の実際の値をグラフに追加するには、[EC2] 名前空間、[By Auto Scaling Group] (Auto Scaling グループ別) の順に選択します。次に、[CPUUtilization] メトリクス、および対象とする Auto Scaling グループのチェックボックスをオンにします。
9. (オプション) このグラフを CloudWatch ダッシュボードに追加するには、[Actions] (アクション)、[Add to dashboard] (ダッシュボードに追加) の順に選択します。

Metric Math を使用して精度メトリクスを作成する

Metric Math により、複数の CloudWatch メトリクスをクエリし、数式を使用して、これらのメトリクスに基づく新しい時系列を作成できます。作成された時系列を CloudWatch コンソールで可視化でき、ダッシュボードに追加できます。Metric Math の詳細については、「Amazon CloudWatch ユーザーガイド」の「Metric Math を使用する」を参照してください。

Metric Math を使用して、Amazon EC2 Auto Scaling が予測スケーリングのために生成するデータを各種の方法でグラフ化できます。これにより、ポリシーのパフォーマンスを経時的にモニタリングし、メトリクスの組み合わせを改善できるかどうかを把握することができます。

例えば、Metric Math 式を使用して、平均絶対パーセント誤差 (MAPE) をモニタリングできます。MAPE メトリクスは、予測値と、特定の予測期間中に観測された実際の値の差をモニタリングするのに役立ちます。MAPE の値の変化は、アプリケーションの性質が変化するにつれて、ポリシーのパフォーマンスが経時的に低下しているかどうかを示します。MAPE の増加は、予測値と実際の値の差が大きいことを示します。

例: Metric Math 式

このタイプのグラフを使用するには、次の例に示すような Metric Math 式を作成します。

```
{
  "MetricDataQueries": [
    {
      "Expression": "TIME_SERIES(AVG(ABS(m1-m2)/m1))",
      "Id": "e1",
      "Period": 3600,
      "Label": "MeanAbsolutePercentageError",
      "ReturnData": true
    },
    {
      "Id": "m1",
      "Label": "ActualLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/EC2",
          "MetricName": "CPUUtilization",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Sum"
      },
      "ReturnData": false
    },
    {
      "Id": "m2",
      "Label": "ForecastedLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/AutoScaling",
          "MetricName": "PredictiveScalingLoadForecast",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            },
            {
              "Name": "PolicyName",
```

```
              "Value": "my-predictive-scaling-policy"
            },
            {
              "Name": "PairIndex",
              "Value": "0"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Average"
      },
      "ReturnData": false
    }
  ]
}
```

単一のメトリクスではなく、MetricDataQueries 用のメトリクスデータクエリ構造の配列があります。MetricDataQueries の各項目は、メトリクスを取得するか、数式を実行します。最初の項目は、数式である e1 です。指定された式は、ReturnData パラメータを true に設定し、最終的に単一の時系列を生成します。他のすべてのメトリクスで、ReturnData 値は false です。

この例では、指定された式は実際の値と予測値を入力値として使用し、新しいメトリクス (MAPE) を返します。m1 は、実際の負荷値を含む CloudWatch メトリクスです (CPU 使用率が、my-predictive-scaling-policy という名前のポリシーに対して最初に指定された負荷メトリクスであると仮定)。m2 は、予測負荷値を含む CloudWatch メトリクスです。MAPE メトリクスの計算構文は次のとおりです。

(絶対値 ((実際の値 - 予測値)/(実際の値))) の平均

精度メトリクスを視覚化してアラームを設定する

精度メトリクスデータを視覚化するには、CloudWatch コンソールの [Metrics] (メトリクス) タブをクリックします。そこからデータをグラフ化できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch グラフへの数式の追加」を参照してください。

[Metrics] (メトリクス) セクションから、モニタリングしているメトリクスにアラームを設定することもできます。[Graphed metrics] (グラフ化したメトリクス) タブで、[Actions] (アクション) 列にある [Create alarm] (アラームを作成) アイコンをクリックします。[Create alarm] (アラームを作成) アイコンは小さなベルです。詳細および通知オプションについては、「Amazon CloudWatch ユーザーガイド」の「メトリクス数式に基づく CloudWatch アラームの作成」と「アラームの変更をユーザーに通知する」を参照してください。

GetMetricData および PutMetricAlarm を使用して、Metric Math によって計算し、その出力に基づいてアラームを作成することもできます。

予定されたアクションを使用して予測値を上書きする

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

例えば、予測されるキャパシティーを超える最小キャパシティーでスケジュールされたアクションを作成できます。実行中に、Amazon EC2 Auto Scaling は Auto Scaling グループの最小キャパシティーを更新します。予測スケーリングはキャパシティーを最適化するので、予測値を超える最小キャパシティーでスケジュールされたアクションが適用されます。これにより、キャパシティーが想定より少なくなるのを防ぎます。予測の上書きを停止するには、2 番目のスケジュールされたアクションを使用して、最小キャパシティーを元の設定に戻します。

次の手順では、将来の期間中の予測を上書きするステップを示します。

トピック

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

> **Important**
>
> このトピックでは、予測を上書きして、予測よりも大きなキャパシティにスケールしようとしていることを前提としています。予測スケーリングポリシーの干渉なしに一時的にキャパシティを減らす必要がある場合は、代わりに予測専用モードを使用します。予測専用モードでは、予測スケーリングは予測を生成し続けますが、自動的にキャパシティを増やすことはありません。その後、リソース使用率をモニタリングし、必要に応じてグループのサイズを手動で減らすことができます。手動スケーリングの詳細については、「Amazon EC2 Auto Scaling の手動スケーリング」を参照してください。

ステップ 1: (オプション) 時系列データを分析する

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

1. 予測を取得する

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

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

   get-predictive-scaling-forecast AWS CLI コマンドを使用して予測を取得するには、コマンドに次のパラメータを指定します。

   - Auto Scaling グループの名前を `--auto-scaling-group-name` パラメータに入力します。
   - ポリシーの名前を `--policy-name` パラメータに入力します。
   - 開始時刻を `--start-time` パラメータに入力して、指定した時刻以降の予測データのみが返されるようにします。
   - 終了時刻を `--end-time` パラメータに入力して、指定された時刻より前の予測データのみが返されるようにします。

```
aws autoscaling get-predictive-scaling-forecast --auto-scaling-group-name my-asg \
    --policy-name cpu40-predictive-scaling-policy \
    --start-time "2021-05-19T17:00:00Z" \
    --end-time "2021-05-19T23:00:00Z"
```

   成功すると、コマンドは次の例のようなデータを返します。

```
{
    "LoadForecast": [
        {
            "Timestamps": [
                "2021-05-19T17:00:00+00:00",
                "2021-05-19T18:00:00+00:00",
                "2021-05-19T19:00:00+00:00",
                "2021-05-19T20:00:00+00:00",
```

```
                "2021-05-19T21:00:00+00:00",
                "2021-05-19T22:00:00+00:00",
                "2021-05-19T23:00:00+00:00"
            ],
            "Values": [
                153.0655799339254,
                128.8288551285919,
                107.1179447150675,
                197.3601844551528,
                626.4039934516954,
                596.9441277518481,
                677.9675713779869
            ],
            "MetricSpecification": {
                "TargetValue": 40.0,
                "PredefinedMetricPairSpecification": {
                    "PredefinedMetricType": "ASGCPUUtilization"
                }
            }
        }
    ],
    "CapacityForecast": {
        "Timestamps": [
            "2021-05-19T17:00:00+00:00",
            "2021-05-19T18:00:00+00:00",
            "2021-05-19T19:00:00+00:00",
            "2021-05-19T20:00:00+00:00",
            "2021-05-19T21:00:00+00:00",
            "2021-05-19T22:00:00+00:00",
            "2021-05-19T23:00:00+00:00"
        ],
        "Values": [
            2.0,
            2.0,
            2.0,
            2.0,
            4.0,
            4.0,
            4.0
        ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}
```

応答には LoadForecast と CapacityForecast の 2 つの予測が含まれています。LoadForecast は、時間ごとの負荷予測を示します。CapacityForecast は、40.0 の TargetValue (平均 CPU 使用率 40%) を維持しながら予測された負荷を処理するために時間単位で必要なキャパシティーの予測値を示します。

2. ターゲット期間を特定する

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

ステップ 2: 2 つのスケジュールされたアクションを作成する

次に、アプリケーションの負荷が予測を上回る特定の期間に、2 つのスケジュールされたアクションを作成します。例えば、マーケティングイベントで一時的にトラフィックがサイトに流入する場合は、1 回限りのアクションをスケジュールして、開始時に最小キャパシティーを更新できます。次に、イベント終了時に最小キャパシティーを元の設定に戻す別のアクションをスケジュールします。

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

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Automatic scaling (自動スケーリング)] タブの [Scheduled actions (スケジュールされたアクション)] で、[Create scheduled action (スケジュールされたアクションの作成)] を選択します。
4. 次のスケジュールされたアクション設定を入力します。

   a. スケジュールされたアクションに [Name (名前)] を入力します。
   b. [Min] (最小) に、Auto Scaling グループの新しい最小キャパシティーを入力します。[Min] (最小) は、グループの最大サイズ以下である必要があります。[Min] (最小) の新しい値が、グループの最大サイズよりも大きい場合、[Max] (最大) を更新する必要があります。
   c. [繰り返し] で、[1 回] を選択します。
   d. [Time zone (タイムゾーン)] でタイムゾーンを選択。タイムゾーンが選択されていない場合は、デフォルトでは、ETC/UTC が使用されます。
   e. [Specific start time] (特定の開始時刻) を定義します。
5. [Create] (作成) を選択します。

コンソールに Auto Scaling グループのスケジュールされたアクションが表示されます。

6. イベントの終了時に、最小キャパシティーを元の設定に戻すように、2 番目のスケジュールされたアクションを設定します。予測スケーリングでは、[Min] (最小) に設定した値が予測値未満の場合のみ、キャパシティーをスケーリングできます。

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

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

例えば、5 月 19 日の午後 5 時から 8 時間、最小キャパシティーを 3 インスタンスに維持するスケジュールを定義しましょう。以下のコマンドは、このシナリオを実装する方法を示しています。

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

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

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

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

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

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

- 2021 年 5 月 20 日の午前 1 時 (UTC) に、2 番目のスケジュールされたアクションが実行されます。これにより、イベントの終了時に最小キャパシティーが元の設定に戻ります。

繰り返し起こるスケジュールに基づくスケーリング

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

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

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

```
30 6 * * 2
```

以下も参照してください。

スケジュールされたアクションを作成、一覧表示、編集、および削除する方法の詳細については、「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。

カスタムメトリクスを使用した高度な予測スケーリングポリシー

予測スケーリングポリシーでは、事前定義されたメトリクスまたはカスタムメトリクスを使用できます。カスタムメトリクスは、事前定義されたメトリクス (CPU、ネットワーク I/O、および Application Load Balancer のリクエスト数) ではアプリケーションの負荷が十分に反映できない場合に役立ちます。

カスタムメトリクスを使用して予測スケーリングポリシーを作成するときは、AWS が提供するその他の CloudWatch メトリクスを指定する、またはユーザーが定義して独自に発行するメトリクスを指定することができます。メトリクス計算を使用して既存のメトリクスを集計し、AWS が自動的に追跡しない新しい時系列に変換することもできます。新しい合計や平均の計算など、データの値を組み合わせることを、集計すると言います。結果のデータは集計と言います。

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

トピック

- ベストプラクティス
- 前提条件

- カスタムメトリクス用の JSON の構築
- 予測スケーリングポリシーでのカスタムメトリクスに関する考慮事項
- 制約事項

ベストプラクティス

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

- 負荷メトリクスの指定では、グループのキャパシティーに関係なく Auto Scaling グループ全体の負荷を表すメトリクスが最も有用です。
- スケーリングメトリクスの指定では、インスタンスあたりの平均スループットまたは使用率のメトリクスでスケールするのが最も有用です。
- スケーリングメトリクスは、キャパシティーに反比例する必要があります。つまり、Auto Scaling グループのインスタンス数が増加すると、スケーリングメトリクスはほぼ同じ割合で減少するようにします。予測スケーリングが期待どおりに動作するようにするには、負荷メトリクスとスケーリングメトリクスに強い相関がある必要もあります。
- ターゲット使用率は、スケーリングメトリクスのタイプと一致する必要があります。CPU 使用率を使用するポリシー設定の場合、これはパーセンテージのターゲットです。リクエスト数やメッセージ数など、スループットを使用するポリシー設定の場合、これは任意の 1 分間のインスタンスあたりのリクエスト数やメッセージ数のターゲットです。
- これらの推奨事項に従わない場合、予測される将来の時系列の値は、多くの場合、誤りになります。データが正しいことを確認するために、Amazon EC2 Auto Scaling コンソールで予測値を表示できます。または、予測スケーリングポリシーを作成した後、GetPredictiveScalingForecast API を呼び出して返された `LoadForecast` および `CapacityForecast` オブジェクトを検査します。
- 予測スケーリングがキャパシティーのアクティブスケーリングを開始する前に予測を評価できるように、予測のみモードで予測スケーリングを設定することを強くお勧めします。

前提条件

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

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

独自のメトリクスを発行するときは、少なくとも 5 分間隔の頻度でデータポイントを発行するようにしてください。Amazon EC2 Auto Scaling は、必要な期間の長さに基づいて CloudWatch からデータポイントを取得します。例えば、負荷メトリクスの指定では、時間単位のメトリクスを使用してアプリケーションの負荷を測定します。CloudWatch は、発行されたメトリクスデータを使用して、各 1 時間の期間内にタイムスタンプがあるすべてのデータポイントを集計することにより、各 1 時間の期間に対して単一のデータ値を提供します。

カスタムメトリクス用の JSON の構築

以下のセクションには、CloudWatch からのデータをクエリするための予測スケーリングを設定する方法の例が記載されています。このオプションの設定には 2 つの異なる手法あり、予測スケーリングポリシーの JSON を構築するために使用する形式は、選択される手法の影響を受けます。メトリクス計算を使用する場合は、実行されるメトリクス計算に基づいて JSON の形式がさらに多様化します。

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

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

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

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

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
```

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

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

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

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

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

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

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

Metric Math 式を使用する

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

トピック

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

Metric Math について

既存のメトリクスデータの集計だけを行いたい場合は、CloudWatch Metric Math により、別のメトリクスを CloudWatch に発行する手間とコストを節約できます。AWS が提供するメトリクスはすべて使用できます。また、アプリケーションの一部として定義したメトリクスを使用することもできます。例えば、インスタンスごとに Amazon SQS キューバックログを計算したい場合があります。これを行うには、キューから取得可能なメッセージのおおよその数を取得し、その数を Auto Scaling グループの実行中のキャパシティーで割ります。

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

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

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

Metric Math を使用するには、次の操作を実行します。

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

メトリクス計算を使用してメトリクスを組み合わせる予測スケーリングポリシーの例 (AWS CLI)

場合によっては、メトリクスを直接指定するのではなく、まず何らかの方法でそのデータを処理する必要がある場合があります。例えば、Amazon SQS キューから作業を取り出すアプリケーションがあり、キュー内の項目数を予測スケーリングの基準として使用したいとします。キューにあるメッ

セージの数だけでは、必要なインスタンスの数は定義されません。インスタンスごとのバックログを計算するために使用できるメトリクスを作成するには、さらに多くの作業が必要です。詳細については、「Amazon に基づくスケーリングポリシー SQS」を参照してください。

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

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
```

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

```
}
```

この例では、ポリシーの ARN が返されます。

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

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

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

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

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

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

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

すると、新しい Auto Scaling グループのインスタンスの CPUUtilization が、自動的に含まれます。

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-
policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 3600))"
          }
        ]
      },
      "CustomizedCapacityMetricSpecification": {
        "MetricDataQueries": [
```

```
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))"
          }
        ]
      }
    }
  ]
}
```

この例では、ポリシーの ARN が返されます。

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

予測スケーリングポリシーでのカスタムメトリクスに関する考慮事項

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

- エラーメッセージが表示された場合は、メッセージを読み、可能な場合は報告されている問題を解決します。
- Blue/Green デプロイシナリオで検索式を使用しようとしているときに問題が発生した場合は、まず、完全一致ではなく部分一致を検索する検索式の作成方法を理解していることを確認してください。また、クエリが、特定のアプリケーションを実行している Auto Scaling グループのみを見つけることを確認します。検索式の構文の詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch 検索式の構文」を参照してください。
- 事前に式を検証しなかった場合は、put-scaling-policy コマンドは、スケーリングポリシーの作成時にそれを検証します。ただし、このコマンドでは、検出されたエラーの正確な原因を特定できない可能性があります。問題を解決するには、get-metric-data コマンドへのリクエストからの応答で受け取ったエラーをトラブルシューティングします。CloudWatch コンソールから式をトラブルシューティングすることもできます。
- コンソールで [Load] (負荷) と [Capacity] (キャパシティー) のグラフを表示すると、[Capacity] (キャパシティー) グラフにはデータがまったく表示されない場合があります。グラフに完全なデータが含まれるようにするには、Auto Scaling グループのグループメトリクスを常に有効にしてくだ

さい。詳細については、「Auto Scaling グループのメトリクスを有効にする (コンソール)」を参照してください。

- キャパシティーメトリクスの指定は、有効期間にわたって異なる Auto Scaling グループで実行されるアプリケーションがある場合にのみ、Blue/Green デプロイで役立ちます。このカスタムメトリクスでは、複数の Auto Scaling グループの総キャパシティーを指定できます。予測スケーリングは、これを使用して、履歴データをコンソールの [Capacity] (キャパシティー) グラフに表示します。
- `MetricDataQueries` で SUM() のような数学関数を使用せずに、独自の SEARCH() 関数を指定する場合、`ReturnData` に `false` を指定する必要があります。これは、検索式が複数の時系列を返す可能性がある一方、数式に基づくメトリクス指定は 1 つの時系列しか返すことができないためです。
- 検索式に含まれるすべてのメトリクスは、同じ解像度である必要があります。

制約事項

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

スケールイン中に終了する Auto Scaling インスタンスを制御する

Amazon EC2 Auto Scaling は終了ポリシーを使用して、インスタンスを終了する順序を決定します。事前定義されたポリシーを使用するか、ユーザー固有の要件を満たすカスタムポリシーを作成できます。カスタムポリシーまたはインスタンスのスケールイン保護を使用することで、Auto Scaling グループが、まだ終了する準備ができていないインスタンスを終了しないようにすることもできます。

内容

Amazon EC2 Auto Scaling が終了ポリシーを使用する場合

以下のセクションでは、Amazon EC2 Auto Scaling が終了ポリシーを使用するシナリオについて説明します。

内容

スケールインイベント

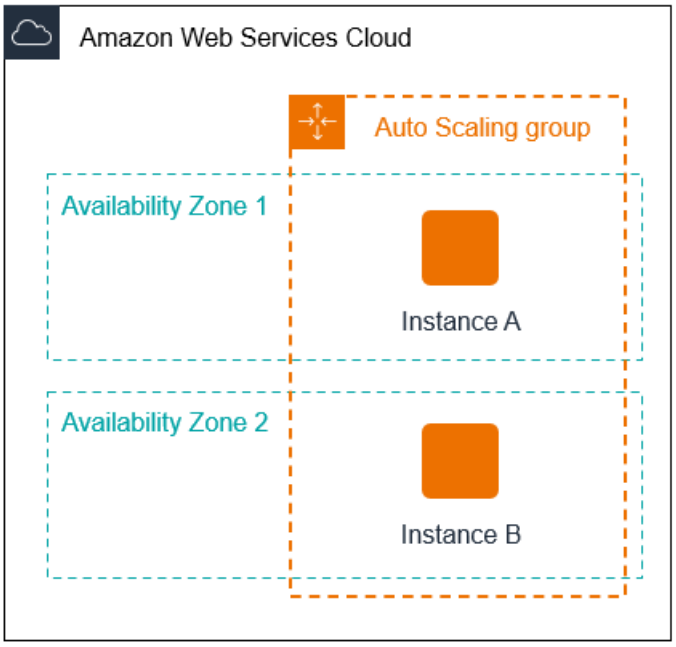
スケールインイベントは、Auto Scaling グループ の希望するキャパシティの新しい値が、グループの現在のキャパシティよりも低い場合にも発生します。

スケールインイベントは、次のシナリオで発生します。

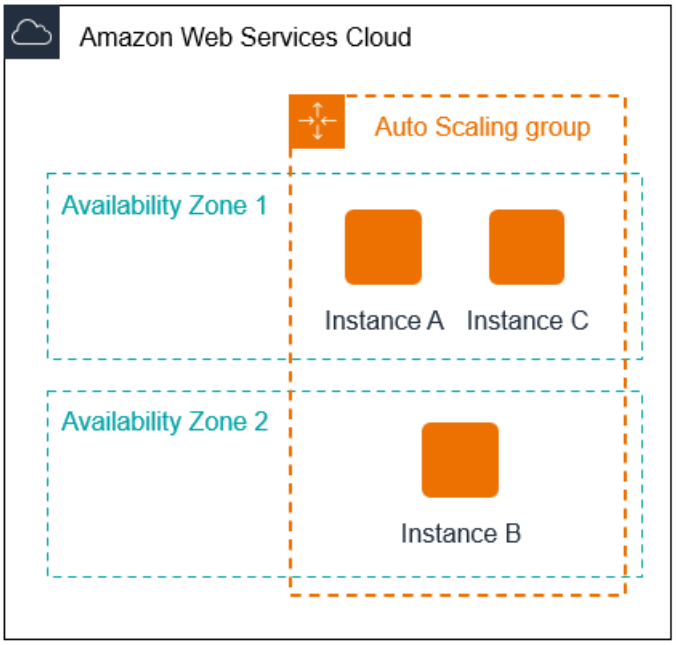
- 動的スケーリングポリシーを使用し、メトリクス値の変更の結果としてグループのサイズが小さくなる場合
- スケジュールされたスケーリングを使用し、スケジュールされたアクションの結果としてグループのサイズが小さくなる場合
- 手動でグループのサイズを縮小します。

次に、スケールインイベントがある場合に終了ポリシーを使用する方法の例を示します。

1. この例の Auto Scaling グループには、1 つのインスタンスタイプ、2 つのアベイラビリティーゾーン、2 つのインスタンスの希望するキャパシティーがあるとします。また、リソース使用率の増減時にインスタンスを追加および削除する、動的スケーリングポリシーもあります。このグループの 2 つのインスタンスは、次の図に示すように 2 つのアベイラビリティーゾーンに分散されます。

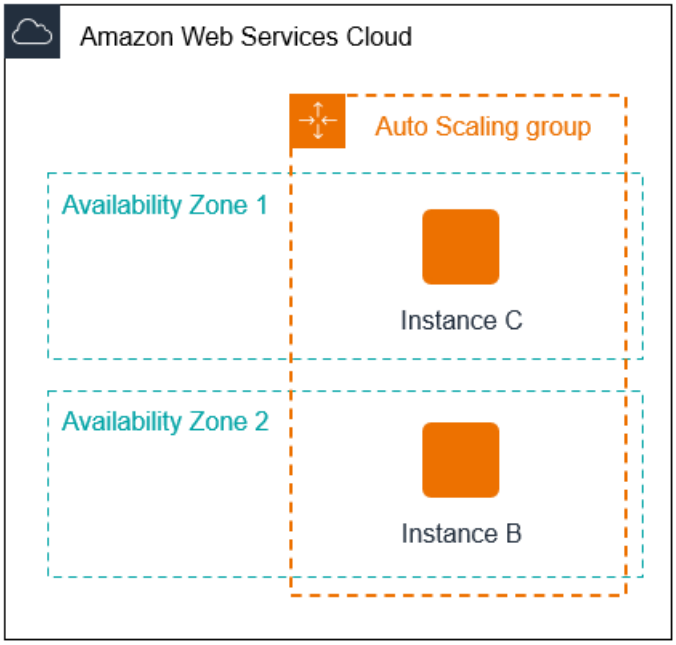


2. Auto Scaling グループがスケールアウトすると、Amazon EC2 Auto Scaling は新しいインスタンスを起動します。Auto Scaling グループには、次の図に示すように、2 つのアベイラビリティーゾーンに分散された 3 つのインスタンスがあります。



3. Auto Scaling グループがスケールインすると、Amazon EC2 Auto Scaling はインスタンスの 1 つを終了します。
4. グループに特定の終了ポリシーを割り当てなかった場合、Amazon EC2 Auto Scaling はデフォルトの終了ポリシーを使用します。2 つのインスタンスを含むアベイラビリティーゾーンを選択し、起動設定から起動されたインスタンス、異なる起動テンプレートから起動されたインスタ

ンス、または現在の起動テンプレートの最も古いバージョンから起動されたインスタンスを終了します。インスタンスが同じ起動テンプレートとバージョンから起動された場合、Amazon EC2 Auto Scaling は次の請求時間に最も近いインスタンスを選択して終了します。



インスタンスの更新

Auto Scaling グループのインスタンスを更新するには、インスタンスの更新を開始します。インスタンスの更新中、Amazon EC2 Auto Scaling はグループ内のインスタンスを終了し、終了したインスタンスの置き換えを起動します。Auto Scaling グループの終了ポリシーは、どのインスタンスを最初に置き換えるかを制御します。

アベイラビリティーゾーンの再調整

Amazon EC2 Auto Scaling は、Auto Scaling グループで有効になっているアベイラビリティーゾーン間で容量を均等に分散します。これにより、アベイラビリティーゾーンの停止による影響を軽減できます。アベイラビリティーゾーン間のキャパシティーの分散のバランスがとれなくなった場合、Amazon EC2 Auto Scaling は、インスタンスが最も少ない有効なアベイラビリティーゾーンでインスタンスを起動し、他の場所でインスタンスを終了することで、Auto Scaling グループのバランスを再調整します。終了ポリシーは、最初に終了を優先するインスタンスを制御します。

アベイラビリティーゾーン間でのインスタンスの分散のバランスが崩れる原因はいくつかあります。

インスタンスの削除

Auto Scaling グループからインスタンスをデタッチ、スタンドバイにインスタンスを置くか、インスタンスを明示的に終了して希望するキャパシティーを減らし、代替インスタンスが起動しないようにすると、グループのバランスが崩れる可能性があります。この場合、Amazon EC2 Auto Scaling はアベイラビリティーゾーンのバランスを再調整して補正します。

最初に指定したアベイラビリティーゾーンとは異なるアベイラビリティーゾーンの使用

Auto Scaling グループを拡張して追加のアベイラビリティーゾーンを含めたり、使用するアベイラビリティーゾーンを変更したりすると、Amazon EC2 Auto Scaling は新しいアベイラビリティーゾーンでインスタンスを起動し、他のゾーンのインスタンスを終了して、Auto Scaling グループがアベイラビリティーゾーンに均等にまたがるようにします。

可用性の停止

可用性の停止はまれにしか発生しません。ただし、1 つのアベイラビリティーゾーンが使用できなくなってから回復すると、Auto Scaling グループでアベイラビリティーゾーン間で不均衡になる可能性があります。Amazon EC2 Auto Scaling はグループを徐々に再調整しようとします。再調整により他のゾーンのインスタンスが終了する可能性があります。

例えば、1 つのインスタンスタイプ、2 つのアベイラビリティーゾーン、2 つのインスタンスの希望するキャパシティーのある Auto Scaling グループがあるとします。1 つのアベイラビリティーゾーンに障害が発生した場合、Amazon EC2 Auto Scaling は正常なアベイラビリティーゾーンに新しいインスタンスを自動的に起動し、異常なアベイラビリティーゾーンにインスタンスを置き換えます。その後、異常なアベイラビリティーゾーンが後で正常な状態に戻ると、Amazon EC2 Auto Scaling はこのゾーンで新しいインスタンスを自動的に起動し、影響を受けていないゾーンのインスタンスを終了します。

> **Note**
>
> 再調整時に、Amazon EC2 Auto Scaling は古いインスタンスを終了する前に新しいインスタンスを起動するため、再調整によってアプリケーションのパフォーマンスや可用性が損なわれることはありません。
> Amazon EC2 Auto Scaling は古いインスタンスを終了する前に新しいインスタンスの起動を試みるため、指定された最大容量またはそれに近い状態になると、再調整アクティビティが妨げられるか、完全に停止する可能性があります。この問題を回避するため、再分散アクティビティの間、グループに対して指定されている最大キャパシティーが一時的に 10% のマージン（または 1 インスタンスのマージンのどちらか大きい方）で増えます。このマージ

ンは、グループが最大キャパシティーに達しているか、それに近い状態であり、ユーザーがゾーンの再設定をリクエストしたため、またはゾーンの可用性の問題を補正するために、グループの再分散が必要な場合にのみ追加されます。この追加キャパシティーは、グループの再分散に要する時間にわたってのみ提供されます。

Amazon EC2 Auto Scaling の終了ポリシーを設定する

終了ポリシーは、Amazon EC2 Auto Scaling が特定の順序でインスタンスを終了するために従う基準を提供します。デフォルトでは、Amazon EC2 Auto Scaling は古い設定を使用しているインスタンスを最初に終了するように設計された終了ポリシーを使用します。終了ポリシーを変更して、最初に終了することが最も重要なインスタンスを制御できます。

Amazon EC2 Auto Scaling は、インスタンスを終了すると、Auto Scaling グループで有効になっているアベイラビリティーゾーン間のバランスを維持しようとします。ゾーン間でのバランスを維持することは、終了ポリシーよりも優先されます。1 つのアベイラビリティーゾーンに他のアベイラビリティーゾーンよりも多くのインスタンスがある場合、Amazon EC2 Auto Scaling はまず不均衡なゾーンに終了ポリシーを適用します。アベイラビリティーゾーンのバランスが取れている場合、すべてのゾーンに終了ポリシーが適用されます。

トピック

デフォルトの終了ポリシーの仕組み

Amazon EC2 Auto Scaling は、インスタンスを終了する必要がある場合、まず、インスタンスが最も多いアベイラビリティーゾーン (またはゾーン) と、スケールインから保護されていないインスタンスを少なくとも 1 つ特定します。次に、特定されたアベイラビリティーゾーン内の保護されていないインスタンスを次のように評価します。

古い設定を使用するインスタンス

- 起動テンプレートを使用するグループの場合 – インスタンスのいずれかが古い設定を使用しているかどうかを、以下の優先順位で判断します。

1. まず、起動設定で起動されたインスタンスを確認します。
2. 次に、現在の起動テンプレートではなく、別の起動テンプレートを使用して起動されたインスタンスを確認します。
3. 最後に、現在の起動テンプレートの最も古いバージョンを使用したインスタンスを確認します。

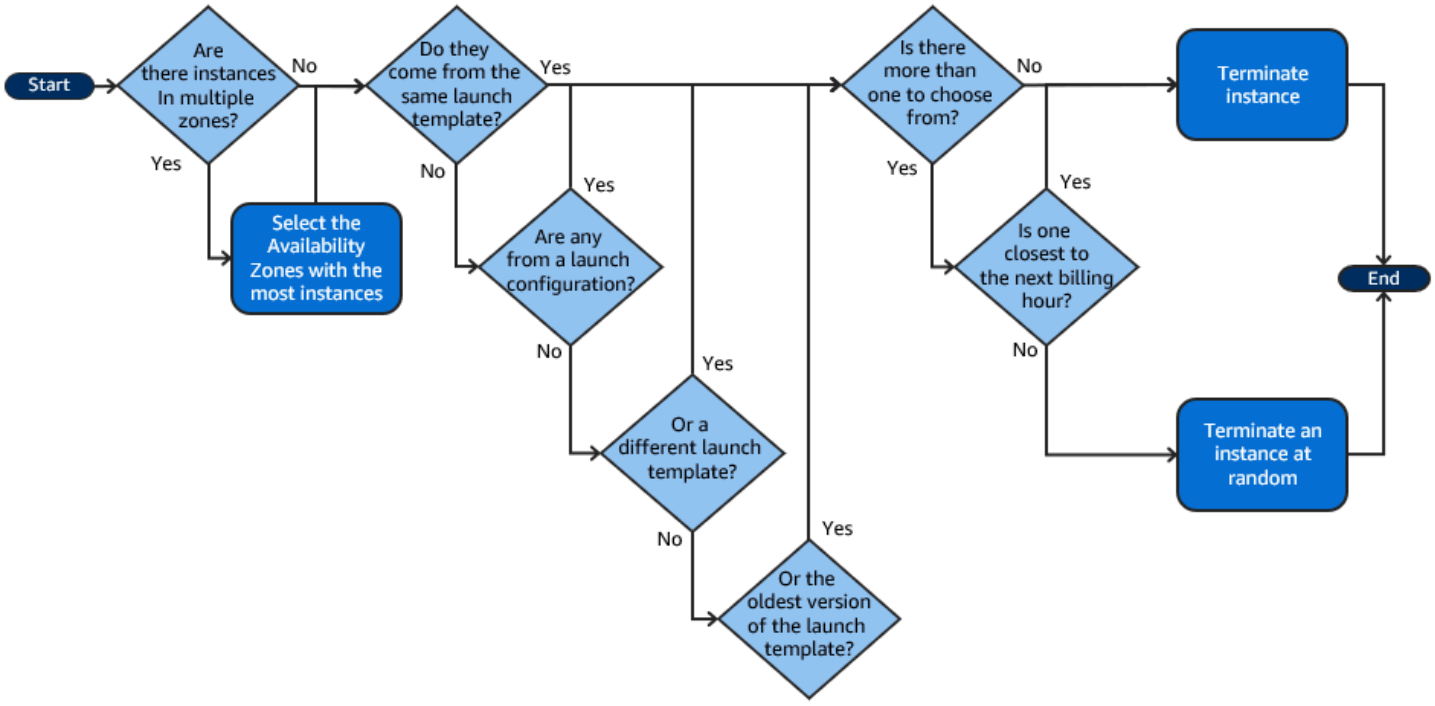
- 起動設定を使用するグループの場合 – インスタンスのいずれかが最も古い起動設定を使用しているかどうかを判断します。

古い設定のインスタンスが見つからない場合、または複数のインスタンスから選択できる場合、Amazon EC2 Auto Scaling は次の請求時間に近いインスタンスの次の基準を考慮します。

次の課金時間に近いインスタンス

前の基準を満たすインスタンスのいずれかが、次の課金時間に最も近いかどうかを判断します。複数のインスタンスの近さが同じ場合は、ランダムに 1 つ終了します。これによって、時間ごとに課金されるインスタンスを最大限に活用できます。ただし、ほとんどのEC2使用量は 1 秒あたりに請求されるため、この最適化によるメリットは少なくなります。詳細については、「Amazon のEC2料金」を参照してください。

次のフロー図は、起動テンプレートを使用するグループでデフォルトの終了ポリシーがどのように機能するかを示しています。



デフォルトの終了ポリシーと混合インスタンスグループ

Amazon EC2 Auto Scaling は、混合インスタンスグループのインスタンスを終了するときに追加の基準を適用します。

Amazon EC2 Auto Scaling がインスタンスを終了する必要がある場合、まずグループの設定に基づいて終了する購入オプション (スポットまたはオンデマンド) を特定します。これにより、グループが時間の経過とともに傾向として、スポットインスタンスとオンデマンドインスタンスが指定された比率になるようにできます。

次に、各アベイラビリティーゾーン内で独立に終了ポリシーを適用します。アベイラビリティーゾーンのバランスを維持するために、終了するアベイラビリティーゾーンで、スポットインスタンスかオンデマンドインスタンスかを決定します。インスタンスタイプに定義された重みを持つ混合インスタンスグループにも同じロジックが適用されます。

各ゾーン内で、デフォルトの終了ポリシーは次のように動作して、特定された購入オプション内でどの保護されていないインスタンスを終了できるかを決定します。

1. Auto Scaling グループの指定された割り当て戦略との整合性を向上させるために、いずれかのインスタンスを終了できるかどうかを決定します。最適化するインスタンスが特定されない場合、または複数のインスタンスから選択できる場合は、評価が続行されます。
2. いずれかのインスタンスが古い設定を使用しているかどうかを次の優先順序で判断します。
   a. まず、起動設定で起動されたインスタンスを確認します。
   b. 次に、現在の起動テンプレートではなく、別の起動テンプレートを使用して起動されたインスタンスを確認します。
   c. 最後に、現在の起動テンプレートの最も古いバージョンを使用したインスタンスを確認します。

   古い設定のインスタンスがない場合、または複数のインスタンスから選択できる場合は、評価が続行されます。
3. いずれかのインスタンスが次の課金時間に最も近いかどうかを判断します。複数のインスタンスの近さが同じ場合は、ランダムに 1 つ選択します。

事前定義された終了ポリシー

次の事前定義された終了ポリシーから選択します。

- **`Default`** – デフォルトの終了のポリシーに従って、インスタンスを終了します。

- **`AllocationStrategy`** – Auto Scaling グループのインスタンスを終了して、残りのインスタンスを、終了するインスタンスのタイプ (スポットインスタンスまたはオンデマンドインスタンス) の配分戦略に合わせます。このポリシーは、優先するインスタンスタイプが変更されたときに便利です。スポット配分戦略が `lowest-price` の場合、N 個の最低価格のスポットプール間で、スポットインスタンスの分散バランスを徐々に再調整できます。スポット配分戦略が `capacity-optimized` の場合、使用可能なスポットキャパシティーがより多いスポットプール間で、スポットインスタンスの分散バランスを徐々に再調整できます。優先度の低いタイプのオンデマンドインスタンスを優先度の高いタイプのオンデマンドインスタンスに徐々に置き換えることもできます。
- **`OldestLaunchTemplate`** – 最も古い起動テンプレートを使用するインスタンスを終了します。このポリシーでは、現在の起動テンプレートを使用していないインスタンスが最初に終了され、その後、現在の起動テンプレートのうち最も古いバージョンを使用しているインスタンスが終了されます。このポリシーは、グループを更新し、以前の設定を使用しているインスタンスを廃止する場合に便利です。
- **`OldestLaunchConfiguration`** – 最も古い起動設定のインスタンスを終了します。このポリシーは、グループを更新し、以前の設定を使用しているインスタンスを廃止する場合に便利です。このポリシーでは、最新以外の起動設定を使用するインスタンスが最初に終了されます。
- **`ClosestToNextInstanceHour`** – 次の課金時間に最も近いインスタンスを終了します。これにより、時間単価のインスタンスを最大限に活用できます。
- **`NewestInstance`** – グループ内の最も新しいインスタンスを終了します。このポリシーは、新しい起動設定をテストするが、新しい設定は本稼働環境には保持しない場合に便利です。
- **`OldestInstance`** – グループ内の最も古いインスタンスを終了します。このオプションは、Auto Scaling グループのインスタンスを新しいEC2インスタンスタイプにアップグレードする場合に便利です。より古いタイプのインスタンスをより新しいタイプのインスタンスに徐々に置き換えることができます。

> **Note**
>
> Amazon EC2 Auto Scaling は、どの終了ポリシーが使用されているかに関係なく、常にアベイラビリティーゾーン間でインスタンスを最初に分散します。その結果、いくつかの新しいインスタンスが古いインスタンスの前に終了される状況が発生する可能性があります。例えば、最近追加されたアベイラビリティーゾーンがある場合、グループに使用されている他のアベイラビリティーゾーンより多くのインスタンスが含まれるアベイラビリティーゾーンがある場合。

Auto Scaling グループの終了ポリシーを変更する

Auto Scaling グループの終了ポリシーを変更するには、次のいずれかの方法を使用します。

Console

Amazon Auto Scaling EC2Auto Scaling コンソールで Auto Scaling グループを最初に作成するときに、終了ポリシーを変更することはできません。デフォルトの終了ポリシーが自動的に使用されます。Auto Scaling グループを作成したら、デフォルトのポリシーを、別の終了ポリシー、または適用する順序で並んでいる複数の終了ポリシーに置き換えることができます。

Auto Scaling グループの終了ポリシーを変更するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[高度な設定]、[編集] の順に選択します。
4. [終了ポリシー] で、1 つまたは複数の終了ポリシーを選択します。複数のポリシーを選択する場合は、適用する順に合わせて選択していきます。

   オプションで、[Custom termination policy] (カスタム終了ポリシー) を選択した後、ニーズを満たす Lambda 関数を選択することもできます。Lambda 関数のために作成したバージョンとエイリアスがある場合は、バージョン/エイリアス ドロップダウンリストから、いずれかのバージョンとエイリアスを選択します。Lambda 関数の未公開バージョンを使用する場合には、[Version/Alias] (バージョン/エイリアス) の設定はデフォルトのままにします。詳細については、「Lambda を使用したカスタム終了ポリシーを作成する」を参照してください。

   > **Note**
   >
   > 複数のポリシーを使用する場合は、その順序を正しく設定する必要があります。
   >
   > - [Default] (デフォルト) のポリシーを使用する場合は、リストの末尾にあるポリシーを選択する必要があります。
   > - [Custom termination policy] (カスタム終了ポリシー) を使用する場合には、リストの最初にあるポリシーを選択します。
5. [Update] (更新) を選択します。

AWS CLI

別のポリシーが指定されていない限り、デフォルトの終了ポリシーが自動的に使用されます。

Auto Scaling グループの終了ポリシーを変更するには

以下のいずれかのコマンドを使用します。

- create-auto-scaling-group
- update-auto-scaling-group

終了ポリシーを個別に使用することも、ポリシーのリストに組み合わせることもできます。例えば、次のコマンドを使用して、最初に OldestLaunchConfiguration ポリシーを使用し、その後で ClosestToNextInstanceHour ポリシーを使用するように Auto Scaling グループを更新します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
termination-policies "OldestLaunchConfiguration" "ClosestToNextInstanceHour"
```

Default の終了ポリシーを使用する場合、終了ポリシーのリストでは最後のポリシーになるように指定します。例えば、--termination-policies "OldestLaunchConfiguration" "Default" と指定します。

カスタム終了ポリシーを使用するには、まず を使用して終了ポリシーを作成する必要があります AWS Lambda。終了ポリシーとして使用する Lambda 関数を指定するには、終了ポリシーのリストで最初の関数を使用します。例えば、--termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" "OldestLaunchConfiguration" と指定します。詳細については、「Lambda を使用したカスタム終了ポリシーを作成する」を参照してください。

Lambda を使用したカスタム終了ポリシーを作成する

Amazon EC2 Auto Scaling は、終了ポリシーを使用して、Auto Scaling グループのサイズを小さくするときに、どのインスタンスを最初に終了するかを優先します (スケールインと呼ばれます)。Auto Scaling グループではデフォルトの終了ポリシーを使用しますが、独自の終了ポリシーを選択または作成することもできます。定義済みの終了ポリシーを選択する方法の詳細については、「Amazon EC2 Auto Scaling の終了ポリシーを設定する」を参照してください。

このトピックでは、Amazon EC2 Auto Scaling が特定のイベントに応答して呼び出す AWS Lambda 関数を使用してカスタム終了ポリシーを作成する方法について説明します。作成した Lambda 関数は、Amazon EC2 Auto Scaling によって送信された入力データ内の情報を処理し、終了する準備ができているインスタンスのリストを返します。

カスタム終了ポリシーを使用すると、終了するインスタンスとタイミングをより適切に制御できます。例えば、Auto Scaling グループがスケールインすると、Amazon EC2 Auto Scaling は中断すべきでない実行中のワークロードがあるかどうかを判断できません。Lambda 関数を使用すると、終了リクエストを検証し、ワークロードが完了するまで待ってから、インスタンス ID を終了のために Amazon EC2 Auto Scaling に返すことができます。

内容

入力データ

Amazon EC2 Auto Scaling は、スケールインイベントのJSONペイロードを生成します。また、インスタンスの最大有効期間またはインスタンス更新機能の結果としてインスタンスが終了する時期にもペイロードを生成します。また、アベイラビリティーゾーン間でグループのバランスを再調整するときに開始できるスケールインイベントのJSONペイロードも生成します。

このペイロードには、Amazon EC2 Auto Scaling が終了する必要がある容量、終了を提案するインスタンスのリスト、終了を開始したイベントに関する情報が含まれています。

次にペイロードの例を示します。

```
{
  "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:<account-
id>:autoScalingGroup:d4738357-2d40-4038-ae7e-b00ae0227003:autoScalingGroupName/my-asg",
  "AutoScalingGroupName": "my-asg",
  "CapacityToTerminate": [
    {
      "AvailabilityZone": "us-east-1b",
```

```
      "Capacity": 2,
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 1,
      "InstanceMarketOption": "spot"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "Capacity": 3,
      "InstanceMarketOption": "on-demand"
    }
  ],
  "Instances": [
    {
      "AvailabilityZone": "us-east-1b",
      "InstanceId": "i-0056faf8da3e1f75d",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-02e1c69383a3ed501",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-036bc44b6092c01c7",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    ...
  ],
  "Cause": "SCALE_IN"
}
```


ペイロードには、Auto Scaling グループの名前、その Amazon リソースネーム (ARN) 、および次の要素が含まれます。

- `CapacityToTerminate` は、特定のアベイラビリティーゾーンで終了するように設定されたスポットまたはオンデマンドのキャパシティーを示します。

- `Instances` は、「」の情報に基づいて Amazon EC2 Auto Scaling が終了を提案するインスタンスを表しますCapacityToTerminate。
- `Cause` は、終了の原因となったイベントである SCALE_IN、INSTANCE_REFRESH、MAX_INSTANCE_LIFETIME、REBALANCE を示します。

次の情報は、Amazon EC2 Auto Scaling が入力データInstancesで を生成する方法における最も重要な要因の概要を示しています。

- スケールインイベントおよびインスタンスの更新に基づく終了によってインスタンスが終了する場合は、アベイラビリティーゾーン間のバランスを維持することが優先されます。そのため、グループに使用されている他のアベイラビリティーゾーンより多くのインスタンスが含まれるアベイラビリティーゾーンがある場合、入力データのインスタンスはそのバランスのとれていないアベイラビリティーゾーンのみからインスタンスに適用されます。グループに使用されているアベイラビリティーゾーンのバランスがとれている場合、入力データにはグループのすべてのアベイラビリティーゾーンのインスタンスが含まれます。
- 混合インスタンスポリシーを使用する場合、各購入オプションの希望する割合に基づいて、スポットおよびオンデマンドのキャパシティーをバランスよく維持することも優先されます。まず、2 つのタイプ (スポットまたはオンデマンド) のどちらを終了すべきかを識別します。次に、アベイラビリティーゾーンのバランスが最も高い結果となるアベイラビリティーゾーンを終了できるインスタンス (特定された購入オプション内) を特定します。

レスポンスデータ

入力データと応答データが連携して、終了するインスタンスのリストを絞り込みます。

指定された入力では、Lambda 関数からの応答は次の例のようになります。

```
{
  "InstanceIDs": [
    "i-02e1c69383a3ed501",
    "i-036bc44b6092c01c7",
    ...
  ]
}
```

`InstanceIDs` は、終了する準備ができているインスタンスを表します。

または、終了する準備ができている別のインスタンスのセットを返すこともできます。これにより、入力データのインスタンスが上書きされます。Lambda 関数が呼び出されたときに終了する準備ができていない場合は、インスタンスを返さないように選択することもできます。

終了する準備ができているインスタンスがない場合、Lambda 関数からの応答は次の例のようになります。

```
{
  "InstanceIDs": [ ]
}
```

考慮事項

カスタム終了ポリシーを使用する場合、次の点を考慮してください。

- レスポンスデータで最初にインスタンスを返しても、その終了は保証されません。Lambda 関数が呼び出されたときに必要な数を超えるインスタンスが返された場合、Amazon EC2 Auto Scaling は、Auto Scaling グループに指定した他の終了ポリシーと照らし合わせて各インスタンスを評価します。複数の終了ポリシーがある場合、リスト内の次の終了ポリシーを適用しようとします。終了に必要な数を超えるインスタンスがある場合は、次の終了ポリシーに移ります。他の終了ポリシーが指定されていない場合は、デフォルトの終了ポリシーを使用して、終了するインスタンスを決定します。
- インスタンスが返されない場合、または Lambda 関数がタイムアウトした場合、Amazon EC2 Auto Scaling は関数を再度呼び出す前にしばらく待機します。スケールインイベントでは、グループの希望するキャパシティが現在のキャパシティよりも小さい限り、試行を続けます。例えば、リフレッシュベースの終了の場合、1 時間試行し続けます。その後、インスタンスの終了に失敗し続けると、インスタンスの更新操作は失敗します。インスタンスの最大有効期間では、Amazon EC2 Auto Scaling は、最大有効期間を超過していると識別されたインスタンスを終了しようとし続けます。
- 関数は繰り返し再試行されるため、Lambda 関数をカスタム終了ポリシーとして使用する前に、コード内の永続的なエラーをテストして修正してください。
- 入力データを独自のインスタンスのリストで上書きして終了し、これらのインスタンスを終了すると、アベイラビリティーゾーンのバランスが崩れると、Amazon EC2 Auto Scaling はアベイラビリティーゾーン間の容量の分散を徐々に再調整します。まず、Lambda 関数を呼び出して、リバランシングを開始するかどうかを判断できるように、終了する準備ができているインスタンスがあるかどうかを確認します。終了する準備ができているインスタンスがある場合、最初に新しいインスタンスを起動します。インスタンスの起動が完了すると、グループの現在のキャパシティが希望するキャパシティよりも大きいことが検出され、スケールインイベントが開始されます。

- スケールイン保護を使用して特定のインスタンスが終了されないように保護する機能にカスタム終了ポリシーが影響をおよぼすことはありません。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

Lambda 関数を作成する

まず Lambda 関数を作成して、Auto Scaling グループの終了ポリシーで Amazon リソースネーム (ARN) を指定できるようにします。

Lambda 関数を作成するには (コンソール)

1. Lambda コンソールで [Functions (関数)] ページを開きます。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループの作成時に使用したのと同じリージョンを選択します。
3. [Create function (関数の作成)] を選択し、[Author from scratch (一から作成)] を選択します。
4. [基本的な情報] の [関数名] に、関数の名前を入力します。
5. [Create function (関数の作成)] を選択します。関数のコードと設定に戻ります。
6. 関数をまだコンソールで開いている状態で、関数コードの下にエディタに貼り付けます。
7. [デプロイ] を選択します。
8. 必要に応じて、Lambda 関数の公開バージョンを作成するには、[Versions (バージョン)] タブをクリックし、次に新しいバージョンを発行します。Lambda でのバージョニングの詳細については、AWS Lambda デベロッパーガイドの「Lambda 関数のバージョン」を参照してください。
9. バージョンを公開することを選択した場合、このバージョンの Lambda 関数と関連付けるには [Aliases (エイリアス)] タブを選択します。Lambda のエイリアスの詳細については、AWS Lambda デベロッパーガイドの「Lambda 関数のエイリアス」を参照してください。
10. 次に [Configuration (設定)] タブと [Permissions (アクセス許可)] を選択します。
11. [Resource-based policy (リソースベースのポリシー)] にスクロールダウンして [アクセス許可の追加] を選択します。リソースベースのポリシーを使用して、関数を呼び出すアクセス許可を、ポリシーで指定されているプリンシパルに付与します。この場合、プリンシパルはEC2Auto Scaling グループに関連付けられている Amazon Auto Scaling サービスにリンクされたロールになります。 Auto Scaling
12. [Policy statement (ポリシーステートメント)] セクションで、権限を設定します。

    a. AWS アカウント を選択します。

b. プリンシパルには、呼び出し元のサービスにリンクされたロールARNの を入力します。たとえば、 です**arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling**。

c. アクション で、lambda:InvokeFunction を選択します。

d. [Statement ID (ステートメント ID)] に **AllowInvokeByAutoScaling** といった一意のステートメント ID を入力します。

e. [Save] を選択します。

13. これらの指示に従ったら、次のステップとして Auto Scaling グループの終了ポリシーで関数ARNの を指定します。詳細については、「Auto Scaling グループの終了ポリシーを変更する」を参照してください。

> **Note**
>
> Lambda 関数を開発するためのリファレンスとして使用できる例については、Amazon EC2 Auto Scaling の GitHub リポジトリを参照してください。

制限

- Auto Scaling グループの終了ポリシーで指定できる Lambda 関数は 1 つだけです。複数の終了ポリシーが指定されている場合は、最初に Lambda 関数を指定する必要があります。
- Lambda 関数を参照するには、非修飾 ARN (サフィックスなし) を使用するか、バージョンまたはエイリアスARNをサフィックスとする修飾 を使用します。非修飾 ARNが使用されている場合 (などfunction:my-function) 、リソースベースのポリシーは、関数の未公開バージョンで作成する必要があります。修飾 ARN が使用されている場合 (function:my-function:1や などfunction:my-function:prod) 、リソースベースのポリシーは、関数の特定の公開バージョンで作成する必要があります。
- サフィックスARNが付いた修飾 $LATEST を使用することはできません。$LATEST サフィックス ARN が付いた修飾 を参照するカスタム終了ポリシーを追加しようとすると、エラーが発生します。
- 入力データで提供されるインスタンスの数は、30,000 インスタンスまでに制限されています。終了できるインスタンスが 30,000 個を超える場合、入力データには インスタンスの最大数が戻されることを示す "HasMoreInstances": true を示します。

- Lambda 関数の最大実行時間は 2 秒 (2000 ミリ秒) です。ベストプラクティスとして、予想される実行時間に基づいて Lambda 関数のタイムアウト値を設定する必要があります。Lambda 関数のデフォルトのタイムアウトは 3 秒ですが、これを減らすことができます。
- ランタイムが 2 秒の制限を超えると、ランタイムがこのしきい値を下回るまで、すべてのスケールインアクションが保留されます。ランタイムが一貫して長い Lambda 関数の場合、結果をキャッシュして後続の Lambda 呼び出し中に取得できるようにするなど、ランタイムを短縮する方法を見つけます。

インスタンスのスケールイン保護を使用してインスタンスの終了を制御する

インスタンスのスケールイン保護により、Amazon EC2 Auto Scaling が終了できるインスタンスを制御できます。この機能の一般的なユースケースとしては、コンテナベースのワークロードのスケーリングがあります。詳細については、「インスタンスの終了を的確に処理するようにアプリケーションを設計する」を参照してください。

デフォルトでは、Auto Scaling グループを作成すると、インスタンスのスケールイン保護は無効になります。つまり、Amazon EC2 Auto Scaling はグループ内の任意のインスタンスを終了できます。

Auto Scaling グループでスケールイン保護設定を有効化しインスタンスを起動すると、その直後からインスタンスの保護が開始されます。インスタンスのスケールイン保護は、インスタンスの状態が `InService` の場合に開始されます。その後、終了できるインスタンスを制御するには、Auto Scaling グループ内で個別インスタンスのスケールイン保護設定を無効にします。そうすることで、引き続き特定のインスタンスを望ましくない終了から保護できます。

トピック

考慮事項

インスタンスのスケールイン保護を使用する場合の考慮事項を次に示します。

- Auto Scaling グループのすべてのインスタンスがスケールインから保護されている場合にスケールインイベントが発生すると、希望するキャパシティは減少します。ただし、Auto Scaling グループはインスタンスのスケールイン保護の設定が無効になるまで、必要な数のインスタンスを終了す

ることはできません。では AWS Management Console、Auto Scaling グループ内のすべてのインスタンスがスケールインイベントが発生したときにスケールインから保護されている場合、Auto Scaling グループのアクティビティ履歴に次のメッセージが含まれます。`Could not scale to desired capacity because all remaining instances are protected from scale in.`

- スケールインから保護されているインスタンスをデタッチすると、インスタンスのスケールイン保護の設定は失われます。インスタンスをグループに再度アタッチすると、グループの現在のインスタンスのスケールイン保護を受け継ぎます。Amazon EC2 Auto Scaling が新しいインスタンスを起動するか、インスタンスをウォームプールから Auto Scaling グループに移動すると、インスタンスは Auto Scaling グループのインスタンススケールイン保護設定を継承します。
- インスタンスのスケールイン保護は、次の状況から Auto Scaling インスタンスを保護することはできません。
  - インスタンスがヘルスチェックに失敗した場合のヘルスチェックの置換。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。
  - スポットインスタンスの中断。キャパシティーが使用できなくなった場合、またはスポット料金が上限価格を超えた場合、スポットインスタンスは終了されます。
  - キャパシティブロック予約は終了します。Amazon は、スケールインから保護されていてもキャパシティブロックインスタンスEC2を再利用します。
  - `terminate-instance-in-auto-scaling-group` コマンドによる手動での終了。詳細については、「Auto Scaling グループのインスタンスを終了する (AWS CLI)」を参照してください。
  - Amazon EC2コンソール、CLIコマンド、およびAPIオペレーションによる手動終了。Auto Scaling インスタンスを手動終了から保護するには、Amazon EC2終了保護を有効にします。（これにより、Amazon EC2 Auto Scaling が `terminate-instance-in-auto-scaling-group` コマンドを使用してインスタンスを終了したり、手動で終了したりするのを防ぐことはできません）。起動テンプレートで Amazon EC2終了保護を有効にする方法については、「」を参照してください詳細設定を使用して起動テンプレートを作成する。

Auto Scaling グループのスケールイン保護を変更する

Auto Scaling グループのインスタンスのスケールイン保護の設定は、有効または無効にすることができます。有効にすると、グループによって起動されたすべての新しいインスタンスで、インスタンスのスケールイン保護が有効になります。

Auto Scaling グループのこの設定を有効または無効にしても、既存のインスタンスには影響しません。

Console

新しい Auto Scaling グループのスケールイン保護を有効にするには

Auto Scaling グループを作成するときに、[グループサイズとスケーリングポリシーを設定する] ページの [インスタンスのスケールイン保護] で、[インスタンスのスケールイン保護を有効にする] チェックボックスを選択します。

既存のグループのスケールイン保護を有効または無効にするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループのチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[高度な設定]、[編集] の順に選択します。
4. [インスタンスのスケールイン保護] で、[インスタンスのスケールイン保護を有効にする] チェックボックスを選択または選択解除して、必要に応じてこのオプションを有効または無効にします。
5. [Update] (更新) を選択します。

AWS CLI

新しい Auto Scaling グループのスケールイン保護を有効にするには

以下の create-auto-scaling-group コマンドを使用して、インスタンスのスケールイン保護を有効にします。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --new-
instances-protected-from-scale-in ...
```

既存のグループのスケールイン保護を有効にするには

次のupdate-auto-scaling-groupコマンドを使用して、指定した Auto Scaling グループのインスタンススケールイン保護を有効にします。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --new-
instances-protected-from-scale-in
```

既存のグループのスケールイン保護を無効にするには

次のコマンドを使用して、指定したグループのインスタンスのスケールイン保護を無効にします。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --no-new-instances-protected-from-scale-in
```

インスタンスのスケールイン保護を変更する

デフォルトで、インスタンスは所属する Auto Scaling グループからインスタンスのスケールイン保護の設定を取得します。ただし、起動後に個々のインスタンスのスケールイン保護を有効または無効にできます。

Console

インスタンスのスケールイン保護を有効または無効にするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Instance management (インスタンス管理)] タブの [Instances (インスタンス)] で、インスタンスを選択します。
4. インスタンスのスケールイン保護を有効にするには、[Actions (アクション)]、[Set scale-in protection (スケールイン保護を設定)] の順に選択します。プロンプトが表示されると、[Set scale-in protection (スケールイン保護を設定)] を選択します。
5. インスタンスのスケールイン保護を無効にするには、[Actions (アクション)]、[Remove scale-in protection (スケールイン保護の削除)] の順に選択します。プロンプトが表示されたら、[Remove scale-in protection (スケールイン保護の削除)] を選択します。

AWS CLI

インスタンスのスケールイン保護を有効にするには

以下の set-instance-protection コマンドを使用して、指定したインスタンスに対するインスタンスのスケールイン保護を有効にします。

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-
group-name my-asg --protected-from-scale-in
```

インスタンスのスケールイン保護を無効にするには

次のコマンドを使用して、指定したインスタンスにおけるインスタンスのスケールイン保護を無効にします。

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-
group-name my-asg --no-protected-from-scale-in
```

> **Note**
>
> インスタンスのスケールイン保護は、誰かが Amazon EC2コンソールまたは を使用してインスタンスを手動で終了する場合など、人為的エラーが発生した場合にインスタンスが終了しないことを保証しません AWS CLI。インスタンスを誤って終了しないようにするには、Amazon EC2 終了保護を使用できます。ただし、終了保護とインスタンスのスケールイン保護が有効になっている場合でも、ヘルスチェックでインスタンスが異常であると判断された場合、またはグループ自体が誤って削除された場合、インスタンスストレージに保存されたデータが失われる可能性があります。あらゆる環境と同様に、ベスト・プラクティスは、データのバックアップを頻繁に行うこと、またはビジネス継続性要件に適している場合にいつでもバックアップすることです。

インスタンスの終了を的確に処理するようにアプリケーションを設計する

このトピックでは、Auto Scaling グループがまだ終了の準備ができていないインスタンスを終了したり、割り当てられたジョブを完了させるには早すぎる状態でインスタンスを終了したりしないようにするために使用できる機能について説明します。これらの 3 つの機能をすべて組み合わせて使用することも、個別に使用して、インスタンスの終了を的確に処理するようにアプリケーションを設計することもできます。

例えば、長時間実行されるジョブの受信メッセージを収集する Amazon SQSキューがあるとします。新しいメッセージが到達すると、Auto Scaling グループのインスタンスがメッセージを取得し、処理を開始します。各メッセージの処理には 3 時間かかります。メッセージの数が増えると、新しいインスタンスが Auto Scaling グループに自動的に追加されます。メッセージの数が少なくなると、既存のインスタンスは自動的に終了します。この場合、Amazon EC2 Auto Scaling は終了する

インスタンスを決定する必要があります。デフォルトでは、Amazon EC2 Auto Scaling は、現在アイドル状態のインスタンスではなく、3 時間の長いジョブの処理に 2.9 時間かかるインスタンスを終了する可能性があります。Amazon EC2 Auto Scaling を使用する際に予期しない終了の問題を回避するには、このシナリオに対応するようにアプリケーションを設計する必要があります。

内容

> **Important**
>
> インスタンスの終了を適切に処理するように Amazon EC2 Auto Scaling でアプリケーションを設計する場合は、以下の点に注意してください。
>
> - インスタンスに異常がある場合、Amazon EC2 Auto Scaling は、使用する機能に関係なく、インスタンスを置き換えます (`ReplaceUnhealthy`プロセスを停止しない限り)。ライフサイクルフックを使用すると、アプリケーションは、正常にシャットダウンしたり、インスタンスの終了前に回復する必要があるデータをコピーしたりできます。
> - 終了ライフサイクルフックは、インスタンスの終了前に実行または完了することが保証されていません。何かが失敗しても、Amazon EC2 Auto Scaling はインスタンスを終了します。

インスタンスのスケールイン保護

インスタンスのスケールイン保護は、インスタンスの終了が、デフォルトで拒否され、かつ、特定のインスタンスについてのみ明示的に許可されるべき重要なアクションである多くの状況で使用できます。例えば、コンテナ化されたワークロードを実行する場合、すべてのインスタンスを保護し、現在のタスクやスケジュールされたタスクがないインスタンスについてのみ保護を解除したいと考えるのが一般的です。Amazon などのサービスは、 製品にインスタンスのスケールイン保護との統合ECS を構築しました。

Auto Scaling グループでスケールイン保護を有効にして、インスタンスの作成時にスケールイン保護を適用したり、既存のインスタンスのためにそれを有効にしたりできます。インスタンスで実行する作業がなくなった場合、保護をオフに切り替えることができます。インスタンスは新しいジョブのためにポーリングを続行し、新しいジョブが割り当てられたら保護を再度有効にすることができます。

アプリケーションは、インスタンスが終了可能かどうかを管理する一元的なコントロールプレーンから、またはインスタンス自体から保護を設定できます。ただし、多数のインスタンスがスケールイン保護を継続的に切り替えている場合、大規模なフリートでスロットリングの問題が発生する可能性があります。

詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。

カスタム終了ポリシー

インスタンスのスケールイン保護と同様に、カスタム終了ポリシーは、Auto Scaling グループが特定のインスタンスを終了しないようにするのに役立ちます。

デフォルトでは、Auto Scaling グループはデフォルトの終了ポリシーを使用して、最初に終了するインスタンスを決定します。どのインスタンスが最初に終了するかをさらに制御したい場合は、Lambda 関数を使用して独自のカスタム終了ポリシーを実装できます。Amazon EC2 Auto Scaling は、終了するインスタンスを決定する必要があるたびに関数を呼び出します。関数によって返されたインスタンスのみが終了します。関数がエラー、タイムアウト、または空のリストを生成した場合、Amazon EC2 Auto Scaling はインスタンスを終了しません。

カスタム終了ポリシーは、インスタンスが十分に冗長であるか、または十分に活用されていないためにインスタンスを終了できることがわかっている場合に役立ちます。これをサポートするには、グループ全体のワークロードをモニタリングするコントロールプレーンを備えたアプリケーションを実装する必要があります。これにより、インスタンスがまだジョブを処理している場合、Lambda 関数はそのインスタンスを含めてはならないことを認識できます。

詳細については、「Lambda を使用したカスタム終了ポリシーを作成する」を参照してください。

終了ライフサイクルフック

終了ライフサイクルフックは、既に終了対象として選択されているインスタンスの寿命を延長します。これにより、現在インスタンスに割り当てられているすべてのメッセージまたはリクエストを完了するため、または進行状況を保存して作業を別のインスタンスに転送するための追加の時間を確保できます。

多くのワークロードでは、終了対象として選択されたインスタンス上のアプリケーションを正常にシャットダウンするには、ライフサイクルフックで十分な場合があります。これはベストエフォート型のアプローチであり、エラーが発生した場合の終了を防ぐために使用することはできません。

ライフサイクルフックを使用するには、インスタンスの終了がいつ選択されるかを知る必要があります。これを知るには 2 つの方法があります。

オプション	説明	最適なケース	ドキュメントへのリンク
インスタンス内	インスタンスメタデータサービス (IMDS) は、インスタンスのステータスをインスタンスから直接ポーリングできる安全なエンドポイントです。メタデータが `Terminated` で返された場合、インスタンスの終了がスケジュールされています。	インスタンスを終了する前に、インスタンスに対してアクションを実行する必要があるアプリケーション。	ターゲットライフサイクル状態を取得する
インスタンス外	インスタンスが終了する際に、イベント通知が生成されます。Amazon EventBridge、Amazon 、またはAmazon を使用してルールを作成しSQS、これらのイベントSNSをキャプチャし、Lambda 関数で などのレスポンスを呼び出すことができます。	インスタンスの外部でアクションを実行する必要があるアプリケーション。	通知ターゲットを作成する

ライフサイクルフックを使用するには、インスタンスの終了準備が完全に整うタイミングを知る必要もあります。Amazon EC2 Auto Scaling は、CompleteLifecycleAction呼び出しを受信するかタイムアウトが経過するかのいずれか早い方まで、インスタンスを終了するEC2ように Amazon に指示しません。

デフォルトでは、インスタンスは終了ライフサイクルフックにより、引き続き 1 時間にわたって実行できます (ハートビートタイムアウト)。ライフサイクルアクションを完了するのに 1 時間では足りない場合は、デフォルトのタイムアウトを設定できます。ライフサイクルアクションが実際に進行中の場合は、RecordLifecycleActionHeartbeatAPI呼び出しでタイムアウトを延長できます。

詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

Amazon EC2 Auto Scaling プロセスの停止と再開

このトピックでは、Auto Scaling グループの 1 つまたは複数のプロセスを中断してから再開して、一時的に特定のオペレーションを無効にする方法について説明します。

プロセスの中断は、スケーリングポリシーやスケジュールされたアクションに干渉されることなく、問題を調査またはトラブルシューティングする必要がある場合に役立ちます。また、Auto Scaling グループを変更している間に、Amazon EC2 Auto Scaling がインスタンスを異常とマークして置き換えるのを防ぐこともできます。

トピック

> **Note**
>
> Amazon EC2 Auto Scaling は、ユーザーが開始した停止に加えて、インスタンスの起動に繰り返し失敗する Auto Scaling グループのプロセスを停止することもできます。これは、管理上の中断と呼ばれます。管理上の中断は一般に、24 時間以上インスタンスの起動を試みているが、インスタンスの起動に成功しない Auto Scaling グループに適用されます。管理上の理由から、Amazon EC2 Auto Scaling によって中断されたプロセスを再開できます。

プロセスのタイプ

中断/再開機能は、以下のプロセスをサポートします。

- `Launch` – グループがスケールアウトしたとき、または Amazon EC2 Auto Scaling がウォームプールにインスタンスを追加するときなど、その他の理由でインスタンスを起動することを選択した場合に、Auto Scaling グループにインスタンスを追加します。 Auto Scaling
- `Terminate` – Auto Scaling グループのスケールイン時、またはインスタンスが最大有効期間を超えたために終了した場合やヘルスチェックに失敗した場合など、他の理由で Amazon EC2 Auto

Scaling がインスタンスの終了を選択した場合に、Auto Scaling グループからインスタンスを削除します。

- `AddToLoadBalancer` – インスタンスが起動されたときに、アタッチされたロードバランサーターゲットグループまたは Classic Load Balancer にインスタンスを追加します。詳細については、「Elastic Load Balancing を使用して Auto Scaling グループ内の受信アプリケーショントラフィックを分散する」を参照してください。
- `AlarmNotification` – 動的スケーリングポリシーに関連付けられているアラームからのCloudWatch通知を受け入れます。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。
- `AZRebalance` - 以前に使用できなかったアベイラビリティーゾーンが正常な状態に戻ったときなど、グループのバランスが取れなくなったときに、指定されたすべてのアベイラビリティーゾーンでグループ内のEC2インスタンスの数を均等に分散します。詳細については、「アクティビティの再分散」を参照してください。
- `HealthCheck` — Amazon EC2または Elastic Load Balancing がインスタンスに異常があることを Amazon EC2 Auto Scaling に指示した場合、インスタンスの状態をチェックし、インスタンスを異常としてマークします。このプロセスは、手動で設定したインスタンスのヘルス ステータスをオーバーライドできます。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。
- `InstanceRefresh` – インスタンスの更新機能を使用して、インスタンスを終了および置換します。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。
- `ReplaceUnhealthy` – 異常とマークされたインスタンスを終了してから、新しいインスタンスを作成して置き換えます。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。
- `ScheduledActions` – AWS Auto Scaling スケーリングプランを作成し、予測スケーリングをオンにしたときに、作成したか、自動的に作成されたスケジュールされたスケーリングアクションを実行します。詳細については、「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。

プロセスの中断に関する考慮事項

プロセスを中断する前に、以下を考慮してください。

- 一時停止`AlarmNotification`すると、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、グループのターゲット追跡、ステップ、簡易スケーリングポリシーを一時

的に停止できます。その代わりに個々のスケーリングポリシーを一時的に停止するには、「Auto Scaling グループのスケーリングポリシーを無効化する」を参照してください。

- Amazon EC2 Auto Scaling がヘルスチェックに基づいてインスタンスを終了することなく、インスタンスを再起動する HealthCheckおよび ReplaceUnhealthyプロセスを停止することもできます。ただし、Amazon EC2 Auto Scaling が残りのインスタンスでヘルスチェックを引き続き実行する必要がある場合は、代わりにスタンバイ機能を使用します。詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。
- Launch プロセスと Terminate プロセス、または AZRebalance を中断してから、インスタンスのデタッチ、または指定されたアベイラビリティーゾーンの変更などで Auto Scaling グループを変更すると、アベイラビリティーゾーン間でのグループのバランスが悪くなる可能性があります。この場合、中断されたプロセスを再開した後、Amazon EC2 Auto Scaling はアベイラビリティーゾーン間でインスタンスを均等に徐々に再分散します。
- Terminate プロセスを停止しても、強制削除オプションを指定して delete-auto-scaling-group コマンドを使用することで、インスタンスを強制的に終了できます。
- Terminate プロセスの中断は、現在 InService 状態にあるインスタンスにのみ適用されます。Pending など、他の状態のインスタンスや、スタンバイから適切に再開できないインスタンスの終了は妨げられません。
- AWS CLI または を使用して Auto Scaling グループを記述する呼び出しに存在する場合、RemoveFromLoadBalancerLowPriorityこのプロセスは無視できますSDKs。このプロセスは非推奨で後方互換性のためにのみ保持されています。

プロセスを中断する

Auto Scaling グループのプロセスを中断するには、次のいずれかの方法を使用します。

Console

プロセスを停止するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[高度な設定]、[編集] の順に選択します。
4. [Suspended processes (中断したプロセス)] で、停止するプロセスを選択します。

5. [Update] (更新) を選択します。

AWS CLI

以下の suspend-processes コマンドを使用して、個々のプロセスを中断します。

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-
processes HealthCheck ReplaceUnhealthy
```

すべてのプロセスを中断するには、以下のように `--scaling-processes` オプションを削除します。

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

プロセスを再開する

Auto Scaling グループの中断されたプロセスを再開するには、次のいずれかの方法を使用します。

Console

停止されたプロセスを再開するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[高度な設定]、[編集] の順に選択します。
4. [Suspended processes] (中断されたプロセス) で、中断されたプロセスを削除します。
5. [Update] (更新) を選択します。

AWS CLI

中断されたプロセスを再開するには、以下の resume-processes コマンドを使用します。

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-
processes HealthCheck
```

中断されたすべてのプロセスを再開するには、以下のように --scaling-processes オプションを削除します。

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

中断されているプロセスが他のプロセスに及ぼす影響

以下のセクションでは、さまざまなプロセスが個々に中断されたときに発生する現象について説明しています。

トピック

Launch は中断されています

- AlarmNotification は引き続きアクティブですが、Auto Scaling グループは、しきい値を超過した状態のアラームに対してスケールアウトアクティビティを開始できません。
- ScheduledActions はアクティブですが、Auto Scaling グループは、実行されるスケジュールされたアクションに対してスケールアウトアクティビティを開始できません。
- AZRebalance は、グループの再配分を停止します。
- ReplaceUnhealthy は引き続き異常なインスタンスを終了しますが、置き換えは開始しません。Launch プロセスを再開すると、Amazon EC2 Auto Scaling Launchは中断期間中に終了したインスタンスを直ちに置き換えます。
- InstanceRefresh はインスタンスを置き換えません。

`Terminate` は中断されています

- `AlarmNotification` は引き続きアクティブですが、Auto Scaling グループは、しきい値を超過した状態のアラームに対してスケールインアクティビティを開始できません。
- `ScheduledActions` はアクティブですが、Auto Scaling グループは、実行されるスケジュールされたアクションに対してスケールインアクティビティを開始できません。
- `AZRebalance` はまだアクティブですが、正しく機能していません。古いインスタンスを終了せずに新しいインスタンスを起動することがあります。これにより、Auto Scaling グループがその最大サイズより 10% まで大きくなることあります。バランスの再調整アクティビティ中にこの状態が一時的に許可されるためです。`Terminate` プロセスを再開するまで、Auto Scaling グループは最大サイズを超えることがあります。
- `ReplaceUnhealthy` は非アクティブですが、`HealthCheck` はアクティブです。`Terminate` が再開されると、`ReplaceUnhealthy` プロセスはすぐに実行を開始します。`Terminate` が中断されている間に異常とマークされたインスタンスがある場合、それらのインスタンスはすぐに置き換えられます。
- `InstanceRefresh` はインスタンスを置き換えません。

`AddToLoadBalancer` は中断されています

- Amazon EC2 Auto Scaling はインスタンスを起動しますが、ロードバランサーのターゲットグループまたは Classic Load Balancer には追加しません。`AddToLoadBalancer` プロセスを再開すると、インスタンスが起動されるときのロードバランサーへの追加が再開されます。ただし、このプロセスが中断されている間に起動されたインスタンスは追加されません。これらのインスタンスを手動で登録する必要があります。

`AlarmNotification` は中断されています

- Amazon EC2 Auto Scaling は、 CloudWatch アラームのしきい値を超過してもスケーリングポリシーを呼び出しません。を再開すると`AlarmNotification`、Amazon EC2 Auto Scaling は現在違反しているアラームしきい値を持つポリシーを考慮します。

`AZRebalance` は中断されています

- Amazon EC2 Auto Scaling は、特定のイベントの後にインスタンスの再配布を試みません。ただし、スケールアウトまたはスケールインのイベントが発生した場合でも、スケーリングプロセスは

アベイラビリティーゾーン間のバランスを調整しようとします。例えば、スケールアウト中に、インスタンスが最も少ないアベイラビリティーゾーンでインスタンスを起動します。AZRebalance が中断されている間にグループのバランスが崩れて再開すると、Amazon EC2 Auto Scaling はグループのバランスを再調整しようとします。最初に Launch を呼び出してから Terminate を呼び出します。

- が停止されている場合、ウォームプールAZRebalanceは影響を受けません。

HealthCheck は中断されています

- Amazon EC2 Auto Scaling は、 EC2および Elastic Load Balancing ヘルスチェックの結果としてインスタンスの異常のマークを停止します。カスタムヘルスチェックは引き続き正常に機能します。HealthCheck を中断した後、必要に応じて、グループ内のインスタンスのヘルス状態を手動で設定し、ReplaceUnhealthy がそれらのインスタンスを置き換えるようにできます。

InstanceRefresh は中断されています

- Amazon EC2 Auto Scaling は、インスタンスの更新の結果としてインスタンスの置き換えを停止します。進行中のインスタンス更新がある場合、操作はキャンセルされず、一時停止されます。

ReplaceUnhealthy は中断されています

- Amazon EC2 Auto Scaling は、異常とマークされたインスタンスの置き換えを停止します。失敗EC2または Elastic Load Balancing のヘルスチェックに失敗したインスタンスは、引き続き異常としてマークされます。ReplaceUnhealthy プロセスを再開するとすぐに、Amazon EC2 Auto Scaling は、このプロセスの中断中に異常とマークされたインスタンスを置き換えます。ReplaceUnhealthy プロセスは最初に Terminate を呼び出し、次に Launch を呼び出します。

ScheduledActions は中断されています

- Amazon EC2 Auto Scaling は、停止期間中に実行が予定されているスケジュールされたアクションを実行しません。を再開するとScheduledActions、Amazon EC2 Auto Scaling は、スケジュールされた時間が経過していないスケジュールされたアクションのみを考慮します。

追加の考慮事項

さらに、Launch または Terminate が中断される場合は、以下の機能が正しく機能しない可能性があります。

- インスタンスの最大有効期間 – Launch または Terminate が中断されている場合、インスタンスの最大有効期間機能でインスタンスを置き換えることはできません。
- スポットインスタンスの中断 – Terminate が中断されていても、Auto Scaling グループにスポットインスタンスがある場合には、スポットキャパシティが使用可能でなくなった場合にそれらのインスタンスを終了することが可能です。Launch が中断されている間、Amazon EC2 Auto Scaling は、別のスポットインスタンスプールから、または同じスポットインスタンスプールから、再び利用可能になると、代替インスタンスを起動できません。
- 容量の再調整 — Terminateが中断され、スポットインスタンスの中断を処理するために容量の再調整を使用する場合、EC2スポット容量が使用できなくなっても Amazon スポットサービスはインスタンスを終了できます。Launch が停止されている場合、Amazon EC2 Auto Scaling は、別のスポットインスタンスプールまたは同じスポットインスタンスプールから、再び利用可能になると、代替インスタンスを起動できません。
- インスタンスのアタッチとデタッチ – Launch と Terminate が中断されると、Auto Scaling グループにアタッチされているインスタンスをデタッチすることはできますが、Launch が中断されている間は、新しいインスタンスをそのグループにアタッチできません。
- スタンバイインスタンス – Launch と Terminate が中断されると、インスタンスを Standby 状態にすることはできますが、Launch が中断されている間は、Standby 状態のインスタンスを service に戻すことはできません。

Amazon EC2 Auto Scaling グループをモニタリングする

モニタリングは、Amazon EC2 Auto Scaling および AWS クラウド ソリューションの信頼性、可用性、およびパフォーマンスを維持する上で重要な部分です。AWS は、Amazon EC2 Auto Scaling をモニタリングしたり、問題が発生したときに報告したり、必要に応じて自動アクションを実行したりするために以下のモニタリングツールが用意されています。

ヘルスチェック

Amazon EC2 Auto Scaling は、Auto Scaling グループのインスタンスに対して定期的にヘルスチェックを実行します。インスタンスがヘルスチェックに合格しない場合、そのインスタンスは異常とマークされ、Amazon EC2 Auto Scaling がインスタンスを置き換えるために新しいインスタンスを起動する間に終了します。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

AWS Health Dashboard

AWS Health Dashboard は情報を表示し、AWS リソースのヘルス状態の変化によってトリガー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、「Amazon EC2 Auto Scaling の AWS Health Dashboard 通知」を参照してください。

CloudTrail

AWS CloudTrail を使用すると、AWS アカウント として、または代理として実行した Amazon EC2 Auto Scaling API へのコールを追跡できます。CloudTrail は、その情報をログファイルの形で指定した Amazon S3 バケットに格納します。これらのログファイルを使用して、Auto Scaling グループの動作をモニタリングできます。ログには、実行されたリクエスト、そのリクエストの作成元のソース IP アドレス、リクエストの実行者、リクエストの実行日時などが含まれています。詳細については、「を使用した Amazon EC2 Auto Scaling API呼び出しの口グ記録 AWS CloudTrail」を参照してください。

> **Amazon EC2 インスタンスのログ収集**
>
> CloudWatch を使用して、EC2 インスタンスのオペレーティングシステムからログを収集できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する」および「CloudWatch Logs に送信されたログデータを表示する」を参照してください。

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その他の AWS サービスについては、「AWS Prescriptive Guidanceの「Logging and monitoring guide for application owners」ガイドを参照してください。

Amazon CloudWatch

Amazon CloudWatch を使用すると、ログを分析し、リアルタイムで AWS リソースとホストアプリケーションのメトリクスをモニタリングできます。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクス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の設定を行うことができます。たとえば、ネットワークアクティビティがメトリクスの期待値よりも急激に高くなった、または低くなったときに、通知を受け取ることができます。このサービスを使用して Auto Scaling グループとインスタンスのメトリックスをモニターリングする方法の詳細については、「Auto Scaling グループとインスタンスの CloudWatch メトリクスを監視する」を参照してください。

CloudWatch は、Amazon EC2 Auto Scaling の AWS API 使用状況メトリクスも追跡します。これらのメトリックスを使用して、API 呼び出し量が定義したしきい値を超えたときに警告するアラームを設定できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「AWS 使用状況メトリクスの使用」を参照してください。

AWS Compute Optimizer

Compute Optimizer は、新しいインスタンスタイプに移行するかどうかの判断に役立つ Amazon EC2 インスタンス推奨を提供します。Auto Scaling グループのインスタンスタイプが最適かどうかを分析し、コストを削減してワークロードのパフォーマンスを向上させるための推奨事項を生成します。詳細については、「でインスタンスタイプのレコメンデーションを取得する AWS Compute Optimizer」を参照してください。

Amazon EventBridge

Amazon EventBridge は、アプリケーションをさまざまなイベントソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、お客様独自のアプリケーション、Software as a Service (SaaS) アプリケーション、AWS のサービスからのリアルタイムデータをストリーム配信し、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、「Auto Scaling イベントの処理に EventBridge を使用する」を参照してください。

AWS Security Hub

AWS Security Hub を使用して、セキュリティのベストプラクティスに関連して Amazon EC2 Auto Scaling の使用状況をモニタリングできます。Security Hub は、検出セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。Security Hub を使用して Amazon EC2 Auto Scaling リソースを評価する方法の詳細については、「AWS Security Hub ユーザーガイド」の「Amazon EC2 Auto Scaling コントロール」を参照してください。

Amazon Simple Notification Service

Amazon EC2 Auto Scaling がインスタンスを起動または終了するときに Amazon SNS 通知を送信するように、Auto Scaling グループを設定できます。詳しくは、「Amazon EC2 Auto Scaling の Amazon SNS 通知オプション」を参照してください。

Auto Scaling グループでのインスタンスのヘルスチェック

Amazon EC2 Auto Scaling は、Auto Scaling グループ内のインスタンスのヘルスステータスを継続的にモニタリングして、希望するキャパシティを維持します。

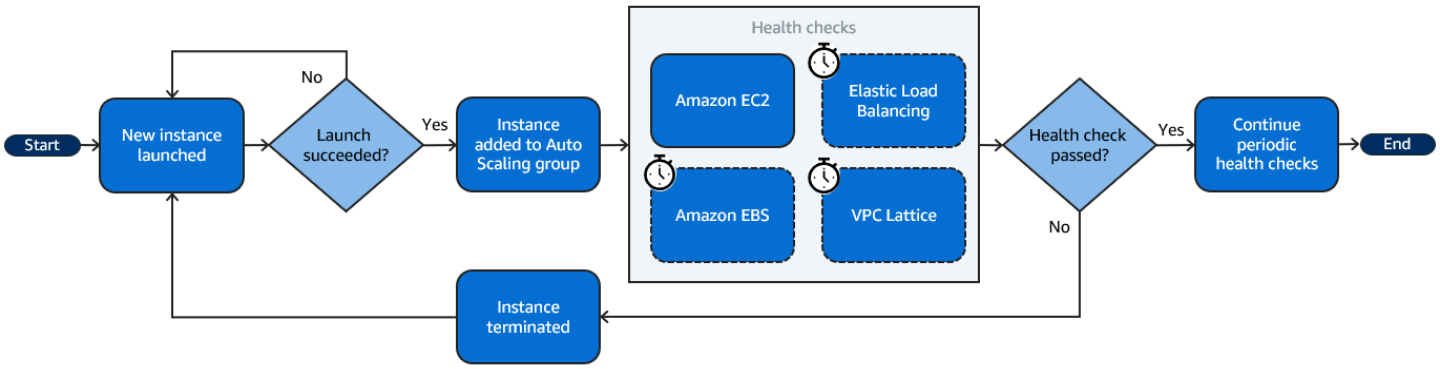
Auto Scaling グループ内のすべてのインスタンスは `Healthy` ステータスで開始します。インスタンスに異常があるという通知を Amazon EC2 Auto Scaling が受け取らない限り、インスタンスは正常であると見なされます。インスタンスが異常になり、置き換える必要がある場合、さまざまなソースから通知を受け取る可能性があります。こうしたソースには、以下が含まれます。

- Amazon EC2
- Elastic Load Balancing
- VPC Lattice
- Amazon EBS
- 定義したカスタムヘルスチェック

Amazon EC2 Auto Scaling が `InService` インスタンスに異常があると判断した場合、そのインスタンスを新しいインスタンスに置き換えて、グループの希望するキャパシティを維持します。新しいインスタンスは、Auto Scaling グループの現在の設定、およびそれに関連する起動テンプレートまたは起動設定を使用して起動します。

次のフロー図は、Auto Scaling グループで新しいインスタンスを起動するプロセスを示しています。インスタンスを起動することから始まります。起動が成功すると、インスタンスが Auto Scaling グ

ループに追加されます。次に、Amazon EC2 Auto Scaling は、組み込みの Amazon EC2 ステータスチェックを使用してインスタンスのヘルスチェックを実行し、猶予期間後にグループに対して有効にしたオプションのヘルスチェックを実行します。これらのヘルスチェックは継続して定期的に実行されます。ヘルスチェックのいずれかが不合格になると、インスタンスは置き換えられます。



異常なインスタンスは、スポットインスタンスの中断やユーザーによる手動終了など、インスタンスが予期せず終了した場合に発生することもあります。この場合も、Amazon EC2 Auto Scaling は希望するキャパシティを維持するために、代替インスタンスを自動的に起動します。

内容

- Auto Scaling グループのヘルスチェックについて
- Auto Scaling グループにヘルスチェックの猶予期間を設定する
- ヘルスチェックを使用して、Amazon EBS ボリュームに障害がある Auto Scaling インスタンスをモニタリングする
- Auto Scaling グループに対してカスタムヘルスチェックを設定する
- ヘルスチェック不合格の理由を表示する
- Amazon EC2 Auto Scaling の異常なインスタンスをトラブルシューティングする

Auto Scaling グループのヘルスチェックについて

このトピックでは、使用可能なヘルスチェックタイプの概要と、Amazon EC2 Auto Scaling ヘルスチェックをアプリケーションと統合する際の主な考慮事項について説明します。

内容

- ヘルスチェックタイプ
- Amazon EC2 ヘルスチェック
- Elastic Load Balancing のヘルスチェック

- VPC Lattice ヘルスチェック
- Amazon EC2 Auto Scaling によってダウンタイムが最小限に抑えられる仕組み
- ウォームプール内のインスタンスのヘルスチェック
- ヘルスチェックの考慮事項

ヘルスチェックタイプ

Amazon EC2 Auto Scaling は、以下のヘルスチェックの 1 つ、または複数を使用することで、`InService` インスタンスのヘルスステータスを判断できます。

ヘルスチェックタイプ	チェックする事柄
Amazon EC2 ステータスチェックと予定されているイベント	• インスタンスが実行中であることをチェックします。 • ハードウェアまたはソフトウェアの根本的な問題で、インスタンスの機能を損なう可能性があるものをチェックします。 これは、Auto Scaling グループに対するデフォルトのヘルスチェックタイプです。
Elastic Load Balancing のヘルスチェック	• ロードバランサーがインスタンスを正常として報告しているかどうかをチェックして、インスタンスがリクエストを処理できることを確認します。 このヘルスチェックタイプを実行するには、Auto Scaling グループに対してこのタイプを有効にする必要があります。
VPC Lattice ヘルスチェック	• VPC Lattice がインスタンスを正常として報告しているかどうかをチェックして、インスタンスがリクエストを処理できることを確認します このヘルスチェックタイプを実行するには、Auto Scaling グループに対してこのタイプを有効にする必要があります。
Amazon EBS ヘルスチェック	• EBS ボリュームに到達可能かどうか、および I/O ステータスチェックに合格するかどうかを確認します。

ヘルスチェックタイプ	チェックする事柄
	このヘルスチェックタイプを実行するには、Auto Scaling グループに対してこのタイプを有効にする必要があります。
カスタムヘルスチェック	• カスタムヘルスチェックに従って、インスタンスのヘルス問題を示す可能性のあるその他の問題がないかチェックします。

Amazon EC2 ヘルスチェック

インスタンスが起動されると、インスタンスは Auto Scaling グループにアタッチされ、`InService` 状態になります。Auto Scaling グループ内のインスタンスの異なるライフサイクル状態に関する詳細については、「Amazon EC2 Auto Scaling インスタンスのライフサイクル」を参照してください。

Amazon EC2 Auto Scaling は、Auto Scaling グループ内のすべてのインスタンスのヘルスステータスを定期的にチェックすることで、それらが実行中で良好な状態であることを確認します。

ステータスチェック

Amazon EC2 Auto Scaling は、Amazon EC2 インスタンスのステータスチェックとシステムステータスチェックの結果を使用して、インスタンスのヘルスステータスを判断します。インスタンスが `running` 以外の Amazon EC2 状態である場合、またはステータスチェックのステータスが `impaired` になった場合、Amazon EC2 Auto Scaling はインスタンスを異常であると見なし、そのインスタンスを置き換えます。これには、インスタンスが以下のいずれかの状態にある場合が含まれます。

- `stopping`
- `stopped`
- `shutting-down`
- `terminated`

Amazon EC2 ステータスチェックに特別な設定は必要なく、常に有効になっています。詳細については、「Amazon EC2 ユーザーガイド」の「ステータスチェックのタイプ」を参照してください。

> **Important**
>
> Amazon EC2 Auto Scaling は、何のアクションも実行せずにステータスチェックを不合格にすることがあります。ステータスチェックが不合格になると、Amazon EC2 Auto Scaling は AWS が問題を解決するまで数分待機します。ステータスチェックのステータスが `impaired` になっても、インスタンスは直ちに `Unhealthy` としてマークされません。ただし、インスタンスが `running` 状態ではなくなったことを Amazon EC2 Auto Scaling が検出すると、この状況は即時不合格として扱われます。この場合、インスタンスは直ちに `Unhealthy` としてマークされ、置き換えられます。

予定されているイベント

Amazon EC2 は、インスタンスのイベントを、特定のタイムスタンプ後に実行されるようにスケジュールすることがあります。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの予定されているイベント」を参照してください。

インスタンスのいずれかが予定されているイベントの影響を受ける場合、Amazon EC2 Auto Scaling は、そのインスタンスを異常と見なして置き換えます。インスタンスのシャットダウンは、タイムスタンプで指定された日付と時刻に到達するまで開始されません。

Elastic Load Balancing のヘルスチェック

Auto Scaling グループに対して Elastic Load Balancing ヘルスチェックを有効にすると、Amazon EC2 Auto Scaling はこれらのヘルスチェックの結果を使用して、インスタンスのヘルスステータスを判断できます。

Auto Scaling グループに対して Elastic Load Balancing ヘルスチェックを有効にする前に、Elastic Load Balancing ロードバランサーを設定し、インスタンスが正常かどうかを判断するためのヘルスチェックを設定する必要があります。詳細については、「Elastic Load Balancing ロードバランサーをアタッチする準備をする」を参照してください。

ロードバランサーを Auto Scaling グループにアタッチすると、次のようになります。

- Amazon EC2 Auto Scaling が、Auto Scaling グループ内のインスタンスをロードバランサーに登録します。
- インスタンスの登録が終了すると、インスタンスは `InService` 状態になり、ロードバランサーで使用できるようになります。

デフォルトで、Amazon EC2 Auto Scaling は Elastic Load Balancing ヘルスチェックの結果を無視しますが、Auto Scaling グループに対してこれらのヘルスチェックを有効にした後、登録されたインスタンスを Elastic Load Balancing が `Unhealthy` と報告すると、Amazon EC2 Auto Scaling は、次回の定期ヘルスチェックでそのインスタンスを `Unhealthy` とマークし、置き換えます。

ロードバランサーに対して Connection Draining が有効になっている場合、Amazon EC2 Auto Scaling は、処理中のリクエストが完了するまで、または最大タイムアウト時間が終了するまで待機してから、異常なインスタンスを終了します。

> **Note**
>
> ロードバランサーをアタッチして Auto Scaling グループについての Elastic Load Balancing ヘルスチェックを有効にする方法については、「Auto Scaling グループに Elastic Load Balancing ロードバランサーをアタッチする」を参照してください。
> グループに対して Elastic Load Balancing ヘルスチェックを有効にすると、Amazon EC2 Auto Scaling は、Elastic Load Balancing から異常として報告されたインスタンスを置き換えることができます。ただし、置き換えは、ロードバランサーが `InService` 状態になってから行われます。詳細については、「ロードバランサーのアタッチメントステータスを確認する」を参照してください。

VPC Lattice ヘルスチェック

デフォルトでは、Amazon EC2 Auto Scaling は、VPC Lattice ヘルスチェックの結果を無視します。オプションで、Auto Scaling グループに対してこれらのヘルスチェックを有効にできます。これらのヘルスチェックを有効化した後、VPC Lattice が登録されたインスタンスを `Unhealthy` として報告すると、Amazon EC2 Auto Scaling は、次回の定期ヘルスチェックでそのインスタンスを `Unhealthy` としてマークし、置き換えます。インスタンスを登録してからその状態をチェックする処理は、Elastic Load Balancing ヘルスチェックの仕組みと同じです。

> **Note**
>
> VPC Lattice ターゲットグループをアタッチし、Auto Scaling グループに対して VPC Lattice ヘルスチェックを有効にする方法については、「VPC Lattice ターゲットグループを Auto Scaling グループにアタッチする」を参照してください。
> グループに対して VPC Lattice ヘルスチェックを有効にすると、Amazon EC2 Auto Scaling は、VPC Lattice から異常として報告されたインスタンスを置き換えることができます。ただし、置き換えは、ターゲットグループが `InService` 状態になってから行われます。詳細

については、「VPC Lattice ターゲットグループのアタッチメントステータスを確認する」を参照してください。

Amazon EC2 Auto Scaling によってダウンタイムが最小限に抑えられる仕組み

デフォルトでは、新しいインスタンスは既存のインスタンスが終了すると同時にプロビジョニングされるため、新しいインスタンスが完全に動作するまで、新しいリクエストが受け入れられなくなる可能性があります。

Amazon EC2 Auto Scaling は、実行されていない (つまり set-instance-health コマンドで `Unhealthy` とマークされた) インスタンスがあることを認識すると、それらを直ちに置き換えます。ただし、他のインスタンスが異常である場合、Amazon EC2 Auto Scaling は不合格状態からの回復に以下のアプローチを使用します。このアプローチは、一時的な問題、または誤設定されたヘルスチェックが原因で発生する可能性があるダウンタイムを最小限に抑えます。

- スケーリングアクティビティが進行中で、Auto Scaling グループのキャパシティが希望するキャパシティより 10% 以上少ない場合、Amazon EC2 Auto Scaling は、進行中のスケーリングアクティビティの終了を待ってから、異常なインスタンスを置き換えます。
- スケールアウト時、Amazon EC2 Auto Scaling はインスタンスが最初のヘルスチェックに合格するのを待ちます。また、デフォルトのインスタンスウォームアップが完了するのを待って、新しいインスタンスの準備が整っていることも確実にします。
- インスタンスがウォームアップを完了し、グループのキャパシティが希望するキャパシティの 90% を超えると、Amazon EC2 Auto Scaling は以下のように異常なインスタンスを置き換えます。
  - Amazon EC2 Auto Scaling が一度に置き換えるインスタンスはグループの希望容量の最大 10% のみで、異常なインスタンスのすべてが置き換えられるまで続行されます。
  - インスタンスを置き換えるときは、新しいインスタンスが最初のヘルスチェックに合格するのを待ちます。また、デフォルトのインスタンスウォームアップが完了するのも待ち、完了後に続行します。

> **Note**
>
> 10% の値が結果として 1 未満になるほど Auto Scaling グループのサイズが小さい場合、通常と異なり、Amazon EC2 Auto Scaling は異常なインスタンスを一度に 1 つずつ置き換えます。これは、グループのダウンタイムの原因になる可能性があります。

また、Auto Scaling グループ内のすべてのインスタンスが異常であると Elastic Load Balancing ヘルスチェックが報告し、ロードバランサーが `InService` 状態である場合、Amazon EC2 Auto Scaling は一度に異常としてマークするインスタンス数を減らすことがあります。そうすることで、他のシナリオに適用される 10% よりも、一度に置き換えられるインスタンスの数を大幅に削減することができます。これにより、問題を修正する時間が確保され、Amazon EC2 Auto Scaling がグループ全体を自動的に終了することはありません。

ウォームプール内のインスタンスのヘルスチェック

Amazon EC2 Auto Scaling はウォームプール内のインスタンスのヘルスチェックも行います。詳細については、「ヘルスチェックのステータスとヘルスチェックの失敗理由を表示する」を参照してください。

ヘルスチェックの考慮事項

Amazon EC2 Auto Scaling ヘルスチェックを使用する際の考慮事項を次に示します。

- 終了処理中のインスタンス、または起動中のインスタンスで何かを実行する必要がある場合は、ライフサイクルフックを使用することができます。これらのフックを使用すると、Amazon EC2 Auto Scaling がインスタンスを起動または終了する時点で、カスタムアクションを実行できます。詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。
- Amazon EC2 Auto Scaling は、そのヘルスチェックから Amazon EC2 ステータスチェックと予定されているイベントを削除する方法を提供しません。インスタンスが置き換えられないようにしたい場合は、個々の Auto Scaling グループについて `ReplaceUnhealthy` プロセスと `HealthCheck` プロセスを停止することをお勧めします。詳細については、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。
- 異常なインスタンスのヘルスステータスを手動で `Healthy` に戻す場合は、set-instance-health コマンドの使用を試みることができます。エラーが発生する場合、その原因はインスタンスが既に終了中であるためだと考えられます。通常、set-instance-health コマンドを使用してインスタンスのヘルスステータスを `Healthy` に戻すことは、`ReplaceUnhealthy` プロセスまたは `Terminate` プロセスが停止している場合にのみ有効です。
- ヘルスチェックの影響を受けずにインスタンスのトラブルシューティングが必要な場合は、インスタンスを `Standby` 状態にすることができます。Amazon EC2 Auto Scaling は、`Standby` 状態のインスタンスに対して、そのインスタンスを稼働状態に戻すまではヘルスチェックを実行しませ

ん。詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

- インスタンスを削除すると、関連付けられたすべての Elastic IP アドレスは関連付けを解除され、新しいインスタンスと自動的に関連付けられることはありません。これらの Elastic IP アドレスと新しいインスタンスは、手動で関連付けるか、ライフサイクルフックベースのソリューションを使用して自動的に関連付ける必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「Elastic IP アドレス」を参照してください。
- 同様に、インスタンスが終了されると、それにアタッチされている EBS ボリュームがデタッチ (または、ボリュームの `DeleteOnTermination` 属性に応じて削除) されます。これらの EBS ボリュームは、新しいインスタンスに手動でアタッチするか、ライフサイクルフックベースのソリューションを使用して自動的にアタッチする必要があります。詳細については、「Amazon EBS ユーザーガイド」の「インスタンスへの Amazon EBS ボリュームのアタッチ」を参照してください。

Auto Scaling グループにヘルスチェックの猶予期間を設定する

Amazon EC2 Auto Scaling ヘルスチェックは、ある `InService` インスタンスが異常であると判断すると、新しいインスタンスに置き換えます。ヘルスチェックの猶予期間は、新しいインスタンスに異常がある場合、そのインスタンスを終了するまでに稼働する最小時間 (秒単位) を指定します。

ユースケース例としては、Elastic Load Balancing のヘルスチェックが失敗し、インスタンスがまだ初期化していることが原因である場合、Amazon EC2 Auto Scaling が実行を回避する要件があります。Elastic Load Balancing のヘルスチェックは、インスタンスがロードバランサーに登録されたときに開始して並行で実行されます。猶予期間によって、新しく起動されたインスタンスが `InService` 状態になった後、これらのヘルスチェックにすぐに合格しなかった場合に、Amazon EC2 Auto Scaling がそれらのインスタンスを `Unhealthy` とマークして不必要に終了しないようになります。

コンソールでは、Auto Scaling グループを作成するときのヘルスチェックの猶予期間がデフォルトで 300 秒になっています。AWS CLI または SDK を使用して Auto Scaling グループを作成するときのデフォルト値は 0 秒です。値が 0 の場合、ヘルスチェックの猶予期間は無効になります。

この値を高く設定しすぎると、Amazon EC2 Auto Scaling ヘルスチェックの効果が低減します。インスタンスの起動にライフサイクルフックを使用する場合は、ヘルスチェックの猶予期間の値を 0 に設定できます。ライフサイクルフックを使用すると、Amazon EC2 Auto Scaling は、インスタンスが常に初期化されてから `InService` 状態になることを確実にするための手段を提供します。詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

猶予期間は以下のインスタンスに適用されます。

- 新しく起動されたインスタンス
- スタンバイ状態になった後で実行状態に戻されるインスタンス
- グループに手動でアタッチされるインスタンス

> **Important**
>
> ヘルスチェックの猶予期間中に Amazon EC2 Auto Scaling が Amazon EC2 `running` 状態ではなくなったインスタンスを検出した場合は、直ちにそのインスタンスを `Unhealthy` としてマークし、置き換えます。例えば、Auto Scaling グループ内のインスタンスを停止すると、そのインスタンスは `Unhealthy` とマークされ、置き換えられます。

グループにヘルスチェックの猶予期間を設定する

ヘルスチェックの猶予期間は、新規または既存の Auto Scaling グループに設定できます。

Console

新しいグループのヘルスチェックの猶予期間を変更するには

Auto Scaling グループを作成するときは、[詳細オプションを設定] ページにある [ヘルスチェック] の [ヘルスチェックの猶予期間] に、時間を秒単位で入力します。これは、インスタンスが `InService` 状態になってからインスタンスのヘルスステータスを確認するまでに Amazon EC2 Auto Scaling が待機する必要がある時間です。

AWS CLI

新しいグループのヘルスチェックの猶予期間を変更するには

create-auto-scaling-group コマンドに `--health-check-grace-period` オプションを追加します。以下の例は、*my-asg* という名前の新しい Auto Scaling グループに対するヘルスチェック猶予期間を *60* 秒の値で設定します。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 60 ...
```

Console

既存グループのヘルスチェックの猶予期間を変更するには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
5. [Health check grace period] (ヘルスチェックの猶予期間) に、秒単位で時間を入力します。これは、インスタンスが `InService` 状態になってからインスタンスのヘルスステータスを確認するまでに Amazon EC2 Auto Scaling が待機する必要がある時間です。
6. [Update] (更新) を選択します。

AWS CLI

既存グループのヘルスチェックの猶予期間を変更するには

update-auto-scaling-group コマンドに `--health-check-grace-period` オプションを追加します。以下の例は、*`my-asg`* という名前の既存の Auto Scaling グループに対するヘルスチェック猶予期間を *`120`* 秒の値で設定します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 120
```

> **Note**
>
> Auto Scaling グループに対してデフォルトのインスタンスウォームアップ時間を設定することも強く推奨されます。詳細については、「Auto Scaling グループに対するインスタンスのデフォルトウォームアップを設定する」を参照してください。

ヘルスチェックを使用して、Amazon EBS ボリュームに障害がある Auto Scaling インスタンスをモニタリングする

Auto Scaling グループの Amazon EBS ヘルスチェックを有効にして、アプリケーションが実行されているシステム全体を Amazon EC2 Auto Scaling がモニタリングするようにできます。

これらのヘルスチェックを有効にすると、Amazon EC2 Auto Scaling は、インスタンスにアタッチされた EBS ボリュームで実行された Amazon EC2 ステータスチェックの結果を受け取ります。ボリュームにアクセスできない、またはボリュームが I/O ステータスチェックに合格しない場合、ヘルスチェックは不合格になり、対応するインスタンスは異常と見なされます。Amazon EC2 Auto Scaling は、異常なインスタンスを検出すると、それを置き換えます。

このトピックでは、アタッチされた EBS ステータスチェックに読者が精通していることを前提としています。そうでない場合は、「Amazon EC2 ユーザーガイド」の「アタッチ済みの EBS ステータスチェック」セクションを参照してください。次のトピックでは、アタッチされた EBS ステータスチェックに依存する Amazon EC2 Auto Scaling ヘルスチェックを有効にする方法について説明します。

> **Note**
>
> すべての Auto Scaling グループの Amazon EBS ヘルスチェックを有効にできます。ただし、これらのヘルスチェックは AWS Nitro System 上に構築されたインスタンスに対してのみ使用できます。

グループに対して Amazon EBS ヘルスチェックを有効にする

新規および既存の Auto Scaling グループに対して Amazon EBS ヘルスチェックを有効にできます。

Console

新しいグループに対する Amazon EBS ヘルスチェックの有効化

Auto Scaling グループを作成するときに、[詳細オプションを設定] ページにある [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Amazon EBS ヘルスチェックを有効にする] を選択します。そして、[ヘルスチェックの猶予期間] に、秒単位で時間を入力します。これは、インスタンスが `InService` 状態になってからインスタンスのヘルスステータスを確認するまでに、Amazon EC2 Auto Scaling が待機する必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

AWS CLI

新しいグループに対する Amazon EBS ヘルスチェックの有効化

create-auto-scaling-group コマンドに `--health-check-type` オプションを追加します。次の例では、*my-asg* という名前の新しい Auto Scaling グループの `--health-check-type` オプションに **EBS** を指定します。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60 ...
```

`--health-check-type` オプションには複数の値を指定できます。例えば、Amazon EBS と Elastic Load Balancing の両方のヘルスチェックタイプを追加するには、次のコマンドを使用します。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS,ELB" --health-check-grace-period 60 ...
```

値名では大文字と小文字が区別されます。

Console

既存のグループに対する Amazon EBS ヘルスチェックの有効化

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling グループ] ページの下部にスプリットペインが開きます。
4. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
5. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Amazon EBS ヘルスチェックを有効にする] を選択します。
6. [ヘルスチェックの猶予期間] に、秒単位で時間を入力します。これは、インスタンスが `InService` 状態になってからインスタンスのヘルスステータスを確認するまでに、Amazon EC2 Auto Scaling が待機する必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

7. [Update] (更新) を選択します。

AWS CLI

既存のグループに対する Amazon EBS ヘルスチェックの有効化

update-auto-scaling-group コマンドに `--health-check-type` オプションを追加します。次の例では、*my-asg* という名前の既存の Auto Scaling グループの `--health-check-type` オプションに **EBS** を指定しています。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60
```

複数のヘルスチェックタイプを使用するには、`--health-check-type` オプションに複数の値 (**EBS,ELB** など) を指定できます。

値名では大文字と小文字が区別されます。

Auto Scaling グループに対して Amazon EBS ヘルスチェックを無効にする

次のトピックでは、Auto Scaling グループに対して Amazon EBS ヘルスチェックを無効にする方法について説明します。Amazon EBS ヘルスチェックが不要になった場合は、次の手順を使用して無効にします。

Console

グループに対する Amazon EBS ヘルスチェックの無効化

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling グループ] ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
4. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Amazon EBS ヘルスチェックを有効にする] の選択を解除します。
5. [Update] (更新) を選択します。

AWS CLI

グループに対する Amazon EBS ヘルスチェックの無効化

Auto Scaling グループのヘルスチェックを更新して Amazon EBS ヘルスチェックを使用しないようにするには、update-auto-scaling-group コマンドを使用します。`--health-check-type` オプションと **`EC2`** の値を含めます。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

他のヘルスチェックタイプ (Elastic Load Balancing など) を無効にせずに Amazon EBS ヘルスチェックを無効にするには、**`EC2`** の代わりにそれらを指定する必要があります。例えば、Elastic Load Balancing ヘルスチェックの場合は、`--health-check-type` オプションに **`ELB`** を指定します。

値名では大文字と小文字が区別されます。

Auto Scaling グループに対してカスタムヘルスチェックを設定する

カスタムヘルスチェックを使用して、Amazon EC2 Auto Scaling が提供する既存のヘルスチェックオプションを補完できます。カスタムヘルスチェックを他のヘルスチェックタイプと組み合わせることで、アプリケーションのニーズに合わせた包括的なヘルスモニタリングシステムを作成できます。

開始するには、カスタムテストを作成して、Auto Scaling グループのインスタンスが正しく動作し、受信トラフィックを処理できることを確認します。設定したヘルスチェックでインスタンスが応答していないことが検出された場合は、その特定のインスタンスを Unhealthy としてマークします。これにより、Amazon EC2 Auto Scaling はそのインスタンスをすぐに置き換えます。

インスタンスのヘルスステータスは、AWS CLI または SDK を使用して Amazon EC2 Auto Scaling に直接送信できます。次の例は、AWS CLI を使用してインスタンスのヘルスステータスを設定し、インスタンスのヘルスステータスを確認する方法を示しています。

次の set-instance-health コマンドを使用して、指定したインスタンスのヘルスステータスを **`Unhealthy`** に設定します。

```
aws autoscaling set-instance-health --instance-id i-1234567890abcdef0 --health-
status Unhealthy
```

デフォルトで、このコマンドはヘルスチェックの猶予期間を守りますが、--no-should-respect-grace-period オプションを含めることでこの動作を上書きし、猶予期間を守らないようにすることも可能です。

次の describe-auto-scaling-groups コマンドを使用して、インスタンスのヘルスステータスが Unhealthy であることを確認します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

次に示すのは、インスタンスのヘルスステータスが Unhealthy であり、インスタンスが終了中であることを示す応答の例です。

```
{
    "AutoScalingGroups": [
        {
            ....
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcdef0"
                    },
                    "InstanceId": "i-1234567890abcdef0",
                    "InstanceType": "t2.micro",
                    "HealthStatus": "Unhealthy",
                    "LifecycleState": "Terminating"
                },
                ...
            ]
        }
    ]
}
```

ヘルスチェック不合格の理由を表示する

次の手順を使用して、ヘルスチェックによって置き換えられたインスタンスに関する情報を表示できます。

デフォルトでは、Amazon EC2 Auto Scaling は、異常があるインスタンスを終了するための新しいスケーリングアクティビティを作成してから、異常があるインスタンスを終了します。インスタンスの終了中、別のスケーリングアクティビティが新しいインスタンスを起動します。この動作を変更して、インスタンスメンテナンスポリシーを使用して、できるだけ早く新しいインスタンスの起動を開始できます。詳細については、「インスタンスメンテナンスポリシー」を参照してください。

Console

ヘルスチェック不合格の理由の表示

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [Activity (アクティビティ)] タブの [Activity history (アクティビティ履歴)] の下の [Status (ステータス)] 列に、Auto Scaling グループがインスタンスを正常に起動したか、終了したかが表示されます。

   正常でないインスタンスを終了した場合、原因列には、終了の日時、およびヘルスチェックが失敗した理由が表示されます。例えば、`At 2022-05-14T20:11:53Z an instance was taken out of service in response to a user health-check` と指定します。このメッセージは、カスタムヘルスチェックによってインスタンスが異常とマークされたことを示します。

   ヘルスチェック不合格に関するヘルプについては、「Amazon EC2 Auto Scaling の異常なインスタンスをトラブルシューティングする」を参照してください。

AWS CLI

ヘルスチェック不合格の理由の表示

以下の describe-scaling-activities コマンドを実行します。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下に示すのは応答の例です。Cause にはヘルスチェック不合格の理由が記載されています。

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
 response to a user health-check.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2a\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

出力の各フィールドについては、「Amazon EC2 Auto Scaling API Reference」(Amazon EC2 Auto Scaling API リファレンス) の「Activity」(アクティビティ) を参照してください。

Auto Scaling グループが削除された後にスケーリングアクティビティの説明を表示するには、describe-scaling-activities コマンドに `--include-deleted-groups` オプションを追加します。

Amazon EC2 Auto Scaling の異常なインスタンスをトラブルシューティングする

以下に示すのは、Amazon EC2 Auto Scaling から返されるエラーメッセージ、考えられる原因、問題の解決のための手順です。

エラーメッセージを取得するには、「ヘルスチェック不合格の理由を表示する」を参照してください。

エラーメッセージ

EC2 インスタンスのステータスチェックが失敗したことにより、インスタンスのサービスは停止されました。

問題: Auto Scaling インスタンスで Amazon EC2 のステータスチェックが失敗します。

原因 1: 何らかの問題により、Auto Scaling グループ内のインスタンスに障害があると Amazon EC2 が認識した場合、Amazon EC2 Auto Scaling はヘルスチェックの一環として、インスタンスを自動的に置き換えます。

解決策 1: インスタンスのステータスチェックが不合格になった場合、通常は、アプリケーションで問題が発生しなくなるまでインスタンス設定を変更することで、問題に自力で対処する必要があります。この問題を解決するには、以下の手順を実行します。

1. Auto Scaling グループには含まれない Amazon EC2 インスタンスを手動で作成して、問題を調査します。障害のあるインスタンスの調査に関する一般的なヘルプについては、「Amazon EC2 ユーザーガイド」の「ステータスチェックに失敗したインスタンスをトラブルシューティングする」と「Amazon EC2 ユーザーガイド」の Windows インスタンスのトラブルシューティングに関するページ参照してください。
2. インスタンスが起動に成功し、正常に機能していることを確認したら、エラーのない新しいインスタンス設定を Auto Scaling グループにデプロイします。
3. AWS アカウントへの課金が継続されないよう、作成したインスタンスを削除します。

EC2 ヘルスチェックでインスタンスが終了または停止済みであることが検出され、インスタンスのサービスが停止されてしまう

問題: 停止、再起動、または終了した Auto Scaling インスタンスは置き換えられます。

原因 1: ユーザーが手動で、インスタンスを停止、再起動、または終了しています。

解決策 1: Auto Scaling グループ内のインスタンスを停止または再起動する必要がある場合は、まずインスタンスをスタンバイ状態にすることをお勧めします。詳細については、「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。

原因 2: Amazon EC2 Auto Scaling は、Amazon EC2 スポットサービスがインスタンスを中断した後、スポットインスタンスの置き換えを試みます。これは、スポット料金がお客様の上限価格を超えるか、またはキャパシティーが使用できなくなるためです。

解決策 2: 特定の時点で要求を満たせるだけのスポットインスタンスが存在するという保証はありません。ただし、以下を試すことができます。

- より高いスポット上限価格を設定します (オンデマンド料金が利用できます)。上限価格を高く設定することで、要求されたキャパシティーを Amazon EC2 スポットサービスが確保し、それを維持できる可能性を高められます。
- 複数のアベイラビリティーゾーンで複数のインスタンスタイプを実行することで、より多くの異なるキャパシティプールからインスタンスを起動できるようにします。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。
- 複数のインスタンスタイプを使用する場合は、キャパシティーの再調整機能を有効にすることも考慮してください。この機能は、実行中のインスタンスが終了する前に Amazon EC2 スポットサービスで、新しいスポットインスタンスを起動させたい場合に便利です。詳細については、「キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する」を参照してください。

原因 3: キャパシティブロックでは、Amazon EC2 は、キャパシティブロックの終了時刻の 30 分前に、実行中のインスタンスを終了します。この突然の終了により、Auto Scaling グループは、キャパシティブロックが終了中であるにもかかわらず、新しいインスタンスを起動して希望するキャパシティを維持しようとします。

解決策 3: この問題を解決するには、以下の手順をお試しください。

- Auto Scaling グループの希望するキャパシティを減らして、新しいインスタンスを起動しようとしないようにします。詳細については、「Amazon EC2 Auto Scaling の手動スケーリング」を参照してください。
- このエラーが頻繁に発生しないように、キャパシティブロックの終了時刻の 30 分前に Auto Scaling グループをスケールインしてください。キャパシティブロックの終了時刻の 30 分前にライフサイクルフックが完了していることを確認します。詳細については、「使用アイテム Capacity Blocks 機械学習ワークロード用の」を参照してください。

ELB システムのヘルスチェックが障害を検出してインスタンスのサービスが停止される

問題: Auto Scaling インスタンスが EC2 ステータスチェックに合格する場合があります。ただし、Auto Scaling グループが登録されている、ターゲットグループや Classic Load Balancer に対するElastic Load Balancing のヘルスチェックで失敗する可能性があります。

原因 1: Auto Scaling グループが Elastic Load Balancing によって提供されるヘルスチェックに依存している場合、Amazon EC2 Auto Scaling は、EC2 のステータスチェックと Elastic Load Balancing のヘルスチェック両方の結果を確認することで、インスタンスのヘルスステータスを判断します。ロードバランサーは、各インスタンスにリクエストを送信して正しい応答を待つか、インスタンスとの接続の確立を試みることで、ヘルスチェックを実行します。インスタンスで実行するアプリケーションに問題があり、ロードバランサーがそのインスタンスを停止中と判断する場合、そのインスタンスは Elastic Load Balancing のヘルスチェックに失敗する場合があります。

解決策 1Elastic Load Balancing のヘルスチェックに合格するには:

- ターゲットグループのヘルスチェックにおいて、適切な設定が行われていることを確認します。ロードバランサーのヘルスチェック設定は、ターゲットグループごとに定義します。詳細については、「ターゲットのヘルスチェックを設定する」を参照してください。
- ロードバランサーで予期される成功コードを記録し、成功時にアプリケーションがそれらのコードを返すかを見て、適切な設定が行われていることを確認します。
- ロードバランサーと Auto Scaling グループのセキュリティグループでの設定が適切であることを確認します。
- ロードバランサーが Auto Scaling グループと同じアベイラビリティーゾーンに構成されていることを確認します。

解決策 2: Auto Scaling グループを更新して、Elastic Load Balancing のヘルスチェックを無効にします。これらのヘルスチェックを無効にする方法については、「Auto Scaling グループに Elastic Load Balancing ロードバランサーをアタッチする」を参照してください。

原因 2: ヘルスチェックの猶予期間とインスタンスのスタートアップ時間の間に不一致があります。

解決策 3: Auto Scaling グループのヘルスチェック猶予期間を編集します。猶予期間は、新たに起動したインスタンスが正常であると Elastic Load Balancing が認識するために必要な、連続した正常なヘルスチェックの回数をサポートするのに十分な時間に設定します。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

追加リソース

ここにはない問題が発生した場合は、以下の AWS re:Post の記事で、その他のトラブルシューティングについてのヘルプを参照してください。

- インスタンスが Amazon EC2 Auto Scaling により終了された理由は何ですか?
- 異常なインスタンスが Amazon EC2 Auto Scaling により終了されない理由は何ですか?

Amazon EC2 Auto Scaling の AWS Health Dashboard 通知

AWS Health Dashboard は、Amazon EC2 Auto Scaling からの通知をサポートします。これらの通知は、アプリケーションに影響を与える可能性のあるリソースのパフォーマンスや可用性の問題の把握と修復のガイダンスを提供します。現在使用できるのは、存在しないセキュリティグループおよび起動テンプレートに固有のイベントのみです。

AWS Health Dashboard は AWS Health サービスの一部です。セットアップは一切必要なく、アカウントで認証されるすべてのユーザーが表示できます。詳細については、「AWS Health ダッシュボードの開始方法」を参照してください。

次のようなメッセージを受信した場合は、アクションを実行するためのアラームとして処理する必要があります。

例: 欠落したセキュリティグループが原因で Auto Scaling グループがスケールアウトされていない

```
 Hello,

 At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
 AWS ##### 123456789012.

 A security group associated with this Auto Scaling group cannot be found. Each time
a
 scale out operation is performed, it will be prevented until you make a change that
 fixes the issue.

 We recommend that you review and update your Auto Scaling group configuration to
change
 the launch template or launch configuration that depends on the unavailable security
 group.

 Sincerely,
```

```
Amazon Web Services
```

例: 欠落した起動テンプレートが原因で Auto Scaling グループがスケールアウトされていない

```
Hello,

At 2021-05-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
AWS ##### 123456789012.

The launch template associated with this Auto Scaling group cannot be found. Each
time
a scale out operation is performed, it will be prevented until you make a change
that
fixes the issue.

We recommend that you review and update your Auto Scaling group configuration and
specify an existing launch template to use.

Sincerely,
Amazon Web Services
```

Auto Scaling グループとインスタンスの CloudWatch メトリクスを監視する

メトリクスは Amazon CloudWatch での基本的な概念です。メトリクスは、&CW; に発行された時系列のデータポイントのセットを表します。メトリクスはモニターリング対象の変数と考え、データポイントは時間の経過と共に変数の値を表します。これらのメトリクスを使用して、システムが正常に実行されていることを確認できます。

Auto Scaling グループに関する情報を収集する Amazon EC2 Auto Scaling メトリクスは、`AWS/AutoScaling` 名前空間にあります。Auto Scaling インスタンスから CPU やその他の使用状況データを収集する Amazon EC2 インスタンスメトリクスは、`AWS/EC2` 名前空間にあります。

Amazon EC2 Auto Scaling コンソールには、グループメトリクスとグループの集計インスタンスメトリクスの一連のグラフが表示されます。必要に応じて、Amazon EC2 Auto Scaling コンソールではなく Amazon CloudWatch から Auto Scaling グループのデータとインスタンスにアクセスできます。

詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

内容

Amazon EC2 Auto Scaling コンソールでモニタリンググラフを表示する

Amazon EC2 コンソールの Amazon EC2 Auto Scaling セクションでは、CloudWatch メトリクスを使用して、個々の Auto Scaling グループの進行状況を分単位でモニタリングできます。

次のタイプのメトリクスをモニタリングできます。

- Auto Scaling メトリクス – Auto Scaling メトリクスは、これを有効にした場合にのみオンになります。詳細については、「Auto Scaling グループのメトリクスを有効にする (コンソール)」を参照してください。Auto Scaling メトリクスが有効になっている場合、モニタリンググラフには Auto Scaling メトリクスについて 1 分単位で発行されたデータが表示されます。
- EC2 メトリクス – Amazon EC2 インスタンスメトリクスは常に有効になっています。詳細モニタリングが有効になっている場合、モニタリンググラフには、インスタンスメトリクスについて 1 分単位で発行されたデータが表示されます。詳細については、「Auto Scaling インスタンスのモニタリングを設定する」を参照してください。

Amazon EC2 Auto Scaling コンソールを使用してモニタリンググラフを表示するには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. メトリクスを表示する Auto Scaling グループの横にあるチェックボックスをオンにします。

   [Auto Scaling グループ] ページの下部に分割ペインが開き、グループに関する情報が表示されます。
3. モニタリングタブを選択します。

   Amazon EC2 Auto Scaling は、[Auto Scaling] メトリクスのモニタリンググラフを表示します。
4. グループの集計インスタンスメトリクスのモニタリンググラフを表示するには、[EC2] を選択します。

グラフアクション

- データポイントにカーソルを合わせると、特定の UTC 時刻のデータがポップアップ表示されます。
- グラフを拡大するには、グラフの右上にあるメニューツール (縦の 3 ドット) から [Enlarge] (拡大する) 選択します。または、グラフの上部にある最大化アイコンをクリックします。
- 定義済みの期間の値を 1 つ選択して、グラフに表示されるデータの期間を調整します。グラフが拡大されている場合は、[Custom] (カスタム) を選択して独自の期間を定義できます。
- メニューツールから [Refresh] (更新) を選択して、グラフのデータを更新します。
- グラフデータの上でカーソルをドラッグし、特定の範囲を選択します。次に、メニューツールで [Apply time range] (時間範囲の適用) を選択します。
- メニューツールから [View logs] (ログを表示) を選択して、CloudWatch コンソールで関連付けられたログストリーム (存在する場合) を表示します。
- CloudWatch でグラフを表示するには、メニューツールから [View in metrics] (メトリクスを表示) を選択します。これにより、そのグラフの CloudWatch ページが表示されます。ここでは、Auto Scaling グループが長期にわたってどのように変化したかをより深く理解するために、詳細情報を表示したり、履歴情報にアクセスしたりできます。

Auto Scaling グループのメトリクスをグラフ表示する

Auto Scaling グループを作成したら、Amazon EC2 Auto Scaling コンソールを開き、[Monitoring] (モニタリング) タブにグループのモニタリンググラフを表示できます。

[Auto Scaling] セクションのグラフメトリクスには、次のメトリクスが含まれます。これらのメトリクスは、終了インスタンス数や保留中のインスタンスの数など、潜在的な問題の指標となる測定値を提供します。これらのメトリクスの定義については、「Amazon EC2 Auto Scaling の Amazon CloudWatch メトリクス」を参照してください。

Display name (表示名)	CloudWatch メトリクス名
最小グループサイズ	`GroupMinSize`
最大グループサイズ	`GroupMaxSize`
希望するキャパシティ	`GroupDesiredCapacity`
稼働中のインスタンス	`GroupInServiceInstances`
保留中のインスタンス	`GroupPendingInstances`

Display name (表示名)	CloudWatch メトリクス名
スタンバイインスタンス	GroupStandbyInstances
終了処理中のインスタンス	GroupTerminatingInstances
インスタンスの合計	GroupTotalInstances

[EC2] セクションで、Amazon EC2 インスタンスの主要なパフォーマンスメトリクスに基づく、次のグラフメトリクスを確認できます。これらの EC2 メトリクスは、グループ内のすべてのインスタンスのメトリクスの集計です。これらのメトリクスの定義は、「Amazon EC2 ユーザーガイド」の「インスタンスの利用可能な CloudWatch メトリクスのリスト表示」を参照してください。

Display name (表示名)	CloudWatch メトリクス名
CPU 使用率	CPUUtilization
ディスク読み取り数	DiskReadBytes
ディスク読み取りオペレーション数	DiskReadOps
ディスク書き込み数	DiskWriteBytes
ディスク書き込みオペレーション数	DiskWriteOps
ネットワーク入力	NetworkIn
ネットワーク出力	NetworkOut
ステータスチェックに失敗 (すべて)	StatusCheckFailed
ステータスチェックに失敗 (インスタンス)	StatusCheckFailed_Instance
ステータスチェックに失敗 (システム)	StatusCheckFailed_System

さらに、Auto Scaling グラフメトリクスには、特定のユースケースに使用できるメトリクスがいくつかあります。

以下のメトリクスは、各インスタンスがグループの希望するキャパシティに寄与するユニット数を定義する重みをグループで使用する場合に役立ちます。これらのメトリクスの定義については、「Amazon EC2 Auto Scaling の Amazon CloudWatch メトリクス」を参照してください。

Display name (表示名)	CloudWatch メトリクス名
稼働中キャパシティのユニット数	`GroupInServiceCapacity`
保留中キャパシティのユニット数	`GroupPendingCapacity`
スタンバイ中キャパシティのユニット数	`GroupStandbyCapacity`
終了処理中キャパシティのユニット数	`GroupTerminatingCapacity`
キャパシティのユニット数合計	`GroupTotalCapacity`

次のメトリクスは、グループでウォームプール機能を使用する場合に役立ちます。これらのメトリクスの定義については、「Amazon EC2 Auto Scaling の Amazon CloudWatch メトリクス」を参照してください。

Display name (表示名)	CloudWatch メトリクス名
ウォームプールの最小サイズ	`WarmPoolMinSize`
ウォームプールの希望するキャパシティ	`WarmPoolDesiredCapacity`
ウォームプールの保留中キャパシティのユニット数	`WarmPoolPendingCapacity`

Display name (表示名)	CloudWatch メトリクス名
ウォームプールの終了処理中キャパシティのユニット数	`WarmPoolTerminatingCapacity`
ウォームプールのウォームアップされたキャパシティのユニット数	`WarmPoolWarmedCapacity`
ウォームプールの起動したキャパシティのユニット数合計	`WarmPoolTotalCapacity`
グループおよびウォームプールの希望するキャパシティ	`GroupAndWarmPoolDesiredCapacity`
グループおよびウォームプールの起動したキャパシティのユニット数合計	`GroupAndWarmPoolTotalCapacity`

関連リソース

- インスタンスごとのメトリクスをモニタリングするには、Amazon EC2 ユーザーガイドの「インスタンスのグラフメトリクス」を参照してください。
- CloudWatch ダッシュボードは、CloudWatch コンソールのカスタマイズ可能なホームページです。これらのページを使用して、異なるリージョンにまたがるリソースも含めて、単一のビューでリソースをモニタリングできます。CloudWatch ダッシュボードを使用して、AWS リソースのメトリクスおよびアラームをカスタマイズした状態で表示することができます。詳細については、『Amazon CloudWatch ユーザーガイド』を参照してください。

Amazon EC2 Auto Scaling の Amazon CloudWatch メトリクス

Amazon EC2 Auto Scaling は `AWS/AutoScaling` 名前空間に以下のメトリクスを公開します。実際に使用できる Auto Scaling グループメトリクスは、グループメトリクスを有効にしているかどうか、およびどのグループメトリクスを有効にしたかによって異なります。グループメトリクスは、追加料金なしで 1 分単位で利用できますが、有効にする必要があります。

Auto Scalingグループメトリクスを有効にすると、Amazon EC2 Auto Scaling は、ベストエフォートベースで毎分、CloudWatch にサンプルデータを送信します。CloudWatch でサービスの中断が発生するまれなケースでは、グループメトリクス履歴のギャップを埋めるためのデータのバックフィルは行われません。

内容

Auto Scaling グループメトリクス

これらのメトリクスを使用して、グループサイズの経時変化など、Auto Scaling グループの履歴をほぼ継続的に把握することができます。

メトリクス	説明
`GroupMinSize`	Auto Scaling グループの最小サイズ。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
`GroupMaxSize`	Auto Scaling グループの最大サイズ。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
`GroupDesiredCapacity`	Auto Scaling グループが保持しようとするインスタンスの数。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
`GroupInServiceInstances`	Auto Scaling グループの一部として実行するインスタンスの数。このメトリクスには保留中もしくは終了処理中のインスタンスは含まれません。

メトリクス	説明
	レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupPendingInstances	保留中のインスタンスの数。保留中のインスタンスは、稼働状態ではありません。このメトリクスには稼働中もしくは終了処理中のインスタンスは含まれません。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupStandbyInstances	Standby 状態にあるインスタンスの数。この状態のインスタンスはまだ実行中ですが、実際には使用されていません。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupTerminatingInstances	終了処理中のインスタンスの数。このメトリクスには稼働中もしくは保留中のインスタンスは含まれません。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupTotalInstances	Auto Scaling グループに含まれるインスタンスの合計数。このメトリクスは稼働中、保留中、および終了処理中のインスタンスの数を特定します。 レポート基準: メトリクス収集が有効になっている場合に報告されます。

各インスタンスタイプの vCPU 数に基づいて重みを割り当てるなど、さまざまなユニットで希望するキャパシティを測定するように混合インスタンスグループを設定すると、次のメトリクスは Auto Scaling グループによって使用されるユニット数をカウントします。希望するキャパシティをさまざまなユニットで測定するように混合インスタンスグループを設定しなかった場合、次のメトリクスが事前入力されますが、これらは前の表で定義されているメトリクスと同じです。詳細については、「混合インスタンスグループを作成するための設定の概要」を参照してください。

メトリクス	説明
GroupInServiceCapacity	Auto Scaling グループの一部として実行されているキャパシティユニットの数。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupPendingCapacity	保留中のキャパシティユニットの数。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupStandbyCapacity	Standby 状態にあるキャパシティユニットの数。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupTerminatingCapacity	終了処理中のキャパシティユニットの数。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupTotalCapacity	Auto Scaling グループ内のキャパシティユニットの合計数。 レポート基準: メトリクス収集が有効になっている場合に報告されます。

Amazon EC2 Auto Scaling は、ウォームプールを持つ Auto Scaling グループの以下のメトリクスもレポートします。詳細については、「ウォームプールを使用してブート時間が長いアプリケーションのレイテンシーを低減する」を参照してください。

メトリクス	説明
WarmPoolMinSize	ウォームプールの最小サイズ。 レポート基準: メトリクス収集が有効になっている場合に報告されます。

メトリクス	説明
WarmPoolDesiredCapacity	Amazon EC2 Auto Scaling がウォームプールで維持しようとするキャパシティの量。 これは、Auto Scaling グループの最大サイズから希望するキャパシティを引いた値に相当します。また、設定されている場合は、Auto Scaling グループの準備されている最大キャパシティから希望するキャパシティを引いた値に相当します。 ただし、ウォームプールの最小サイズが、最大サイズ (または設定されている場合は、準備された最大キャパシティ) と Auto Scaling グループの希望するキャパシティの差以上である場合、希望するウォームプールのキャパシティは WarmPoolMinSize に等しくなります。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
WarmPoolPendingCapacity	保留中のウォームプール内のキャパシティ量。このメトリクスには実行中、停止中、または終了処理中のインスタンスは含まれません。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
WarmPoolTerminatingCapacity	終了処理中のウォームプールのキャパシティの量。このメトリクスには実行中、停止中、または保留中のインスタンスは含まれません。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
WarmPoolWarmedCapacity	スケールアウト中に Auto Scaling グループに入れることができるキャパシティの量。このメトリクスには保留中もしくは終了処理中のインスタンスは含まれません。 レポート基準: メトリクス収集が有効になっている場合に報告されます。

メトリクス	説明
WarmPoolTotalCapacity	実行中、停止中、保留中、または終了処理中のインスタンスを含む、ウォームプールの合計キャパシティ。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupAndWarmPoolDesiredCapacity	Auto Scaling グループとウォームプールを合わせた希望するキャパシティ。 レポート基準: メトリクス収集が有効になっている場合に報告されます。
GroupAndWarmPoolTotalCapacity	Auto Scaling グループとウォームプールを合わせた合計キャパシティ。これには、実行中、停止中、保留中、終了処理中、または稼働中のインスタンスが含まれます。 レポート基準: メトリクス収集が有効になっている場合に報告されます。

Auto Scaling グループメトリクスのディメンション

以下のディメンションを使用して、前の表に示したメトリクスを絞り込むことができます。

ディメンション	説明
AutoScalingGroupName	Auto Scaling グループの名前をフィルターします。

予測スケーリングのメトリクスとディメンション

AWS/AutoScaling 名前空間には、予測スケーリングに関する以下のメトリクスが含まれます。

メトリクスは、1 時間の解像度で使用できます。

予測値を実際の値と比較することで、予測精度を評価できます。これらのメトリクスを使用した予測精度の評価についての詳細は、「CloudWatch による予測スケーリングメトリクスのモニタリング」(CloudWatch で予測スケーリングメトリクスをモニタリングする) を参照してください。

メトリクス	説明	ディメンション
PredictiveScalingLoadForecast	アプリケーションによって生成されることが予想される負荷の量。 Average、Minimum、Maximum の統計は有用ですが、Sum の統計は有用ではありません。 レポート基準: 初期予測の作成後にレポートされます。	AutoScalingGroupName, PolicyName, PairIndex
PredictiveScalingCapacityForecast	アプリケーションの需要を満たすために必要であると予想されるキャパシティ。これは、Auto Scaling インスタンスを維持するための負荷予測と目標使用率レベルに基づいています。 Average、Minimum、Maximum の統計は有用ですが、Sum の統計は有用ではありません。 レポート基準: 初期予測の作成後にレポートされます。	AutoScalingGroupName, PolicyName
PredictiveScalingMetricPairCorrelation	スケーリング指標と負荷指標のインスタンスごとの平均の間における相関関係。予測スケーリングは高い相関関係を前提とします。したがって、この指標の値が小さい場合、指標ペアを使用しない方がよいでしょう。 Average、Minimum、Maximum の統計は有用ですが、Sum の統計は有用ではありません。 レポート基準: 初期予測の作成後にレポートされます。	AutoScalingGroupName, PolicyName, PairIndex

> **Note**
>
> `PairIndex` ディメンションは、Amazon EC2 Auto Scaling によって割り当てられた際に、負荷スケーリングメトリクスペアのインデックスに関連付けられた情報を返します。現在、有効な値は `0` のみです。

Auto Scaling グループのメトリクスを有効にする (コンソール)

グループメトリクスを有効にするには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [Monitoring] (モニタリング) タブで、ページ上部の [Auto Scaling] の下にある [Auto Scaling group metrics collection] (Auto Scaling グループのメトリクスのコレクション) を選択し、[Enable] (有効化) チェックボックスをオンにします。

グループメトリクスを無効にするには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling スケーリンググループを選択します。
3. [モニタリング] タブで、[Auto Scaling group metrics collection (Auto Scaling グループメトリクスの収集)] の [Enable (有効)] チェックボックスをオフにします。

Auto Scaling グループのメトリクスを有効にする (AWS CLI)

Auto Scaling グループメトリクスを有効にするには

enable-metrics-collection コマンドを使用して、1 つ以上のグループメトリクスを有効にします。例えば、次のコマンドは、指定した Auto Scaling グループの単一のメトリクスを有効にします。

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \
  --metrics GroupDesiredCapacity --granularity "1Minute"
```

--metrics メトリクスを省略した場合、すべてのメトリクスが有効になります。

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \
  --granularity "1Minute"
```

Auto Scaling グループメトリクスを無効にするには

disable-metrics-collection コマンドを使用して、すべてのグループメトリクスを無効にします。

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

Auto Scaling インスタンスのモニタリングを設定する

Amazon EC2 は、インスタンスから未加工データを収集して、Auto Scaling グループの CPU やその他の使用率データを記述する、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらのメトリクスをモニタリングする間隔を、1 分または 5 分のいずれかに設定できます。

インスタンスモニタリングは、インスタンスが起動されるたびに、基本モニタリング (5 分ごと) または詳細モニタリング (1 分ごと) で有効になります。詳細モニタリングでは追加料金が適用されます。詳細については、[Amazon EC2 ユーザーガイド」の「Amazon CloudWatch の料金」および「CloudWatch を使用したインスタンスのモニタリング」を参照してください。

Auto Scaling グループを作成する前に、アプリケーションに適したモニタリングタイプを許可する起動テンプレートまたは起動設定を作成する必要があります。グループにスケーリングポリシーを追加する場合は、負荷の変動に迅速に対応するために、詳細モニタリングを使用して EC2 インスタンスのメトリクスデータを 1 分単位で取得することを強くお勧めします。

内容

- 詳細モニタリングを有効にするには (コンソール)
- 詳細モニタリングを有効にする (AWS CLI)
- 基本モニタリングと詳細モニタリングを切り替える
- CloudWatch エージェントを使用したその他のメトリクスの収集

詳細モニタリングを有効にするには (コンソール)

デフォルトでは、AWS Management Console を使用して起動テンプレートまたは起動設定を作成した場合、基本モニタリングが有効になります。

起動テンプレートの詳細モニタリングを有効にするには

AWS Management Console を使用して起動テンプレートを作成する場合、[Advanced details] (詳細情報) セクションの [Detailed CloudWatch monitoring] (詳細 CloudWatch モニタリング) で、[Enable] (有効) を選択します。それ以外の場合は、基本モニタリングが有効です。詳細については、「詳細設定を使用して起動テンプレートを作成する」を参照してください。

起動設定で詳細モニタリングを有効にするには

AWS Management Console を使用して起動設定を作成するときは、[追加設定] セクションで、[Enable EC2 instance detailed monitoring within CloudWatch (CloudWatch 内で EC2 インスタンスの詳細モニタリングを有効にする)] を選択します。それ以外の場合は、基本モニタリングが有効です。詳細については、「「起動設定を作成する」」を参照してください。

詳細モニタリングを有効にする (AWS CLI)

デフォルトでは、AWS CLI を使用して起動テンプレートを作成すると、基本モニタリングが有効になります。詳細モニタリングは、AWS CLI を使用して起動設定を作成する場合に有効になります。

起動テンプレートの詳細モニタリングを有効にするには

起動テンプレートの場合は、create-launch-template コマンドを使用して、起動テンプレートを作成するための情報が含まれる JSON ファイルを渡します。モニタリング属性を `"Monitoring": {"Enabled":true}` に設定して詳細モニタリングを有効にするか、または `"Monitoring": {"Enabled":false}` に設定して基本モニタリングを有効にします。

起動設定で詳細モニタリングを有効にするには

起動設定の場合は、create-launch-configuration コマンドを `--instance-monitoring` オプションで使用します。このオプションを `true` に設定して詳細モニタリングを有効に、または `false` に設定して基本モニタリングを有効にします。

```
--instance-monitoring Enabled=true
```

基本モニタリングと詳細モニタリングを切り替える

新しい EC2 インスタンスで有効になるモニタリングのタイプを変更するには、起動テンプレートを更新するか、Auto Scaling グループを更新して新しい起動テンプレートまたは起動設定を使用します。既存のインスタンスは、以前に有効化されたモニタリングタイプを使用し続けます。すべてのインスタンスを更新するには、すべてのインスタンスを終了して Auto Scaling グループに置き換えま

す。インスタンスを個々に更新するには、monitor-instances および unmonitor-instances を使用します。

> **Note**
>
> インスタンスの更新機能と最大有効期間の機能を使用すると、Auto Scaling グループ内のすべてのインスタンスを置き換えて、新しい設定を使用する新しいインスタンスを起動することもできます。詳細については、「Auto Scaling グループ内のインスタンスを置き換えます。」を参照してください。

基本モニタリングと詳細モニタリングを切り替えたときは、以下を実行します。

Auto Scaling グループのステップスケーリングポリシーまたはシンプルスケーリングポリシーに関連付けられた CloudWatch アラームがある場合は、put-metric-alarm コマンドを使用して各アラームを更新します。各間隔をモニタリングタイプに合わせます (基本モニタリングでは 300 秒、詳細モニタリングでは 60 秒)。詳細モニタリングから基本モニタリングに変更しても、5 分間アラームを更新しないと、1 分ごとに統計を確認し続けます。5 回のうち最大 4 回は利用可能なデータが検出されない可能性があります。

CloudWatch エージェントを使用したその他のメトリクスの収集

使用可能なメモリや、使用されているメモリなどのオペレーティングシステムレベルのメトリクスを収集するには、CloudWatch エージェントをインストールする必要があります。追加料金が発生する場合があります。CloudWatch エージェントを使用して、Amazon EC2 インスタンスからシステムメトリクスとログファイルの両方を収集できます。詳細については、Amazon CloudWatch ユーザーガイドの「CloudWatch エージェントにより収集されるメトリクス」を参照してください。

を使用した Amazon EC2 Auto Scaling API呼び出しの口グ記録 AWS CloudTrail

Amazon EC2 Auto Scaling は、ユーザーAWS CloudTrail、ロール、または AWS のサービス. CloudTrail captures がイベントとして Auto Scaling をAPI呼び出す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。キャプチャされた呼び出しには、 AWS Management Console からの呼び出しと、Auto Scaling APIオペレーションへのコード呼び出しが含まれます。によって収集された情報を使用して CloudTrail、Auto Scaling に対するリクエスト、リクエスト元の IP アドレス、リクエスト日時などの詳細を確認できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか。
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail アカウント AWS アカウント を作成すると、 は でアクティブになり、 CloudTrail イベント履歴に自動的にアクセスできます。 CloudTrail イベント履歴は、過去 90 日間に記録された管理イベントの表示可能、検索可能、ダウンロード可能、およびイミュータブルなレコードを で提供します AWS リージョン。詳細については、「 ユーザーガイド」の CloudTrail 「イベント履歴の使用」を参照してください。 AWS CloudTrail イベント履歴の表示には料金はかかりません CloudTrail。

AWS アカウント 過去 90 日間のイベントの継続的な記録については、証跡を作成します。

CloudTrail 証跡

証跡により CloudTrail 、 はログファイルを Amazon S3 バケットに配信できます。を使用して作成された証跡はすべてマルチリージョン AWS Management Console です。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント AWS リージョン 内のすべての でアクティビティをキャプチャするため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成 CloudTrail することで、 から Amazon S3 バケットに継続的な管理イベントのコピーを 1 つ無料で配信できますが、Amazon S3 ストレージ料金が発生します。 CloudTrail 料金の詳細については、AWS CloudTrail 「 料金表」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

での Auto Scaling 管理イベント CloudTrail

管理イベントは、 のリソースで実行される管理オペレーションに関する情報を提供します AWS アカウント。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。デフォルトでは、 は管理イベント CloudTrail を記録します。

Amazon EC2 Auto Scaling は、すべての Auto Scaling コントロールプレーンオペレーションを管理イベントとして記録します。Auto Scaling がログに記録する Amazon EC2 Auto Scaling コントロールプレーンオペレーションのリストについては、「Amazon EC2 Auto Scaling APIリファレンス」を参照してください。 CloudTrail

Auto Scaling イベントの例

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APIオペレーション、オペレーションの日時、リクエストパラメータなどに関する情報が含まれます。 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、イベントは特定の順序では表示されません。

次の例は、 `CreateLaunchConfiguration`オペレーションを示す CloudTrail イベントを示しています。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-08-21T17:05:42Z"
            }
        }
    },
    "eventTime": "2018-08-21T17:07:49Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "CreateLaunchConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Coral/Jakarta",
    "requestParameters": {
        "ebsOptimized": false,
        "instanceMonitoring": {
            "enabled": false
        },
        "instanceType": "t2.micro",
```

```
        "keyName": "EC2-key-pair-oregon",
        "blockDeviceMappings": [
            {
                "deviceName": "/dev/xvda",
                "ebs": {
                    "deleteOnTermination": true,
                    "volumeSize": 8,
                    "snapshotId": "snap-01676e0a2c3c7de9e",
                    "volumeType": "gp2"
                }
            }
        ],
        "launchConfigurationName": "launch_configuration_1",
        "imageId": "ami-6cd6f714d79675a5",
        "securityGroups": [
            "sg-00c429965fd921483"
        ]
    },
    "responseElements": null,
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


CloudTrail レコードの内容の詳細については、「 AWS CloudTrail ユーザーガイド」のCloudTrail「レコードの内容」を参照してください。

での Auto Scaling RemoveAction 呼び出し CloudWatch

Auto Scaling がアラームから自動スケーリングアクションを削除する CloudWatch ように に指示API すると、Auto Scaling が を呼び CloudWatch RemoveAction出すことが AWS CloudTrail ログに表示される場合があります。これは、スケーラブルなターゲットの登録を解除したり、スケーリングポリシーを削除したり、アラームが存在しないスケーリングポリシーを呼び出す場合に発生する可能性があります。

Amazon EC2 Auto Scaling の Amazon SNS 通知オプション

アプリケーションに影響を与える重要なイベントを通知するように Auto Scaling グループを設定できます。通知を使用すると、ポーリングが不要になり、ポーリングによる `RequestLimitExceeded` エラーが発生することもなくなります。

Amazon EC2 Auto Scaling に関する通知を受け取るには、次の 2 つの方法があります。

- Amazon Simple Notification Service – Auto Scaling グループがインスタンスを起動または終了したときに Amazon SNS で通知を送信できます。Amazon SNS 通知は、オン/オフの切り替えのみが可能です。詳細については、「Amazon SNS と Amazon EC2 Auto Scaling」を参照してください。
- Amazon EventBridge – EventBridge は、指定された条件に一致し、Amazon SNS を含むさまざまなターゲットに送信される、より高度なイベント駆動型の通知を提供します。EventBridge で、より広範な Auto Scaling イベントをモニタリングし、より正確なモニタリングを実現することも可能です。詳細については、「Auto Scaling イベントの処理に EventBridge を使用する」を参照してください。

オプションで、ライフサイクルフックの通知を使用して、起動時または終了時にインスタンスに対してカスタムアクションを実行できます。ライフサイクルフックで使用する通知を設定する方法の詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

Amazon SNS と Amazon EC2 Auto Scaling

このセクションでは、Amazon SNS を使用して、Auto Scaling グループがインスタンスを起動または終了するタイミングをモニタリングする方法について説明します。

例えば、`autoscaling: EC2_INSTANCE_TERMINATE` 通知タイプを使用するように Auto Scaling グループを設定する場合、Auto Scaling グループがインスタンスを終了すると、E メール通知が送信されます。この E メールには、インスタンス ID やインスタンスを終了した理由など、終了したインスタンスの詳細が含まれます。

Amazon EC2 Auto Scaling がグループにインスタンスを追加または削除すると、これらの変更に関する通知がインスタンスごとに 1 つずつ送信されます。ただし、これらの通知の配信はベストエフォート型であるため、例えば、その後のヘルスチェックが失敗した場合など、最初の通知の後もインスタンスが引き続き失敗する可能性があります。ヘルスチェックプロセスの詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

Amazon SNS 全般の詳細については、「Amazon Simple Notification Service デベロッパーガイド」を参照してください。

内容

- Amazon SNS トピックを作成します。
- Amazon SNS トピックを購読します。
- Amazon SNS サブスクリプションを確認する
- 通知を送信するように Auto Scaling グループを設定する
- 通知をテストする
- 通知設定を削除する

- 暗号化された Amazon SNS トピックのキーポリシー

SNS 通知

Amazon EC2 Auto Scaling は、以下のイベントが発生したときの Amazon SNS 通知の送信をサポートしています。

イベント	説明
`autoscaling:EC2_INSTANCE_LAUNCH`	インスタンスの起動成功
`autoscaling:EC2_INSTANCE_LAUNCH_ERROR`	インスタンスの起動失敗
`autoscaling:EC2_INSTANCE_TERMINATE`	インスタンスの削除成功
`autoscaling:EC2_INSTANCE_TERMINATE_ERROR`	インスタンスの削除失敗

メッセージには、次に示す情報が含まれます。

- `Event` – イベント。
- `AccountId` - Amazon Web Services のアカウント ID
- `AutoScalingGroupName` - Auto Scaling グループの名前。
- `AutoScalingGroupARN` – Auto Scaling グループの ARN。
- `EC2InstanceId` - EC2 インスタンスの ID。

例えば、以下のようになります。

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:region:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup constraints
 to ...
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-id","Availability Zone":"zone"}
Origin: AutoScalingGroup
Destination: EC2
```

Amazon EC2 Auto Scaling の Amazon SNS 通知を設定する

Amazon SNS を使用して E メール通知を送信するには、最初にトピックを作成してから、そのトピックと共に E メールアドレスを登録する必要があります。

Amazon SNS トピックを作成します。

SNS トピックとは、論理アクセスポイント、つまり Auto Scaling グループが通知を送信するために使用する通信チャネルです。トピックの名前を指定することにより、トピックを作成します。

トピック名を作成する際には、名前が次の要件を満たしている必要があります。

- 1 ~ 256 文字
- 大文字および小文字の ASCII 文字、数字、アンダースコア、またはハイフンが含まれている

詳細については、Amazon Simple 通知サービス デベロッパーガイドの「Amazon SNS トピックの作成」を参照してください。

Amazon SNS トピックを購読します。

Auto Scaling グループがトピックに送信した通知を受信するには、そのトピックにエンドポイントを登録する必要があります。この手順では、エンドポイント に、Amazon EC2 Auto Scaling からの通知を受信する E メールアドレスを指定します。

詳細については、Amazon Simple 通知サービス デベロッパーガイドの「Amazon SNS トピックへのサブスクライブ」を参照してください。

Amazon SNS サブスクリプションを確認する

Amazon SNS は、前のステップで指定した E メールアドレスに確認メールを送信します。

次のステップに進む前に、AWS Notifications からの E メールを開き、リンクを選択してサブスクリプションを確認するようにしてください。

AWSからの確認メッセージが届きます。Amazon SNS は、通知を受信し、通知を E メールとして指定された E メールアドレスに送信するように設定されました。

通知を送信するように Auto Scaling グループを設定する

Auto Scaling グループは、インスタンスの起動または終了などのスケーリングイベントが発生したときに Amazon SNS に通知を送信するように設定することができます。Amazon SNS は、インスタンスに関する情報が含まれた通知を、ユーザーが指定する E メールアドレスに送信します。

Auto Scaling グループの Amazon SNS 通知を設定する (コンソール)

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの隣にあるチェックボックスを選択します。

   ページ下部に分割ウィンドウが開き、選択したグループの情報が表示されます。
3. [Activity] (アクティビティ) タブで、[Activity notifications] (アクティビティ通知)、[Create notification] (通知の作成) の順に選択します。
4. [Create notifications] ペインで、以下の作業を行います。

   a. [SNS Topic] (SNS トピック) で、SNS トピックを選択します。

   b. [Event types] (イベントタイプ) で、通知を送信するイベントを選択します。

   c. [Create] (作成) を選択します。

Auto Scaling グループの Amazon SNS 通知を設定する (AWS CLI)

以下の put-notification-configuration コマンドを使用します。

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-
asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH"
 "autoscaling:EC2_INSTANCE_TERMINATE"
```

通知をテストする

起動イベントの通知を生成するには、Auto Scaling グループの希望容量を 1 増やして、Auto Scaling グループを更新します。インスタンスを起動してから数分以内に通知を受け取ります。

希望する容量を変更するには (コンソール)

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling グループ] ページの下部に分割ペインが開き、選択したグループに関する情報が表示されます。
3. [詳細] タブで、[グループの詳細]、[編集] の順に選択します。
4. [Desired capacity (希望するキャパシティ)] の場合は、現在の値を 1 ずつ増やします。この値が [Maximum capacity (最大容量)] を超える場合は、[Maximum capacity (最大容量)] の値を 1 ずつ増やす必要があります。
5. [Update] (更新) を選択します。
6. 数分後、イベントの通知が届きます。このテスト用に起動した追加のインスタンスが必要なくなった場合は、[Desired capacity (希望する容量)] を 1 減らすことができます。数分後、イベントの通知が届きます。

通知設定を削除する

Amazon EC2 Auto Scaling 通知設定が使用されなくなった場合は、設定を削除できます。

Amazon EC2 Auto Scaling 通知設定を削除する (コンソール)

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。

2. Auto Scaling スケーリンググループを選択します。
3. [Activity] (アクティビティ) タブで、削除する通知の横にあるチェックボックスをオンにしてから、[Actions] (アクション)、[Delete] (削除) の順に選択します。

Amazon EC2 Auto Scaling 通知設定を削除する (AWS CLI)

次の delete-notification-configuration コマンドを使用します。

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --
topic-arn arn
```

Auto Scaling グループに関連付けられている Amazon SNS トピックとすべてのサブスクリプションの削除については、「Amazon Simple Notification Service デベロッパーガイド」の「Amazon SNS サブスクリプションおよびトピックを削除する」を参照してください。

暗号化された Amazon SNS トピックのキーポリシー

指定した Amazon SNS トピックは、AWS Key Management Service で作成されたカスタマーマネージドキーで暗号化される場合があります。暗号化されたトピックに発行するための許可を Amazon EC2 Auto Scaling に付与するには、まず KMS キーを作成してから、次のステートメントを KMS キーのポリシーに追加する必要があります。サンプルの ARN を、キーへのアクセスが許可されている適切なサービスにリンクされたロールの ARN に置き換えます。詳細については、「Amazon Simple Notification Service デベロッパーガイド」の「AWS KMS 許可を設定する」を参照してください。

この例では、ポリシーステートメントにより、[AWSServiceRoleForAutoScaling] という名前のサービスにリンクされたロールに、カスタマーマネージドキーを使用するための許可が付与されます。Amazon EC2 Auto Scaling のサービスにリンクされたロールの詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。

```
{
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling"
  },
  "Action": [
    "kms:GenerateDataKey*",
```

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

Amazon EC2 Auto Scaling が暗号化されたトピックにパブリッシュできるようにするキーポリシーでは、aws:SourceArn および aws:SourceAccount 条件キーはサポートされていません。

AWS Amazon EC2 Auto Scaling と統合された のサービス

Amazon EC2 Auto Scaling は、他の AWS サービスと統合できます。各サービスと Amazon EC2 Auto Scaling の連携の詳細については、以下の統合オプションを参照してください。

トピック

- キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する
- キャパシティ予約を使用して特定のアベイラビリティーゾーンでキャパシティを予約する
- を使用してコマンドラインから Auto Scaling グループを作成する AWS CloudShell
- AWS CloudFormation で Auto Scaling グループを作成する
- でインスタンスタイプのレコメンデーションを取得する AWS Compute Optimizer
- Elastic Load Balancing を使用して Auto Scaling グループ内の受信アプリケーショントラフィックを分散する
- VPC Lattice ターゲットグループを使用してトラフィックフローを管理する
- Auto Scaling イベントの処理に EventBridge を使用する
- Amazon VPC を使用して Auto Scaling インスタンスのネットワーク接続を提供する

キャパシティの再調整を使用して Amazon EC2 スポットの中断に対処する

スポットインスタンスの可用性に影響する変更をモニタリングし、自動的に応答するように Amazon EC2 Auto Scaling を設定できます。キャパシティの再調整は、実行中のインスタンスが Amazon EC2 により中断される前に、新しいスポットインスタンスでフリートを事前に拡張することにより、ワークロードの可用性を維持するのに役立ちます。

キャパシティの再調整の目標は、ワークロードの処理を中断されることなく継続することです。スポットインスタンスの中断リスクが高い場合、Amazon EC2 スポットサービスは、Amazon EC2 Auto Scaling に EC2 インスタンスの再調整レコメンデーションを通知します。

Auto Scaling グループに対してキャパシティの再調整を有効にすると、Amazon EC2 Auto Scaling は、再調整レコメンデーションを受け取ったグループ内のスポットインスタンスを積極的に置き換えようとします。これは、ワークロードを中断リスクが低い新しいスポットインスタンスに再調整する機会を提供します。ワークロードは、既存のインスタンスが中断される前に Amazon EC2 Auto Scaling が新しいスポットインスタンスを起動している間も、作業の処理を続行できます。

キャパシティの再調整が無効化されていると、Amazon EC2 Auto Scaling は、Amazon EC2 スポットサービスがインスタンスを中断し、それらのヘルスチェックが失敗するまでスポットインスタンスを置き換えません。Amazon EC2 は常に、インスタンスを中断する前に EC2 インスタンスの再調整レコメンデーションと、スポットインスタンスの中断 2 分前の通知の両方を提供します。

内容

概要

Auto Scaling グループでキャパシティの再調整を使用するための基本的な手順は、以下のとおりです。

1. 複数のインスタンスタイプとアベイラビリティーゾーンを使用するように Auto Scaling グループを設定します。そうすることで、Amazon EC2 Auto Scaling は各アベイラビリティーゾーンでスポットインスタンスの利用可能なキャパシティを検討できるようになります。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。
2. 必要に応じてライフサイクルフックを追加して、再調整通知を受け取るインスタンス内でアプリケーションを正常にシャットダウンします。詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

   以下は、ライフサイクルフックを使用するいくつかの理由です。

   - Amazon SQS ワーカーの正常なシャットダウンのために
   - ドメインネームシステム (DNS) からの登録解除を完了するには
   - システムログまたはアプリケーションログをプルし、Amazon Simple Storage Service (Amazon S3) にアップロードするには
3. ライフサイクルフックのカスタムアクションを開発します。カスタムアクションを呼び出すには、インスタンスの終了の準備が整ったタイミングを知る必要があります。インスタンスのライフサイクル状態を検出することでタイミングを確認します。
   - インスタンスの外部でアクションを呼び出すには、EventBridge ルールを作成し、イベントパターンがルールに一致したときに実行するアクションを自動化できます。

- インスタンス内でアクションを呼び出すには、シャットダウンスクリプトを実行し、インスタンスのメタデータを介してライフサイクルスの状態を取得するようにインスタンスを設定します。

カスタムアクションが 2 分以内に終了するように設計することが重要です。これにより、インスタンス終了前にタスクを完了するのに十分な時間が確保されます。

これらの手順を完了すると、キャパシティの再調整を使用できるようになります。

キャパシティーの再調整の動作

キャパシティの再調整では、インスタンスが再調整レコメンデーションを受け取ると、Amazon EC2 Auto Scaling は次のように動作します。

- 新しいスポットインスタンスの起動時、Amazon EC2 Auto Scaling は、新しいインスタンスがヘルスチェックに合格するまで待機してから、以前のインスタンスを終了させます。複数のインスタンスを置き換える場合、以前の各インスタンスの終了は、新しいインスタンスが起動され、ヘルスチェックに合格してから開始されます。
- Amazon EC2 Auto Scaling は以前のインスタンスを終了する前に新しいインスタンスの起動を試みるため、指定された最大キャパシティに到達している、またはそれに近い状態は、再調整アクティビティを妨げたり、それらを完全に停止させる可能性があります。この問題を回避するために、Amazon EC2 Auto Scaling は一時的にグループの最大サイズを必要なキャパシティの最大 10% まで超過できます。
- Auto Scaling グループにライフサイクルフックを追加しなかった場合、Amazon EC2 Auto Scaling は、新しいインスタンスがヘルスチェックに合格すると同時に以前のインスタンスの終了を開始します。
- ライフサイクルフックを追加した場合は、前のインスタンスの終了を開始するまでの時間が、ライフサイクルフックに指定したタイムアウト値だけ延長されます。
- スケーリングポリシーまたはスケジュールされたスケーリングを使用している場合、スケーリングアクティビティは並行して実行されます。スケーリングアクティビティが進行中で、Auto Scaling グループが新しい希望キャパシティを下回っている場合、Amazon EC2 Auto Scaling はまずスケールアウトを行ってから、以前のインスタンスを終了します。

あるアベイラビリティーゾーンに該当するインスタンスタイプのキャパシティがない場合、Amazon EC2 Auto Scaling は他の有効なアベイラビリティーゾーンで成功するまでスポットインスタンスの起動を試行し続けます。

最悪のシナリオでは、新しいインスタンスの起動に失敗するか、ヘルスチェックに失敗すると、Amazon EC2 Auto Scaling はインスタンスの再起動を試行し続けます。　新しいインスタンスの起動試行中、以前のインスタンスは最終的に中断され、2 分間の中断通知により強制的に終了されます。

考慮事項

キャパシティの再調整を使用する場合は、次の点を考慮してください。

スポットの中断に耐えられるようにアプリケーションを設計する

アプリケーションはインスタンス数の動的な変化と、スポット インスタンスが早期に中断される可能性に対処する必要があります。例えば、Auto Scaling グループが Elastic Load Balancing ロードバランサーの背後にある場合、Amazon EC2 Auto Scaling は、インスタンスがロードバランサーから登録解除されるまで待機してから、終了ライフサイクルフックを呼び出します。インスタンスの登録解除とライフサイクルアクションの完了にかかる時間が長すぎる場合、Amazon EC2 Auto Scaling がインスタンスを終了させる前にライフサイクルアクションの完了を待機する間に、インスタンスが中断される可能性があります。

2 分間のスポットインスタンス中断通知の前に、Amazon EC2 が再調整のレコメンデーションシグナルを送信できるとは限りません。場合によっては、再調整のレコメンデーションシグナルが、2 分間の中断通知と同時に到着する可能性があります。この場合、Amazon EC2 Auto Scaling はライフサイクルフックを呼び出し、新しいスポットインスタンスをすぐに起動しようとします。

代替スポットインスタンスが中断されるリスクの増大を回避する

`lowest-price` 割り当て戦略を使用している場合、代替スポットインスタンスが中断されるリスクが高くなることがあります。これは、代替スポットインスタンスが起動後すぐに中断される可能性が高い場合も、その時点で利用可能なキャパシティを持つ最低料金のプールでインスタンスを起動することが原因です。中断のリスクが高まるのを避けるため、`lowest-price` の割り当て戦略を使用しないことを強くお勧めします。代わりに、`price-capacity-optimized` の使用をお勧めします。この戦略では、中断される可能性が最も低く、可能な限り低価格のスポットプールで代替スポットインスタンスを起動します。したがって、近い将来に中断される可能性は低くなります。

Amazon EC2 Auto Scaling は、可用性が同じかそれ以上の場合にのみ、新しいインスタンスを起動します

キャパシティ再調整の目的の 1 つは、スポットインスタンスの可用性を改善することです。既存のスポットインスタンスが再調整のレコメンデーションを受け取った場合、Amazon EC2 Auto

Scaling は、新しいインスタンスが既存のインスタンスと同等かそれ以上の可用性を提供する場合にのみ新しいインスタンスを起動します。新しいインスタンスの中断のリスクが既存のインスタンスよりもひどい場合、Amazon EC2 Auto Scaling は新しいインスタンスを起動しません。ただし、Amazon EC2 Auto Scaling は引き続き Amazon EC2 スポットサービスから提供された情報に基づいてスポットキャパシティプールを評価し、可用性が向上した場合は新しいインスタンスを起動します。

Amazon EC2 Auto Scaling が新しいインスタンスをプロアクティブに起動しないと、既存のインスタンスが中断する可能性があります。中断が発生すると、Amazon EC2 Auto Scaling は、スポットインスタンスの中断通知を受け取ると直ちに新しいインスタンスの起動を試みます。これは、新しいインスタンスが中断されるリスクが高いかどうかに関係なく起こります。

キャパシティーの再調整は、スポットインスタンスの中断率を増加させるものではありません

キャパシティの再調整を有効にしても、スポットインスタンスの中断率 (Amazon EC2 がキャパシティを取り戻す必要があるときに再利用されるスポットインスタンスの数) は増加しません。ただし、インスタンスに中断のリスクがあることをキャパシティの再調整が検出した場合、Amazon EC2 Auto Scaling は直ちに新しいインスタンスの起動を試みます。したがって、リスクのあるインスタンスが中断された後に Amazon EC2 Auto Scaling が新しいインスタンスを起動するのを待つ場合よりも多くのインスタンスが置き換えられる可能性があります。

キャパシティの再調整が有効になっているインスタンスをさらに置き換える可能性がありますが、事後対応ではなくプロアクティブに対応できるというメリットがあります。これにより、インスタンスが中断される前にアクションを実行できる時間が増えます。スポットインスタンスの中断通知では、通常、インスタンスを正常にシャットダウンするための猶予期間が最大 2 分しかありません。キャパシティの再調整で新しいインスタンスが事前に起動されるため、リスクのあるインスタンスで既存のプロセスを完了できる可能性が高まります。また、インスタンスのシャットダウン手順を開始し、リスクのあるインスタンスで新しい作業がスケジュールされないようにしたり、新しく起動したインスタンスがアプリケーションを引き継ぐように準備することもできます。キャパシティの再調整のプロアクティブな置き換えにより、正常な継続性の恩恵を受けることができます。

次の理論的な例は、キャパシティの再調整を使用するリスクとメリットを示しています。

- 午後 2 時 – インスタンス A に対する再調整のレコメンデーションを受け取りました。Amazon EC2 Auto Scaling が直ちに代替インスタンス B の起動の試行を開始するため、ユーザーはシャットダウン手順を開始する時間を確保できます。
- 午後 2 時 30 分 – インスタンス B の再調整のレコメンデーションを受け取り、インスタンス C に置き換えられました。これにより、ユーザーはシャットダウン手順を開始する時間を確保できます。

- 午後 2 時 32 分 – キャパシティの再調整が有効になっておらず、インスタンス A のスポットインスタンスの中断通知が午後 2 時 32 分に受信されていたとすれば、アクションを実行するための猶予時間は最大でも 2 分だけでした。ただし、インスタンス A はこの時点まで実行され続けていたはずです。

AWS Management Console または AWS CLI を使用してキャパシティーの再調整を有効にする

Auto Scaling グループのキャパシティーの再調整を有効にするには、AWS Management Console または AWS CLI を使用します。Amazon EC2 Auto Scaling は、再調整のレコメンデーションを受け取ったグループ内のスポットインスタンスをプロアクティブに置き換えようとします。

キャパシティの再調整を有効にする (コンソール)

Auto Scaling グループを作成または更新するときに、キャパシティの再調整を有効または無効にできます。

新しい Auto Scaling グループのキャパシティの再調整を有効にするには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. [Auto Scaling グループの作成] を選択します。
3. ステップ 1 で、起動テンプレートまたは構成を選択し、Auto Scaling グループの名前を入力し、起動テンプレートを選択してから、[Next] (次へ) を選択して次のステップに進みます。
4. [ステップ 2: インスタンスの起動オプションを選択] の [インスタンスタイプの要件] で、混合インスタンスグループを作成するための設定を選択します。この混合インスタンスグループには、起動できるインスタンスタイプ、インスタンス購入オプション、スポットインスタンスとオンデマンドインスタンスの配分戦略が含まれます。これらの設定はデフォルトで設定されていません。これらを設定するには、[Override launch template] (起動テンプレートを上書きする) を選択する必要があります。混合インスタンスグループの作成に関する詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。
5. [ネットワーク] で、必要なオプションを選択します。使用するサブネットが、異なるアベイラビリティーゾーンにあることを確認します。
6. [配分戦略] セクションで、スポット配分戦略を選択します。[容量の再分散] で、チェックボックスをオンまたはオフにして、容量の再分散を有効または無効にします。このオプションは、[イ

ンスタンスの購入オプション] セクションで Auto Scaling グループのスポットインスタンスとして起動するようにリクエストした場合にのみ表示されます。

7. Auto Scaling グループを作成します。
8. (オプション) 必要に応じてライフサイクルフックを追加します。詳細については、「Auto Scaling グループにライフサイクル フックを追加する」を参照してください。

既存の Auto Scaling グループのキャパシティの再調整を有効または無効にするには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。ページの下部にスプリットペインが開きます。
3. [Details] (詳細) タブで、[Allocation strategies] (配分戦略)、[Edit] (編集) の順に選択します。
4. [配分戦略] セクションで、[容量の再分散] チェックボックスをオンまたはオフにして、容量の再分散を有効または無効にします。
5. [Update] (更新) を選択します。

キャパシティの再調整を有効にする (AWS CLI)

以下の例では、AWS CLIを使用してキャパシティの再調整を有効または無効にする方法を示しています。

[create-auto-scaling-group] または [update-auto-scaling-group] コマンドに次のパラメータを指定して使用します。

- `--capacity-rebalance` / `--no-capacity-rebalance` – キャパシティーの再調整が有効かどうかを示すブール値。

[create-auto-scaling-group] コマンドを呼び出す前に、Auto Scaling グループで使用するように設定されている起動テンプレートの名前が必要です。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。

> **Note**
>
> 次の手順は、JSON または YAML でフォーマットされた設定ファイルの使用方法を示しています。AWS CLIバージョン 1 を使用する場合は、JSON 形式の設定ファイルを指定する必要

があります。AWS CLIバージョン 2 を使用する場合は、YAML または JSON のいずれかでフォーマットされた設定ファイルを指定できます。

JSON

新しい Auto Scaling グループを作成して設定するには

- 次の create-auto-scaling-group コマンドを使用して、新しい Auto Scaling グループを作成し、キャパシティの再調整を有効にします。このコマンドは、Auto Scaling グループの唯一のパラメーターとして JSON ファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

[混合インスタンス・ポリシー] を指定する CLI 設定ファイルがまだない場合は、作成します。

次の行を設定ファイルのトップレベルの JSON オブジェクトに追加します。

```
{
    "CapacityRebalance": true
}
```

次は、config.json ファイルの例です。

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacity": 12,
    "MinSize": 12,
    "MaxSize": 15,
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 0,
            "OnDemandPercentageAboveBaseCapacity": 25,
            "SpotAllocationStrategy": "price-capacity-optimized"
        },
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
```

```
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        }
    },
    "TargetGroupARNs": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff",
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

新しい Auto Scaling グループを作成して設定するには

- 次の create-auto-scaling-group コマンドを使用して、新しい Auto Scaling グループを作成し、キャパシティの再調整を有効にします。このコマンドは、Auto Scaling グループの唯一のパラメーターとして YAML ファイルを参照します。

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

YAML でフォーマットされた設定ファイルに以下の行を追加します。

```
CapacityRebalance: true
```

次は、config.yaml ファイルの例です。

```
---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25
    SpotAllocationStrategy: price-capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
TargetGroupARNs:
- arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-
group/943f017f100becff
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

既存の Auto Scaling グループのキャパシティの再調整を有効にするには

- 以下の update-auto-scaling-group コマンドを使用して、キャパシティの再調整を有効にします。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --capacity-rebalance
```

Auto Scaling グループのキャパシティの再調整を有効になっていることを確認するには

- 以下の describe-auto-scaling-group コマンドを使用して、キャパシティの再調整が有効になっていることを確認し、詳細を表示します。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

以下に、応答の例を示します。

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "CapacityRebalance": true
        }
    ]
}
```

キャパシティの再調整を無効にするには

キャパシティの再調整を無効にするには、[update-auto-scaling-group] コマンドに`--no-capacity-rebalance`オプションを付けて使用します。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --no-capacity-rebalance
```

関連リソース

キャパシティの再調整の特徴の詳細とチュートリアルについては、AWSコンピューティング ブログの [EC2 Auto Scaling の新しいキャパシティの再調整の特徴を使用して、スポットインスタンスのライフサイクルをプロアクティブに管理] のブログ記事をご覧ください。

EC2 インスタンスの再調整のレコメンデーションの詳細については、「Amazon EC2 ユーザーガイド」の「EC2 インスタンスの再調整に関する推奨事項」を参照してください。

ライフスタイルフックの詳細については、以下のリソースを参照してください。

- チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定 (EventBridge の使用)
- チュートリアル: データスクリプトとインスタンスメタデータを使用してライフサイクル状態を取得する

制約事項

- Amazon EC2 Auto Scaling は、インスタンスがスケールインから保護されていない場合のみ、再調整通知を受信したインスタンスを置き換えることができます。ただし、スケールイン保護はスポットの中断による終了を防ぐことはできません。詳細については、「インスタンスのスケールイン保護を使用してインスタンスの終了を制御する」を参照してください。
- キャパシティの再調整のサポートは、中東 (アラブ首長国連邦) リージョンを除くAmazon EC2 Auto Scaling が使用可能なすべての商用 AWS リージョンで利用できます。

キャパシティ予約を使用して特定のアベイラビリティーゾーンでキャパシティを予約する

Amazon EC2 オンデマンドキャパシティ予約は、特定のアベイラビリティーゾーンでコンピューティングキャパシティを予約するのに役立ちます。キャパシティ予約の使用を開始するには、特定のアベイラビリティーゾーンにキャパシティ予約を作成します。次に、インスタンスをリザーブドキャパシティに起動し、そのキャパシティの使用率をリアルタイムで表示して、必要に応じてキャパシティを増減することができます。

キャパシティ予約は、open または `targeted` のいずれかで設定されます。キャパシティ予約が open の場合、一致する属性を持つすべての新規および既存のインスタンスは、キャパシティ予約のキャパシティ内で自動的に実行されます。キャパシティ予約 が `targeted` の場合、インスタンスはそれがリザーブドキャパシティで実行されるように具体的に設定する必要があります。

キャパシティ予約の設定

キャパシティ予約設定は、オンデマンドキャパシティを使用する前にキャパシティ予約でリザーブドキャパシティを優先することで、キャパシティ予約を効率的に使用できます。次のキャパシティ予約設定オプションから選択できます。

- デフォルト — Auto Scaling は、起動テンプレートのキャパシティ予約設定またはオープンキャパシティ予約を使用します。
- なし — Auto Scaling はキャパシティーの予約にインスタンスを起動しません。インスタンスはオンデマンド容量で実行されます。
- キャパシティ予約のみ – Auto Scaling は、キャパシティ予約またはキャパシティ予約グループでのみインスタンスを起動します。容量が利用できない場合、インスタンスは起動に失敗します。
- キャパシティ予約 - Auto Scaling は、キャパシティ予約またはキャパシティ予約グループにインスタンスを起動します。キャパシティーが利用できない場合、インスタンスはオンデマンドキャパシティーで実行されます。

キャパシティ予約のみまたはキャパシティ予約を最初に選択した場合は、キャパシティ予約ターゲットを指定できます。

> **Note**
>
> キャパシティーの予約 設定を選択する必要があります。キャパシティ予約のターゲットはオプションです。

キャパシティ予約の設定と起動テンプレートに関する考慮事項

キャパシティ予約のみまたはキャパシティ予約を最初に選択した場合は、次の点を考慮してください。

- キャパシティ予約のみまたはキャパシティ予約を最初に選択すると、Auto Scaling は起動テンプレートのキャパシティ予約ターゲットではなく、Auto Scaling グループで指定されたキャパシティ予約ターゲットを使用します。
- キャパシティ予約のみまたはキャパシティ予約を最初に選択し、キャパシティ予約ターゲットを指定しない場合、Auto Scaling は起動テンプレートのキャパシティ予約ターゲットまたはオープンキャパシティ予約を使用します。

キャパシティ予約のターゲット仕様

キャパシティ予約のみまたはキャパシティ予約を最初に選択すると、次のキャパシティ予約のターゲットオプションを使用できます。

- Open – Auto Scaling は、開いている任意のキャパシティ予約にインスタンスを起動します。キャパシティ予約のみを選択し、キャパシティーが利用できない場合、インスタンスは起動できません。キャパシティ予約を最初に選択し、キャパシティが利用できない場合、インスタンスはオンデマンドキャパシティで起動します。
- キャパシティ予約の指定 – Auto Scaling は、指定されたキャパシティ予約でインスタンスを起動します。キャパシティ予約のみを選択し、キャパシティーが利用できない場合、インスタンスは起動できません。キャパシティ予約を最初に選択し、キャパシティが利用できない場合、インスタンスはオンデマンドキャパシティで起動します。
- キャパシティ予約リソースグループの指定 – Auto Scaling は、指定されたキャパシティ予約リソースグループのオープンキャパシティ予約にインスタンスを起動します。キャパシティ予約のみを選択し、キャパシティーが利用できない場合、インスタンスは起動できません。キャパシティ予約を最初に選択し、キャパシティが利用できない場合、インスタンスはオンデマンドキャパシティで起動します。

Auto Scaling グループでキャパシティ予約を使用する

Auto Scaling グループでキャパシティ予約を使用するには、Auto Scaling グループにキャパシティ予約設定を追加するか、起動テンプレートでキャパシティ予約ターゲットを指定できます。選択した方法では、最初のキャパシティ予約またはキャパシティ予約リソースグループを作成する必要があります。

キャパシティ予約を作成するには、「Amazon EC2 ユーザーガイド」の「キャパシティ予約の作成」を参照してください。キャパシティ予約グループを作成するには、「Amazon EC2 ユーザーガイド」の「キャパシティ予約グループの作成」を参照してください。

`targeted` キャパシティ予約とキャパシティ予約グループを使用する Auto Scaling グループを作成するすべての手順については、「」を参照してくださいキャパシティ予約を使用する起動テンプレートで Auto Scaling グループで`targeted`キャパシティ予約を使用する。

Auto Scaling グループを作成または編集し、キャパシティ予約設定を使用する

Auto Scaling グループを作成または編集するときに、次のいずれかの方法を使用してキャパシティ予約設定を使用します。

Console

新しいグループでキャパシティ予約設定を使用するには (コンソール)

1. Amazon EC2 起動ウィザードを使用して Auto Scaling グループを作成する 「」の手順に従って、ステップ 3 までの手順の各ステップを完了します。
2. グループサイズとスケーリングの設定ページの「キャパシティーの追加設定」、「キャパシティーの予約」 設定で、キャパシティーの予約 設定を選択します。キャパシティ予約の設定の詳細については、「」を参照してくださいキャパシティ予約の設定。
3. Amazon EC2 起動ウィザードを使用して Auto Scaling グループを作成する のステップを続行します。

AWS CLI

新しいグループでキャパシティ予約設定を使用するには (AWS CLI)

create-auto-scaling-group コマンドに `--capacity-reservation-specification`パラメータを追加します。

1. キャパシティーの予約 設定を指定します。詳細については、「キャパシティ予約の設定」を参照してください。
2. キャパシティ予約ターゲットを指定します。キャパシティ予約のみまたはキャパシティ予約を最初に選択し、キャパシティ予約ターゲットを指定しない場合、Auto Scaling は起動テンプレートのキャパシティ予約ターゲットまたはオープンキャパシティ予約を使用します。

Console

既存のグループでキャパシティ予約設定を使用するには (コンソール)

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. 画面の上部のナビゲーションバーで、Auto Scaling グループを作した AWS リージョン を選択します。
3. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
4. 詳細タブのキャパシティ予約設定で、編集を選択します。

5. 「追加のキャパシティ設定」の「キャパシティ予約設定」で、キャパシティ予約設定を選択します。 キャパシティ予約の設定の詳細については、「」を参照してくださいキャパシティ予約の設定。
6. [Update] (更新) を選択します。

AWS CLI

既存のグループでキャパシティ予約設定を使用するには (AWS CLI)

update-auto-scaling-group コマンドに `--capacity-reservation-specification`パラメータを追加します。

1. キャパシティーの予約 設定を指定します。詳細については、「キャパシティ予約の設定」を参照してください。
2. キャパシティ予約ターゲットを指定します。キャパシティ予約のみまたはキャパシティ予約を最初に選択し、キャパシティ予約ターゲットを指定しない場合、Auto Scaling は起動テンプレートのキャパシティ予約ターゲットまたはオープンキャパシティ予約を使用します。

キャパシティ予約を使用する起動テンプレートで Auto Scaling グループで**`targeted`**キャパシティ予約を使用する

このトピックでは、`targeted` キャパシティ予約にオンデマンドインスタンスを起動する Auto Scaling グループを作成する方法を説明します。これにより、特定のキャパシティ予約をいつ使用するかをより細かく制御できます。

基本的なステップは次のとおりです。

1. 同じインスタンスタイプ、プラットフォーム、インスタンス数を持つ複数のアベイラビリティーゾーンでキャパシティ予約を作成します。
2. AWS Resource Groups を使用してキャパシティ予約をグループ化します。
3. キャパシティ予約と同じアベイラビリティーゾーンを使用して、リソースグループを対象とする起動テンプレートを使用して Auto Scaling グループを作成します。

内容

- ステップ 1: キャパシティ予約を作成する
- ステップ 2: キャパシティ予約グループを作成する

ステップ 1: キャパシティ予約を作成する

この手順では、targetedキャパシティ予約を使用します。

> **Note**
>
> 最初にキャパシティ予約を作成するときは、targeted 予約のみを作成できます。

Console

キャパシティ予約を作成するには

1. EC2 で Amazon https://console.aws.amazon.com/ec2/ コンソールを開きます。
2. [キャパシティ予約]、[作成キャパシティ予約] の順に選択します。
3. 「キャパシティ予約を作成」のページで、[インスタンスの詳細] セクションで、以下の設定を指定します。起動するインスタンスのインスタンスタイプ、プラットフォーム、アベイラビリティーゾーンは、ここで指定するインスタンスタイプ、プラットフォーム、アベイラビリティーゾーンと一致する必要があります。一致しない場合、キャパシティ予約 は適用されません。

   a. [インスタンスのタイプ] では、リザーブドキャパシティーに起動するインスタンスのタイプを選択します。
   b. [プラットフォーム] では、インスタンスのオペレーティングシステムを選択します。
   c. [アベイラビリティーゾーン]で、キャパシティを予約したい最初のアベイラビリティーゾーンを選択します。
   d. [合計キャパシティ] で、必要なインスタンス数を選択します。Auto Scaling グループに必要なインスタンスの総数を、使用する予定のアベイラビリティーゾーンの数で割って計算します。
4. [キャパシティ予約の詳細] で、[キャパシティ予約の終了] について、次のオプションのいずれかを選択します。

   - [特定の時間]– 指定した日時にキャパシティ予約を自動的にキャンセルできます。

- [手動] – 明示的にキャンセルするまでキャパシティを予約できます。

5. [インスタンスの適格性] については、[ターゲット: キャパシティ予約を対象とするインスタンスのみ] を選択します。
6. (オプション) [タグ] には、キャパシティ予約に関連付けるタグを指定します。
7. [作成] を選択します。
8. 新しく作成したキャパシティ予約の ID をメモしておきます。キャパシティ予約グループを設定するために必要です。

Auto Scaling グループに対して有効にする各アベイラビリティーゾーンに対してこの手順を繰り返し、[アベイラビリティーゾーン] オプションの値のみを変更します。

AWS CLI

キャパシティ予約を作成するには

次の create-capacity-reservation コマンドを使用して、キャパシティ予約を作成します。`--availability-zone`、`--instance-type`、`--instance-platform`、および `--instance-count` のサンプルの値を置き換えます。

```
aws ec2 create-capacity-reservation \
    --availability-zone us-east-1a \
    --instance-type c5.xlarge \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --instance-match-criteria targeted
```

キャパシティ予約 ID の結果の例

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234567890abcdef1",
        "OwnerId": "123456789012",
        "CapacityReservationArn": "arn:aws:ec2:us-east-1:123456789012:capacity-
reservation/cr-1234567890abcdef1",
        "InstanceType": "c5.xlarge",
        "InstancePlatform": "Linux/UNIX",
        "AvailabilityZone": "us-east-1a",
        "Tenancy": "default",
        "TotalInstanceCount": 3,
        "AvailableInstanceCount": 3,
```

```
        "EbsOptimized": false,
        "EphemeralStorage": false,
        "State": "active",
        "StartDate": "2023-07-26T21:36:14+00:00",
        "EndDateType": "unlimited",
        "InstanceMatchCriteria": "targeted",
        "CreateDate": "2023-07-26T21:36:14+00:00"
    }
}
```

新しく作成したキャパシティ予約の ID をメモしておきます。キャパシティ予約グループを設定するために必要です。

Auto Scaling グループに対して有効にする各アベイラビリティーゾーンに対してこのコマンドを繰り返し、`--availability-zone` オプションの値のみを変更します。

ステップ 2: キャパシティ予約グループを作成する

キャパシティ予約の作成が完了したら、 AWS Resource Groups サービスを使用してそれらをグループ化できます。 AWS Resource Groups は、さまざまな用途で複数の異なるタイプのグループをサポートしています。Amazon EC2 は、サービスにリンクされたリソースグループと呼ばれる専用グループを使用して、キャパシティ予約のグループをターゲットにします。このサービスにリンクされたリソースグループを操作するには、 AWS CLI または SDK を使用できますが、コンソールは使用できません。サービスにリンクされたリソース グループの詳細については、「AWS Resource Groups User Guide」の 「Service configurations for resource groups」を参照してください。

を使用してキャパシティ予約グループを作成するには AWS CLI

create-group コマンドを使用して、キャパシティ予約のみを含めることができるリソースグループを作成します。この例では、プレイスメントグループ名は*my-cr-group*です。

```
aws resource-groups create-group \
    --name my-cr-group \
    --configuration '{"Type":"AWS::EC2::CapacityReservationPool"}'
 '{"Type":"AWS::ResourceGroups::Generic", "Parameters": [{"Name": "allowed-resource-
types", "Values": ["AWS::EC2::CapacityReservation"]}]}'
```

以下に、応答の例を示します。

```
{
    "Group": {
```

```
        "GroupArn": "arn:aws:resource-groups:us-east-1:123456789012:group/my-cr-group",
        "Name": "my-cr-group"
    },
    "GroupConfiguration": {
        "Configuration": [
            {
                "Type": "AWS::EC2::CapacityReservationPool"
            },
            {
                "Type": "AWS::ResourceGroups::Generic",
                "Parameters": [
                    {
                        "Name": "allowed-resource-types",
                        "Values": [
                            "AWS::EC2::CapacityReservation"
                        ]
                    }
                ]
            }
        ],
        "Status": "UPDATE_COMPLETE"
    }
}
```

リソースグループのARNを書き留めます。Auto Scaling グループの起動テンプレートを設定するために必要です。

AWS CLI を使用してキャパシティ予約を新しく作成したグループに関連付けるには

次の group-resources コマンドを使用して、キャパシティ予約を新しく作成したキャパシティ予約グループに関連付けます。--resource-arns オプションでは、ARNs を使用してキャパシティ予約を指定します。関連する リージョン、アカウント ID、および前にメモした予約 ARNs を使用して IDs を作成します。この例では、IDs *cr-1234567890abcdef1*と の予約*cr-54321abcdef567890*は、 という名前のグループでグループ化されます*my-cr-group*。

```
aws resource-groups group-resources \
    --group my-cr-group \
    --resource-arns \
      arn:aws:ec2:region:account-id:capacity-reservation/cr-1234567890abcdef1 \
      arn:aws:ec2:region:account-id:capacity-reservation/cr-54321abcdef567890
```

以下に、応答の例を示します。

```
{
   "Succeeded": [
   "arn:aws:ec2:us-east-1:123456789012:capacity-reservation/cr-1234567890abcdef1",
   "arn:aws:ec2:us-east-1:123456789012:capacity-reservation/cr-54321abcdef567890"
   ],
   "Failed": [],
   "Pending": []
}
```

リソースグループの変更または削除については、AWS 「Resource Groups API Reference」を参照してください。

ステップ 3: 起動テンプレートを作成する

起動テンプレートを使用するには、 ステップ 1: キャパシティ予約を作成すると を完了しますステップ 2: キャパシティ予約グループを作成する。次に、起動テンプレートを作成します。

Console

起動テンプレートを作成するには

1. EC2 で Amazon https://console.aws.amazon.com/ec2/ コンソールを開きます。
2. ナビゲーションペインで、[インスタンス] の [テンプレートの起動] を選択します。
3. [起動テンプレートの作成] を選択します。名前を入力し、起動テンプレートの最初のバージョンの説明を加えます。
4. [Auto Scaling ガイダンス] で、[チェックボックス] を選択します。
5. 起動テンプレートを作成します。使用する予定のキャパシティ予約に一致する AMI とインスタンスタイプを選択し、オプションでキーペア、1 つ以上のセキュリティグループ、インスタンスに追加の EBS ボリュームまたはインスタンスストアボリュームを選択します。
6. [高度な設定]を展開し、以下の操作を行います。
   a. [キャパシティ予約]で、[グループ別のターゲット] を選択します。
   b. [キャパシティ予約 - グループ別のターゲット] で、前のセクションで作成したキャパシティ予約グループを選択し、[保存] を選択します。
7. [起動テンプレートの作成] を選択します。
8. 確認ページで、[Auto Scaling グループの作成] を選択します。

AWS CLI

起動テンプレートを作成するには

次の create-launch-template コマンドを使用して、キャパシティ予約が特定のリソースグループをターゲットにするように指定する起動テンプレートを作成します。--launch-template-name のサンプル値を置き換えます。*c5.xlarge* をキャパシティ予約で使用したインスタンスタイプに置き換え、*ami-0123456789EXAMPLE* 使用するAMIの ID を指定します。を、前のセクションの冒頭で作成したリソースグループの ARN *arn:aws:resource-groups:region:account-id:group/my-cr-group*に置き換えます。

```
aws ec2 create-launch-template \
    --launch-template-name my-launch-template \
    --launch-template-data \
        '{"InstanceType": "c5.xlarge",
          "ImageId": "ami-0123456789EXAMPLE",
          "CapacityReservationSpecification":
            {"CapacityReservationTarget":
                { "CapacityReservationResourceGroupArn": "arn:aws:resource-
groups:region:account-id:group/my-cr-group" }
            }
        }'
```

以下に、応答の例を示します。

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-0dd77bd41dEXAMPLE",
        "LaunchTemplateName": "my-launch-template",
        "CreateTime": "2023-07-26T21:42:48+00:00",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

ステップ 4: Auto Scaling グループを作成する

Console

通常どおり Auto Scaling グループを作成しますが、VPC サブネットを選択するときは、作成した`targeted`キャパシティ予約に一致する各アベイラビリティーゾーンからサブネットを選択します。その後、Auto Scaling グループがこれらのアベイラビリティーゾーンのいずれかでオンデマンドインスタンスを起動すると、そのインスタンスがそのアベイラビリティーゾーンのリザーブドキャパシティで実行されます。希望するキャパシティが満たされる前にリソースグループがキャパシティ予約を使い果たした場合、リザーブドキャパシティを超えるものは通常のオンデマンドキャパシティとして起動されます。

Auto Scaling グループを作成するには

1. Word で Amazon EC2 コンソールを開き、ナビゲーションペインから Auto Scaling Groups を選択します。 https://console.aws.amazon.com/ec2/
2. 画面上部のナビゲーションバーで、起動テンプレートの作成時に使用した AWS リージョンものと同じ を選択します。
3. [Auto Scaling グループの作成] を選択します。
4. [起動テンプレートまたは起動設定を選択する] ページで [Auto Scaling グループ名] にAuto Scaling グループの名前を入力します。
5. [起動テンプレート] で、既存の起動テンプレートを選択します。
6. [起動テンプレートのバージョン] で、スケールアウト時に Auto Scaling グループで使用する起動テンプレートのバージョン (デフォルト、最新、または特定のバージョン) を選択します。
7. インスタンス起動オプションの選択ページで、インスタンスタイプ要件セクションをスキップして、起動テンプレートで指定された EC2 インスタンスタイプを使用します。
8. Network の VPC で、VPC を選択します。Auto Scaling グループは、起動テンプレートで指定したセキュリティグループと同じ VPC で作成する必要があります。起動テンプレートでセキュリティグループを指定しなかった場合は、キャパシティ予約と同じアベイラビリティーゾーンにサブネットがある任意の VPC を選択できます。
9. [アベイラビリティーゾーンとサブネット]では、キャパシティ予約がどのアベイラビリティーゾーンにあるかに基づいて、含めたいサブネットを各アベイラビリティーゾーンから選択します。
10. [次へ] を 2 回選択します。

11. [グループサイズとスケーリングポリシーを設定] ページの [必要なキャパシティ] に、起動するインスタンスの初期数を入力します。この数値を最小キャパシティまたは最大キャパシティ制限の範囲外の値に変更する場合は、[最小キャパシティ] または [最大キャパシティ] の値を更新する必要があります。詳細については、「Auto Scaling グループのスケーリング制限を設定する」を参照してください。
12. [Skip to review] を選択します。
13. [Review (レビュー)]ページで、[Create Auto Scaling group (Auto Scaling グループを作成)] を選択します。

AWS CLI

Auto Scaling グループを作成するには

次の create-auto-scaling-group コマンドを使用して、起動テンプレートの名前とバージョンを `--launch-template`オプションの値として指定します。`--auto-scaling-group-name`、`--min-size`、`--max-size`、および `--vpc-zone-identifier` のサンプルの値を置き換えます。

`--availability-zones` オプションには、キャパシティ予約を作成したアベイラビリティーゾーンを指定します。例えば、キャパシティ予約で `us-east-1a` と `us-east-1b` のアベイラビリティーゾーンの場合は、同じゾーンに Auto Scaling グループを作成する必要があります。その後、Auto Scaling グループがこれらのアベイラビリティーゾーンのいずれかでオンデマンドインスタンスを起動すると、そのインスタンスがそのアベイラビリティーゾーンのリザーブドキャパシティで実行されます。希望するキャパシティが満たされる前にリソースグループがキャパシティ予約を使い果たした場合、リザーブドキャパシティを超えるものは通常のオンデマンドキャパシティとして起動されます。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-launch-template,Version='1' \
    --min-size 6 \
    --max-size 6 \
    --vpc-zone-identifier "subnet-5f46ec3b,subnet-0ecac448" \
    --availability-zones us-east-1a us-east-1b
```

関連リソース

実装例については、次の AWS samples GitHub リポジトリの AWS CloudFormation テンプレートを参照してください: aws-auto-scaling-backedhttps://github.com/aws-samples/-by-on-demand-capacity-reservations/。

キャパシティ予約の詳細について調べる際は、以下の関連トピックが役に立ちます。

- オンデマンドキャパシティ予約
  - 「Amazon EC2 ユーザーガイド」の「キャパシティ予約の作成」
  - 「Amazon EC2 ユーザーガイド」の「オンデマンドキャパシティ予約」
  - AWS クラウド運用と移行ブログの「Amazon EC2 オンデマンドキャパシティ予約のグループをターゲットにする」
- キャパシティブロック (期間が定義されたキャパシティ予約)
  - 「Amazon Word ユーザーガイド」の「ML のキャパシティブロックEC2」
  - 使用アイテム Capacity Blocks 機械学習ワークロード用の

を使用してコマンドラインから Auto Scaling グループを作成する AWS CloudShell

サポートされている AWS リージョンでは、 から直接起動するブラウザベースの事前認証済みシェル AWS CloudShell に対して、 を使用して AWS CLI コマンドを実行できます AWS Management Console。任意のシェル (Bash、 PowerShell、または Z シェル) を使用して、 サービスに対して AWS CLI コマンドを実行できます。

次の 2 つの方法のいずれか AWS Management Console を使用して、 AWS CloudShell から を起動できます。

- コンソールのナビゲーションバー AWS CloudShell のアイコンを選択します。これは検索ボックスの右側にあります。
- コンソールナビゲーションバーの検索ボックスを使用して CloudShell を検索し、CloudShell オプションを選択します。

が新しいブラウザウィンドウで初めて AWS CloudShell 起動すると、ウェルカムパネルが表示され、主要な機能が一覧表示されます。このパネルを閉じると、シェルがコンソール認証情報を構成および

転送する間、ステータスのアップロードが提供されます。コマンドプロンプトが表示されたら、シェルは対話的な操作の準備ができています。

このサービスの詳細については、「AWS CloudShell ユーザーガイド」を参照してください。

AWS CloudFormation で Auto Scaling グループを作成する

Amazon EC2 Auto Scaling は AWS CloudFormation と統合されています。これは、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスです。必要なすべての AWS リソース (Auto Scaling グループなど) を説明するテンプレートを作成すれば、AWS CloudFormation がお客様に代わってこれらのリソースのプロビジョニングや設定を処理します。

AWS CloudFormation を使用する際には、Amazon EC2 Auto Scaling リソースに対し一貫したセットアップを繰り返すために、テンプレートを再利用できます。リソースを一度記述するだけで、同じリソースを複数の AWS アカウント とリージョンで何度でもプロビジョニングできます。

Amazon EC2 Auto Scaling とAWS CloudFormation テンプレート

Amazon EC2 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 や YAML でフォーマットされたテキストファイルです。これらのテンプレートには、AWS CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、AWS CloudFormation デザイナー を使えば、AWS CloudFormation テンプレートを使いこなすことができます。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。

Amazon EC2 Auto Scaling に独自のスタックテンプレートの作成を開始するには、次のタスクを完了してください。

- 「AWS::EC2::LaunchTemplate」を使用して起動テンプレートを作成します。
- 「AWS::AutoScaling::AutoScalingGroup」を使用して Auto Scaling グループを作成します。

Auto Scaling グループを Application Load Balancer の背後にデプロイする方法を表示するウォークスルーについては、「AWS CloudFormation ユーザーガイド」の「チュートリアル: スケーラブルなロードバランシングウェブサーバーの作成」を参照してください。

Auto Scaling グループおよび関連リソースを作成するテンプレートスニペットのその他の便利な例については、「AWS CloudFormation ユーザーガイド」の次のセクションを参照してください。

- Amazon EC2 Auto Scaling リソースタイプのリファレンス
- AWS CloudFormation で Amazon EC2 Auto Scaling リソースを設定する

AWS CloudFormation の詳細はこちら

AWS CloudFormation の詳細については、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

でインスタンスタイプのレコメンデーションを取得する AWS Compute Optimizer

AWS は、 の機能を使用して、パフォーマンスの向上、コストの削減、またはその両方に役立つ Amazon EC2 インスタンスタイプのレコメンデーションを提供します AWS Compute Optimizer。これらの推奨事項を使用して、Auto Scaling グループ内の新しいインスタンスタイプに移行するかどうかを判断できます。

推奨事項を作成するために、Compute Optimizer は既存インスタンスの仕様と最近のメトリックス履歴を分析します。次に、コンパイルされたデータを使用して、既存のパフォーマンスワークロードを処理するために最適化されている Amazon EC2 インスタンスタイプを推奨します。推奨事項は、時間あたりのインスタンス料金とともに返されます。

> **Note**
>
> Compute Optimizer から推奨事項を取得するには、まず Compute Optimizer にオプトインする必要があります。詳細については、「AWS Compute Optimizer ユーザーガイド」の「AWS Compute Optimizerの使用開始」を参照してください。

内容

- 制限
- 結果

- 推奨事項の表示
- 推奨事項の評価に関する考慮事項

制限

Compute Optimizer は、M、C、R、T、X のインスタンスタイプを起動して実行するように設定された Auto Scaling グループ内のインスタンスの推奨事項を生成します。ただし、 AWS Graviton2 プロセッサを搭載した -g インスタンスタイプ (C6g など) と、ネットワーク帯域幅のパフォーマンスが高い -n インスタンスタイプ (M5n など) のレコメンデーションは生成されません。

また、Auto Scaling グループは、単一のインスタンスタイプを実行するように設定する必要があります (つまり、インスタンスタイプが混在しない) 。また、スケーリングポリシーがアタッチされておらず、希望キャパシティ、最小キャパシティ、最大キャパシティ (インスタンス数が固定されている Auto Scaling グループ) に対して同じ値を持つ必要があります。Compute Optimizer は、これらの構成要件のすべてを満たす Auto Scaling グループのインスタンス推奨事項を生成します。

結果

Compute Optimizer は、Auto Scaling グループの調査結果を次のように分類します。

- 最適化されていない – Compute Optimizer がワークロードのパフォーマンスを向上できる推奨事項を特定した場合、Auto Scaling グループは、最適化されていないとみなされます。
- 最適化 – 選択したインスタンスタイプに基づいて、ワークロードを実行するためにグループが正しくプロビジョニングされていると Compute Optimizer が判断した場合、Auto Scaling グループは、最適化されていると見なされます。最適化されたリソースについては、Compute Optimizer が新世代のインスタンスタイプを推奨することがあります。
- なし – Auto Scaling グループの推奨事項はありません。これは、Compute Optimizer を 12 時間未満にオプトインした場合、または Auto Scaling グループの実行が 30 時間未満の場合、または Auto Scaling グループまたはインスタンスタイプが Compute Optimizer でサポートされていない場合に発生する可能性があります。詳細については、「制限」セクションを参照してください。

推奨事項の表示

Compute Optimizer にオプトインすると、Auto Scaling グループに対して生成された結果と推奨事項を表示できます。最近オプトインした場合、推奨事項が最大 12 時間、反映されないことがあります。

Auto Scaling グループに対して生成された推奨事項を表示する

1. https://console.aws.amazon.com/compute-optimizer/ で Compute Optimizer コンソールを開きます。

   ダッシュボードページが開きます。

2. [View recommendations for all Auto Scaling groups (すべての Auto Scaling グループの推奨事項を表示する)] を選択します。
3. Auto Scaling スケーリンググループを選択します。
4. [View detail (詳細を表示)] を選択します。

   デフォルトのテーブル設定に基づいて、事前構成されたビューに最大 3 つの異なるインスタンスの推奨事項が表示されるように、ビューが変更されます。また、Auto Scaling グループの recent CloudWatch メトリクスデータ (平均CPU使用率、平均ネットワーク入力、平均ネットワーク出力) も提供します。

推奨事項の 1 つを使用するかどうかを決定します。パフォーマンスの向上のために最適化するか、コスト削減のために最適化するか、これら 2 つの組み合わせのために最適化するかを決定します。

Auto Scaling グループのインスタンスタイプを変更するには、起動テンプレートを更新するか、Auto Scaling group を更新して新しい起動設定を使用します。既存のインスタンスでは、引き続き以前の設定を使用します。既存のインスタンスを更新するには、これらのインスタンスを終了して Auto Scaling グループに置き換えるようにするか、オートスケーリングにより終了ポリシーに基づいて古いインスタンスを新しいインスタンスに徐々に置き換えるようにします。

> **Note**
>
> インスタンスの最大有効期間とインスタンスの更新機能を使用すると、Auto Scaling グループ内の既存のインスタンスを置き換えて、新しい起動テンプレートや起動設定を使用する新しいインスタンスを起動することもできます。詳細については、インスタンスの最大存続期間に基づいて Auto Scaling インスタンスを置き換えるおよびインスタンスの更新を使用して Auto Scaling グループのインスタンスを更新するを参照してください。

推奨事項の評価に関する考慮事項

新しいインスタンスタイプに移行する前に、次の点を考慮してください。

- 推奨情報は使用状況を予測するものではありません。推奨事項は、直近の 14 日間の使用履歴に基づいています。将来の使用ニーズを満たすことが予想されるインスタンスタイプをかならず選択してください。
- グラフ化されたメトリクスを参考にして、実際の使用量がインスタンスのキャパシティーよりも低いかどうかを判断します。 CloudWatch でメトリクスデータ (平均、ピーク、パーセンタイル) を表示して、EC2 インスタンスのレコメンデーションをさらに評価することもできます。例えば、1 日の中で CPU の割合メトリクスがどのように変化するか、また、対応する必要があるピークがあるかどうかに注目してください。詳細については、「Amazon CloudWatch ユーザーガイド」の「使用可能なメトリクスの表示」を参照してください。
- Compute Optimizer は、バーストパフォーマンスインスタンス (T3、T3a、および T2 インスタンス) の推奨事項を提供する場合があります。ベースラインを超えて定期的にバーストする場合は、新しいインスタンスタイプの vCPUs に基づいて引き続きバーストできることを確認してください。詳細については、「Amazon CPU ユーザーガイド」の「バーストパフォーマンスインスタンスのワードクレジットとベースラインパフォーマンス」を参照してください。 EC2
- リザーブドインスタンスを購入した場合、オンデマンドインスタンスはリザーブドインスタンスとして請求される場合があります。現在のインスタンスタイプを変更する前に、まずリザーブドインスタンスの使用率と適用範囲に対する影響を評価します。
- 可能であれば、新世代のインスタンスへの交換を検討します。
- 別のインスタンスファミリーに移行する場合は、仮想化、アーキテクチャー、ネットワークタイプなどの点で、現在のインスタンスタイプと新しいインスタンスタイプに互換性があることを確認してください。詳細については、「Amazon Word ユーザーガイド」の「インスタンスのサイズ変更の互換性」を参照してください。 EC2
- 最後に、推奨事項ごとに提供されるパフォーマンスリスク評価を検討します。パフォーマンスリスクは、推奨されるインスタンスタイプがワークロードのパフォーマンス要件を満たすかどうかを検証するために費やす必要のある作業量を示します。また、変更前と変更後に厳格な負荷テストおよびパフォーマンステストを行うことをお勧めします。

追加リソース

このページのトピックに加えて、次のリソースも参照してください。

- Amazon EC2 インスタンスタイプ
- AWS Compute Optimizer ユーザーガイド

Elastic Load Balancing を使用して Auto Scaling グループ内の受信アプリケーショントラフィックを分散する

Elastic Load Balancing は、実行中のすべてのEC2インスタンスに受信アプリケーショントラフィックを自動的に分散します。Elastic Load Balancing は、どのインスタンスにも負荷がかからないように、トラフィックを最適にルーティングすることで受信したリクエストを管理します。Auto Scaling グループで Elastic Load Balancing を使用するには、Auto Scaling グループにロードバランサーをアタッチする。これにより、グループがロードバランサーに登録され、ロードバランサーは、Auto Scaling グループへのすべての受信ウェブトラフィックの 1 つのお問合せポイントとして機能します。

Auto Scaling グループで Elastic Load Balancing を使用する場合、ロードバランサーに個々のEC2インスタンスを登録する必要はありません。Auto Scaling グループによって起動されたインスタンスは、自動的にロードバランサーのメンバーとなります。同様に、Auto Scaling グループによって終了されたインスタンスは、ロードバランサーから自動的に登録解除されます。

ロードバランサーを Auto Scaling グループにアタッチした後、Elastic Load Balancing メトリクス (ターゲットあたりの Application Load Balancer のリクエスト数など)を使用して、需要の変化に応じてグループ内のインスタンス数をスケールするように Auto Scaling グループを設定できます。

必要に応じて、Elastic Load Balancing ヘルスチェックを Auto Scaling グループに追加して、Amazon EC2 Auto Scaling がこれらの追加のヘルスチェックに基づいて異常なインスタンスを識別して置き換えられるようにできます。それ以外の場合は、ターゲットグループの正常なホスト数が許可を下回った場合に通知する CloudWatch アラームを作成できます。

内容

Elastic Load Balancing のタイプ

Elastic Load Balancing は、Auto Scaling グループで使用できる四つのタイプのロードバランサーを提供します:それらは、Application Load Balancer、Network Load Balancer、Gateway Load Balancer、Classic Load Balancer です。

ロードバランサーの設定方法は、種類によって大きく異なります。Application Load Balancer、Network Load Balancer、Gateway Load Balancer で、インスタンスはターゲットグループにターゲットとしてメンバーとされ、トラフィックをターゲットグループに送信します。Classic Load Balancer で、インスタンスはロードバランサーに直接メンバーとされます。

Application Load Balancer

アプリケーションレイヤー (HTTP/HTTPS) でルーティングとロードバランシングを行い、パスベースのルーティングをサポートします。Application Load Balancer は、仮想プライベートクラウド () 内のEC2インスタンスなど、1 つ以上の登録済みターゲットのポートにリクエストをルーティングできますVPC。

Network Load Balancer

Layer-4 ヘッダーから抽出されたアドレス情報に基づいて、トランスポートレイヤー (TCP/UDP Layer-4) でルーティングとロードバランスを行います。Network Load Balancer は、ロードバランサーの有効期間中、トラフィックバーストを処理し、クライアントの出典 IP を保持して、固定 IP を使用します。

Gateway Load Balancer

アプライアンス· インスタンスのフリートにトラフィックを分散します。ファイアウォール、侵入検知および防止システム、その他のアプライアンスなど、サードパーティー製の仮想アプライアンスのスケール、可用性、およびシンプルさを提供します。Gateway Load Balancer は、GENEVEプロトコルをサポートする仮想アプライアンスで動作します。追加の技術統合が必要なため、Gateway Load Balancer を選択する前に、必ずユーザーガイドを参照してください。

Classic Load Balancer

トランスポートレイヤー () でルーティングと負荷分散を行いますTCP/SSL), or at the application layer (HTTP/HTTPS。

利用可能なさまざまなタイプのロードバランサーの詳細については、次のリソースを参照してください。

- Elastic Load Balancing とは?

- Application Load Balancer とは?
- Network Load Balancer とは?
- Gateway Load Balancer とは ?
- Classic Load Balancer とは ?

Elastic Load Balancing ロードバランサーをアタッチする準備をする

Auto Scaling グループに Elastic Load Balancing ロードバランサーをアタッチするには、次の前提条件を満たす必要があります。

- Auto Scaling グループにトラフィックをルーティングするために使用されるロードバランサーおよびターゲットグループをすでに作成している必要があります。

  ロードバランサーおよびターゲットグループを作成するには、次の 2 つの方法があります。

  - Elastic Load Balancing の使用 — Elastic Load Balancing ドキュメントの手順に従い、Auto Scaling グループを作成する前にロードバランサーおよびターゲットグループを作成、設定します。Amazon EC2インスタンスを登録するステップをスキップします。ターゲットグループを EC2 Auto Scaling グループにアタッチすると、Amazon Auto Scaling は自動的にインスタンスの登録 (および登録解除) を処理します。詳細については、Elastic Load Balancing ユーザーガイドの Elastic Load Balancing で使用開始 を参照してください。
  - Amazon EC2 Auto Scaling の使用 – Amazon EC2 Auto Scaling コンソールから基本的な設定を使用して、ロードバランサーとターゲットグループを作成、設定、アタッチします。詳細については、「コンソールから Application Load Balancer または Network Load Balancer を設定する」を参照してください。
- ロードバランサーを作成する前に、必要なロードバランサーのタイプを確認してください。詳細については、「Elastic Load Balancing のタイプ」を参照してください。
- ロードバランサーとそのターゲットグループは AWS アカウント、Auto Scaling グループと同じ 、VPC、および リージョンに存在する必要があります。
- ターゲットグループは、`instance` のターゲットタイプを指定する必要があります。Auto Scaling グループを使用する場合、`ip` のターゲットタイプを指定することはできません。
- Auto Scaling グループの起動テンプレートに、ロードバランサーからの必要なインバウンドトラフィックを許可する正しいセキュリティグループが含まれていない場合、起動テンプレートを更新する必要があります。推奨されるルールは、ロードバランサーのタイプと、ロードバランサーが使用するバックエンドのタイプによって異なります。例えば、トラフィックをウェブサーバーにルーティングするには、ロードバランサーからのポート 80 でのインバウンドHTTPアクセスを許可し

ます。起動テンプレートが変更されても、既存のインスタンスは新しい設定に更新されません。既存のインスタンスを更新するには、インスタンスの更新を開始してインスタンスを置き換えることができます。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。

- 起動テンプレートのセキュリティグループでは、ヘルスチェックを実行するために Elastic Load Balancing の正しいポート上のロードバランサーからのアクセスも許可される必要があります。
- Gateway Load Balancer の背後に仮想アプライアンスをデプロイする場合、起動テンプレートの Amazon マシンイメージ (AMI) は、GENEVEプロトコルAMIをサポートする の ID を指定して、Auto Scaling グループが Gateway Load Balancer とトラフィックを交換できるようにする必要があります。また、起動テンプレートのセキュリティグループは、ポート 6081 でUDPトラフィックを許可する必要があります。

> **Tip**
>
> 完了に時間がかかるブートストラップスクリプトがある場合、必要に応じて起動ライフサイクルフックを Auto Scaling グループに追加して、ブートストラップスクリプトが正常に完了し、インスタンス上のアプリケーションでトラフィックを受け入れる準備ができるまで、ロードバランサーの背後でのインスタンス登録を遅延させることができます。Amazon Auto Scaling EC2Auto Scaling コンソールで Auto Scaling グループを最初に作成するときにライフサイクルフックを追加することはできません。ただし、ライフサイクルフックはグループを作成した後に追加できます。詳細については、「Amazon EC2 Auto Scaling のライフサイクルフック」を参照してください。

ターゲットのヘルスチェックを設定する

Elastic Load Balancing ロードバランサーで登録されたターゲットのヘルスチェックを設定して、トラフィックを適切に処理できるようにします。具体的な手順は、使用しているロードバランサーのタイプによって異なります。詳細については、以下のリソースを参照してください。

- Application Load Balancer —「Application Load Balancer のユーザーガイド」の「ターゲットグループのヘルスチェック」を参照してください。
- Network Load Balancer —「Network Load Balancer のユーザーガイド」の「ターゲットグループのヘルスチェック」を参照してください。
- Gateway Load Balancer —「Gateway Load Balancer のユーザーガイド」の「ターゲットグループのヘルスチェック」を参照してください。

- Classic Load Balancer —「Classic Load Balancer のユーザーガイド」の「Configure health checks for your Classic Load Balancer」を参照してください。

デフォルトでは、Amazon EC2 Auto Scaling はインスタンスを異常と見なさず、Elastic Load Balancing のヘルスチェックに合格しなかった場合は置き換えます。Auto Scaling グループのデフォルトのヘルスチェックは、EC2ヘルスチェックのみです。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

Amazon EC2 Auto Scaling が Elastic Load Balancing によって異常と報告されたインスタンスを置き換えるようにするには、Elastic Load Balancing ヘルスチェックを使用するように Auto Scaling グループを設定できます。これにより、Amazon EC2 Auto Scaling は、インスタンスがEC2ヘルスチェックまたは Elastic Load Balancing ヘルスチェックのいずれかに合格しなかった場合、インスタンスを異常と見なします。複数のロードバランサー ターゲットグループまたは Classic Load Balancer をグループにアタッチする場合、インスタンスが正常と見なされるためには、すべてのロードバランサーが、インスタンスは正常であるとして報告する必要があります。ロードバランサーの 1 つがインスタンスを異常として報告した場合は、他のロードバランサーがこれを正常として報告した場合でも、Auto Scaling グループ はそのインスタンスを置き換えます。

Auto Scaling グループに対してこれらのヘルスチェックを有効にする方法の詳細については、「Auto Scaling グループに Elastic Load Balancing ロードバランサーをアタッチする」を参照してください。

> **Note**
>
> これらのヘルスチェックをできるだけ早く開始するには、設定されているグループのヘルスチェック猶予期間が長すぎず、Elastic Load Balancing のヘルスチェックによりターゲットがリクエストを処理できるかどうかを判断できるのに十分な長さに設定されていることを確認してください。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。

Auto Scaling グループに Elastic Load Balancing ロードバランサーをアタッチする

このトピックでは、Auto Scaling グループに Elastic Load Balancing ロードバランサーをアタッチする方法について説明します。また、Elastic Load Balancing ヘルスチェックを有効にして、Elastic

Load Balancing が異常と報告するインスタンスを Amazon EC2 Auto Scaling で置き換える方法についても説明します。

デフォルトでは、Amazon EC2 Auto Scaling は Amazon EC2ヘルスチェックに基づいて異常または到達不可能なインスタンスのみを置き換えます。Elastic Load Balancing ヘルスチェックを有効にすると、EC2Auto Scaling グループにアタッチした Elastic Load Balancing ロードバランサーのいずれかが異常と報告した場合、Amazon Auto Scaling は実行中のインスタンスを置き換えることができます。

Application Load Balancer を Auto Scaling グループにアタッチするチュートリアルについては、「チュートリアル: スケーリングとロードバランシングを使用するアプリケーションのセットアップ」を参照してください。

> **Important**
>
> 先に進む前に、前のセクションのすべての前提条件を満たしてください。

内容

ターゲットグループまたは Classic Load Balancer をアタッチする

Auto Scaling グループを作成または更新する際は、1 つ以上のターゲットグループまたは Classic Load Balancer をアタッチできます。Application Load Balancer、Network Load Balancer、または Gateway Load Balancer をアタッチする場合は、ロードバランサーではなくターゲットグループをアタッチします。

このセクションの手順に従い、コンソールを使用して次の操作を実行します。

- Auto Scaling グループにターゲットグループまたは Classic Load Balancer をアタッチする
- Elastic Load Balancing のヘルスチェックを有効にする

新しい Auto Scaling グループを作成しているときに、既存のロードバランサーをアタッチするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。

2. 画面の上部のナビゲーションバーで、ロードバランサーを作成した AWS リージョン を選択します。
3. [Auto Scaling グループの作成] を選択します。
4. ステップ 1 と 2 では、必要に応じてオプションを選択し、「ステップ 3: アドバンストオプションを設定する」へ進みます。
5. [ロードバランシング] では、[既存のロードバランサーにアタッチ] を選択します。
6. [既存のロードバランサーにアタッチ] で、次のいずれかの操作を行います。

   a. Application Load Balancerでは、Network Load Balancers、および Gateway Load Balancer:

      [ロードバランサーのターゲットグループから選択] を選択して、[Existing load balancer target groups] (既存のロードバランサーターゲットグループ) でターゲットグループを選択します。

   b. Classic Load Balancerでは:

      [Classic Load Balancerから選択] を選択して、[Classic Load Balancer] でロードバランサーを選択します。
7. (オプション) [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Elastic Load Balancing のヘルスチェックをオンにする] を選択します。
8. (オプション) [ヘルスチェックの猶予期間] に秒単位で時間を入力します。これは、Amazon EC2 Auto Scaling が `InService` 状態になった後にインスタンスのヘルスステータスをチェックするまで待機する必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。
9. Auto Scaling グループの作成に進みます。Auto Scaling グループの作成後、インスタンスは自動的にロードバランサーにメンバーとされます。

Auto Scaling グループの作成後に既存のロードバランサーをアタッチするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. 統合 タブで、ロードバランシング、編集 を選択します。
4. [Load balancing] (ロードバランシング) で、次のいずれかの操作を行います。

a. [Application, Network または Gateway Load Balancer のターゲットグループ] で、そのチェックボックスを選択してターゲットグループを選択します。

b. [Classic Load Balancer] で、そのチェックボックスを選択してロードバランサーを選択します。

5. [Update] (更新) を選択します。

ロードバランサーのアタッチが完了したら、必要に応じて、そのターゲットグループで使用するヘルスチェックを有効にできます。

Elastic Load Balancing のヘルスチェックを有効にするには

1. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
2. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Elastic Load Balancing のヘルスチェックをオンにする] を選択します。
3. [ヘルスチェックの猶予期間] に、秒単位で時間を入力します。これは、Amazon EC2 Auto Scaling が `InService` 状態になった後にインスタンスのヘルスステータスをチェックするまで待機する必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。
4. [Update] (更新) を選択します。

> **Note**
>
> ロードバランサーがアタッチされている間、 AWS CLIを使用してロードバランサーのステータスを監視できます。Amazon EC2 Auto Scaling がインスタンスを正常に登録し、少なくとも 1 つの登録済みインスタンスがヘルスチェックに合格すると、 のステータスが表示されますInService。詳細については、「ロードバランサーのアタッチメントステータスを確認する」を参照してください。

ターゲットグループまたは Classic Load Balancer をデタッチする

ロードバランサーが不要になった場合、以下の手順に従って、Auto Scaling グループからデタッチします。

グループからロードバランサーをデタッチするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[ロードバランシング]、[編集] の順に選択します。
4. [ロードバランシング] で、次のいずれかの操作を行います。

   a. [Application, Network または Gateway Load Balancer のターゲットグループ] で、ターゲットグループの横にある削除 (X) アイコンを選択します。

   b. [Classic Load Balancer] で、ロードバランサーの横にある削除 (X) アイコンを選択します。
5. [Update] (更新) を選択します。

ターゲットグループのデタッチが完了したら、Elastic Load Balancing のヘルスチェックを無効にできます。

Elastic Load Balancing のヘルスチェックを無効にするには

1. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
2. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Elastic Load Balancing のヘルスチェックをオンにする] の選択を解除します。
3. [Update] (更新) を選択します。

コンソールから Application Load Balancer または Network Load Balancer を設定する

以下の手順に従い、Auto Scaling グループを作成するときに、Application Load Balancer または Network Load Balancer を作成してアタッチします。

新しい Auto Scaling グループの作成時に、新しいロードバランサーを作成してアタッチするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. [Auto Scaling グループの作成] を選択します。

3. ステップ 1 と 2 では、必要に応じてオプションを選択し、「ステップ 3: アドバンストオプションを設定する」へ進みます。
4. [ロードバランシング] で、[新しいロードバランサーにアタッチ] を選択します。
   a. [新しいロードバランサーにアタッチ] で、[ロードバランサーのタイプ] に、Application Load Balancer または Network Load Balancer のいずれを作成するかを選択します。
   b. [ロードバランサーの名前] は、ロードバランサーの名前を入力するか、デフォルトの名前のままにします。
   c. [ロードバランサーのスキーム] で、インターネット向け 公開ロードバランサーを作成するか、内部向けロードバランサーのデフォルトのままにするかを選択します。
   d. アベイラビリティーゾーンとサブネットで、EC2インスタンスの起動を選択した各アベイラビリティーゾーンのパブリックサブネットを選択します。(これらは、ステップ 2 で事前設定されます)。
   e. [リスナーとルーティング] をクリックし、リスナーのポート番号を更新し (必要な場合)、[デフォルトルーティング] で、[ターゲットグループの作成] を選択します。または、ドロップダウンリストから既存のターゲットグループを選択することができます。
   f. 最後のステップで、[ターゲットグループの作成] を選択した場合、[新しいターゲットグループ名] にターゲットグループの名前を入力するか、デフォルトの名前のままにします。
   g. ロードバランサーにタグを追加するには、[タグの追加] を選択して、タグごとにタグのキーと値を指定します。
5. (オプション) [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Elastic Load Balancing のヘルスチェックをオンにする] を選択します。
6. (オプション) [ヘルスチェックの猶予期間] に秒単位で時間を入力します。これは、Amazon EC2 Auto Scaling が `InService` 状態になった後にインスタンスのヘルスステータスをチェックするまで待機する必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。
7. Auto Scaling グループの作成に進みます。Auto Scaling グループの作成後、インスタンスは自動的にロードバランサーにメンバーとされます。

> **Note**
>
> Auto Scaling グループを作成したら、Elastic Load Balancing コンソールを使用して追加のリスナーを作成できます。これは、 HTTPSや などの安全なプロトコルでUDPリス

ナーを作成する必要がある場合に便利です。異なるポートを使用していれば、既存のロードバランサーにもっとリスナーを追加できます。

ロードバランサーのアタッチメントステータスを確認する

ロードバランサーは、アタッチされた後、グループのインスタンスを登録している間は、Adding 状態になります。グループのすべてのインスタンスが登録済みになると、Added 状態になります。最低 1 つの登録されたインスタンスがヘルスチェックを通過した後、InService 状態になります。ロードバランサーが InService状態の場合、Amazon EC2 Auto Scaling は異常と報告されたインスタンスを終了して置き換えることができます。メンバーとされたインスタンスがヘルスチェックに合格しない場合 (例えば、誤って設定されたヘルスチェックが原因)、ロードバランサーは InService 状態に入力しません。Amazon EC2 Auto Scaling はインスタンスを終了して置き換えません。

ロードバランサーをデタッチすると、グループのインスタンスの登録解除中、Removing 状態になります。登録解除してもインスタンスは引き続き実行されます。Application Load Balancer、Network Load Balancer、および Gateway Load Balancer では、Connection Draining がデフォルトで有効になっています。Connection Draining が有効になっている場合、Elastic Load Balancing は、処理中のリクエストが完了するまで、または最大タイムアウトの期限が切れるまで (どちらか早い方) 待機してから、インスタンスの登録を解除します。

(AWS Command Line Interface AWS CLI) または を使用して、アタッチメントのステータスを確認できます AWS SDKs。アタッチメントステータスはコンソールから確認できません。

を使用してアタッチメントのステータス AWS CLI を確認するには

次のdescribe-traffic-sourcesコマンドは、指定された Auto Scaling グループのすべてのトラフィックソースのアタッチメントステータスを返します。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

この例では、Auto Scaling ARN グループにアタッチされている Elastic Load Balancing ターゲットグループの と、 State要素内のターゲットグループのアタッチステータスを返します。

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/1234567890123456",
```

```
            "State": "InService",
            "Type": "elbv2"
        }
    ]
}
```

アベイラビリティーゾーンを追加する

理的な冗長性による安全性と信頼性を活用するには、作業しているリージョンの複数のアベイラビリティーゾーンにまたがって Auto Scaling グループを配置し、ロードバランサーをアタッチしてそれらのアベイラビリティーゾーンに受信トラフィックを分散させます。

1 つのアベイラビリティーゾーンが異常または使用できなくなった場合、Amazon EC2 Auto Scaling は影響を受けていないアベイラビリティーゾーンで新しいインスタンスを起動します。異常なアベイラビリティーゾーンが正常な状態に戻ると、Amazon EC2 Auto Scaling は Auto Scaling グループのすべてのアベイラビリティーゾーンにアプリケーションインスタンスを自動的に均等に再分散します。Amazon EC2 Auto Scaling は、インスタンス数が最も少ないアベイラビリティーゾーンで新しいインスタンスを起動することでこれを行います。ただし、試行が失敗した場合、Amazon EC2 Auto Scaling は成功するまで他のアベイラビリティーゾーンで起動を試みます。

Elastic Load Balancing は、ロードバランサーに有効な各アベイラビリティーゾーンにロードバランサー ノードを作成します。ロードバランサーのクロスゾーン ロードバランシングを有効にすると、各ロードバランサー ノードは、有効化されたすべてのアベイラビリティーゾーンのメンバーとされたインスタンスに均等にトラフィックを分配します。クロスゾーン負荷分散が無効の場合は、各ロードバランサーノードは、そのアベイラビリティーゾーンの登録されたインスタンスにのみリクエストを均等に分散します。

Auto Scaling グループを作成しているときは、少なくとも 1 つのアベイラビリティーゾーンを指定する必要があります。その後、Auto Scaling グループにアベイラビリティーゾーンを追加し、そのアベイラビリティーゾーンをロードバランサーに対して有効にすることで (ロードバランサーがサポートしている場合)、アプリケーションの可用性を拡張できます。

制限

ロードバランサーで有効になっているアベイラビリティーゾーンを更新するには、次の制限事項に注意する必要があります。

- ロードバランサー用のアベイラビリティーゾーンを有効にすると、そのアベイラビリティーゾーンからサブネットを 1 つ指定します。ロードバランサーでは、アベイラビリティーゾーンごとに最大で 1 つのサブネットを有効にできることに注意してください。

- インターネット向けロードバランサーの場合、ロードバランサーに指定するサブネットには最低八個の利用可能な IP アドレスが必要です。
- Application Load Balancer の場合、少なくても 二つのアベイラビリティーゾーンを有効にする必要があります。
- Network Load Balancers の場合、有効になっているアベイラビリティーゾーンを無効にすることはできませんが、追加のアベイラビリティーゾーンを有効にすることはできます。
- Gateway Load Balancers の場合、有効になっているアベイラビリティーゾーンを無効にすることはできませんが、追加のアベイラビリティーゾーンを有効にすることはできます。

次の手順に従い、追加のアベイラビリティーゾーンのサブネットに Auto Scaling グループとロードバランサーを拡張します。

アベイラビリティーゾーンを追加するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling グループ ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[ネットワーク]、[編集] の順に選択します。
4. [サブネット] で、Auto Scaling グループに追加したいアベイラビリティーゾーンに対応するサブネットの削除 (X) アイコンを選択します。
5. [更新] を選択します。
6. ロードバランサーのアベイラビリティーゾーンを更新して、Auto Scaling グループと同じゾーンを共有するには、以下のステップを完了します:

   a. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。

   b. ロードバランサーを選択します。

   c. 次のいずれかを行います。

      - Application Load Balancer および Network Load Balancer の場合:

        1. [説明] タブで、[アベイラビリティーゾーン] で、[サブネットを編集する] を選択します。
        2. [サブネットを編集する] ページの [アベイラビリティーゾーン] で、追加するアベイラビリティーゾーンのチェックボックスを選択します。そのゾーンにサブネットが

1 つしかない場合は、そのサブネットが選択されています。そのゾーンに複数のサブネットがある場合は、いずれかのサブネットを選択します。

- の Classic Load Balancer の場合VPC：

  1. [インスタンス] タブで、[アベイラビリティーゾーンの編集] を選択します。
  2. [サブネットを追加および削除する] ページの [使用可能なサブネット] で、追加 (+) アイコンを使用してサブネットを選択します。サブネットが [選択済みサブネット] に移動します。

d. [Save] を選択します。

関連リソース

Amazon EC2 Auto Scaling は、アベイラビリティーゾーンを変更すると グループのバランスを再調整します。これは、一部のインスタンスを置き換えて再配布することを意味します。詳細については、「例: 複数のアベイラビリティーゾーン全体にインスタンスを分散させる」を参照してください。

ロードバランサーのために有効になっていないアベイラビリティーゾーンにターゲットを登録している場合、そのロードバランサーはそれらのターゲットにトラフィックをルーティングしません。詳細については、Elastic Load Balancing ユーザーガイドの How Elastic Load Balancing works を参照してください。

アベイラビリティーゾーンの削除

Auto Scaling グループおよびロードバランサーからアベイラビリティーゾーンを削除するには、以下の手順に従います。

アベイラビリティーゾーンを削除するには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling グループ ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[ネットワーク]、[編集] の順に選択します。
4. [サブネット] で、Auto Scaling グループから削除したいアベイラビリティーゾーンに対応するサブネットの削除 (X) アイコンを選択します。そのゾーンに複数のサブネットがある場合は、それぞれの 削除 (X) アイコンを選択します。

5. [更新] を選択します。
6. ロードバランサーのアベイラビリティーゾーンを更新して、Auto Scaling グループと同じゾーンを共有するには、以下のステップを完了します:

   a. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
   b. ロードバランサーを選択します。
   c. 次のいずれかを行います。
      - Application Load Balancer の場合:

        1. [説明] タブで、[アベイラビリティーゾーン] で、[サブネットを編集する] を選択します。
        2. [サブネットの編集] ページの [アベイラビリティーゾーン] で、チェックボックスをオフにするとアベイラビリティーゾーンのサブネットが削除されます。

      - の Classic Load Balancer の場合VPC：

        1. [インスタンス] タブで、[アベイラビリティーゾーンの編集] を選択します。
        2. [サブネットをの追加と削除] ページの[利用可能なサブネット] で、その削除 (-) アイコンを使用してサブネットを削除します。サブネットが [利用可能なサブネット] の下に移動します。

   d. [Save] を選択します。

Auto Scaling グループからターゲットグループまたは Classic Load Balancer をデタッチする

ロードバランサーが不要になった場合、以下の手順に従って、Auto Scaling グループからデタッチします。

グループからロードバランサーをデタッチするには

1. で Amazon EC2コンソールを開きhttps://console.aws.amazon.com/ec2/、ナビゲーションペインから Auto Scaling Groups を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   [Auto Scaling groups] (Auto Scaling グループ) ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[ロードバランシング]、[編集] の順に選択します。
4. [ロードバランシング] で、次のいずれかの操作を行います。

a. [Application, Network または Gateway Load Balancer のターゲットグループ] で、ターゲットグループの横にある削除 (X) アイコンを選択します。

b. [Classic Load Balancer] で、ロードバランサーの横にある削除 (X) アイコンを選択します。

5. [Update] (更新) を選択します。

ターゲットグループのデタッチが完了したら、Elastic Load Balancing のヘルスチェックを無効にできます。

Elastic Load Balancing のヘルスチェックを無効にするには

1. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
2. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[Elastic Load Balancing のヘルスチェックをオンにする] の選択を解除します。
3. [Update] (更新) を選択します。

を使用した Elastic Load Balancing の使用例 AWS CLI

AWS Command Line Interface （ AWS CLI) を使用して、ロードバランサーとターゲットグループをアタッチ、デタッチ、記述し、Elastic Load Balancing ヘルスチェックを追加および削除し、有効にするアベイラビリティーゾーンを変更します。

このトピックでは、Amazon EC2 Auto Scaling の一般的なタスクを実行する AWS CLI コマンドの例を示します。

> **Important**
>
> その他の例については、「AWS CLI コマンドリファレンス」の「`aws elbv2`」と「`aws elb`」を参照してください。

内容

- ターゲットグループまたは Classic Load Balancer をデタッチする
- Elastic Load Balancing のヘルスチェックを削除する
- レガシーコマンド

ターゲットグループまたは Classic Load Balancer をアタッチする

次のcreate-auto-scaling-groupコマンドを使用して Auto Scaling グループを作成し、同時に Amazon リソースネーム () を指定してターゲットグループをアタッチしますARN。ターゲットグループは、Application Load Balancer、Network Load Balancer、または Gateway Load Balancer に関連付けることができます。

`--auto-scaling-group-name`、`--vpc-zone-identifier`、`--min-size`、および `--max-size` のサンプルの値を置き換えます。`--launch-template` オプションの場合、*`my-launch-template`* と *`1`* を Auto Scaling グループの起動テンプレートの名前とバージョンに置き換えます。`--traffic-sources` オプションでは、サンプルを Application Load Balancer、Network Load Balancer、または Gateway Load Balancer ARNARNのターゲットグループの に置き換えます。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1"
```

attach-traffic-sources コマンドを使用して、作成後に追加のターゲットグループを Auto Scaling グループにアタッチします。

次のコマンドは、同じグループに別のターゲットグループを追加します。

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2"
```

別の方法として、Classic Load Balancer をグループにアタッチするには、次の例のように、create-auto-scaling-group または attach-traffic-sources を使用するときに `--traffic-sources` オプションおよび `--type` オプションを指定します。*`my-classic-load-balancer`* を Classic Load Balancer の名前に置き換えます。`--type` オプションの場合、**`elb`** の値を指定します。

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

ターゲットグループまたは Classic Load Balancer の説明を表示する

Auto Scaling グループにアタッチされたロードバランサーまたはターゲットグループを記述するには、次のdescribe-traffic-sourcesコマンドを使用します。*my-asg* をグループの名前に置き換えます。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

この例では、Auto Scaling ARN グループにアタッチした Elastic Load Balancing ターゲットグループの を返します。

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1",
            "State": "InService",
            "Type": "elbv2"
        },
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2",
            "State": "InService",
            "Type": "elbv2"
        }
    ]
}
```

出力された State フィールドの説明については、「ロードバランサーのアタッチメントステータスを確認する」を参照してください。

Elastic Load Balancing のヘルスチェックを追加する

Auto Scaling グループがインスタンスで実行するヘルスチェックに Elastic Load Balancing ヘルスチェックを追加するには、次のupdate-auto-scaling-groupコマンドを使用して、 --health-check-type オプションの値**ELB**として を指定します。*my-asg* をグループの名前に置き換えます。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
```

```
  --health-check-type "ELB"
```

新しいインスタンスは、多くの場合、ヘルスチェックに合格する前に簡単なウォームアップの時間が必要です。猶予期間で十分なウォームアップ時間が提供されない場合、インスタンスはトラフィックを処理する準備ができていないように見えることがあります。Amazon EC2 Auto Scaling は、これらのインスタンスを異常と見なして置き換える場合があります。

ヘルスチェックの猶予期間を更新するには、次の例のように、update-auto-scaling-group を使用するときに `--health-check-grace-period` オプションを使用します。を秒数*300*に置き換えて、新しいインスタンスが異常であることが判明した場合、インスタンスを終了する前に稼働状態を維持します。

```
--health-check-grace-period 300
```

詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

アベイラビリティーゾーンを変更する

アベイラビリティーゾーンの変更には、注意が必要な制限がいくつかあります。詳細については、「アベイラビリティーゾーンを追加する」を参照してください。

Application Load Balancer または Network Load Balancer のアベイラビリティーゾーンを変更するには

1. ロードバランサーのアベイラビリティーゾーンを変更する前に、まず Auto Scaling グループのアベイラビリティーゾーンを更新して、使用しているインスタンスタイプが指定したゾーンで使用できることを確認することをお勧めします。

   Auto Scaling グループのアベイラビリティーゾーンを更新するには、次のupdate-auto-scaling-groupコマンドを使用します。サンプルサブネットを、有効にするアベイラビリティーゾーン内のサブネットIDsの IDsに置き換えます。指定したサブネットは、以前に有効であったサブネットに置き換わります。*my-asg* をグループの名前に置き換えます。

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
     --vpc-zone-identifier "subnet-41767929,subnet-cb663da2,subnet-8360a9e7"
   ```

2. 次のdescribe-auto-scaling-groupsコマンドを使用して、新しいサブネットのインスタンスが起動したことを確認します。インスタンスが起動した場合は、インスタンスとそのステータスのリストが表示されます。*my-asg* をグループの名前に置き換えます。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. 次の set-subnets コマンドを使用して、ロードバランサーのサブネットを指定します。サンプルサブネットを、有効にするアベイラビリティーゾーン内のサブネットIDsの IDsに置き換えます。アベイラビリティーゾーンごとに 1 つだけサブネットを指定できます。指定したサブネットは、以前に有効であったサブネットに置き換わります。をロードバランサーARNの *my-lb-arn*に置き換えます。

```
aws elbv2 set-subnets --load-balancer-arn my-lb-arn \
  --subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
```

Classic Load Balancer のアベイラビリティーゾーンを変更するには

1. ロードバランサーのアベイラビリティーゾーンを変更する前に、まず Auto Scaling グループのアベイラビリティーゾーンを更新して、使用しているインスタンスタイプが指定したゾーンで使用できることを確認することをお勧めします。

   Auto Scaling グループのアベイラビリティーゾーンを更新するには、次のupdate-auto-scaling-groupコマンドを使用します。サンプルサブネットを、有効にするアベイラビリティーゾーン内のサブネットIDsの IDsに置き換えます。指定したサブネットは、以前に有効であったサブネットに置き換わります。*my-asg* をグループの名前に置き換えます。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --vpc-zone-identifier "subnet-41767929,subnet-cb663da2"
```

2. 次のdescribe-auto-scaling-groupsコマンドを使用して、新しいサブネットのインスタンスが起動したことを確認します。インスタンスが起動した場合は、インスタンスとそのステータスのリストが表示されます。*my-asg* をグループの名前に置き換えます。

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. 次の attach-load-balancer-to-subnets コマンドを使用して、Classic Load Balancer の新しいアベイラビリティーゾーンを有効にします。サンプルのサブネット ID を、有効にするアベイラビリティーゾーンのサブネットの ID に置き換えます。*my-lb* を、使用しているロードバランサーの名前に置き換えます。

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb \
```

```
  --subnets subnet-cb663da2
```

アベイラビリティーゾーンを無効にするには、次の detach-load-balancer-from-subnets コマンドを使用します。サンプルのサブネット ID を、無効にするアベイラビリティーゾーンのサブネットの ID に置き換えます。*my-lb* を、使用しているロードバランサーの名前に置き換えます。

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-lb \
  --subnets subnet-8360a9e7
```

ターゲットグループまたは Classic Load Balancer をデタッチする

次のdetach-traffic-sourcesコマンドは、不要になったターゲットグループを Auto Scaling グループからデタッチします。

`--auto-scaling-group-name` オプションの場合、*my-asg* を使用しているグループの名前に置き換えます。`--traffic-sources` オプションでは、サンプルを Application Load Balancer、Network Load Balancer、または Gateway Load Balancer ARNARNのターゲットグループの に置き換えます。

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/1234567890123456"
```

Classic Load Balancer をグループからデタッチするには、次の例のように `--traffic-sources` オプションと `--type` オプションを指定します。*my-classic-load-balancer* を Classic Load Balancer の名前に置き換えます。`--type` オプションの場合、**elb** の値を指定します。

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

Elastic Load Balancing のヘルスチェックを削除する

Auto Scaling グループから Elastic Load Balancing ヘルスチェックを削除するには、次のupdate-auto-scaling-groupコマンドを使用して、 `--health-check-type` オプションの値**EC2**として を指定します。*my-asg* をグループの名前に置き換えます。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
```

```
  --health-check-type "EC2"
```

詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

レガシーコマンド

次の例は、レガシーCLIコマンドを使用してロードバランサーとターゲットグループをアタッチ、デタッチ、および記述する方法を示しています。これらは、お客様が使用する際の参照用として、このドキュメントに残してあります。レガシーCLIコマンドは引き続きサポートされますが、複数のトラフィックソースタイプをアタッチおよびデタッチできる新しい「トラフィックソースCLI」コマンドを使用することをお勧めします。同じ Auto Scaling グループで、レガシーCLIコマンドと「トラフィックソースCLI」コマンドの両方を使用できます。

ターゲットグループまたは Classic Load Balancer (レガシー) をアタッチする

ターゲットグループをアタッチするには

次のcreate-auto-scaling-groupコマンドは、ターゲットグループがアタッチされた Auto Scaling グループを作成します。Application Load Balancer、Network Load Balancer、または Gateway Load Balancer Load Balancer のターゲットグループの Amazon リソースネーム (ARN) を指定します。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-
targets/1234567890123456" \
  --min-size 1 --max-size 5
```

次の attach-load-balancer-target-groups コマンドは、ターゲットグループを既存の Auto Scaling グループにアタッチします。

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-
targets/1234567890123456"
```

Classic Load Balancer をアタッチするには

次のcreate-auto-scaling-groupコマンドは、Classic Load Balancer がアタッチされた Auto Scaling グループを作成します。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --load-balancer-names "my-load-balancer" \
  --min-size 1 --max-size 5
```

次のattach-load-balancersコマンドは、指定された Classic Load Balancer を既存の Auto Scaling グループにアタッチします。

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg \
  --load-balancer-names my-lb
```

ターゲットグループまたは Classic Load Balancer (レガシー) の説明を表示する

ターゲットグループの説明を表示するには

Auto Scaling グループに関連付けられているターゲットグループを記述するには、describe-load-balancer-target-groups コマンドを使用します。次の例では、 のターゲットグループを一覧表示しますmy-asg。

```
aws autoscaling describe-load-balancer-target-groups --auto-scaling-group-name my-asg
```

Classic Load Balancer の説明を表示するには

Auto Scaling グループに関連付けられている Classic Load Balancer を記述するには、 describe-load-balancers コマンドを使用します。次の例では、 の Classic Load Balancer を一覧表示しますmy-asg。

```
aws autoscaling describe-load-balancers --auto-scaling-group-name my-asg
```

ターゲットグループまたは Classic Load Balancer (レガシー) をデタッチする

ターゲットグループをデタッチするには

次の detach-load-balancer-target-groups コマンドは、不要になったターゲットグループを Auto Scaling グループからデタッチします。

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-name my-asg \
```

```
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-
targets/1234567890123456"
```

Classic Load Balancer をデタッチするには

次のdetach-load-balancersコマンドは、不要になった Classic Load Balancer を Auto Scaling グループからデタッチします。

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg \
  --load-balancer-names my-lb
```

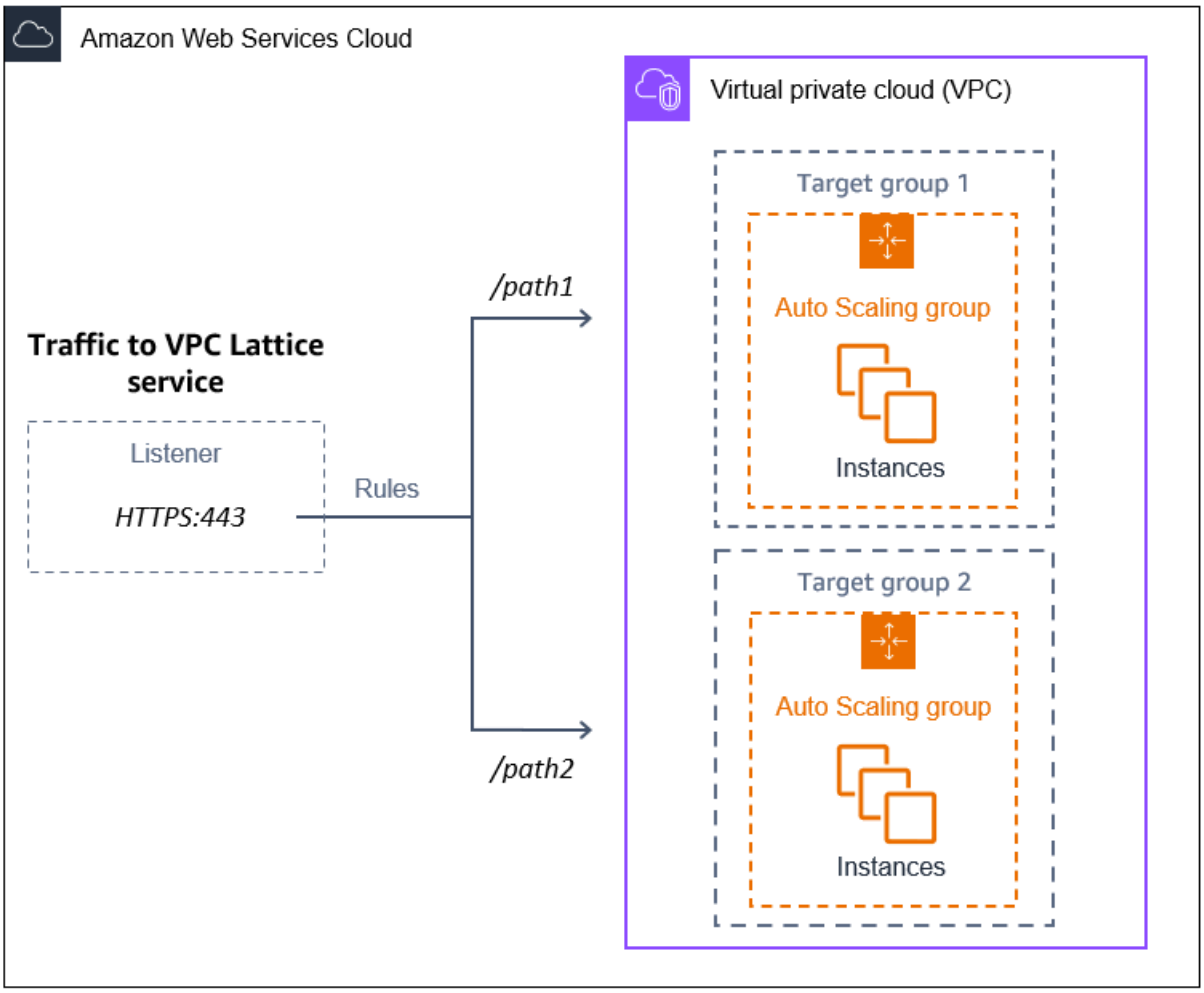
VPC Lattice ターゲットグループを使用してトラフィックフローを管理する

Amazon VPC Lattice を使用すると、Auto Scaling グループや Lambda 関数など、個別のリソースで実行されるアプリケーションとサービス間のトラフィックおよび API コールのフローを管理できます。VPC Lattice は、複数のアカウントおよび仮想プライベートクラウド (VPC) にわたって、すべてのサービスを接続、保護、およびモニタリングできるアプリケーションネットワーキングサービスです。VPC Lattice の詳細については、「VPC Lattice とは?」を参照してください。

VPC Lattice の使用を開始するには、まず、サービスネットワークに関連付けられた VPC 内のリソースが互いに接続できるようにするために必要な VPC Lattice リソースを作成します。これらのリソースには、サービス、リスナー、リスナールール、およびターゲットグループが含まれます。

Auto Scaling グループを VPC Lattice サービスに関連付けるには、インスタンス ID で登録したインスタンスへのリクエストをルーティングするサービスのターゲットグループを作成し、リクエストをターゲットグループに送信するリスナーを追加します。次に、ターゲットグループを Auto Scaling グループにアタッチします。Amazon EC2 Auto Scaling は、EC2 インスタンスをターゲットとしてターゲットグループに自動的に登録します。その後、Amazon EC2 Auto Scaling がインスタンスを終了する必要がある場合、終了前にターゲットグループからインスタンスを自動的に登録解除します。

ターゲットグループをアタッチすると、そのグループが Auto Scaling グループへのすべての受信リクエストのエントリポイントになります。続いて、下図の例が示すように、VPC Lattice サービスで指定したリスナールールを使用して、受信リクエストを適切なターゲットグループにルーティングします。



トラフィックが VPC Lattice 経由で Auto Scaling グループにルーティングされると、VPC Lattice はラウンドロビン負荷分散を使用してグループ内のインスタンス間でリクエストの負荷を分散させます。また、VPC Lattice は登録済みのインスタンスの正常性をモニタリングし、トラフィックを正常なインスタンスにのみルーティングすることができます。

受信リクエストに対してインスタンスを使用できるようにしておくために、必要に応じて VPC Lattice ヘルスチェックを Auto Scaling グループに追加します。このようにしておくと、いずれかの EC2 インスタンスに障害が発生した場合でも、Auto Scaling グループで新しいインスタンスが起動され、自動的に置き換えられます。VPC Lattice ヘルスチェックの動作は、Elastic Load Balancing ヘルスチェックの動作と似ています。Auto Scaling グループのデフォルトのヘルスチェックは EC2 ヘルスチェックのみです。

VPC Lattice の詳細については、AWS ブログの「Amazon VPC Lattice でサービス間の接続、セキュリティ、モニタリングを簡素化 – 一般提供開始」を参照してください。

内容

- Auto Scaling グループに VPC Lattice ターゲットグループをアタッチする準備を行う
- VPC Lattice ターゲットグループを Auto Scaling グループにアタッチする
- VPC Lattice ターゲットグループのアタッチメントステータスを確認する

Auto Scaling グループに VPC Lattice ターゲットグループをアタッチする準備を行う

Auto Scaling グループに VPC Lattice ターゲットグループをアタッチする前に、以下の前提条件を満たす必要があります。

- VPC Lattice サービスネットワーク、サービス、リスナー、およびターゲットグループの作成を既に行っている必要があります。詳細については、「Amazon Lattice ユーザーガイド」の次のトピックを参照してください。
  - サービスネットワーク
  - サービス
  - リスナー
  - ターゲットグループ
- ターゲットグループは、Auto Scaling グループと同じ AWS アカウント、VPC、およびリージョンに存在する必要があります。
- ターゲットグループは、`instance` のターゲットタイプを指定する必要があります。Auto Scaling グループを使用する場合、`ip` のターゲットタイプを指定することはできません。
- ターゲットグループを Auto Scaling グループにアタッチするには、十分な IAM アクセス許可が付与されている必要があります。次のポリシー例は、ターゲットグループをアタッチおよびデタッチするために必要な最小限のアクセス許可を示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:AttachTrafficSources",
                "autoscaling:DetachTrafficSources",
                "autoscaling:DescribeTrafficSources",
                "vpc-lattice:RegisterTargets",
                "vpc-lattice:DeregisterTargets"
```

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

- Auto Scaling グループの起動テンプレートに VPC Lattice の正しい設定 (互換性のあるセキュリティグループなど) が含まれていない場合は、起動テンプレートを更新する必要があります。起動テンプレートが変更されても、既存のインスタンスは新しい設定に更新されません。既存のインスタンスを更新するには、インスタンスの更新を開始してインスタンスを置き換えることができます。詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。
- Auto Scaling グループで VPC Lattice ヘルスチェックを有効にする前に、アプリケーションベースのヘルスチェックを設定して、アプリケーションが期待どおりに応答していることを確認できます。詳細については、「VPC Lattice ユーザーガイド」の「ターゲットグループのヘルスチェック」を参照してください。

セキュリティグループ: インバウンドルールとアウトバウンドルール

セキュリティグループは、関連付けられた EC2 インスタンスのファイアウォールとして機能し、インバウンドトラフィックとアウトバウンドトラフィックの両方をインスタンスレベルでコントロールします。

> **Note**
>
> ネットワーク設定は非常に複雑であるため、VPC Lattice で使用するセキュリティグループを新たに作成することを強くお勧めします。そうすることで、AWS Support に問い合わせる際にも、サポートを受けやすくなります。次のセクションは、ユーザーがこの推奨事項に従うことを前提とした内容になっています。
>
> Auto Scaling グループで使用できる VPC Lattice のセキュリティグループを作成する方法の詳細については、「VPC Lattice ユーザーガイド」の「セキュリティグループを使用してトラフィックを制御する」を参照してください。トラフィックフローの問題をトラブルシューティングするには、「VPC Lattice ユーザーガイド」で詳細を確認してください。

セキュリティグループの作成方法の詳細については、「Amazon EC2 ユーザーガイド」の「セキュリティグループの作成」を参照し、次の表を使用して選択するオプションを決定してください。

オプション	値
名前	覚えやすい名前。
説明	セキュリティグループの識別に役立つ説明。
VPC	Auto Scaling グループと同じ VPC。

インバウンドルール

セキュリティグループを作成するときには、インバウンドルールはありません。インバウンドルールをセキュリティグループに追加するまで、VPC Lattice サービスネットワーク内のクライアントからインスタンスに送信されるインバウンドトラフィックは許可されません。

VPC Lattice サービスネットワーク内のクライアントが Auto Scaling グループ内のインスタンスに接続できるようにするには、Auto Scaling グループのセキュリティグループを正しく設定する必要があります。この場合、特定の IP アドレスではなく、VPC Lattice の AWS マネージドプレフィックスリストの名前からのトラフィックを許可するインバウンドルールを指定します。VPC Lattice のプレフィックスリストは、VPC Lattice が使用する IP アドレスの範囲を CIDR 表記で表したものです。詳細については、「Amazon VPC ユーザーガイド」の「AWS マネージドプレフィックスリストの操作」を参照してください。

セキュリティグループにルールを追加する方法の詳細については、「Amazon VPC ユーザーガイド」の「ルールをセキュリティグループに追加する」を参照し、次の表を使用して、選択するオプションを決定してください。

オプション	値
HTTP ルール	Type: HTTP ソース: com.amazonaws.*region*.vpc-lattice
HTTPS ルール	タイプ: HTTPS

オプション	値	
	ソース: com.amazonaws.*region*.vpc-lattice	

セキュリティグループはステートフルです。VPC Lattice サービスネットワーク内のクライアントから Auto Scaling グループ内のインスタンスへのトラフィックを許可し、以前に離れたクライアントに応答を返します。

アウトバウンドルール

デフォルトでは、セキュリティグループにはすべてのアウトバウンドトラフィックを許可するアウトバウンドルールが含まれています。必要に応じて、このデフォルトルールを削除し、特定のセキュリティニーズに対応するアウトバウンドルールを追加できます。

制約事項

- 混合インスタンスグループはサポートされていません。混合インスタンスポリシーが適用されている Auto Scaling グループに VPC Lattice ターゲットグループをアタッチしようとすると、エラーメッセージ「現在、混合インスタンスを含む Auto Scaling グループは VPC Lattice サービスと統合できません」が表示されます。これは、負荷分散アルゴリズムにより、利用可能なすべてのリソースに負荷が均等に分散されたことで、インスタンス同士が同じ負荷を処理できるほどに類似していると想定されたためです。

VPC Lattice ターゲットグループを Auto Scaling グループにアタッチする

このトピックでは、VPC Lattice ターゲットグループを Auto Scaling グループにアタッチする方法について説明します。また、VPC Lattice ヘルスチェックを有効にして、VPC Lattice から異常として報告されたインスタンスを Amazon EC2 Auto Scaling で置き換える方法についても説明します。

デフォルトでは、Amazon EC2 Auto Scaling は、Amazon EC2 ヘルスチェックに基づいて、異常なインスタンスまたはアクセスできないインスタンスのみを置き換えます。VPC Lattice ヘルスチェックを有効にすると、Auto Scaling グループにアタッチした VPC Lattice ターゲットグループのいずれかから、実行中のインスタンスが異常として報告された場合、Amazon EC2 Auto Scaling は、そのインスタンスを置き換えることができます。詳細については、「Auto Scaling グループでのインスタンスのヘルスチェック」を参照してください。

> **Important**
>
> 先に進む前に、前のセクションのすべての前提条件を満たしてください。

VPC Lattice ターゲットグループをアタッチする

グループを作成または更新するときに、1 つ以上のターゲットグループを Auto Scaling グループにアタッチできます。

Console

このセクションの手順に従い、コンソールを使用して次の操作を実行します。

- VPC Lattice ターゲットグループを Auto Scaling グループにアタッチする
- VPC Lattice のヘルスチェックを有効にする

VPC Lattice ターゲットグループを新しい Auto Scaling グループにアタッチするには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 画面の上部のナビゲーションバーで、ターゲットグループを作成した AWS リージョン を選択します。
3. [Auto Scaling グループの作成] を選択します。
4. ステップ 1 と 2 で、希望するオプションを選択し、「ステップ 3: 詳細オプションを設定する」へ進みます。
5. [VPC Lattice 統合オプション] で、[VPC Lattice サービスへアタッチ] を選択します。
6. [VPC Lattice ターゲットグループを選択] で、ターゲットグループを選択します。
7. (オプション) [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[VPC Lattice ヘルスチェックを有効にする] を選択します。
8. (オプション) [ヘルスチェックの猶予期間] に秒単位で時間を入力します。これは、インスタンスが `InService` 状態になった後で、Amazon EC2 Auto Scaling がインスタンスのヘルスステータスのチェックを待つ必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。
9. Auto Scaling グループの作成に進みます。Auto Scaling グループの作成後、インスタンスは自動的に VPC Lattice ターゲットグループに登録されます。

VPC Lattice ターゲットグループを既存の Auto Scaling グループにアタッチするには

次の手順に従って、ターゲットグループを既存の Auto Scaling グループにアタッチします。

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. Auto Scaling グループの横にあるチェックボックスを選択します。

   ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[VPC Lattice 統合オプション]、[編集] を順に選択します。
4. [VPC Lattice 統合オプション] で、[VPC Lattice サービスへアタッチ] を選択します。
5. [VPC Lattice ターゲットグループを選択] で、ターゲットグループを選択します。
6. [Update] (更新) を選択します。

ターゲットグループのアタッチが完了したら、必要に応じて、そのターゲットグループで使用するヘルスチェックを有効にできます。

VPC Lattice ヘルスチェックを有効にするには

1. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
2. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[VPC Lattice ヘルスチェックを有効にする] を選択します。
3. [ヘルスチェックの猶予期間] に、秒単位で時間を入力します。これは、インスタンスが `InService` 状態になった後で、Amazon EC2 Auto Scaling がインスタンスのヘルスステータスのチェックを待つ必要がある時間です。詳細については、「Auto Scaling グループにヘルスチェックの猶予期間を設定する」を参照してください。
4. [Update] (更新) を選択します。

AWS CLI

このセクションの手順に従い、AWS CLI を使用して次の操作を実行します。

- VPC Lattice ターゲットグループを Auto Scaling グループにアタッチする
- VPC Lattice のヘルスチェックを有効にする

VPC Lattice ターゲットグループを Auto Scaling グループにアタッチするには

次の create-auto-scaling-group コマンドを使用して Auto Scaling グループを作成し、同時に、そのグループの Amazon リソースネーム (ARN) を指定して VPC Lattice ターゲットグループをアタッチします。

--auto-scaling-group-name、--vpc-zone-identifier、--min-size、および --max-size のサンプルの値を置き換えます。--launch-template オプションの場合、*my-launch-template* と *1* を、VPC Lattice ターゲットグループに登録したインスタンス用に作成した起動テンプレートの名前とバージョンに置き換えます。--traffic-sources オプションの場合、サンプル ARN を VPC Lattice ターゲットグループの ARN に置き換えます。

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

VPC Lattice ターゲットグループを既に作成していたら、次の attach-traffic-sources コマンドを使用して、そのターゲットグループを Auto Scaling グループにアタッチします。

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

VPC Lattice のヘルスチェックを有効にするには

VPC Lattice ターゲットグループにアプリケーションベースのヘルスチェックを設定している場合は、これらのヘルスチェックを有効にできます。--health-check-type オプションと **VPC_LATTICE** の値を指定した create-auto-scaling-group または update-auto-scaling-group コマンドを使用します。Auto Scaling グループで実行するヘルスチェックの猶予期間を指定する場合は、--health-check-grace-period オプションを含め、その値を秒単位で指定します。

```
--health-check-type "VPC_LATTICE" --health-check-grace-period 60
```

VPC Lattice ターゲットグループをデタッチする

VPC Lattice が不要になった場合、次の手順に従って、Auto Scaling グループからターゲットグループをデタッチします。

Console

このセクションの手順に従い、コンソールを使用して次の操作を実行します。

- Auto Scaling グループから VPC Lattice ターゲットグループをデタッチする
- VPC Lattice のヘルスチェックを無効にする

Auto Scaling グループから VPC Lattice ターゲットグループをデタッチするには

1. https://console.aws.amazon.com/ec2/ でAmazon EC2 コンソールを開き、ナビゲーションペインで [Auto Scaling グループ] を選択します。
2. 既存のグループの横にあるチェックボックスをオンにします。

   ページの下部にスプリットペインが開きます。
3. [詳細] タブで、[VPC Lattice 統合オプション]、[編集] を順に選択します。
4. [VPC Lattice 統合オプション] で、ターゲットグループの横にある [削除] アイコン (X) を選択します。
5. [Update] (更新) を選択します。

ターゲットグループのデタッチが完了したら、VPC Lattice ヘルスチェックを無効にできます。

VPC Lattice ヘルスチェックを無効にするには

1. [詳細] タブで、[ヘルスチェック]、[編集] の順に選択します。
2. [ヘルスチェック] の [追加のヘルスチェックタイプ] で、[VPC Lattice ヘルスチェックを有効にする] の選択を解除します。
3. [Update] (更新) を選択します。

AWS CLI

このセクションの手順に従い、AWS CLI を使用して次の操作を実行します。

- Auto Scaling グループから VPC Lattice ターゲットグループをデタッチする
- VPC Lattice のヘルスチェックを無効にする

ターゲットグループが不要になったら、detach-traffic-sources コマンドを使用して、Auto Scaling グループからそのターゲットグループをデタッチします。

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

Auto Scaling グループのヘルスチェックを更新して VPC Lattice ヘルスチェックを使用しないようにするには、update-auto-scaling-group コマンドを使用します。`--health-check-type` オプションと **EC2** の値を含めます。

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

VPC Lattice ターゲットグループのアタッチメントステータスを確認する

VPC Lattice ターゲットグループを Auto Scaling グループにアタッチすると、グループ内のインスタンスの登録中、`Adding` 状態になります。グループのすべてのインスタンスが登録済みになると、`Added` 状態になります。最低 1 つの登録されたインスタンスがヘルスチェックを通過した後、`InService` 状態になります。ターゲットグループが `InService` 状態にある場合、Amazon EC2 Auto Scaling は異常として報告されたインスタンスを終了し、置き換えることができます。登録したインスタンスがどれもヘルスチェックを通過しなかった場合 (ヘルスチェックの設定ミスなどによる)、ターゲットグループは `InService` 状態になりません。Amazon EC2 Auto Scaling はインスタンスを終了し置き換えをすることはありません。

サービスのターゲットグループをデタッチすると、グループ内のインスタンスの登録解除中、`Removing` 状態になります。登録解除してもインスタンスは引き続き実行されます。デフォルトでは、Connection Draining (登録解除の遅延) が有効になっています。Connection Draining が有効になっている場合、VPC Lattice は、処理中のリクエストが完了するまで、または最大タイムアウト時間が終了するまで (どちらか早い方) 待機してから、インスタンスの登録を解除します。

アタッチメントステータスは、AWS Command Line Interface (AWS CLI) または AWS SDK を使用して確認できます。アタッチメントステータスはコンソールから確認できません。

AWS CLI を使用してアタッチメントステータスを確認するには

次の describe-traffic-sources コマンドは、指定した Auto Scaling グループのすべてのトラフィックソースのアタッチメントステータスを返します。

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

この例では、Auto Scaling グループにアタッチされている VPC Lattice ターゲットグループの ARN と、State 要素内のターゲットグループのアタッチメントステータスを返します。

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:vpc-lattice:region:account-
id:targetgroup/tg-0e2f2665eEXAMPLE",
            "State": "InService",
            "Type": "vpc-lattice"
        }
    ]
}
```

Auto Scaling イベントの処理に EventBridge を使用する

Amazon EventBridge は、以前はCloudWatch Eventsと呼ばれていましたが、リソースをモニタリングし、他の AWS のサービスを使用するターゲットアクションを起動するための、イベント駆動型のルールを設定に役立ちます。

Amazon EC2 Auto Scaling からのイベントは、ほぼリアルタイムに EventBridge に提供されます。これらのさまざまなイベントへの対応でプログラム的なアクションや通知を呼び出す EventBridge ルールを設定できます。例えば、インスタンスの起動または終了プロセス中に AWS Lambda 関数を呼び出して、事前設定されたタスクを実行することができます。

EventBridge ルールのターゲットは、AWS Lambda 関数、Amazon SNS トピック、API 送信先、他の AWS アカウント 内のイベントバスなど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 Amazon EventBridge ターゲット を参照してください 。

Amazon SNS トピックと EventBridge ルールを使用する例を用いて EventBridge ルールを作成することで開始します。その後、ユーザーがインスタンスの更新を開始すると、チェックポイントに到達するたびに Amazon SNS が E メールで通知します。詳細については、「インスタンスの更新イベント用の EventBridge ルールを作成する」を参照してください。

内容

- Amazon EC2 Auto Scaling イベントリファレンス
- ウォームプールのイベントとパターンの例

- Amazon EventBridge ルールを使用してアクションを自動化する

Amazon EC2 Auto Scaling イベントリファレンス

Amazon EventBridge を使用すると、受信イベントを照合し、処理のためにターゲットにルーティングするルールを作成できます。

内容

ライフサイクルアクションイベント

Auto Scaling グループにライフサイクルフックを追加すると、インスタンスが待機状態に移行するときに、Amazon EC2 Auto Scaling が EventBridge にイベントを送信します。イベントは、ベストエフォートベースで生成されます。

イベントタイプ

スケールアウトライフサイクルアクション

次のイベント例は、起動ライフサイクルフックを理由として、Amazon EC2 Auto Scaling がインスタンスを `Pending:Wait` 状態に移行したことを示しています。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
```

```
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

スケールインライフサイクルアクション

次のイベント例は、終了ライフサイクルフックを理由として、Amazon EC2 Auto Scaling がインスタンスを `Terminating:Wait` 状態に移行したことを示しています。

> **Important**
>
> Auto Scaling グループがスケールイン時にインスタンスをウォームプールに返す場合、インスタンスをウォームプールに返すことによって `EC2 Instance-terminate Lifecycle Action` イベントが生成される場合もあります。インスタンスがスケールインで待機状態に移行したときに配信されるイベントには、`Destination` の値として `WarmPool` が含まれます。詳細については、「Instance reuse policy」を参照してください。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken":"87654321-4321-4321-4321-210987654321",
```

```
    "AutoScalingGroupName":"my-asg",
    "LifecycleHookName":"my-lifecycle-hook",
    "EC2InstanceId":"i-1234567890abcdef0",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata":"additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

成功したスケーリングイベント

次の例は、成功したスケーリングイベントのイベントタイプを示しています。イベントは、ベストエフォートベースで生成されます。

イベントタイプ

- 成功したスケールアウトイベント
- 成功したスケールインイベント

成功したスケールアウトイベント

次のイベント例は、Amazon EC2 Auto Scaling がインスタンスを正常に起動したことを示しています。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Launching a new EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
```

```
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

成功したスケールインイベント

次のイベント例は、Amazon EC2 Auto Scaling がインスタンスを正常に終了したことを示しています。

> **Important**
>
> Auto Scaling グループがスケールイン時にインスタンスをウォームプールに返す場合、インスタンスをウォームプールに返すことによって EC2 Instance Terminate Successful イベントが生成される場合もあります。インスタンスがウォームプールに正常に戻ったときに配信されるイベントには、Destination の値として WarmPool が含まれます。詳細については、「Instance reuse policy」を参照してください。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
```

```
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Terminating EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

失敗したスケーリングイベント

次の例は、失敗したスケーリングイベントのイベントタイプを示しています。イベントは、ベストエフォートベースで生成されます。

イベントタイプ

- 失敗したスケールアウトイベント
- 失敗したスケールインイベント

失敗したスケールアウトイベント

次のイベント例は、Amazon EC2 Auto Scaling がインスタンスの起動に失敗したことを示しています。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
```

```
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

失敗したスケールインイベント

次のイベント例は、Amazon EC2 Auto Scaling がインスタンスの終了に失敗したことを示しています。

> **Important**
>
> Auto Scaling グループがスケールイン時にインスタンスをウォームプールに返すときに、インスタンスをウォームプールに返すことに失敗すると、EC2 Instance Terminate Unsuccessful イベントが生成される場合もあります。インスタンスがウォームプールに戻るのに失敗した場合に配信されるイベントには、Destination の値として WarmPool が含まれます。詳細については、「Instance reuse policy」を参照してください。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
```

```
  "detail-type": "EC2 Instance Terminate Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

インスタンス更新イベント

次の例は、インスタンス更新機能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。

イベントタイプ

- チェックポイントに達しました
- インスタンスの更新が開始されました
- インスタンスの更新が成功しました
- インスタンスの更新に失敗しました
- インスタンスの更新がキャンセルされました
- インスタンス更新のロールバックが開始されました

- インスタンスの更新のロールバックが成功しました
- インスタンスの更新のロールバックが失敗しました

チェックポイントに達しました

置き換えられたインスタンスの数が、チェックポイントに定義された割合のしきい値に達すると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
    "AutoScalingGroupName": "my-asg",
    "CheckpointPercentage": "50",
    "CheckpointDelay": "300"
  }
}
```

インスタンスの更新が開始されました

インスタンスの更新のステータスが `InProgress` に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
```

```
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

インスタンスの更新が成功しました

インスタンスの更新のステータスが Successful に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

インスタンスの更新に失敗しました

インスタンスの更新のステータスが Failed に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
```

```
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

インスタンスの更新がキャンセルされました

インスタンスの更新のステータスが Cancelled に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Cancelled",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

インスタンス更新のロールバックが開始されました

インスタンスの更新のステータスが RollbackInProgress に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Started",
  "source": "aws.autoscaling",
```

```
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

インスタンスの更新のロールバックが成功しました

インスタンスの更新のステータスが RollbackSuccessful に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

インスタンスの更新のロールバックが失敗しました

インスタンスの更新のステータスが Failed に変わると、Amazon EC2 Auto Scaling は次のイベントを送信します。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
```

```
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

ウォームプールのイベントとパターンの例

Amazon EC2 Auto Scaling は、Amazon EventBridge の事前定義されたパターンを複数サポートしています。これは、イベントパターンが作成される方法を簡素化します。フォーム上のフィールド値を選択すると、EventBridge がパターンを生成します。現在、Amazon EC2 Auto Scaling は、ウォームプールがある Auto Scaling グループによって発行されるイベントに対して事前定義されたパターンをサポートしていません。パターンは、JSON オブジェクトとして入力する必要があります。このセクションと「ウォームプールイベント向けの EventBridge ルールを作成する」トピックでは、イベントパターンを使用してイベントを選択し、それらをターゲットに送信する方法を説明します。

Amazon EC2 Auto Scaling が EventBridge に送信するウォームプール関連のイベントをフィルタリングする EventBridge ルールを作成するには、イベントの `detail` セクションの `Origin` および `Destination` フィールドを含めます。

`Origin`および`Destination` の値には以下を指定できます。

`EC2` | `AutoScalingGroup` | `WarmPool`

内容

イベントの例

Auto Scaling グループにライフサイクルフックを追加すると、インスタンスが待機状態に移行するときに、Amazon EC2 Auto Scaling が EventBridge にイベントを送信します。詳細については、

「Auto Scaling グループのウォームプールでライフサイクルフックを使用する」を参照してください。

このセクションには、Auto Scaling グループにウォームプールがある場合のこれらのイベントの例が含まれています。イベントは、ベストエフォートベースで出力されます。

> **Note**
>
> スケーリングが成功したときに Amazon EC2 Auto Scaling が EventBridge に送信するイベントについては、「成功したスケーリングイベント」を参照してください。スケーリングが失敗した場合のイベントについては、「失敗したスケーリングイベント」を参照してください。

イベント例

- スケールアウトライフサイクルアクション
- スケールインライフサイクルアクション

スケールアウトライフサイクルアクション

インスタンスがスケールアウトイベントの待機状態に移行するときに配信されるイベントには、`detail-type` の値として `EC2 Instance-launch Lifecycle Action` が含まれます。`detail` オブジェクト内の `Origin` および `Destination` 属性の値は、インスタンスがどこから来てどこへ行くのかを示します。

このスケールアウトイベントの例では、新しいインスタンスが起動し、ウォームプールに追加されるため、その状態が `Warmed:Pending:Wait` に変わります。詳細については、「ウォームプール内のインスタンスのライフサイクル状態の移行」を参照してください。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
```

```
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "WarmPool"
  }
}
```

このスケールアウトイベントの例では、インスタンスがウォームプールから Auto Scaling グループに追加されるため、インスタンスの状態は Pending:Wait に変わります。詳細については、「ウォームプール内のインスタンスのライフサイクル状態の移行」を参照してください。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-19T00:35:52.359Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "WarmPool",
    "Destination": "AutoScalingGroup"
  }
}
```

スケールインライフサイクルアクション

インスタンスがスケールインイベントの待機状態に移行するときに配信されるイベントには、detail-type の値として EC2 Instance-terminate Lifecycle Action が含まれま

す。detail オブジェクト内の Origin および Destination 属性の値は、インスタンスがどこから来てどこへ行くのかを示します。

このスケールインイベントの例では、インスタンスがウォームプールに戻されるため、インスタンスの状態は Warmed:Pending:Wait に変わります。詳細については、「ウォームプール内のインスタンスのライフサイクル状態の移行」を参照してください。

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2022-03-28T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "42694b3d-4b70-6a62-8523-09a1eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-termination-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "WarmPool"
  }
}
```

イベントパターンの例

前のセクションでは、Amazon EC2 Auto Scaling によって生成されるイベント例を説明します。

EventBridge イベントパターンは、一致するイベントと同じ構造をしています。イベントのパターンでは、照合する対象のフィールドを引用符で囲み、検出したい値を指定します。

イベント内の次のフィールドは、ルールで定義され、アクションをトリガーするイベントパターンを形成します。

`"source": "aws.autoscaling"`

イベントが Amazon EC2 Auto Scaling Group からのものであることを特定します。

`"detail-type": "EC2 Instance-launch Lifecycle Action"`

イベントタイプを特定します。

`"Origin": "EC2"`

インスタンスがどこから来ているかを識別します。

`"Destination": "WarmPool"`

インスタンスがどこに行くかを識別します。

次のサンプルイベントパターンを使用して、ウォームプールに入るインスタンスに関連付けられているすべての EC2 Instance-launch Lifecycle Action イベントをキャプチャします。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
   }
}
```

次のサンプルイベントパターンを使用して、スケールアウトイベントのためにウォームプールから取り出されるインスタンスに関連付けられたすべての EC2 Instance-launch Lifecycle Action イベントをキャプチャします。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "WarmPool" ],
      "Destination": [ "AutoScalingGroup" ]
   }
}
```

次のサンプルイベントパターンを使用して、Auto Scaling グループに直接起動するインスタンスに関連付けられているすべての EC2 Instance-launch Lifecycle Action イベントをキャプチャします。

```
{
```

```
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "AutoScalingGroup" ]
   }
}
```

次のサンプルイベントパターンを使用して、スケールインでウォームプールに戻されるインスタンスに関連付けられているすべての EC2 Instance-terminate Lifecycle Action イベントをキャプチャします。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],
  "detail": {
      "Origin": [ "AutoScalingGroup" ],
      "Destination": [ "WarmPool" ]
   }
}
```

次のイベントパターンのサンプルを使用して、送信元や送信先に関わらず、EC2 Instance-launch Lifecycle Actionに関連付けられているすべてのイベントをキャプチャします。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

Amazon EventBridge ルールを使用してアクションを自動化する

Amazon EC2 Auto Scaling がイベントを発行すると、イベント通知が JSON ファイルとして Amazon EventBridge に送信されます。EventBridge ルールを作成し、イベントパターンがルールに一致したときに実行するアクションを自動化できます。EventBridge がルールで定義されているパターンに一致するイベントパターンを検出すると、EventBridge はルールで指定されているターゲットを呼び出します。

このセクションの手順例を開始点として使用できます。

以下のドキュメントも役に立つかもしれません。

- Lambda 関数を使用して、インスタンスの起動中、またはインスタンスの終了前にそれらでカスタムアクションを実行するには、「チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定」を参照してください。
- CloudTrail のログに記録された API コールで Lambda 関数を呼び出すには、「Amazon EventBridge ユーザーガイド」の「チュートリアル: EventBridge を使用して、AWS API コールのログを記録する」を参照してください。
- イベントルールの作成方法に関する詳細については、「Amazon EventBridge ユーザーガイド」の「イベントに反応する Amazon EventBridge ルールの作成」を参照してください。

トピック

- インスタンスの更新イベント用の EventBridge ルールを作成する
- ウォームプールイベント向けの EventBridge ルールを作成する

インスタンスの更新イベント用の EventBridge ルールを作成する

以下の例は、E メール通知を送信する EventBridge ルールを作成します。これは、インスタンス更新中、チェックポイントに到達した時に Auto Scaling グループがイベントを発行するたびに実行されます。Amazon SNS を使用して E メール通知を設定する手順も含まれています。Amazon SNS を使用して E メール通知を送信するには、最初にトピックを作成してから、そのトピックと共に E メールアドレスを登録する必要があります。

インスタンスの更新機能に関する詳細については、「インスタンスの更新を使用して Auto Scaling グループのインスタンスを更新する」を参照してください。

Amazon SNS トピックを作成します。

SNS トピックは論理アクセスポイント、つまり Auto Scaling グループ が通知を送信するために使用する通信チャネルです。トピックの名前を指定することにより、トピックを作成します。

トピック名は、以下の要件を満たしている必要があります。

- 1 ~ 256 文字
- 大文字および小文字の ASCII 文字、数字、アンダースコア、またはハイフンが含まれている

詳細については、Amazon Simple 通知サービス デベロッパーガイドの「Amazon SNS トピックの作成」を参照してください。

Amazon SNS トピックを購読します。

Auto Scaling グループがトピックに送信した通知を受信するには、そのトピックにエンドポイントを登録する必要があります。この手順では、エンドポイント に、Amazon EC2 Auto Scaling からの通知を受信する E メールアドレスを指定します。

詳細については、Amazon Simple 通知サービス デベロッパーガイドの「Amazon SNS トピックへのサブスクライブ」を参照してください。

Amazon SNS サブスクリプションを確認する

Amazon SNS は、前のステップで指定した E メールアドレスに確認メールを送信します。

次のステップに進む前に、必ずAWS 通知の E メールを開き、リンクを選択して受信登録を確認してください。

AWSからの確認メッセージが届きます。Amazon SNS は、通知を受信し、通知を E メールとして指定された E メールアドレスに送信するように設定されました。

Amazon SNS トピックにイベントをルートする

選択したイベントに一致するルールを作成し、それらを Amazon SNS トピックにルーティングして、購読済みの E メールアドレスに通知します。

Amazon SNS トピックに通知を送信するルールを作成する

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで ルール] を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. [Define rule detail] (詳細の定義) で、次の操作を行います。

    a. ルールの [Name (名前)] を入力し、必要に応じて説明を入力します。

        ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

    b. [イベントバス] として、[デフォルト] を選択します。アカウント内の AWS のサービスで生成されたイベントは、常に、そのアカウントのデフォルトのイベントバスに送られます。

    c. ルールタイプ では、イベントパターンを持つルール] を選択します。

    d. [Next] を選択します。

5. [Build event pattern] (イベントパターンの作成) で、次の操作を行います。

   a. [Event source] (イベントソース) で、[AWS events or EventBridge partner events] (イベントまたは EventBridge パートナーイベント) を選択します。

   b. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。

      i. イベントソース で AWS のサービス を選択します。

      ii. [AWS のサービス] には、[Auto Scaling] を選択します。

      iii. [Event type (イベントタイプ)] で、[Instance Refresh (インスタンス更新)] を選択します。

      iv. デフォルトでは、ルールは更新イベントのすべてのインスタンスに一致します。インスタンスの更新中、チェックポイントに到達したときに通知するルールを作成するには、[Specific instance event(s)] (特定のインスタンスイベント) を選択し、[EC2 Auto Scaling Instance Refresh Checkpoint Reached] (EC2 Auto Scaling インスタンス更新チェックポイントに到達) を選択します。

      v. デフォルトでは、このルールはリージョン内のすべての Auto Scaling グループと一致します。ルールを特定の Auto Scaling グループに一致させるには、[Specific group name (特定のグループ名)] を選択して 1 つ以上の Auto Scaling グループを選択します。

      vi. [Next] を選択します。

6. [Select target(s)] (ターゲットを選択) で、以下の操作を行います。

   a. [Target types] (ターゲットタイプ) には、[AWS のサービス] を選択します。

   b. [Select a target] (ターゲットの選択) には、[SNS topic] (SNS トピック) を選択します。

   c. [Topic] (トピック) には、お使いの Amazon SNS トピックを選択します。

   d. (オプション) [Additional settings] (追加設定) で、その他の設定を行うこともできます。EventBridge ルールの作成の詳細については、Amazon EventBridge ユーザーガイドの「イベントに反応する Amazon EventBridge ルールの作成」を参照してください 。

   e. [Next] を選択します。

7. (オプション) 必要な場合は、[Tags] (タグ) で 1 つ以上のタグを作成したルールに割り当て、[Next] (次へ) を選択します。

8. [Review and create] (レビューと作成) で、ルールの詳細を確認し、必要に応じて変更します。その後、[Create rule] (ルールの作成) を選択します。

ウォームプールイベント向けの EventBridge ルールを作成する

以下の例は、プログラム的なアクションを呼び出す EventBridge ルールを作成します。これは、ウォームプールへの新しいインスタンスの追加時に Auto Scaling グループがイベントを発行するたびに実行されます。

ルールを作成する前に、AWS Lambda関数を作成して、ルールがターゲットとして使用されるようにします。この関数をルールのターゲットとして指定する必要があります。以下の手順では、ウォームプールへの新しいインスタンスの追加時にアクションを実行する EventBridge ルールの作成手順のみが説明されています。着信イベントがルールに一致するときに呼び出すシンプルな Lambda 関数の作成方法を説明する初歩的なチュートリアルについては、「チュートリアル:Lambda 関数を呼び出すライフサイクルフックの設定」を参照してください。

ウォームプールの作成と使用に関する詳細については、「ウォームプールを使用してブート時間が長いアプリケーションのレイテンシーを低減する」を参照してください。

Lambda 関数を呼び出すイベントルールを作成する

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで ルール] を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. [Define rule detail] (詳細の定義) で、次の操作を行います。

   a. ルールの [Name (名前)] を入力し、必要に応じて説明を入力します。

      ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

   b. [イベントバス] として、[デフォルト] を選択します。アカウント内の AWS のサービスがイベントを生成すると、イベントは常にアカウントのデフォルトイベントバスに送られます。

   c. ルールタイプ では、イベントパターンを持つルール] を選択します。

   d. [Next] を選択します。
5. [Build event pattern] (イベントパターンの作成) で、次の操作を行います。

   a. [Event source] (イベントソース) で、[AWS events or EventBridge partner events] (イベントまたは EventBridge パートナーイベント) を選択します。

   b. [Event pattern] では、[Custom pattern (JSON editor)] (カスタムパターン (JSON エディタ)) を選択し、以下のパターンを [Event pattern] (イベントパターン) ボックスに貼り付けて、#####のテキストを Auto Scaling グループの名前に置き換えます。

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "AutoScalingGroupName": [ "my-asg" ],
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
   }
}
```

他のイベントに一致するルールを作成するには、イベントパターンを変更します。詳細については、「イベントパターンの例」を参照してください。

c. [Next] を選択します。

6. [Select target(s)] (ターゲットを選択) で、以下の操作を行います。

   a. [Target types] (ターゲットタイプ) には、[AWS のサービス] を選択します。

   b. [Select a target] (ターゲットを選択) では、[Lambda function] (Lambda 関数) を選択します。

   c. [Function] (機能) には、イベントの送信先となる関数を選択します。

   d. (オプション) [Configure version/alias] (バージョン/エイリアスを設定) で、ターゲットの Lambda 関数のバージョンとエイリアスの設定を入力します。

   e. (オプション)[Additional settings] (追加設定) で、アプリケーションに適切な追加設定を入力します。EventBridge ルールの作成の詳細については、Amazon EventBridge ユーザーガイドの「イベントに反応する Amazon EventBridge ルールの作成」を参照してください 。

   f. [Next] を選択します。

7. (オプション) 必要な場合は、[Tags] (タグ) で 1 つ以上のタグを作成したルールに割り当て、[Next] (次へ) を選択します。

8. [Review and create] (レビューと作成) で、ルールの詳細を確認し、必要に応じて変更します。その後、[Create rule] (ルールの作成) を選択します。

Amazon VPC を使用して Auto Scaling インスタンスのネットワーク接続を提供する

Amazon Virtual Private Cloud (Amazon VPC) は、定義した論理的に分離された仮想ネットワークで Auto Scaling グループなどの AWS リソースを起動できるサービスです。

Amazon VPC のサブネットは、VPC の IP アドレス範囲のセグメントで定義されるアベイラビリティーゾーン内のサブディビジョンです。サブネットを使うと、インスタンスをセキュリティや運用上の必要に応じてグループ化することができます。サブネットは作成されたアベイラビリティーゾーン内に完全に包含されるものです。サブネットで Auto Scaling インスタンスを起動します。

インターネットとサブネット内のインスタンス間の通信を有効にするには、インターネットゲートウェイを作成して VPC にアタッチする必要があります。インターネットゲートウェイを使用すると、サブネット内のリソースを Amazon EC2 ネットワークエッジ経由でインターネットに接続できます。サブネットのトラフィックがインターネットゲートウェイにルーティングされる場合、そのサブネットはパブリックサブネットと呼ばれます。サブネットのトラフィックがインターネットゲートウェイにルーティングされない場合、そのサブネットはプライベートサブネットと呼ばれます。インターネットに接続する必要があるリソースにはパブリックサブネットを、インターネットに接続する必要がないリソースにはプライベートサブネットを使用してください。VPC 内のインスタンスへのインターネットアクセスを許可する方法の詳細については、「Amazon VPC ユーザーガイド」の「インターネットへのアクセス」を参照してください。

内容

デフォルトのVPC

2013 年 12 月 4 AWS アカウント 日以降に を作成した場合、または新しい で Auto Scaling グループを作成する場合は AWS リージョン、デフォルトの VPC が作成されます。デフォルトの VPC には、各アベイラビリティーゾーンにデフォルトのサブネットが付属しています。デフォルトの VPC がある場合、Auto Scaling グループはデフォルトでデフォルトの VPC に作成されます。

Amazon VPCs コンソールの VPCs ページで VPC を表示できます。

デフォルトのVPCの詳細については、Amazon VPCs ユーザーガイドの「デフォルトのVPC」を参照してください。

デフォルト以外のVPC

の VPCs Dashboard ページに移動し、VPC の作成 を選択すると、追加の VPC を作成できます。 AWS Management Console

詳細については、「Amazon VPC ユーザーガイド」を参照してください。

> **Note**
>
> VPCは、その中のすべてのアベイラビリティーゾーンにまたがります AWS リージョン。VPC にサブネットを追加するときは、複数のアベイラビリティーゾーンを選択して、それらのサブネットでホストされているアプリケーションが高可用性であることを確認します。アベイラビリティーゾーンは、 AWS リージョンの冗長電源、ネットワーク、および接続を備えた 1 つ以上の個別のデータセンターです。アベイラビリティーゾーンは、本番環境アプリケーションの可用性、耐障害性、およびスケーラビリティを向上させるために役立ちます。

VPC サブネットを選択する際の考慮事項

Auto Scaling グループの VPC サブネットを選択するときは、次の考慮事項に注意してください。

- Elastic Load Balancing ロードバランサーを Auto Scaling グループにアタッチする場合、インスタンスはパブリックサブネットまたはプライベートサブネットで起動できます。ただし、DNS 解決をサポートするには、ロードバランサーをパブリックサブネットに作成する必要があります。
- SSH 経由で Auto Scaling インスタンスに直接アクセスする場合、インスタンスはパブリックサブネットでのみ起動できます。

- AWS Systems Manager Session Manager を使用して no-ingress Auto Scaling インスタンスにアクセスする場合、インスタンスはパブリックサブネットまたはプライベートサブネットのいずれかで起動できます。
- プライベートサブネットを使用している場合は、パブリック NAT ゲートウェイを使用して Auto Scaling インスタンスがインターネットにアクセスすることを許可できます。
- デフォルトでは、デフォルト VPC のデフォルトサブネットはパブリックサブネットです。

VPC での IP アドレス指定

VPC で Auto Scaling インスタンスを起動すると、インスタンスには、インスタンスが起動されるサブネットの CIDR 範囲からプライベート IP アドレスが自動的に割り当てられます。これにより、インスタンスは VPC 内の他のインスタンスと通信できるようになります。

インスタンスにパブリック IPv4 アドレスを割り当てるように起動テンプレートまたは起動設定を設定できます。インスタンスにパブリック IP アドレスを割り当てると、インスタンスはインターネットや他の AWS サービスと通信できます。

IPv6 アドレスを自動的に割り当てるように設定されたサブネットでインスタンスを起動すると、インスタンスは IPv4 アドレスと IPv6 アドレスの両方を受け取ります。それ以外の場合は、IPv4 アドレスのみを受け取ります。詳細については、「Amazon IPv6 ユーザーガイド」の「Word アドレス」を参照してください。 EC2

CIDR またはサブネットの VPC 範囲を指定する方法については、「Amazon VPC ユーザーガイド」を参照してください。

Amazon EC2 Auto Scaling では、追加のネットワークインターフェイスを指定する起動テンプレートを使用すると、インスタンスの起動時に追加のプライベート IP アドレスを自動的に割り当てることができます。各ネットワークインターフェイスには、インスタンスが起動されるサブネットの CIDR 範囲から 1 つのプライベート IP アドレスが割り当てられます。この場合、システムはパブリック IPv4 アドレスをプライマリネットワークインターフェイスに自動割り当てできなくなります。Auto Scaling インスタンスに使用可能な Elastic IP アドレスを関連付けない限り、パブリック IPv4 アドレス経由でインスタンスに接続することはできません。

VPC のネットワークインターフェイス

VPC の各インスタンスには、デフォルトのネットワークインターフェイス (プライマリネットワークインターフェイス) があります。プライマリネットワークインターフェイスをインスタンスからデタッチすることはできません。VPC の任意のインスタンスに追加のネットワークインターフェイス

を作成してアタッチできます。使用できる Elastic Network Interface の最大数はインスタンスタイプによって異なります。

起動テンプレートを使用してインスタンスを起動するときは、追加のネットワークインターフェイスを指定できます。ただし、複数のネットワークインターフェイスを使用して Auto Scaling インスタンスを起動すると、インスタンスと同じサブネットに各インターフェイスが自動的に作成されます。これは、Amazon EC2 Auto Scaling が起動テンプレートで定義されたサブネットを無視して、Auto Scaling グループで指定されているものを選択するためです。詳細については、「Auto Scaling グループの起動テンプレートの作成」を参照してください。

同じサブネットから複数のネットワークインターフェイスを作成または、インスタンスにアタッチすると、非対称ルーティングなどのネットワーク問題が発生する場合があります。特に Amazon Linux 以外のバリアントを使用しているインスタンスに発生する場合があります。このタイプの設定が必要な場合は、OS 内でセカンダリネットワークインターフェイスを設定する必要があります。例については、 AWS ナレッジセンターの「Ubuntu EC2 インスタンスでセカンダリネットワークインターフェイスを機能させるにはどうすればよいですか？」を参照してください。

インスタンスのプレイスメントテナンシー

デフォルトでは、VPC 内のすべてのインスタンスは共有テナンシーインスタンスとして実行されます。Amazon EC2 Auto Scaling は、Dedicated Instances と Dedicated Hosts もサポートしています。詳細については、「詳細設定を使用して起動テンプレートを作成する」を参照してください。

AWS Outposts

AWS Outposts は、インターネットゲートウェイ、仮想プライベートゲートウェイ、Amazon VPC Transit Gateway、VPC エンドポイントなど、 AWS リージョンでアクセス可能な VPC コンポーネントを使用して、Amazon VPC を リージョンから Outpost に拡張します。Outpost はリージョン内のアベイラビリティーゾーンに設置されており、そのアベイラビリティーゾーンの耐障害性のために使用できる拡張機能です。

詳細については、AWS Outposts ユーザーガイドをご参照ください。

Outpost 内の Application Load Balancer からのトラフィックに対応する Auto Scaling グループをデプロイする方法の例については、「AWS Outpostsで Application Load Balancer を構成」というブログ記事を参照してください。

VPCs について学ぶためのその他のリソース

VPCs とサブネットの詳細については、以下のトピックを参照してください。

- VPC のプライベートサブネット
  - 例: プライベートサブネットと VPC にサーバーがある NAT
  - NAT ゲートウェイ
- VPC のパブリックサブネット
  - 例: テスト環境のVPC
  - 例: ウェブサーバーとデータベースサーバーのVPC
- Application Load Balancer のサブネット
  - ロードバランサーのサブネット
- 一般的なVPC情報
  - Amazon VPC ユーザーガイド
  - VPCs ピアリングを使用して VPC を接続する
  - Elastic Network Interface
  - プライベート接続のための VPC エンドポイントを使用する

Amazon EC2 Auto Scaling のセキュリティ

AWS ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された AWS のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS とお客様とが共有する責務です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ - AWS は、AWS Cloud で AWS の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を負います。また、AWSは、使用するサービスを安全に提供します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon EC2 Auto Scaling に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる対象範囲の AWS サービス」を参照してください。
- クラウド内のセキュリティ – お客様の責任は使用する AWS の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、Amazon EC2 Auto Scaling 使用時における責任共有モデルの適用法を理解するのに役立ちます。以下のトピックで、セキュリティおよびコンプライアンスの目的を満たすように、Amazon EC2 Auto Scaling を設定する方法について説明します。Amazon EC2 Auto Scaling リソースのモニタリングやセキュリティ確保に役立つ他の AWS サービスの使用方法についても説明します。

トピック

Amazon EC2 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として、Amazon EC2 Auto Scaling は AWS グローバルネットワークセキュリティによって保護されています。AWSセキュリティサービスと AWS がインフラストラクチャ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、セキュリティの柱 - AWS Well-Architected Frameworkのインフラストラクチャ保護を参照してください。

AWS が公開している API コールを使用し、ネットワーク経由で Amazon EC2 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。

Amazon EC2 Auto Scaling に仮想プライベートクラウド (VPC) エンドポイントを使用することもできます。インターフェイス VPC エンドポイントは、パブリックインターネットにさらされることなく、Amazon VPC リソースがプライベート IP アドレスを使用して Amazon EC2 Auto Scaling にアクセスできるようになります。詳細については、「Amazon EC2 Auto Scaling エンドポイント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。

関連リソース

Amazon EC2 が提供するサービストラフィックを分離する機能の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のインフラストラクチャセキュリティ」を参照してください。

Amazon EC2 Auto Scaling のレジリエンス

AWS グローバルインフラストラクチャは AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。AWS リージョン は、低レイテンシー、高スループット、そして高度な冗長ネットワークで接続される物理的に独立、隔離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンを使用すると、中断することなくゾーン間で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できます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョン とアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

アベイラビリティーゾーンの地理的な冗長性を活用するには、以下を行います。

- Auto Scaling グループを複数のアベイラビリティーゾーンにわたって配置します。
- 各アベイラビリティーゾーンで少なくとも 1 つのインスタンスを維持します。
- ロードバランサーをアタッチして、受信トラフィックを同じアベイラビリティーゾーンに分散させます。Application Load Balancer を使用する場合は、クロスゾーン負荷分散を有効にして、各 EC2 インスタンスに同程度のトラフィックがかかるようにしてください。これにより、フェイルオーバーイベント時の既存インスタンスへの負荷増加の影響を抑えることができ、クロスゾーン負荷分散を行わない場合よりも耐障害性が向上します。
- Elastic Load Balancing のヘルスチェックが正しく設定されていること、また Auto Scaling グループで有効になっていることを確認してください。その後、インスタンスがヘルスチェックに失敗すると、Elastic Load Balancing はそのインスタンスへのトラフィックの送信を停止して、トラフィックを正常なインスタンスに再ルーティングします。一方、Amazon EC2 Auto Scaling は異常なインスタンスを置き換えます。

Amazon EC2 Auto Scaling は、以下の方法でアプリケーションの耐障害性のニーズをサポートします。

- インスタンスの正常性やアクセス性に問題がないかを確認します。インスタンスに異常があると、そのインスタンスを終了させて新しいインスタンスを起動します。
- 動的スケーリングポリシーが有効な場合は、受信トラフィックに応じてキャパシティーを自動的にスケーリングします。
- スケーリングポリシーをサポートする Amazon CloudWatch メトリクスの信頼性の問題を検出し、データポイントが欠落している場合など、信頼できるメトリクスが利用できない場合はスケールインアクティビティを一時停止します。
- グループのスケーリングに伴い、有効化された各アベイラビリティーゾーンで同等の数のインスタンスを維持しようと試みます。
- 高可用性を維持するためにアベイラビリティーゾーンを使用します。アベイラビリティーゾーンが異常になったとき、Amazon EC2 Auto Scaling は次の処理を実行します。

  - Auto Scaling グループに対して有効になっている別のアベイラビリティーゾーンで新しいインスタンスを起動します。
  - 異常のあるアベイラビリティーゾーンが正常な状態に戻ったときに、有効なすべてのアベイラビリティーゾーンにインスタンスを再配布します。
- 特定のアベイラビリティーゾーンでインスタンスが起動しなかった場合、有効な他のアベイラビリティーゾーンでインスタンスの起動を継続的に試みます。
- Auto Scaling グループに関連付けられたロードバランサーにインスタンスの登録と解除を自動的に行います。そのため、インスタンスを個別に登録したり解除したりする必要はありません。
- Amazon EC2 Auto Scaling サービス API のコントロールプレーンが停止しても、既存の Auto Scaling グループのスケーリングには影響しません。

関連リソース

Amazon EBS が提供するデータ回復力のニーズをサポートする機能については、「Amazon EBS ユーザーガイド」の「Resilience in Amazon Elastic Block Store」を参照してください。

Amazon EC2 Auto Scaling でのデータ保護

AWS 責任共有モデル、Amazon EC2 Auto Scaling でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。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FAQ」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のAWS 「 責任共有モデルとGDPR」ブログ記事を参照してください。

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

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必要で、TLS 1.3 をお勧めします。
- で API およびユーザーアクティビティのログ記録を設定します AWS CloudTrail。 CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」の CloudTrail 証跡の使用」を参照してください。

- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な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 FIPS AWS 経由で にアクセスするときに API 140-3 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。使用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの機密情報は、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、EC2、または Word を使用して Amazon Word Auto Scaling または他の AWS のサービス を使用する場合も同様ですAPI AWS CLI 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。URL を外部サーバーに提供する場合は、そのサーバーへのリクエストを検証するために認証情報を URL に含めないことを強くお勧めします。

Amazon EC2 インスタンスを起動するときに、インスタンスの起動時にユーザーデータをインスタンスに渡すことで、追加の設定を行うことができます。また、インスタンスに渡されるユーザーデータに機密情報を含めないことを強くお勧めします。

AWS KMS keys を使用して Amazon EBS ボリュームを暗号化する

クラウドに保存されている Amazon EBS ボリュームデータを で暗号化するように Auto Scaling グループを設定できます AWS KMS keys。Amazon EC2 Auto Scaling は、データを暗号化するための AWS マネージドキーとカスタマーマネージドキーをサポートしています。起動設定を使用するときは、カスタマーマネージドキーを指定するための `KmsKeyId` オプションを利用できないことに注意してください。カスタマーマネージドキーを指定するには、その代わりに起動テンプレートを使用してください。詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。 AWS KMS 暗号化キーを作成、保存、管理する方法については、 AWS Key Management Service デベロッパーガイドを参照してください。

起動テンプレートまたは起動設定を設定する前に、EBS-backed AMI でカスタマーマネージドキーを設定したり、デフォルトで暗号化を使用して、作成した新しい EBS ボリュームとスナップショットコピーの暗号化を強制したりすることもできます。詳細については、「Amazon EBS ユーザーガイド」の「Word-backed AMIs による暗号化の使用」および「Amazon EBS ユーザーガイド」の「デフォルトでの暗号化」を参照してください。 EC2

> **Note**
>
> 暗号化にカスタマーマネージドキーを使用するときに Auto Scaling インスタンスの起動に必要になるキーポリシーを設定する方法については、「暗号化されたボリュームで使用するために必要な AWS KMS キーポリシー」を参照してください。

関連リソース

Amazon EBS が提供するデータ保護ガイドラインについては、「Amazon EBS ユーザーガイド」の「Amazon Elastic Block Store でのデータ保護」を参照してください。

暗号化されたボリュームで使用するために必要な AWS KMS キーポリシー

Amazon EC2 Auto Scaling は、サービスにリンクされたロールを使用して、アクセス許可を他の に委任します AWS のサービス。Amazon EC2 Auto Scaling サービスにリンクされたロールは事前定義されており、Amazon EC2 Auto Scaling が AWS のサービス ユーザーに代わって他の を呼び出すために必要なアクセス許可が含まれています。事前定義されたアクセス許可には、 へのアクセスも含まれます AWS マネージドキー。ただし、顧客管理キーへのアクセスは含まれていないため、これらのキーを完全に制御できます。

このトピックでは、Amazon EBS 暗号化のカスタマーマネージドキーを指定するときに Auto Scaling インスタンスを起動するために必要なキーポリシーを設定する方法について説明します。

> **Note**
>
> Amazon EC2 Auto Scaling では、アカウント内の暗号化されたボリュームを保護するためにデフォルトを使用する AWS マネージドキー ための追加の認可は必要ありません。

内容

概要

Amazon EBS Auto Scaling がインスタンスを起動するときに AWS KMS keys 、Amazon EC2 の暗号化に以下を使用できます。

- AWS マネージドキー – Amazon EBS が作成、所有、管理するアカウントの暗号化キー。これは、新しいアカウントのデフォルトの暗号化キーです。カスタマーマネージドキーを指定しない限り、AWS マネージドキー は暗号化に使用されます。
- カスタマーマネージド型キー – お客様が作成、所有、および管理するカスタム暗号化キー。詳細については、デベロッパーガイドの「AWS Key Management Service キーの作成」を参照してください。

  注:キーは対称である必要があります。Amazon EBS は、非対称カスタマーマネージドキーをサポートしていません。

カスタマーマネージド型キーは、暗号化されたスナップショットを作成するとき、または暗号化されたボリュームを指定する起動テンプレートを作成するとき、またはデフォルトで暗号化を有効にするときに設定します。

キーポリシーを設定する

KMS キーには、Amazon EC2 Auto Scaling がカスタマーマネージドキーで暗号化された Amazon EBS ボリュームを使用してインスタンスを起動できるようにするキーポリシーが必要です。

このページの例を使用して、Amazon EC2 Auto Scaling にカスタマーマネージドキーへのアクセスを許可するキーポリシーを設定します。カスタマー管理型 キーのキーポリシーは、キーの作成時または後で変更できます。

Amazon EC2 Auto Scaling を使用するには、少なくとも 2 つのポリシーステートメントをキーポリシーに追加する必要があります。

- 最初のステートメントでは、 `Principal`要素で指定された IAM ID がカスタマーマネージドキーを直接使用できます。これには AWS KMS `Encrypt`、キーに対して `Decrypt`、、`GenerateDataKey*`、、および `ReEncrypt*DescribeKey`オペレーションを実行するアクセス許可が含まれます。
- 2 番目のステートメントでは、 `Principal`要素で指定された IAM ID が `CreateGrant`オペレーションを使用して、 AWS KMS または別のプリンシパルと統合された に独自のアクセス許可のサブセットを委任 AWS のサービス する許可を生成できます。これにより、それらのサービスはお客様に代わって、キーを使用して、暗号化されたリソースを作成できるようになります。

キーポリシーに新しいポリシーステートメントを追加する場合は、ポリシーの既存のステートメントを変更しないでください。

次の各例では、キー ID やサービスにリンクされたロールの名前など、置き換える必要がある引数は次のように表示されます。*user placeholder text*。 ほとんどの場合、サービスにリンクされたロールの名前を Amazon EC2 Auto Scaling サービスにリンクされたロールの名前に置き換えることができます。

詳細については、以下のリソースを参照してください。

- を使用してキーを作成するには AWS CLI、「create-key」を参照してください。
- でキーポリシーを更新するには AWS CLI、put-key-policy」を参照してください。
- キー ID と Amazon リソースネーム (ARN) を検索するには、「 AWS Key Management Service デベロッパーガイド」の「キー ID とARNの検索」を参照してください。
- Amazon EC2 Auto Scaling サービスにリンクされたロールの詳細については、「」を参照してくださいAmazon EC2 Auto Scaling のサービスにリンクされたロール。
- Amazon EBS 暗号化と KMS 全般の詳細については、「Amazon EBS ユーザーガイド」および「 デベロッパーガイド」の「Amazon Word 暗号化」を参照してください。 EBS AWS Key Management Service

例 1: カスタマー管理キーへのアクセスを許可するキーポリシーセクション

次の 2 つのポリシーステートメントをカスタマーマネージドキーのキーポリシーに追加し、例の ARN を、キーへのアクセスが許可されている適切なサービスにリンクされたロールの ARN に置き換えます。この例では、ポリシーセクションは、AWSServiceRoleForAutoScaling という名前のサービスにリンクされたロールに、カスタマーマネージドキーを使用するためのアクセス許可を付与します。

```
{
   "Sid": "Allow service-linked role use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
       ]
   },
   "Action": [
       "kms:Encrypt",
```

```
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

```
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": true
        }
     }
}
```

例 2: カスタマー管理キーへのクロスアカウントアクセスを許可するキーポリシーセクション

カスタマー管理キーを Auto Scaling グループとは異なるアカウントで作成する場合は、キーへのクロスアカウントアクセスを許可するキーポリシーと組み合わせてグラントを使用する必要があります。

これには 2 つのステップがあり、以下の順序で完了する必要があります。

1. まず、カスタマー管理キーのキーポリシーに以下の 2 つのポリシーステートメントを追加します。サンプル ARN を他のアカウントの ARN に置き換え、必ず を置き換えてください。*111122223333* Auto Scaling グループ AWS アカウント を作成する の実際のアカウント ID を指定します。これにより、指定されたアカウントの IAM ユーザーまたはロールに、次の CLI コ

マンドを使用してキーの許可を作成するアクセス許可を付与できます。ただし、これ自体がキーへのアクセス許可をユーザーに提供するのではありません。

```
{
   "Sid": "Allow external account 111122223333 use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::111122223333:root"
       ]
   },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
   ],
   "Resource": "*"
}
```

```
{
   "Sid": "Allow attachment of persistent resources in external
 account 111122223333",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::111122223333:root"
       ]
   },
   "Action": [
       "kms:CreateGrant"
   ],
   "Resource": "*"
}
```

2. 次に、Auto Scaling グループを作成するアカウントから、関連する許可を適切なサービスリンクロールに委任するグラントを作成します。グラントの `Grantee Principal`要素は、適切なサービスにリンクされたロールの ARN です。`key-id` はキーのARNです。

以下は、アカウントで CLI という名前のサービスにリンクされたロールを付与する create-grant AWSServiceRoleForAutoScaling コマンドの例です。*111122223333* アカウントでカスタマーマネージドキーを使用するアクセス許可 *444455556666*.

```
aws kms create-grant \
  --region us-west-2 \
  --key-id arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
  --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \
  --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

このコマンドが成功するには、リクエストを行うユーザーが CreateGrant アクションに対する許可を持っている必要があります。

次の IAM ポリシーの例では、アカウントで IAM ID (ユーザーまたはロール) を許可します。*111122223333* アカウントでカスタマーマネージドキーの許可を作成する *444455556666*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

別の で KMS キーのグラントを作成する方法の詳細については AWS アカウント、「 AWS Key Management Service デベロッパーガイド」の「 のグラン AWS KMSト」を参照してください。

> **Important**
>
> 被付与者のプリンシパルとして指定されたサービスにリンクされたロール名は、既存のロールの名前でなければなりません。許可を作成した後、その許可によって Amazon EC2

Auto Scaling が指定された KMS キーを使用できるように、サービスにリンクされたロールを削除して再作成しないでください。

AWS KMS コンソールでキーポリシーを編集する

以前のセクションの例では、キーポリシーにステートメントを追加する方法のみを示しています。これは、キーポリシーを変更する 1 つの方法にすぎません。キーポリシーを変更する最も簡単な方法は、キーポリシーに AWS KMS コンソールのデフォルトビューを使用し、IAM アイデンティティ (ユーザーまたはロール) を適切なキーポリシーのキーユーザーの 1 つにすることです。詳細については、「 AWS Key Management Service デベロッパーガイド」の AWS Management Console 「デフォルトビューの使用」を参照してください。

Important

以下の点に注意してください。コンソールのデフォルトのビューポリシーステートメントには、カスタマーマネージドキーでオペレーションを実行する AWS KMS Revokeアクセス許可が含まれています。アカウントのカスタマーマネージドキー AWS アカウント へのアクセスを許可し、このアクセス許可を付与した権限を誤って取り消した場合、外部ユーザーは暗号化されたデータやデータの暗号化に使用されたキーにアクセスできなくなります。

Amazon EC2 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) し、誰に Amazon EC2 Auto Scaling リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

Amazon EC2 Auto Scaling を使用するには、アカウントにサインインするための AWS アカウントとセキュリティ認証情報が必要です。詳細については、IAM ユーザーガイド」のAWS 「セキュリティ認証情報」を参照してください。

IAM ドキュメントの詳細については、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を持つことができますが、アクセス許可がない限り、Amazon EC2 Auto Scaling リソースを作成またはアクセスすることはできません。例えば、Auto Scaling グループの作成、起動テンプレートを使用したインスタンスの起動などを実行するための許可が必要です。

以下のセクションでは、Amazon IAM EC2 Auto Scaling 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 Word を使用して Amazon EC2 Auto Scaling リソースを保護する方法について詳しく説明します。

Amazon EC2 トピックを最初に読むことをお勧めします。「Amazon EC2 ユーザーガイド」の「Amazon Word の Identity and Access Management」を参照してください。 EC2 このセクションのトピックを読んだら、Amazon EC2 が提供するアクセスコントロールのアクセス許可と、それらが Amazon EC2 Auto Scaling リソースのアクセス許可にどのように適合するかを理解しておく必要があります。

トピック

- Amazon EC2 Auto Scaling と IAM の連携方法
- Amazon EC2 Auto Scaling API のアクセス許可
- AWS Amazon EC2 Auto Scaling の マネージドポリシー
- Amazon EC2 Auto Scaling のサービスにリンクされたロール
- Amazon EC2 Auto Scaling のアイデンティティベースのポリシーの例
- サービス間の混乱した代理の防止
- Auto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する
- Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール

Amazon EC2 Auto Scaling と IAM の連携方法

IAM を使用して Amazon EC2 Auto Scaling へのアクセスを管理する前に、Amazon IAM Auto Scaling で使用できる EC2 の機能について学びます。

Amazon IAM Auto Scaling で使用できる EC2 の機能

IAM 機能	Amazon EC2 Auto Scaling のサポート
アイデンティティベースポリシー	あり

IAM 機能	Amazon EC2 Auto Scaling のサポート
リソースベースのポリシー	なし
ポリシーアクション	あり
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
ACLs	なし
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

Amazon EC2 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、Word IAMユーザーガイドの「 が AWS のサービス IAM と連携する」を参照してください。

Amazon EC2 Auto Scaling のアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

ID ベースのポリシーは、JSON ユーザー、ユーザーのグループ、ロールなど、ID にアタッチできる IAM アクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。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「カスタマー管理ポリシーによるカスタム Word アクセス許可の定義」を参照してください。 IAM

IAM アイデンティティベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否され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、JSONIAM ユーザーガイド」の「Word ポリシー要素のリファレンス」を参照してください。 IAM

Amazon EC2 Auto Scaling 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースにアタッチする JSON ポリシードキュメントです。リソースベースのポリシーの例としては、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、リソースベースのポリシーのプリンシパルとして、アカウント全体または別のアカウントの IAM エンティティを指定できます。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、IAM ユーザーガイド」の「Word でのクロスアカウントリソースアクセス」を参照してください。 IAM

Amazon EC2 Auto Scaling のポリシーアクション

ポリシーアクションのサポート: あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action`要素は、ポリシーでアクセスを許可または拒否するために使用できるアクションを記述します。ポリシーアクションの名前は通常、 associated AWS API オペレーションと同じです。一致する API オペレーションを持たないアクセス許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon EC2 Auto Scaling アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon EC2 Auto Scaling で定義されるアクション」を参照してください。

Amazon EC2 Auto Scaling のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
autoscaling
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
  "autoscaling:action1",
  "autoscaling:action2"
]
```

ワイルドカード (*) を使用して複数のアクションを指定できます。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": "autoscaling:Describe*"
```

Amazon EC2 Auto Scaling のポリシーリソース

ポリシーリソースのサポート: あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON Policy 要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

ARNs を使用して、Word IAMポリシーが適用される Auto Scaling グループと起動設定を識別できます。

Auto Scaling グループには、次のARNがあります。

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
```

起動設定には次のARNがあります。

```
"Resource": "arn:aws:autoscaling:region:account-
id:launchConfiguration:uuid:launchConfigurationName/lc-name"
```

`CreateAutoScalingGroup` アクションで Auto Scaling グループを指定するには、次の例に示すように、UUID をワイルドカード (*) に置き換える必要があります。

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

`CreateLaunchConfiguration` アクションで起動設定を指定するには、次の例に示すように、UUID をワイルドカード (*) に置き換える必要があります。

```
"Resource": "arn:aws:autoscaling:region:account-
id:launchConfiguration:*:launchConfigurationName/lc-name"
```

Amazon EC2 Auto Scaling リソースタイプとその ARNs の詳細については、「サービス認可リファレンス」の「Amazon EC2 Auto Scaling で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon EC2 Auto Scaling で定義されるアクション」を参照してください。

> **Note**
>
> IAM を使用して Auto Scaling グループへのアクセスを制御する ARNs ポリシーの例については、「」を参照してください削除できる Auto Scaling グループを制御する。

すべての Amazon EC2 Auto Scaling アクションがリソースレベルのアクセス許可をサポートしているわけではありません。リソースレベルの許可をサポートしていないアクションの場合、ワイルドカード (*) をリソースとして使用する必要があります。

次の Amazon EC2 Auto Scaling アクションは、リソースレベルのアクセス許可をサポートしていません。

- DescribeAccountLimits
- DescribeAdjustmentTypes
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeAutoScalingNotificationTypes
- DescribeInstanceRefreshes
- DescribeLaunchConfigurations
- DescribeLifecycleHooks
- DescribeLifecycleHookTypes
- DescribeLoadBalancers
- DescribeLoadBalancerTargetGroups
- DescribeMetricCollectionTypes
- DescribeNotificationConfigurations
- DescribePolicies
- DescribeScalingActivities
- DescribeScalingProcessTypes
- DescribeScheduledActions
- DescribeTags
- DescribeTerminationPolicyTypes
- DescribeWarmPool

Amazon EC2 Auto Scaling のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルが、どのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの条件演算子を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば、IAM ユーザー名でタグ付けされている場合にのみ、リソースへのアクセス許可を IAM ユーザーに付与できます。詳細については、IAM ユーザーガイドの「Word ポリシー要素: 変数とタグ」を参照してください。 IAM

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

Amazon EC2 Auto Scaling は、サポートされているアクションへのアクセスを制御し、Auto Scaling グループの設定を強制するために使用できる以下の条件キーをサポートしています。

- `autoscaling:InstanceTypes`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/key-name: tag-value`
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`

次の条件キーは、起動構成リクエストの作成に固有のものです。

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:MetadataHttpEndpoint`
- `autoscaling:MetadataHttpPutResponseHopLimit`

- `autoscaling:MetadataHttpTokens`
- `autoscaling:SpotPrice`

Amazon EC2 Auto Scaling は、リクエスト内のタグまたは Auto Scaling グループに存在するタグに基づいてアクセス許可を定義するために使用できる以下のグローバル条件キーもサポートしています。詳細については、「Auto Scaling グループとインスタンスにタグを付ける」を参照してください。

- `aws:RequestTag/key-name: tag-value`
- `aws:ResourceTag/key-name: tag-value`
- `aws:TagKeys: [tag-key, ...]`

条件キーを使用できる Amazon EC2 Auto Scaling API アクションについては、「サービス認可リファレンス」の「Amazon EC2 Auto Scaling で定義されるアクション」を参照してください。Amazon EC2 Auto Scaling の条件キーの詳細については、「Amazon EC2 Auto Scaling の条件キー」を参照してください。

> **Note**
>
> 条件キーを使用してサポートされているアクションへのアクセスを制御し、Auto Scaling グループの設定を適用する IAM ポリシーの例については、次のリソースを参照してください。
>
> - 起動テンプレートとバージョン番号を要求する — この例では、Auto Scaling グループを作成または更新するときに、起動テンプレートと起動テンプレートのバージョン番号を指定する必要があります。
> - 作成できる Auto Scaling グループのサイズを制御する — この例では、特定のタグがついた Auto Scaling グループを作成または更新するときに、`MinSize` と `MaxSize` のプロパティに指定できる値に制約を課しています。
> - 削除できるスケーリングポリシーを制御する — この例では、特定のタグがついていない Auto Scaling グループのみ、スケーリングポリシーの削除を許可するよう強制しています。

Amazon ACLs Auto Scaling のEC2

ACLsのサポート： いいえ

アクセスコントロールリスト (ACLs) は、リソースへのアクセス許可を持つプリンシパル (アカウントメンバー、ユーザー、またはロール) を制御します。ACLs はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式を使用しません。

Amazon ABAC Auto Scaling を使用したEC2

ABAC (ポリシー内のタグ): 一部

属性ベースのアクセスコントロール (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初のステップです。次に、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する場合に、オペレーションを許可するように ABAC ポリシーを設計します。

ABAC は、急速に成長している環境や、ポリシー管理が煩雑になる状況に役立ちます。

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。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、Word IAMユーザーガイドのABAC 認可によるアクセス許可の定義」を参照してください。ABAC の設定手順を含むチュートリアルを表示するには、Word ユーザーガイドの「属性ベースのアクセスコントロール (WordABAC) を使用する」を参照してください。 IAM

ABAC はタグをサポートするリソースで使用できますが、すべてがタグをサポートしているわけではありません。起動設定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、Auto Scaling グループはタグをサポートしています。

詳細については、「Auto Scaling グループとインスタンスにタグを付ける」を参照してください。

Amazon EC2 Auto Scaling での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、Word ユーザーガイドのIAMAWS のサービス と IAM を連携させる 」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。ロールの切り替えの詳細については、Word ユーザーガイドの「ユーザーから IAM ロールへの切り替え (コンソール) 」を参照してください。 IAM

AWS CLI or AWS API を使用して、一時的な認証情報を手動で作成できます。その後、これらの一時的な認証情報を使用して アクセスできます AWS。 AWS では、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お勧めします。詳細については、IAM の一時的なセキュリティ認証情報」を参照してください。

Amazon EC2 Auto Scaling のサービスロール

サービスロールのサポート: あり

サービスロールは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイド」の「 にアクセス許可を委任するロールを作成する AWS のサービス」を参照してください。

Amazon SNS トピックまたは Amazon SQS キューを通知するライフサイクルフックを作成するときは、ユーザーに代わって Amazon EC2 Auto Scaling が Amazon SNS または Amazon SQS にアクセスすることを許可するロールを指定する必要があります。IAM コンソールを使用して、ライフサイクルフックのサービスロールを設定します。コンソールは、マネージドポリシーを使用して十分なアクセス許可セットを持つロールを作成するのに役立ちます。詳細については、Amazon SNS を使用した通知の受信およびAmazon SQS を使用した通知の受信を参照してください。

Auto Scaling グループを作成するときに、オプションでサービスロールを渡して、Amazon EC2 インスタンスが AWS のサービス ユーザーに代わって他の にアクセスできるようにします。Amazon EC2 インスタンスのサービスロール (起動テンプレートまたは起動設定の Amazon EC2 インスタンスプロファイルとも呼ばれます) は、インスタンスの起動時に Auto Scaling グループ内のすべての EC2 インスタンスに割り当てられる特殊なタイプのサービスロールです。IAM コンソールと を使用して AWS CLI 、このサービスロールを作成または編集できます。詳細については、「Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、Amazon EC2 Auto Scaling の機能が破損する可能性があります。Amazon EC2 Auto Scaling が指示する場合以外は、サービスロールを編集しないでください。

Amazon EC2 Auto Scaling のサービスにリンクされたロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

Amazon EC2 Auto Scaling サービスにリンクされたロールの作成または管理の詳細については、「」を参照してくださいAmazon EC2 Auto Scaling のサービスにリンクされたロール。

Amazon EC2 Auto Scaling API のアクセス許可

「」で説明されているように、必要な Amazon EC2 Auto Scaling API アクションを呼び出すアクセス許可をユーザーに付与する必要がありますAmazon EC2 Auto Scaling のポリシーアクション。さらに、一部の Amazon EC2 Auto Scaling アクションでは、ユーザーに other AWS APIs から特定のアクションを呼び出すアクセス許可を付与する必要があります。

他の AWS APIs から必要なアクセス許可

Amazon EC2 Auto Scaling API のアクセス許可に加えて、ユーザーは、関連するアクションを正常に実行するために other AWS APIs からの次のアクセス許可を持っている必要があります。

Auto Scaling グループの作成 (`autoscaling:CreateAutoScalingGroup`)

- `iam:CreateServiceLinkedRole` — デフォルトのサービスにリンクされたロールがまだ存在しない場合に作成します。
- `iam:PassRole` – 起動時に IAM ロールをサービスまたは EC2 インスタンスに渡す。デフォルト以外のサービスにリンクされたロール、ライフサイクルフックの IAM ロール、またはインスタンスプロファイル (IAM ロールのコンテナ) を指定する起動テンプレートが提供される場合に必要です。

- ec2:RunInstances — 起動テンプレートが提供されている場合にインスタンスを起動します。
- ec2:CreateTags — タグ仕様を含む起動テンプレートが提供されている場合、起動時にインスタンスとボリュームにタグを付けます。

ライフサイクルフックの作成 (autoscaling:PutLifecycleHook)

- iam:PassRole – IAM ロールをサービスに渡す。IAM ロールが指定されている場合は必須です。

VPC Lattice ターゲットグループをアタッチする (autoscaling:AttachTrafficSources)

- vpc-lattice:RegisterTargets — インスタンスをターゲットグループに自動的に登録します。

VPC Lattice ターゲットグループをデタッチする (autoscaling:DetachTrafficSources)

- vpc-lattice:DeregisterTargets — ターゲットグループへのインスタンスの登録を自動的に解除します。

起動設定を作成する (autoscaling:CreateLaunchConfiguration)

- ec2:DescribeImages
- ec2:DescribeInstances
- ec2:DescribeInstanceAttribute
- ec2:DescribeKeyPairs
- ec2:DescribeSecurityGroups
- ec2:DescribeSpotInstanceRequests
- ec2:DescribeVpcClassicLink
- iam:PassRole - 起動時に IAM インスタンスに EC2 ロールを渡します。起動設定でインスタンスプロファイル (IAM ロールのコンテナ) が指定されている場合に必要です。

AWS Amazon EC2 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で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。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS 管理ポリシーを更新する可能性が最も高くなります。

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

Amazon EC2 Auto Scaling 管理ポリシー

(IAM) ID AWS Identity and Access Management (ユーザーまたはロール) には、次の管理ポリシーをアタッチできます。各ポリシーは、Amazon API Auto Scaling の EC2 アクションのすべてまたは一部へのアクセスを提供します。

- AutoScalingConsoleFullAccess – を使用して Amazon EC2 Auto Scaling へのフルアクセスを許可します AWS Management Console。このポリシーは、起動設定を使用しているときは機能しますが、起動テンプレートを使用しているときは機能しません。
- AutoScalingConsoleReadOnlyAccess – を使用して Amazon EC2 Auto Scaling への読み取り専用アクセスを許可します AWS Management Console。このポリシーは、起動設定を使用しているときは機能しますが、起動テンプレートを使用しているときは機能しません。
- AutoScalingFullAccess – AWS CLI または IAM からの完全な Amazon EC2 Auto Scaling アクセスを必要とするが AWS Management Console 、アクセスを必要としない Amazon EC2 Auto Scaling for SDKs ID へのフルアクセスを許可します。
- AutoScalingReadOnlyAccess – AWS CLI または EC2 のみを呼び出す IAM ID の Amazon SDKs Auto Scaling への読み取り専用アクセスを許可します。

コンソールから起動テンプレートを使用する場合は、起動テンプレートに固有の追加のアクセス許可を付与する必要があります。詳細については、「Auto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する」で詳しく説明します。Amazon EC2 Auto Scaling コンソールには、起動テンプレートと起動テンプレートを使用してインスタンスに関する情報を表示できるように、`ec2`アクションのアクセス許可が必要です。

AutoScalingServiceRolePolicy AWS 管理ポリシー

このポリシーは、Amazon EC2 Auto Scaling がユーザーに代わってアクション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。詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。

このポリシーのアクセス許可を確認するには、「 AWS マネージドポリシーリファレンス」のAutoScalingServiceRolePolicy」を参照してください。

AWS マネージドポリシーに対する Amazon EC2 Auto Scaling の更新

Amazon EC2 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。このページの変更に関する自動通知については、Amazon RSS Auto Scaling ドキュメント履歴ページの EC2 フィードにサブスクライブしてください。

変更	説明	日付
Amazon EC2 Auto Scaling が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AutoScalingServiceRolePolicy ポリシーには、 AWS Resource Groups ListGroupResources API アクションを呼び出して、指定されたリソースグループのメンバーであるリソースのすべてのリソース名 (ARNs) を取得するアクセス許可が含まれるようになりました。詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。	2024 年 11 月 20 日
Amazon EC2 Auto Scaling が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AutoScalingServiceRolePolicy ポリシーは、検証VPCを改善するために EC2 のすべてのセキュリティグループを取得する Amazon GetSecurityGroupsForVpc API アクションと、特定のインスタンス要件を満たすインスタンスタイプに関する情報を取得する Amazon Word EC2 Word API アクションを	2024 年 2 月 29 日

変更	説明	日付
	呼び出すアクセス許可を付与するようになりました。GetInstanceTypesFromInstanceRequirements詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。	

変更	説明	日付
Amazon EC2 Auto Scaling が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AutoScalingServiceRolePolicy ポリシーは、API Lattice との統合に必要な VPC アクションにアクセスするためのアクセス許可をサービスに付与するようになりました。 • GetTargetGroup および ListTargetGroup アクション。VPC Lattice ターゲットグループに関する情報を取得するために必要です。 • RegisterTargets および DeregisterTargets アクション。VPC Lattice ターゲットグループからインスタンスを登録および登録解除するために必要です。 • ListTargets 。 Amazon EC2 Auto Scaling が Word VPCLattice ターゲットグループに登録されたインスタンスのヘルス情報を取得できるようにします。 詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。	2022 年 12 月 6 日

変更	説明	日付
Amazon EC2 Auto Scaling が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	起動テンプレートの作成時に AWS Systems Manager パラメータを AMI ID のエイリアスとして使用できるように、`AutoScalingServiceRolePolicy` ポリシーは AWS Systems Manager GetParameters API アクションを呼び出すアクセス許可を付与するようになりました。詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。	2022 年 3 月 28 日
Amazon EC2 Auto Scaling が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	予測スケーリングをサポートするために、`AutoScalingServiceRolePolicy` ポリシーに GetMetricData CloudWatch API アクションを呼び出すアクセス許可が含まれるようになりました。詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。	2021 年 5 月 19 日
Amazon EC2 Auto Scaling が変更の追跡を開始	Amazon EC2 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。	2021 年 5 月 19 日

Amazon EC2 Auto Scaling のサービスにリンクされたロール

Amazon EC2 Auto Scaling は、 AWS のサービス ユーザーに代わって他の 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。サービスにリンクされたロールは、に直接リンクされた一意のタイプの IAM ロールです AWS のサービス。

サービスにリンクされたロールは、他の AWS のサービス にアクセス許可を委任するためのセキュア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。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使用」を参照してください。また、サービスにリンクされたロールを使用すると、すべての API 呼び出しを表示できます AWS CloudTrail。これは、Amazon EC2 Auto Scaling がユーザーに代わって実行するすべてのアクションを追跡できるため、モニタリングと監査の要件に役立ちます。詳細については、「を使用した Amazon EC2 Auto Scaling API呼び出しの口グ記録 AWS CloudTrail」を参照してください。

以下のセクションでは、Amazon EC2 Auto Scaling サービスにリンクされたロールを作成および管理する方法を説明します。まず、IAM アイデンティティ (ユーザーやロールなど) がサービスにリンクされたロールを作成、編集、または削除できるようにするアクセス許可を設定します。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使用」を参照してください。

内容

概要

Amazon EC2 Auto Scaling サービスにリンクされたロールには、次の 2 つのタイプがあります。

- AWSServiceRoleForAutoScaling という名前のアカウントのデフォルトのサービスにリンクされたロール。このロールは、自動的に Auto Scaling グループに割り当てられます。ただし、別のサービスにリンクされたロールを指定している場合を除きます。

- AWSServiceRoleForAutoScaling_ など、ロールの作成時に指定するカスタムサフィックスを持つサービスにリンクされたロール*mysuffix*.

カスタムサフィックス付きのサービスにリンクされたロールのアクセス許可は、デフォルトのサービスにリンクされたロールのアクセス許可と同じです。いずれの場合も、ロールを編集することはできません。また、Auto Scaling グループが使用中の場合は削除することもできません。唯一の違いは、ロール名サフィックスです。

AWS Key Management Service キーポリシーを編集するときにどちらのロールも指定して、Amazon EC2 Auto Scaling によって起動されるインスタンスをカスタマーマネージドキーで暗号化できます。ただし、特定のカスタマー管理キーへのきめ細かなアクセスを許可する場合は、サービスにリンクされたロールカスタムサフィックスを使用する必要があります。カスタムサフィックス付きのサービスにリンクされたロールを使用すると、以下のことが可能です。

- カスタマー管理キーをより詳細にコントロールする
- Word CloudTrailログでどの Auto Scaling グループが API 呼び出しを行ったかを追跡する機能

一部のユーザーにのみアクセスを許可するカスタマー管理キー を作成する場合は、以下のステップに従って、カスタムサフィックス付きのサービスにリンクされたロールを使用できます。

1. カスタムサフィックス付きのサービスにリンクされたロールを作成します。詳細については、「サービスにリンクされたロールを作成する (マニュアル)」を参照してください。
2. サービスにリンクされたロールにカスタマー管理キー へのアクセスを許可します。サービスにリンクされたロールにキーの使用を許可するキーポリシーの詳細については、「暗号化されたボリュームで使用するために必要な AWS KMS キーポリシー」を参照してください。
3. ユーザーに、作成したサービスにリンクされたロールへのアクセスを許可します。IAM ポリシーの作成の詳細については、「」を参照してください渡すことができるサービスにリンクされたロールを制御する (PassRole を使用)。ユーザーが、サービスにリンクされたロールを渡すためのアクセス許可なしでそのロールを指定しようとすると、エラーが表示されます。

サービスにリンクされたロールによって付与されるアクセス許可

Amazon EC2 Auto Scaling は、AWSServiceRoleForAutoScaling という名前のサービスリンクロールまたはカスタムサフィックスサービスリンクロールを使用します。

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

- `autoscaling.amazonaws.com`

ロールのアクセス許可ポリシー、 AutoScalingServiceRolePolicyでは、Amazon EC2 Auto Scaling が以下のアクションを実行できるようにします。

- `ec2` – EC2 インスタンスを作成、説明、変更、開始/停止、および終了します。
- `iam` – IAM ロールを EC2 インスタンスに渡して、インスタンスで実行されているアプリケーションがロールの一時的な認証情報にアクセスできるようにします。
- `iam` – AWSServiceRoleForEC2Spot サービスにリンクされたロールを作成して、Amazon EC2 Auto Scaling がユーザーに代わってスポットインスタンスを起動できるようにします。
- `elasticloadbalancing` — Elastic Load Balancing を使用してインスタンスを登録および登録解除し、登録されたターゲットの正常性をチェックします。
- `cloudwatch` – スケーリングポリシーの CloudWatch アラームを作成、説明、変更、削除し、予測スケーリングに使用されるメトリクスを取得します。
- `sns` – インスタンスの起動または終了時に Amazon SNS に通知を発行します。
- `events` – ユーザーに代わって EventBridge ルールを作成、説明、更新、削除します。
- `ssm` – 起動テンプレートで Systems Manager パラメータを AMI ID のエイリアスとして使用する場合は、Parameter Store からパラメータを読み取ります。
- `vpc-lattice` – VPC Lattice でインスタンスを登録および登録解除し、登録されたターゲットの状態を確認します。
- `resource-groups` – 指定されたリソースグループのメンバーであるリソースのすべてのリソース名 (ARNs) を取得します。

Amazon EC2 Auto Scaling サービスにリンクされたロールでサポートされているリージョン

Amazon EC2 Auto Scaling は、サービスが利用可能なすべての で AWS リージョン サービスにリンクされたロールの使用をサポートしています。

サービスにリンクされたロールの作成、編集、削除

サービスにリンクされたロールを作成する (自動)

Amazon EC2 Auto Scaling は、Auto Scaling グループを初めて作成するときに AWSServiceRoleForAutoScaling サービスにリンクされたロールを作成します。ただし、カスタムサ

フィックスサービスにリンクされたロールを手動で作成し、グループの作成時に指定する場合は除きます。

> **Important**
>
> サービスにリンクされたロールを作成するには、IAM アクセス許可が必要です。それ以外の場合、自動作成は失敗します。詳細については、IAM ユーザーガイド」の「サービスにリンクされたロールのアクセス許可」およびこのガイドサービスにリンクされたロールの作成の「」を参照してください。

Amazon EC2 Auto Scaling は、2018 年 3 月にサービスにリンクされたロールのサポートを開始しました。それ以前に Auto Scaling グループを作成した場合、Amazon EC2 Auto Scaling はアカウントに AWSServiceRoleForAutoScaling ロールを作成しました。詳細については、IAM ユーザーガイド」の「 に新しいロールが表示され AWS アカウントました」を参照してください。

サービスにリンクされたロールを作成する (マニュアル)

サービスにリンクされたロールを作成するには (コンソール)

1. IAM で https://console.aws.amazon.com/iam/ コンソールを開きます。
2. ナビゲーションペインで [ロール]、[ロールの作成] の順に選択します。
3. [Select trusted entity] (信頼されたエンティティの選択) で、[AWS のサービス] を選択します。
4. このロールを使用するサービスを選択する で、EC2 Auto Scaling と EC2 Auto Scaling のユースケースを選択します。
5. [Next: Permissions (次へ: アクセス許可)]、[Next: Tags (次へ: タグ)]、[Next: Review (次へ: レビュー)] の順に選択します。注意: サービスにリンクされたロールの作成時にタグ付けを行うことはできません。
6. レビューページで、ロール名を空白のままにして AWSServiceRoleForAutoScaling という名前のサービスにリンクされたロールを作成するか、サフィックスを入力して AWSServiceRoleForAutoScaling_ という名前のサービスにリンクされたロールを作成します。***suffix***.
7. (オプション) [ロールの説明] で、サービスにリンクされたロールの説明を編集します。
8. [ロールの作成] を選択します。

サービスリンクロールの作成 (AWS CLI)

次の create-service-linked-role CLI コマンドを使用して、AWSServiceRoleForAutoScaling_ という名前の Amazon EC2 Auto Scaling のサービスにリンクされたロールを作成します。***suffix***.

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --
custom-suffix suffix
```

このコマンドの出力には、サービスにリンクされたロールの ARN が含まれます。これを使用して、サービスにリンクされたロールにカスタマーマネージドキーへのアクセス権を付与できます。

```
{
    "Role": {
        "RoleId": "ABCDEF0123456789ABCDEF",
        "CreateDate": "2018-08-30T21:59:18Z",
        "RoleName": "AWSServiceRoleForAutoScaling_suffix",
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_suffix",
        "Path": "/aws-service-role/autoscaling.amazonaws.com/",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Principal": {
                        "Service": [
                            "autoscaling.amazonaws.com"
                        ]
                    },
                    "Effect": "Allow"
                }
            ]
        }
    }
}
```

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作成」を参照してください。

サービスにリンクされたロールを編集する

Amazon EC2 Auto Scaling 用に作成されたサービスにリンクされたロールは編集できません。サービスにリンクされたロールを作成した後、ロールの名前またはアクセス許可を変更することはできません。ただし、ロールの説明は編集できます。詳細については、「Word ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。 IAM

サービスにリンクされたロールを削除する

Auto Scaling グループを使用していない場合、そのサービスにリンクされたロールを削除することをお勧めします。ロールを削除すると、使用されていないエンティティやアクティブにモニタリングおよび維持されていないエンティティがなくなります。

サービスにリンクされたロールを削除するには、まずその関連依存リソースを削除します。これにより、リソースに対する Amazon EC2 Auto Scaling 許可を誤って取り消しないように保護できます。サービスにリンクされたロールが複数の Auto Scaling グループで使用されている場合、サービスにリンクされたロールを削除する前に、そのロールを使用するすべての Auto Scaling グループを削除する必要があります。詳細については、「Auto Scaling インフラストラクチャを削除する」を参照してください。

IAM を使用して、サービスにリンクされたロールを削除できます。詳細については、IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

AWSServiceRoleForAutoScaling サービスにリンクされたロールを削除すると、Auto Scaling グループを作成し、別のサービスにリンクされたロールを指定しないときに、Amazon EC2 Auto Scaling によってロールが再度作成されます。

Amazon EC2 Auto Scaling のアイデンティティベースのポリシーの例

デフォルトでは、 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、Amazon IAM EC2 Auto Scaling Word アクションを実行するためのアクセス許可を IAM ID (ユーザーやロールなど) に付与する API ポリシーを作成して割り当てる必要があります。

これらの IAM ポリシードキュメントの例を使用して JSON ポリシーを作成する方法については、Word 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。

以下に示しているのは、アクセス許可ポリシーの例です。

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:UpdateAutoScalingGroup",
          "autoscaling:DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
      }
   },
   {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
   }]
}
```

このサンプルポリシーは、グループが **purpose=testing** タグを使用している場合に限り、Auto Scaling グループを作成、更新、削除する許可を付与します。Describe アクションはリソースレベルの許可をサポートしないため、条件のない別のステートメントで指定する必要があります。起動テンプレートを使用してインスタンスを起動するには、ユーザーに ec2:RunInstances アクセス許可も必要です。詳細については、「Auto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する」を参照してください。

> **Note**
>
> 独自のカスタム IAM ポリシーを作成して、IAM ID (ユーザーまたはロール) が Amazon EC2 Auto Scaling アクションを実行するためのアクセス許可を許可または拒否できます。これらのカスタムポリシーは、指定されたアクセス許可を必要とする IAM ID にアタッチできます。次の例では、いくつかの一般的ユースケースの許可を示します。
> 一部の Amazon EC2 Auto Scaling API アクションでは、アクションによって作成または変更できる特定の Auto Scaling グループをポリシーに含めることができます。個々の Auto Scaling グループ ARNs を指定することで、これらのアクションのターゲットリソースを制限できます。ただし、ベストプラクティスとして、特定のタグを持つ Auto Scaling グループに対するアクションを許可 (または拒否) するタグベースのポリシーを使用することをお勧めします。

例

- 作成できる Auto Scaling グループのサイズを制御する
- 使用できるタグキーとタグ値を制御する
- 削除できる Auto Scaling グループを制御する
- 削除できるスケーリングポリシーを制御する
- インスタンスの更新アクションへのアクセスを制御する
- サービスにリンクされたロールの作成
- 渡すことができるサービスにリンクされたロールを制御する (PassRole を使用)

作成できる Auto Scaling グループのサイズを制御する

次のポリシーでは、リクエスタが最小サイズとして **1** 未満または最大サイズとして **10** より大きい値を指定しない限り、タグ **environment=development** を持つすべての Auto Scaling グループを作成および更新する許可が付与されます。可能な限りタグを使用して、アカウント内の Auto Scaling グループへのアクセスを制御できるようにします。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:UpdateAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/environment": "development" },
          "NumericGreaterThanEqualsIfExists": { "autoscaling:MinSize": 1 },
          "NumericLessThanEqualsIfExists": { "autoscaling:MaxSize": 10 }
      }
   }]
}
```

または、タグを使用して Auto Scaling グループへのアクセスを制御していない場合は、ARNs を使用して、IAM ポリシーが適用される Auto Scaling グループを識別できます。

Auto Scaling グループには、次のARNがあります。

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/my-asg"
```

複数の ARNs をリストに囲んで指定することもできます。Resource 要素で Amazon EC2 Auto Scaling リソースのARNsを指定する方法の詳細については、「」を参照してくださいAmazon EC2 Auto Scaling のポリシーリソース。

使用できるタグキーとタグ値を制御する

IAM ポリシーの条件を使用して、Auto Scaling グループに適用できるタグキーとタグ値を制御することもできます。リクエスタが特定のタグを指定する場合に限り、Auto Scaling グループを作成またはタグ付けする許可を付与するには、aws:RequestTag 条件キーを使用します。特定のタグキーのみ許可するには、aws:TagKeys 修飾子とともに ForAllValues 条件キーを使用します。

次のポリシーでは、リクエストでキー **environment** にタグを指定することをリクエスタに要求されます。**"?*"** 値は、タグキーに何らかの値を含めることを強制します。ワイルドカードを含める場合は、StringLike 条件演算子を使用する必要があります。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "StringLike": { "aws:RequestTag/environment": "?*" }
      }
   }]
}
```

次のポリシーでは、リクエスタが Auto Scaling グループにタグ付けできるタグは **purpose=webserver** および **cost-center=cc123** であることを指定し、**purpose** タグおよび **cost-center** タグのみが許可されます (他のタグは指定できません)。

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

```
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "aws:RequestTag/purpose": "webserver",
              "aws:RequestTag/cost-center": "cc123"
          },
          "ForAllValues:StringEquals": { "aws:TagKeys": ["purpose", "cost-center"] }
      }
   }]
}
```

次のポリシーでは、リクエスタがリクエストで少なくとも 1 つのタグを指定することを要求し、**cost-center** および **owner** キーのみ使用できます。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner"] }
      }
   }]
}
```

> **Note**
>
> 条件においては、条件キーでは大文字と小文字が区別されず、条件値では大文字と小文字が区別されます。したがって、タグキーの大文字と小文字を区別するには、条件の値としてタグキーが指定される aws:TagKeys 条件キーを使用します。

削除できる Auto Scaling グループを制御する

次のポリシーは、グループにタグ **environment=development** が付けられている場合にのみ、Auto Scaling グループの削除を許可します。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "*",
      "Condition": {
          "StringEquals": { "aws:ResourceTag/environment": "development" }
      }
   }]
}
```

または、条件キーを使用して Auto Scaling グループへのアクセスを制御していない場合は、代わりに Resource要素でリソースのARNsを指定してアクセスを制御できます。

次のポリシーは、APIが DeleteAutoScalingGroup で始まる Auto Scaling グループに対してのみ、Word アクションを使用するアクセス許可をユーザーに付与しますdevteam-。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/devteam-*"
   }]
}
```

複数の ARNs をリストに囲んで指定することもできます。UUID を含めると、特定の Auto Scaling グループへのアクセスが許可されます。新しいグループの UUID は、同じ名前の削除されたグループの UUID とは異なります。

```
"Resource": [
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
```

```
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]
```

削除できるスケーリングポリシーを制御する

次のポリシーでは、DeletePolicy アクションを使用してスケーリングポリシーを削除する許可が付与されます。ただし、処理対象の Auto Scaling グループに **environment=production** タグがある場合、そのアクションを拒否します。可能な限りタグを使用して、アカウント内の Auto Scaling グループへのアクセスを制御できるようにします。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*"
   },
   {
      "Effect": "Deny",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
      }
   }]
}
```

インスタンスの更新アクションへのアクセスを制御する

次のポリシーは、処理対象の Auto Scaling グループにタグ **environment=testing** が付けられている場合にのみ、インスタンスの更新を開始、ロールバック、キャンセルするアクセス許可を付与します。Describe アクションはリソースレベルの許可をサポートしないため、条件のない別のステートメントで指定する必要があります。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:StartInstanceRefresh",
          "autoscaling:CancelInstanceRefresh",
```

```
            "autoscaling:RollbackInstanceRefresh"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "autoscaling:ResourceTag/environment": "testing" }
        }
    },
    {
        "Effect": "Allow",
        "Action": "autoscaling:DescribeInstanceRefreshes",
        "Resource": "*"
    }]
}
```

StartInstanceRefresh 呼び出しで希望する設定を指定するには、次のような関連するアクセス許可が必要になる場合があります。

- ec2:RunInstances – 起動テンプレートを使用して EC2 インスタンスを起動するには、ユーザーは IAM ポリシーで アクセスec2:RunInstances許可を持っている必要があります。詳細については、「Auto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する」を参照してください。
- ec2:CreateTags – 作成時にEC2インスタンスとボリュームにタグを追加する起動テンプレートから Word インスタンスを起動するには、ユーザーは IAM ポリシーで アクセスec2:CreateTags許可を持っている必要があります。詳細については、「インスタンスおよびボリュームにタグ付けするために必要なアクセス許可」を参照してください。
- iam:PassRole – インスタンスプロファイル (EC2 ロールのコンテナ) を含む起動テンプレートから IAM インスタンスを起動するには、ユーザーは IAM ポリシーで アクセスiam:PassRole許可も持っている必要があります。詳細と IAM ポリシーの例については、「」を参照してくださいAmazon IAM インスタンスで実行されるアプリケーションの EC2 ロール。
- ssm:GetParameters – AWS Systems Manager パラメータを使用する起動テンプレートから EC2 インスタンスを起動するには、ユーザーは IAM ポリシーで アクセスssm:GetParameters許可も持っている必要があります。詳細については、「起動テンプレートAMIIDsで の代わりに AWS Systems Manager パラメータを使用する」を参照してください。

サービスにリンクされたロールの作成

Amazon EC2 Auto Scaling では、 のユーザーが Amazon EC2 Auto Scaling API アクションを初めて 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するためのアクセス許可

が必要です。サービスにリンクされたロールがまだ存在しない場合、Amazon EC2 Auto Scaling はそれをアカウントに作成します。サービスにリンクされたロールは、Amazon EC2 Auto Scaling が AWS のサービス ユーザーに代わって他の を呼び出すことができるように、アクセス許可を付与します。

この自動ロール作成を成功させるには、ユーザーには `iam:CreateServiceLinkedRole` アクションへのアクセス許可が必要です。

```
"Action": "iam:CreateServiceLinkedRole"
```

以下は、ユーザーが Amazon EC2 Auto Scaling のサービスにリンクされたロールを Amazon EC2 Auto Scaling 用に作成できるようにするアクセス許可ポリシーの例です。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
      "Condition": {
          "StringLike": { "iam:AWSServiceName":"autoscaling.amazonaws.com" }
      }
   }]
}
```

渡すことができるサービスにリンクされたロールを制御する (PassRole を使用)

Auto Scaling グループを作成または更新し、リクエストでカスタムサフィックスサービスにリンクされたロールを指定するユーザーは、`iam:PassRole` 許可が必要です。

異なるサービスにリンクされたロールに異なるキーへのアクセスを許可する場合は、 アクセス`iam:PassRole`許可を使用して AWS KMS カスタマーマネージドキーのセキュリティを保護できます。組織の必要に応じて、開発チームに 1 つの キー、QA チームにもう 1 つの キー、そして財務チームにもう 1 つの キー を持つことができます。まず、AWSServiceRoleForAutoScaling_devteamkeyaccess という名前のサービスにリンクされたロールなど、必要なキーにアクセスできるサービスにリンクされたロールを作成します。次に、ユーザーやロールなどの IAM ID にポリシーをアタッチします。

次のポリシーは、名前が **devteam-** で始まる Auto Scaling グループに **AWSServiceRoleForAutoScaling_devteamkeyaccess** ロールを渡すための許可を付与しま

す。Auto Scaling グループを作成する IAM ID が別のサービスにリンクされたロールを指定しようとすると、エラーが発生します。サービスにリンクされたロールを指定しない場合、代わりにデフォルトの AWSServiceRoleForAutoScaling ロールが使用されます。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_devteamkeyaccess",
      "Condition": {
          "StringEquals": { "iam:PassedToService": [ "autoscaling.amazonaws.com" ] },
          "StringLike": { "iam:AssociatedResourceARN":
 [ "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/devteam-*" ] }
      }
   }]
}
```

カスタムサフィックス付きのサービスにリンクされたロールの詳細については、「Amazon EC2 Auto Scaling のサービスにリンクされたロール」を参照してください。

サービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。

では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。

これを防ぐために、 は、アカウント内のリソースへのアクセスが許可されているサービスプリンシパルを持つすべてのサービスのデータを保護するのに役立つツール AWS を提供します。Amazon EC2 Auto Scaling サービスロールの信頼ポリシーでは、 aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用することをお勧めします。これらのキーは、Amazon EC2 Auto Scaling がリソースに別のサービスに付与するアクセス許可を制限します。

フィールドSourceArnと SourceAccountフィールドの値は、Amazon EC2 Auto Scaling が AWS Security Token Service（ AWS STS) を使用してユーザーに代わってロールを引き受ける場合に設定されます。

aws:SourceArn または aws:SourceAccount グローバル条件キーを使用するには、Amazon ARN Auto Scaling が保存するリソースの Amazon リソースネーム (EC2) またはアカウントに値を設定します。可能な限り、より具体的な aws:SourceArn を使用してください。単語の未知部分について、値を ARN またはワイルドカード (*) 付きの ARN ARNパターンに設定します。リソースのARNがわからない場合は、aws:SourceAccount代わりに を使用してください。

次の例は、Amazon EC2 Auto Scaling で aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示しています。

例: **aws:SourceArn** 条件キーおよび **aws:SourceAccount** 条件キー

サービスがお客様に代わってアクションを実行するために引き受けるロールは、サービスロールと呼ばれます。Amazon EventBridge 以外の場所に通知を送信するライフサイクルフックを作成する場合は、サービスロールを作成して、Amazon EC2 Auto Scaling がユーザーに代わって Amazon SNS トピックまたは Amazon SQS キューに通知を送信できるようにする必要があります。クロスサービスアクセスに関連付ける Auto Scaling グループを 1 つだけにする場合は、サービスロールの信頼ポリシーを次のように指定できます。

この信頼ポリシーの例では、条件文を使用して、サービスロールの AssumeRole 機能を、指定されたアカウントの指定された Auto Scaling グループに影響を与えるアクションのみに制限します。aws:SourceArn および aws:SourceAccount の条件は個別に評価されます。サービスロールを使用するリクエストでは、両方の条件が満たされている必要があります。

このポリシーを使用する前に、リージョン、アカウント ID、UUID、およびグループ名をアカウントの有効な値に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "autoscaling.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
```

```
        "aws:SourceArn":
 "arn:aws:autoscaling:region:account_id:autoScalingGroup:uuid:autoScalingGroupName/my-
asg"
      },
      "StringEquals": {
        "aws:SourceAccount": "account_id"
      }
    }
  }
}
```

前の例では、以下のようになっています。

- Principal 要素は、サービス(autoscaling.amazonaws.com) のサービスプリンシパルを指定します。
- Action 要素は、sts:AssumeRole アクションを指定します。
- Condition 要素は、aws:SourceArn および aws:SourceAccount グローバル条件キーを指定します。ソースの ARN にはアカウント ID が含まれているため、 aws:SourceAccountで を使用する必要はありませんaws:SourceArn。

追加情報

詳細については、IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」、「混乱した代理問題」、「ロール信頼ポリシーの変更 (コンソール) 」を参照してください。 https://docs.aws.amazon.com/IAM/latest/UserGuide/confused-deputy.html

Auto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する

Amazon EC2 Auto Scaling は、Auto Scaling グループでの Amazon EC2 起動テンプレートの使用をサポートしています。起動テンプレートから Auto Scaling グループを作成することをユーザーに許可することをお勧めします。これにより、ユーザーは Amazon EC2 Auto Scaling と Amazon EC2 の最新機能を使用できるようになります。例えば、ユーザーは混合インスタンスポリシーを使用するための起動テンプレートを指定する必要があります。

AmazonEC2FullAccess ポリシーを使用して、アカウント内の Amazon EC2 Auto Scaling リソース、起動テンプレート、およびその他の EC2 リソースを操作するための完全なアクセス権をユーザーに付与できます。または、このトピックで説明されているように、独自のカスタム IAM ポリシーを作成して、起動テンプレートを操作するためのきめ細かなアクセス許可をユーザーに付与することもできます。

独自の用途に合わせてカスタマイズできるサンプルポリシー

次に、独自の用途に合わせてカスタマイズできる基本アクセス許可ポリシーの例を表示します。ポリシーは、グループが **purpose=testing** タグを使用している場合に限り、すべての Auto Scaling グループを作成、更新、削除することを許可します。その後、すべての Describe アクションに許可を与えます。Describe アクションはリソースレベルの許可をサポートしないため、条件のない別のステートメントで指定する必要があります。

このポリシーを持つIAM ID (ユーザーまたはロール) には、起動テンプレートを使用して Auto Scaling グループを作成または更新するアクセス許可があります。これは、 ec2:RunInstancesアクションを使用するアクセス許可も付与されるためです。

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
            }
         },
         {
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*",
                "ec2:RunInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

Auto Scaling グループを作成または更新するユーザーには、関連する次のような権限が必要になる場合があります。

- ec2:CreateTags – 作成時にインスタンスとボリュームにタグを追加するには、ユーザーは IAM ポリシーで アクセスec2:CreateTags許可を持っている必要があります。詳細については、「インスタンスおよびボリュームにタグ付けするために必要なアクセス許可」を参照してください。
- iam:PassRole – インスタンスプロファイル (EC2 ロールのコンテナ) を含む起動テンプレートから IAM インスタンスを起動するには、ユーザーは IAM ポリシーで アクセスiam:PassRole許可も持っている必要があります。詳細と IAM ポリシーの例については、「」を参照してくださいAmazon IAM インスタンスで実行されるアプリケーションの EC2 ロール。
- ssm:GetParameters – AWS Systems Manager パラメータを使用する起動テンプレートから EC2 インスタンスを起動するには、ユーザーは IAM ポリシーで アクセスssm:GetParameters許可も持っている必要があります。詳細については、「起動テンプレートAMIIDsで の代わりに AWS Systems Manager パラメータを使用する」を参照してください。

インスタンス起動時に完了するアクションに対するこれらのアクセス許可は、ユーザーが Auto Scaling グループを操作するときにチェックされます。詳細については、「ec2:RunInstances と iam:PassRole の許可の検証」を参照してください。

次の例は、IAM ユーザーが起動テンプレートを使用して持つアクセスを制御するために使用できるポリシーステートメントを示しています。

トピック

- 特定のタグを持つ起動テンプレートを要求する
- 起動テンプレートとバージョン番号を要求する
- インスタンスメタデータサービスバージョン 2 (IMDSv2) の使用を要求する
- Amazon EC2 リソースへのアクセスを制限する
- インスタンスおよびボリュームにタグ付けするために必要なアクセス許可
- 起動テンプレートの追加アクセス許可
- ec2:RunInstances と iam:PassRole の許可の検証
- 関連リソース

特定のタグを持つ起動テンプレートを要求する

アクセスec2:RunInstances許可を付与するときは、ユーザーが特定のタグまたは特定の IDs を持つ起動テンプレートのみを使用して、起動テンプレートでインスタンスを起動するときにアクセス許可を制限できるように指定できます。また、起動テンプレートを使用するすべてのユーザーがインス

タンスの起動時に参照および使用できる AMI やその他のリソースを制御するには、 RunInstances 呼び出しに対して追加のリソースレベルのアクセス許可を指定します。

次の例では、指定されたリージョンにある起動テンプレートを持ち、タグ **purpose=testing** を持つ ec2:RunInstances アクションへのアクセス許可を制限します。また、AMIs、インスタンスタイプ、ボリューム、キーペア、ネットワークインターフェイス、セキュリティグループなど、起動テンプレートで指定されたリソースへのアクセス権をユーザーに付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:account-id:launch-template/*",
            "Condition": {
                "StringEquals": { "aws:ResourceTag/purpose": "testing" }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region::image/ami-*",
                "arn:aws:ec2:region:account-id:instance/*",
                "arn:aws:ec2:region:account-id:subnet/*",
                "arn:aws:ec2:region:account-id:volume/*",
                "arn:aws:ec2:region:account-id:key-pair/*",
                "arn:aws:ec2:region:account-id:network-interface/*",
                "arn:aws:ec2:region:account-id:security-group/*"
            ]
        }
    ]
}
```

起動テンプレートでタグベースのポリシーを使用する方法の詳細については、「Amazon EC2 ユーザーガイド」のIAM アクセス許可を使用して起動テンプレートへのアクセスを制御する」を参照してください。

起動テンプレートとバージョン番号を要求する

IAM アクセス許可を使用して、Auto Scaling グループを作成または更新するときに起動テンプレートと起動テンプレートのバージョン番号を指定する必要があるように強制することもできます。

次の例では、起動テンプレートおよび起動テンプレートのバージョン番号が指定されている場合にのみ、ユーザーが Auto Scaling グループを作成および更新できるようにします。このポリシーを持つユーザーが、$Latest または $Default の起動テンプレートのバージョンを指定するためのバージョン番号を省略したり、代わりに起動設定を使用しようとしたりすると、アクションは失敗します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": { "autoscaling:LaunchConfigurationName": "false" }
            }
        }
    ]
}
```

インスタンスメタデータサービスバージョン 2 (IMDSv2) の使用を要求する

セキュリティを強化するために、IMDSv2 を必要とする起動テンプレートの使用を要求するようにユーザーのアクセス許可を設定できます。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスの設定」を参照してください。

次の例では、インスタンスが IMDSv2 (で示される) の使用を要求するようにオプトインされていない限り、ユーザーが `ec2:RunInstances`アクションを呼び出せないことを指定します`"ec2:MetadataHttpTokens":"required"`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireImdsV2",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
            }
        }
    ]
}
```

> **Tip**
>
> インスタンスのメタデータオプションが設定された、新しい起動テンプレートまたは新しいバージョンの起動テンプレートを使用している、代替の Auto Scaling インスタンスを強制的に起動することで、インスタンスの更新を開始できます。詳細については、「Auto Scaling インスタンスの更新」を参照してください。

Amazon EC2 リソースへのアクセスを制限する

次の例では、Amazon EC2 リソースへのアクセスを制限することで、ユーザーが起動できるインスタンスの設定を制御します。起動テンプレートで指定されたリソースにリソースレベルのアクセス許可を指定するには、`RunInstances` アクションステートメントにそのリソースを含める必要があります

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region:account-id:launch-template/*",
                "arn:aws:ec2:region::image/ami-04d5cc9b88example",
                "arn:aws:ec2:region:account-id:subnet/subnet-1a2b3c4d",
                "arn:aws:ec2:region:account-id:volume/*",
                "arn:aws:ec2:region:account-id:key-pair/*",
                "arn:aws:ec2:region:account-id:network-interface/*",
                "arn:aws:ec2:region:account-id:security-group/sg-903004f88example"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:account-id:instance/*",
            "Condition": {
                "StringEquals": { "ec2:InstanceType": ["t2.micro", "t2.small"] }
            }
        }
    ]
}
```

この例には、 2 つのステートメントがあります。

- 最初のステートメントでは、ユーザーは特定のセキュリティグループ (**subnet-1a2b3c4d**) と特定のAMI () を使用して、特定のサブネット (**sg-903004f88example**) でインスタンスを起動する必要があります**ami-04d5cc9b88example**。また、ユーザーは起動テンプレートで指定されているリソース (ネットワークインターフェイス、キーペア、ボリューム) にもアクセスできるようになります。
- 2 番目のステートメントでは、ユーザーは **t2.micro** または **t2.small** インスタンスタイプのみを使用してインスタンスを起動でき、コストを管理できます。

  ただし、起動テンプレートを使用してインスタンスを起動する権限を持つユーザーが、他のインスタンスタイプを起動するのを完全に防ぐ有効な方法は現在のところありません。これは、起動テンプレートで指定されたインスタンスタイプを上書きして、属性ベースのインスタンスタイプ選択を使用して定義されたインスタンスタイプを使用できるためです。

ユーザーが起動できるインスタンスの設定を制御するために使用できるリソースレベルのアクセス許可の完全なリストについては、「サービス認可リファレンス」の「Amazon EC2 のアクション、リソース、および条件キー」を参照してください。

インスタンスおよびボリュームにタグ付けするために必要なアクセス許可

次の例では、インスタンスとボリュームの作成時に、以下のタグを付けることをユーザーに許可します。このポリシーは、起動テンプレートにタグが指定されている場合に必要です。詳細については、「Amazon EC2 ユーザーガイド」の「作成時にリソースにタグを付けるアクセス許可を付与する」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:region:account-id:*/*",
            "Condition": {
                "StringEquals": { "ec2:CreateAction": "RunInstances" }
            }
        }
    ]
}
```

起動テンプレートの追加アクセス許可

コンソールユーザーには、ec2:DescribeLaunchTemplatesおよびec2:DescribeLaunchTemplateVersionsアクションへのアクセス許可を与える必要があります。これらのアクセス許可なしでは、起動テンプレートとネットワークオプションが Auto Scaling グループウィザードに読み込まれないため、ユーザーはウィザードをステップ実行できず、起動テンプレートを使用してインスタンスを起動することができません。これらの追加のアクションは、IAM ポリシーステートメントの Action要素で指定できます。

`ec2:RunInstances` と `iam:PassRole` の許可の検証

ユーザーは、Auto Scaling グループが使用する起動テンプレートのバージョンを指定できます。許可に応じて、これは特定の番号付きバージョン、または起動テンプレートの $Latest もしくは $Default バージョンになります。後者の場合は特に注意してください。これにより、制限することを意図していた ec2:RunInstances および iam:PassRole の許可がオーバーライドされる可能性があります。

このセクションでは、Auto Scaling グループで最新またはデフォルトバージョンの起動テンプレートを使用するシナリオについて説明します。

ユーザーが CreateAutoScalingGroup、UpdateAutoScalingGroup、または StartInstanceRefreshAPIs を呼び出すと、Amazon EC2 Auto Scaling は、リクエストを続行する前に、その時点で最新バージョンまたはデフォルトバージョンの起動テンプレートのバージョンに対してアクセス許可をチェックします。これにより、インスタンスの起動時に完了するアクション (ec2:RunInstances や iam:PassRole アクションなど) の許可が検証されます。これを実現するために、Amazon EC2 RunInstancesドライランコールを発行して、ユーザーが実際にリクエストを行うことなく、アクションに必要なアクセス許可を持っているかどうかを検証します。レスポンスが返されると、Amazon EC2 Auto Scaling によって読み取られます。ユーザーのアクセス許可で特定のアクションが許可されていない場合、Amazon EC2 Auto Scaling はリクエストに失敗し、不足しているアクセス許可に関する情報を含むエラーをユーザーに返します。

最初の検証とリクエストが完了すると、Amazon EC2 Auto Scaling は、サービスにリンクされたロールのアクセス許可を使用して、インスタンスが起動されるたびに、変更されても最新バージョンまたはデフォルトバージョンでインスタンスを起動します。つまり、起動テンプレートを使用しているユーザーは、 アクセスiam:PassRole許可がない場合でも、IAM ロールをインスタンスに渡すように更新できる可能性があります。

$Latest または $Default バージョンを使用するようにグループを設定するアクセス許可を持つ人を制限したい場合は、autoscaling:LaunchTemplateVersionSpecified 条件キーを使用します。これにより、Auto Scaling グループは、ユーザーが CreateAutoScalingGroupおよび APIs UpdateAutoScalingGroup を呼び出すときに、特定の番号付きバージョンのみを受け入れるようになります。この条件キーを IAM ポリシーに追加する方法の例については、「」を参照してください起動テンプレートとバージョン番号を要求する。

$Latest または $Default 起動テンプレートのバージョンを使用するように構成されているAuto Scalingグループについては、デフォルトの起動テンプレートのバージョンをユーザーが指定できる ec2:ModifyLaunchTemplate アクションを含め、起動テンプレートのバージョンを作成および管理できるユーザーを制限することを検討してください。詳細については、「Amazon EC2 ユーザーガイド」の「バージョニング許可の制御」を参照してください。

関連リソース

起動テンプレートと起動テンプレートのバージョンを表示、作成、削除するアクセス許可の詳細については、「Amazon EC2 ユーザーガイド」のIAM アクセス許可を使用して起動テンプレートへのアクセスを制御する」を参照してください。

RunInstances 呼び出しへのアクセスを制御するために使用できるリソースレベルのアクセス許可の詳細については、「サービス認可リファレンス」の「Amazon EC2 のアクション、リソース、および条件キー」を参照してください。

Amazon IAM インスタンスで実行されるアプリケーションの EC2 ロール

Amazon EC2 インスタンスで実行されるアプリケーションには、他の にアクセスするための認証情報が必要です AWS のサービス。これらの認証情報を安全な方法で提供するには、IAM ロールを使用します。このロールは、アプリケーションが他の AWS リソースにアクセスするときに使用できる一時的なアクセス許可を付与します。アプリケーションに許可される操作は、ロールのアクセス許可で決定されます。

Auto Scaling グループ内のインスタンス用に、起動テンプレートまたは起動設定を作成し、インスタンスに関連付けるインスタンスプロファイルを選択する必要があります。インスタンスプロファイルは、IAMの起動時に Amazon EC2 が Word ロールをインスタンスに渡すことを許可する IAM ロールのコンテナです。まず、 AWS リソースへのアクセスに必要なすべてのアクセス許可を持つ IAM ロールを作成します。次に、インスタンスプロファイルを作成し、そのプロファイルにロールを割り当てます。

> **Note**
>
> ベストプラクティスとして、アプリケーション AWS のサービス が必要とする他の に対する最小限のアクセス許可を持つようにロールを作成することを強くお勧めします。

内容

前提条件

Amazon IAM で実行されているアプリケーションが引き受けることができる EC2 ロールを作成します。適切なアクセス許可を選択すると、その後にロールが与えられたアプリケーションが、必要な特定の API 呼び出しを行うことができます。

AWS CLI または IAM の 1 つではなく AWS SDKs コンソールを使用する場合、コンソールは自動的にインスタンスプロファイルを作成し、対応するロールと同じ名前を付けます。

IAM ロールを作成するには (コンソール)

1. IAM で https://console.aws.amazon.com/iam/ コンソールを開きます。
2. 左側のナビゲーションペインで、[Roles] を選択します。
3. [ロールの作成] を選択します。
4. [Select trusted entity] (信頼されたエンティティの選択) で、[AWS のサービス] を選択します。
5. ユースケースでは、EC2 を選択し、次へを選択します。
6. 可能な場合は、アクセス許可ポリシーとして使用するポリシーを選択するか、[ポリシーの作成] を選択して新しいブラウザタブを開き、新しいポリシーをゼロから作成します。詳細については、IAM ユーザーガイド」の「Word ポリシーの作成」を参照してください。 IAM ポリシーを作成したら、そのタブを閉じて元のタブに戻ります。サービスに割り当てるアクセス許可ポリシーの横のチェックボックスをオンにします。
7. (オプション) アクセス許可の境界を設定します。これは、サービスロールに利用できる高度な機能です。詳細については、IAM ユーザーガイドの「Word エンティティのアクセス許可の境界」を参照してください。 IAM
8. [Next (次へ)] を選択します。
9. [Name, review, and create] (名前、確認、および作成) ページの [Role name] (ロール名) に、このロールの目的を識別するために役立つロール名を入力します。この名前は AWS アカウント内で一意である必要があります。他の AWS リソースがロールを参照する可能性があるため、ロールの作成後にロールの名前を編集することはできません。
10. ロールを確認したら、[Create role] (ロールを作成) を選択します。

IAMのアクセス許可

IAM アイデンティティベースのポリシーを使用して、新しい IAM ロールへのアクセスを制御します。アクセス`iam:PassRole`許可は、インスタンスプロファイルを指定する起動テンプレートを使用して Auto Scaling グループを作成または更新する IAM ID (ユーザーまたはロール) に必要です。

次のポリシー例では、名前が で始まる IAM ロールのみを渡すアクセス許可を付与します**`qateam-`**。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/qateam-*",
```

```
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        }
    ]
}
```

> **Important**
>
> Amazon EC2 Auto Scaling が起動テンプレートを使用する Auto Scaling グループの `iam:PassRole`アクションのアクセス許可を検証する方法については、「」を参照してくださいec2:RunInstances と iam:PassRole の許可の検証。

起動テンプレートの作成

を使用して起動テンプレートを作成するときは AWS Management Console、詳細セクションで、IAM インスタンスプロファイルからロールを選択します。詳細については、「詳細設定を使用して起動テンプレートを作成する」を参照してください。

の create-launch-template コマンドを使用して起動テンプレートを作成するときは AWS CLI、次の例に示すように、IAM ロールのインスタンスプロファイル名を指定します。

```
aws ec2 create-launch-template --launch-template-name my-lt-with-instance-profile --
version-description version1 \
--launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","IamInstanceProfile":
{"Name":"my-instance-profile"}}'
```

以下も参照してください。

Amazon EC2 の IAM ロールについて学び、使用を開始するのに役立つ詳細については、以下を参照してください。

- IAM 「Amazon EC2 ユーザーガイド」の「Amazon Word の Word ロールEC2」

- Word ユーザーガイドの「インスタンスプロファイルの使用」と「Word ロールを使用して Amazon Word インスタンスで実行されているアプリケーションにアクセス許可を付与するIAMEC2IAM」

Amazon EC2 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AWS のサービス プログラムによる範囲内」「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択してください。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading AWS Artifact Reports 」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。 では、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。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスの機能をデプロイする手順を示します。
- Amazon Web Services での HIPAA Security and Compliance 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**
>
> すべての AWS のサービス が HIPAA 対応であるわけではありません。詳細については、HIPAA 対象サービスリファレンス」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把握できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「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、不審なアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、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に対応するのに役立ちます。
- 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。

PCI コンプライアンス DSS

Amazon EC2 Auto Scaling は、マーチャントまたはサービスプロバイダーによるクレジットカードデータの処理、保存、および送信をサポートし、Payment Card Industry (PCI) Data Security Standard (DSS) に準拠していることが確認されています。PCI Compliance Package のコピーをリクエストする方法など、 AWS PCI DSS の詳細については、DSSPCI Level 1」を参照してください。

AWS ワークロードの DSS PCI コンプライアンスの達成については、次のコンプライアンスガイドを参照してください。

- での Payment Card Industry Data Security Standard (PCI DSS) 3.2.1 AWS

Amazon EC2 Auto Scaling エンドポイントとインターフェイス VPC エンドポイント

インターフェイス VPC エンドポイントを使用するように Amazon EC2 Auto Scaling を設定することで、VPC のセキュリティ体制を強化できます。インターフェイスエンドポイントは、VPC と Amazon EC2 Auto Scaling 間のすべてのネットワークトラフィックを AWS ネットワークに制限することにより、プライベートで Amazon EC2 Auto Scaling API にアクセスすることを可能にす

る、AWS PrivateLink を活用します。インターフェイスエンドポイントでは、インターネットゲートウェイ、NAT デバイス、または仮想プライベートゲートウェイも必要ありません。

AWS PrivateLink の設定は要件ではありませんが、推奨されます。AWS PrivateLink と VPC エンドポイントの詳細については、AWS PrivateLink ガイドの「AWS PrivateLink とは」を参照してください。

トピック

- 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

以下のサービス名を使用して、Amazon EC2 Auto Scaling のエンドポイントを作成します。

```
com.amazonaws.region.autoscaling
```

詳細については、「AWS PrivateLink ガイド」の「Access an AWS service using an interface VPC endpoint」(インターフェイス VPC エンドポイントを使用してのサービスにアクセスする) を参照してください。

Amazon EC2 Auto Scaling 設定を変更する必要はありません。Amazon EC2 Auto Scaling は、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、使用されている方のエンドポイントを使用して、AWS の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

Amazon EC2 Auto Scaling API へのアクセスをコントロールするために、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": "autoscaling:DeleteScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

詳細については、『AWS PrivateLink ガイド』の「Control access to VPC endpoints using endpoint policies (エンドポイントポリシーを使用して VPC エンドポイントへのアクセスをコントロールする)」を参照してください。

でのこのサービスの使用 AWS SDK

AWS Software Development Kit (SDKs) は、多くの一般的なプログラミング言語で使用できます。各SDKにはAPI、開発者が好みの言語でアプリケーションを簡単に構築できるようにする 、コード例、およびドキュメントが用意されています。

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

このサービスに固有の例については、「を使用した Auto Scaling のコード例 AWS SDKs」を参照してください。

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

を使用した Auto Scaling のコード例 AWS SDKs

次のコード例は、 AWS ソフトウェア開発キット () で Auto Scaling を使用する方法を示しています SDK。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

開始方法

こんにちは、Auto Scaling

次のコード例は、Auto Scaling の使用を開始する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
```

```
public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

        if (response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
        }
    }
}
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMakeファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_autoscaling.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
&autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }
```

```
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
```

```
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
```

```
    hello_autoscaling(boto3.client("autoscaling"))
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Ruby

SDK Ruby 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
```

```
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- API 詳細については、「 AWS SDK for Ruby APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Rust

SDK Rust の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
```

```
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API 詳細については、AWS SDKRust APIリファレンスのDescribeAutoScalingGroups「」の「」を参照してください。

コードの例

- を使用した Auto Scaling の基本的な例 AWS SDKs
  - こんにちは、Auto Scaling
  - を使用した Auto Scaling の基本について説明します。 AWS SDK
  - を使用した Auto Scaling のアクション AWS SDKs
    - AttachInstances で を使用する CLI
    - または AttachLoadBalancerTargetGroupsで を使用する AWS SDK CLI
    - AttachLoadBalancers で を使用する CLI
    - CompleteLifecycleAction で を使用する CLI
    - または CreateAutoScalingGroupで を使用する AWS SDK CLI
    - CreateLaunchConfiguration で を使用する CLI
    - CreateOrUpdateTags で を使用する CLI
    - または DeleteAutoScalingGroupで を使用する AWS SDK CLI
    - DeleteLaunchConfiguration で を使用する CLI
    - DeleteLifecycleHook で を使用する CLI
    - DeleteNotificationConfiguration で を使用する CLI
    - DeletePolicy で を使用する CLI

- DeleteScheduledAction で を使用する CLI
- DeleteTags で を使用する CLI
- DescribeAccountLimits で を使用する CLI
- DescribeAdjustmentTypes で を使用する CLI
- または DescribeAutoScalingGroupsで を使用する AWS SDK CLI
- または DescribeAutoScalingInstancesで を使用する AWS SDK CLI
- DescribeAutoScalingNotificationTypes で を使用する CLI
- DescribeLaunchConfigurations で を使用する CLI
- DescribeLifecycleHookTypes で を使用する CLI
- DescribeLifecycleHooks で を使用する CLI
- DescribeLoadBalancers で を使用する CLI
- DescribeMetricCollectionTypes で を使用する CLI
- DescribeNotificationConfigurations で を使用する CLI
- DescribePolicies で を使用する CLI
- または DescribeScalingActivitiesAWS SDKで を使用する CLI
- DescribeScalingProcessTypes で を使用する CLI
- DescribeScheduledActions で を使用する CLI
- DescribeTags で を使用する CLI
- DescribeTerminationPolicyTypes で を使用する CLI
- DetachInstances で を使用する CLI
- DetachLoadBalancers で を使用する CLI
- または DisableMetricsCollectionで を使用する AWS SDK CLI
- または EnableMetricsCollectionで を使用する AWS SDK CLI
- EnterStandby で を使用する CLI
- ExecutePolicy で を使用する CLI
- ExitStandby で を使用する CLI
- PutLifecycleHook で を使用する CLI
- PutNotificationConfiguration で を使用する CLI
- PutScalingPolicy で を使用する CLI
- PutScheduledUpdateGroupAction で を使用する CLI

    - RecordLifecycleActionHeartbeat で を使用する CLI
    - ResumeProcesses で を使用する CLI
    - または SetDesiredCapacityAWS SDKで を使用する CLI
    - SetInstanceHealth で を使用する CLI
    - SetInstanceProtection で を使用する CLI
    - SuspendProcesses で を使用する CLI
    - または TerminateInstanceInAutoScalingGroupで を使用する AWS SDK CLI
    - または UpdateAutoScalingGroupAWS SDKで を使用する CLI
- を使用した Auto Scaling のシナリオ AWS SDKs
    - を使用して回復力のあるサービスを構築および管理します。 AWS SDK

を使用した Auto Scaling の基本的な例 AWS SDKs

次のコード例は、 で AWS Amazon EC2 Auto Scaling の基本を使用する方法を示していますSDKs。

例

- こんにちは、Auto Scaling
- を使用した Auto Scaling の基本について説明します。 AWS SDK
- を使用した Auto Scaling のアクション AWS SDKs
    - AttachInstances で を使用する CLI
    - または AttachLoadBalancerTargetGroupsで を使用する AWS SDK CLI
    - AttachLoadBalancers で を使用する CLI
    - CompleteLifecycleAction で を使用する CLI
    - または CreateAutoScalingGroupで を使用する AWS SDK CLI
    - CreateLaunchConfiguration で を使用する CLI
    - CreateOrUpdateTags で を使用する CLI
    - または DeleteAutoScalingGroupで を使用する AWS SDK CLI
    - DeleteLaunchConfiguration で を使用する CLI
    - DeleteLifecycleHook で を使用する CLI
    - DeleteNotificationConfiguration で を使用する CLI
    - DeletePolicy で を使用する CLI

- DeleteScheduledAction で を使用する CLI
- DeleteTags で を使用する CLI
- DescribeAccountLimits で を使用する CLI
- DescribeAdjustmentTypes で を使用する CLI
- または DescribeAutoScalingGroupsで を使用する AWS SDK CLI
- または DescribeAutoScalingInstancesで を使用する AWS SDK CLI
- DescribeAutoScalingNotificationTypes で を使用する CLI
- DescribeLaunchConfigurations で を使用する CLI
- DescribeLifecycleHookTypes で を使用する CLI
- DescribeLifecycleHooks で を使用する CLI
- DescribeLoadBalancers で を使用する CLI
- DescribeMetricCollectionTypes で を使用する CLI
- DescribeNotificationConfigurations で を使用する CLI
- DescribePolicies で を使用する CLI
- または DescribeScalingActivitiesAWS SDKで を使用する CLI
- DescribeScalingProcessTypes で を使用する CLI
- DescribeScheduledActions で を使用する CLI
- DescribeTags で を使用する CLI
- DescribeTerminationPolicyTypes で を使用する CLI
- DetachInstances で を使用する CLI
- DetachLoadBalancers で を使用する CLI
- または DisableMetricsCollectionで を使用する AWS SDK CLI
- または EnableMetricsCollectionで を使用する AWS SDK CLI
- EnterStandby で を使用する CLI
- ExecutePolicy で を使用する CLI
- ExitStandby で を使用する CLI
- PutLifecycleHook で を使用する CLI
- PutNotificationConfiguration で を使用する CLI
- PutScalingPolicy で を使用する CLI
- PutScheduledUpdateGroupAction で を使用する CLI

- RecordLifecycleActionHeartbeat で を使用する CLI
- ResumeProcesses で を使用する CLI
- または SetDesiredCapacityAWS SDKで を使用する CLI
- SetInstanceHealth で を使用する CLI
- SetInstanceProtection で を使用する CLI
- SuspendProcesses で を使用する CLI
- または TerminateInstanceInAutoScalingGroupで を使用する AWS SDK CLI
- または UpdateAutoScalingGroupAWS SDKで を使用する CLI

こんにちは、Auto Scaling

次のコード例は、Auto Scaling の使用を開始する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();
```

```
        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

        if (response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
        }
    }
}
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMakeファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_autoscaling.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
```

```
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
```

```
        });
    }
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Ruby

SDK Ruby 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
```

```
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- API 詳細については、「 AWS SDK for Ruby APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Rust

SDK Rust の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
```

```
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API 詳細については、DescribeAutoScalingGroups「」のAWS SDK「Rust APIリファレンス」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用した Auto Scaling の基本について説明します。 AWS SDK

次のコード例は、以下を実行する方法を示しています。

- 起動テンプレートとアベイラビリティーゾーンを使用して Amazon EC2 Auto Scaling グループを作成し、実行中のインスタンスに関する情報を取得します。
- Amazon CloudWatch メトリクス収集を有効にします。
- グループの希望するキャパシティを更新し、インスタンスが起動するのを待ちます。
- グループ内の最も古いインスタンスを削除します。
- ユーザーのリクエストやキャパシティの変更に応じて発生するスケーリングアクティビティを一覧表示します。
- CloudWatch メトリクスの統計を取得し、リソースをクリーンアップします。

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when
deleting the
        // launch template at the end of the application.
        var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);
```

```
        // Confirm that the template was created by asking for a description of
it.
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones.First().ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);
```

```
        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
        Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups!.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups is not null)
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
                var instances = group.Instances;
```

```
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is
{instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities is not null)
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are
{activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
        var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        Console.WriteLine($"Metrics collected for {groupName}:");
        metrics.ForEach(metric =>
        {
            Console.Write($"Metric name: {metric.MetricName}\t");
            Console.WriteLine($"Namespace: {metric.Namespace}");
        });

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        Console.WriteLine("Details for the metrics collected:");
        dataPoints.ForEach(detail =>
        {
            Console.WriteLine(detail);
        });

        // Disable metrics collection.
```

```
        Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
```

```
        var deletedLaunchTemplateName = await
 ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following
 operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
 Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the
 group.");
```

```
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
 for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
 have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
 group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
```

```
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

起動テンプレートとメトリクスを管理するためにシナリオによって呼び出される関数を定義します。これらの関数は、Auto Scaling、Amazon EC2、および CloudWatch アクションをラップします。

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }


    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
        var templateSpecification = new LaunchTemplateSpecification
```

```
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };

        var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
```

```
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Amazon.AutoScaling.Model.Activity>>
 DescribeScalingActivitiesAsync(
        string groupName)
    {
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        return response.Activities;
    }



    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
```

```
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>?> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        var groups = response.AutoScalingGroups;

        return groups;
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }


    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
 _amazonAutoScaling.DisableMetricsCollectionAsync(request);
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
```

```
        };

        var response = await
 _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
 {desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
```

```
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
```

```
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
```

```
    /// Delete an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
 launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }
```

```
        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}


namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
```

```
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
    public async Task<List<Amazon.CloudWatch.Model.Metric>>
 GetCloudWatchMetricsAsync(string groupName)
    {
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);

        return response.Metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = $"{groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
```

```
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);

        return response.Datapoints;
    }

}
```

- API 詳細については、「 AWS SDK for .NET APIリファレンス」の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
 Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
```

```
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." <<
std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
"
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    else { // 2.  Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
                 "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already
exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
                ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "EC2 instances can be created in the following
Availability Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
```

```
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
```

```
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
            "Do you want to collect metrics about the A"
            "Auto Scaling group during this demo (y/n)?  ");
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }
```

```
   std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
             "' from 1 to 3." << std::endl;
   askQuestion("Press enter to continue:  ", alwaysTrueTest);
   // 8. Update the Auto Scaling group, setting a new maximum size.
   {
       Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
       request.SetAutoScalingGroupName(groupName);
       request.SetMaxSize(3);

       Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
               autoScalingClient.UpdateAutoScalingGroup(request);

       if (!outcome.IsSuccess()) {
           std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
           return false;
       }
   }

   if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                          autoScalingClient)) {
       if (!autoScalingGroups.empty()) {
           const auto &instances = autoScalingGroups[0].GetInstances();
           std::cout
                   << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                   << std::endl;
           logAutoScalingGroupInfo(autoScalingGroups);
       }
       else {
           std::cerr
                   << "No EC2 launch groups were retrieved from DescribeGroup
request."
                   << std::endl;
           cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
           return false;
       }
   }
```

```
   std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
             << std::endl;
   std::cout << "Let's update the desired capacity in '" << groupName <<
             "' from 1 to 2." << std::endl;
   askQuestion("Press enter to continue:  ", alwaysTrueTest);
   //  9. Update the Auto Scaling group, setting a new desired capacity.
   {
       Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
       request.SetAutoScalingGroupName(groupName);
       request.SetDesiredCapacity(2);

       Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
               autoScalingClient.SetDesiredCapacity(request);

       if (!outcome.IsSuccess()) {
           std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
           return false;
       }
   }

   if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                          autoScalingClient)) {
       if (!autoScalingGroups.empty()) {
           std::cout
                   << "Here is the current state of the group." << std::endl;
           logAutoScalingGroupInfo(autoScalingGroups);
       }
       else {
           std::cerr
                   << "No EC2 launch groups were retrieved from DescribeGroup
request."
                   << std::endl;
           cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
           return false;
       }
   }

   std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
```

```
   waitForInstances(groupName, autoScalingGroups, autoScalingClient);

   std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
             << std::endl;

   std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
"."
             << std::endl;
   std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
             << "to replace the terminated EC2 instance."
             << std::endl;
   std::cout << "The currently running EC2 instances are:" << std::endl;

   if (autoScalingGroups.empty()) {
       std::cerr << "Error describing groups. No groups returned." << std::endl;
       cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
       return false;
   }

   int instanceNumber = 1;
   Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
           autoScalingGroups[0].GetInstances());
   for (const Aws::String &instanceID: instanceIDs) {
       std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
       ++instanceNumber;
   }

   instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                           1,

static_cast<int>(instanceIDs.size()));

   // 10. Terminate an EC2 instance in the Auto Scaling group.
   {
       Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
       request.SetInstanceId(instanceIDs[instanceNumber - 1]);
       request.SetShouldDecrementDesiredCapacity(false);

       Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
```

```
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group
'"
              << groupName << "'."
              << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
```

```
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);

    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);
```

```
        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                            const
 Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
```

```
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." <<
std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
&details: instancesDetails) {
```

```
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const
 Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client)
 {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
```

```
        (askYesNoQuestion(
                Aws::String("Delete the Auto Scaling group '") + groupName +
                "'  (y/n)?"))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                    autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                        << "The minimum size and desired capacity of the Auto
Scaling group "
                        << "was set to zero before terminating the instances."
                        << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                          << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }

        Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
        if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                               autoScalingClient)) {
            if (!autoScalingGroups.empty()) {
                Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                        autoScalingGroups[0].GetInstances());
                for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                    request.SetInstanceId(instanceID);
                    request.SetShouldDecrementDesiredCapacity(true);


Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =

autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                    request);
```

```
                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
                }
            }

            std::cout
                    << "Waiting for the EC2 instances to terminate before
deleting the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }

        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
```

```
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was
 deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                        const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
```

```
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 詳細については、「 AWS SDK for C++ APIリファレンス」の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
```

```
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred
for the group");
```

```
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling
group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
```

```
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
   }

   public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .maxRecords(10)
                   .build();

           DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
           List<AutoScalingGroup> groups = response.autoScalingGroups();
           for (AutoScalingGroup group : groups) {
               System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
           }

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           String instanceId = "";
           DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .build();

           DescribeAutoScalingGroupsResponse response = autoScalingClient
                   .describeAutoScalingGroups(scalingGroupsRequest);
           List<AutoScalingGroup> groups = response.autoScalingGroups();
           for (AutoScalingGroup group : groups) {
               System.out.println("The group name is " +
group.autoScalingGroupName());
               System.out.println("The group ARN is " +
group.autoScalingGroupARN());
               List<Instance> instances = group.instances();
```

```
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();
```

```
autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
        try {
            DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
            System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
            System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();
```

```
            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
```

```
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンス」の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
 in the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
 vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)
```

```
    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of
 the groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
```

```
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
```

```
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
```

```
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API 詳細については、AWS SDKKotlin APIリファレンスの の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;
```

```
class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an
 Auto Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
```

```
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you
 created in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
```

```
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the
 group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups:
 {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
```

```
        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
```

```
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that
 have occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']}
 recorded:\n";
```

```
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);
```

```
        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- API 詳細については、「 AWS SDK for PHP APIリファレンス」の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
 None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
 instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
 new one: "
    )
```

```
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
```

```
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
        print(f"Metrics enabled for {group_name}.")
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
```

```
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
    print(f"Here's the state of {group_name}:")
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities),
group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)

    if use_metrics:
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value":
group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
```

```
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
                metric_data = svc_helper.get_metric_statistics(
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
```

```
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
 demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

起動テンプレートとメトリクスを管理するためにシナリオによって呼び出される関数を定義します。これらの関数は Amazon EC2および CloudWatch アクションをラップします。

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
 instances
```

```
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
            logger.info("Launch template %s retrieved successfully.",
template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.",
template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.

        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
```

```
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID
%s.",
                template_name,
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:

self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
            )
            return metrics
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


def print_simplified_group(group: dict) -> None:
```

```
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)


def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
 list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
```

```
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
            f"Here are the details of the instance{'s' if len(instances) > 1 else
''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- API 詳細については、 AWS SDK for Python (Boto3) APIリファレンスの以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Rust

SDK Rust の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
```

```
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
 group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step
 0. Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match
 AutoScalingScenario::prepare_scenario(&shared_config).await {
        Ok(scenario) => scenario,
```

```
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
```

```
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
    )
    .await;

    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
the group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
instance",
```

```
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
        );
    }

    // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }
```

```
    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;


use std::{
    error::Error,
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
 seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
 a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
```

```
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}
```

```
pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }
```

```
        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
```

```
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
```

```
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
template.
        //   * Recommended: InstanceType='t1.micro',
ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
            .create_launch_template()
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
```

```
        //   You can use EC2.describe_availability_zones() to get a list of AZs
(you have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
            .min_size(1)
            .set_availability_zones(Some(availability_zones))
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
                ));
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
```

```
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };

        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
                    "Failed to enable metrics collections for group",
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
```

```
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
            (Ok(_), Ok(_)) => Ok(()),
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
        } else {
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
```

```
                    .describe_auto_scaling_groups()

.auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting:
{}",
                                    group.status().unwrap_or("unknown error")
                                )));
                                return Err(errors);
                            }
                        }
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing
after three rounds",
            )])
        }
    }

    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
```

```
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });
```

```
        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
```

```
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
            trace!(
                "Waiting for no scaling found {} activities",
                activities.len()
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
{
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
            group = self.get_group().await?;
            count = count_group_instances(&group);
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
```

```
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
 instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
```

```
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
```

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
        //   UpdateAutoScalingGroup with MinSize=0
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(0)
            .desired_capacity(0)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
```

```
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- API 詳細については、AWS SDKRust APIリファレンスの の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity

- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用した Auto Scaling のアクション AWS SDKs

次のコード例は、 を使用して個々の Auto Scaling アクションを実行する方法を示しています AWS SDKs。各例には へのリンクが含まれており GitHub、コードの設定と実行の手順を確認できます。

これらの抜粋は Auto Scaling を呼び出しAPI、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。アクションは を使用した Auto Scaling のシナリオ AWS SDKs のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「Amazon EC2 Auto Scaling APIリファレンス」を参照してください。

例

- AttachInstances で を使用する CLI
- または AttachLoadBalancerTargetGroupsで を使用する AWS SDK CLI
- AttachLoadBalancers で を使用する CLI
- CompleteLifecycleAction で を使用する CLI
- または CreateAutoScalingGroupで を使用する AWS SDK CLI
- CreateLaunchConfiguration で を使用する CLI
- CreateOrUpdateTags で を使用する CLI
- または DeleteAutoScalingGroupで を使用する AWS SDK CLI
- DeleteLaunchConfiguration で を使用する CLI
- DeleteLifecycleHook で を使用する CLI
- DeleteNotificationConfiguration で を使用する CLI
- DeletePolicy で を使用する CLI
- DeleteScheduledAction で を使用する CLI
- DeleteTags で を使用する CLI
- DescribeAccountLimits で を使用する CLI

- DescribeAdjustmentTypes で を使用する CLI
- または DescribeAutoScalingGroupsで を使用する AWS SDK CLI
- または DescribeAutoScalingInstancesで を使用する AWS SDK CLI
- DescribeAutoScalingNotificationTypes で を使用する CLI
- DescribeLaunchConfigurations で を使用する CLI
- DescribeLifecycleHookTypes で を使用する CLI
- DescribeLifecycleHooks で を使用する CLI
- DescribeLoadBalancers で を使用する CLI
- DescribeMetricCollectionTypes で を使用する CLI
- DescribeNotificationConfigurations で を使用する CLI
- DescribePolicies で を使用する CLI
- または DescribeScalingActivitiesAWS SDKで を使用する CLI
- DescribeScalingProcessTypes で を使用する CLI
- DescribeScheduledActions で を使用する CLI
- DescribeTags で を使用する CLI
- DescribeTerminationPolicyTypes で を使用する CLI
- DetachInstances で を使用する CLI
- DetachLoadBalancers で を使用する CLI
- または DisableMetricsCollectionで を使用する AWS SDK CLI
- または EnableMetricsCollectionで を使用する AWS SDK CLI
- EnterStandby で を使用する CLI
- ExecutePolicy で を使用する CLI
- ExitStandby で を使用する CLI
- PutLifecycleHook で を使用する CLI
- PutNotificationConfiguration で を使用する CLI
- PutScalingPolicy で を使用する CLI
- PutScheduledUpdateGroupAction で を使用する CLI
- RecordLifecycleActionHeartbeat で を使用する CLI
- ResumeProcesses で を使用する CLI
- または SetDesiredCapacityAWS SDKで を使用する CLI

- SetInstanceHealth で を使用する CLI
- SetInstanceProtection で を使用する CLI
- SuspendProcesses で を使用する CLI
- または TerminateInstanceInAutoScalingGroupで を使用する AWS SDK CLI
- または UpdateAutoScalingGroupAWS SDKで を使用する CLI

`AttachInstances` で を使用する CLI

以下のコード例は、`AttachInstances` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループにインスタンスをアタッチするには

この例では、指定されたインスタンスを、指定された Auto Scaling グループにアタッチします。

```
aws autoscaling attach-instances \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

このコマンドでは何も出力されません。

- API 詳細については、AWS CLI 「 コマンドリファレンスAttachInstances」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したインスタンスを、指定した Auto Scaling グループにアタッチします。Auto Scaling は、Auto Scaling グループの希望する容量を自動的に増やします。

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- API 詳細については、「 コマンドレットリファレンスAttachInstances」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **AttachLoadBalancerTargetGroups**で を使用する AWS SDK CLI

以下のコード例は、AttachLoadBalancerTargetGroups の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- レジリエントなサービスの構築と管理

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
```

```
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスAttachLoadBalancerTargetGroups」の「」を参照してください。

CLI

AWS CLI

ターゲットグループを Auto Scaling グループにアタッチする方法

この例では、指定されたターゲットグループを、指定された Auto Scaling グループにアタッチします。

```
aws autoscaling attach-load-balancer-target-groups \
    --auto-scaling-group-name my-asg \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」のElastic Load Balancing とAmazon Auto Scaling」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスAttachLoadBalancerTargetGroups」の「」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    const client = new AutoScalingClient({});
    await client.send(
```

```
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
```

- API 詳細については、「 AWS SDK for JavaScript APIリファレンスAttachLoadBalancerTargetGroups」の「」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
```

```
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- API 詳細については、「 for AWS SDK Python (Boto3) APIリファレンスAttachLoadBalancerTargetGroups」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`AttachLoadBalancers` で を使用する CLI

以下のコード例は、`AttachLoadBalancers` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループに Classic Load Balancer をアタッチするには

この例では、指定された Classic Load Balancer を、指定された Auto Scaling グループにアタッチします。

```
aws autoscaling attach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」のElastic Load Balancing と Amazon Auto Scaling」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスAttachLoadBalancers」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したロードバランサーを、指定した Auto Scaling グループにアタッチします。

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- API 詳細については、「 コマンドレットリファレンスAttachLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`CompleteLifecycleAction` で を使用する CLI

以下のコード例は、`CompleteLifecycleAction` の使用方法を示しています。

CLI

AWS CLI

ライフサイクルアクションを完了するには

この例では、インスタンスの起動または終了を完了できるように、指定されたライフサイクルアクションが完了したことを Amazon EC2 Auto Scaling に通知します。

```
aws autoscaling complete-lifecycle-action \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-result CONTINUE \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスCompleteLifecycleAction」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したライフサイクルアクションを完了します。

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- API 詳細については、「 コマンドレットリファレンスCompleteLifecycleAction」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **CreateAutoScalingGroup**で を使用する AWS SDK CLI

以下のコード例は、CreateAutoScalingGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ
- レジリエントなサービスの構築と管理

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
```

```
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };

        var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスCreateAutoScalingGroup」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスCreateAutoScalingGroup」の「」を参照してください。

CLI

AWS CLI

例 1: Auto Scaling グループを作成する方法

次の create-auto-scaling-group の例では、リージョン内の複数のアベイラビリティーゾーンのサブネット内に Auto Scaling グループを作成します。インスタンスは、指定された起動テンプレートのデフォルトバージョンで起動されます。終了ポリシーやヘルスチェック構成など、他のほとんどの構成にはデフォルトが使用されることに注意してください。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループ」を参照してください。 EC2 Auto Scaling

例 2: Application Load Balancer、Network Load Balancer、または Gateway Load Balancer をアタッチする方法

この例では、予想されるトラフィックをサポートするロードバランサーARNのターゲットグループの を指定します。ヘルスチェックタイプは、Elastic Load Balancing がインスタンスを異常として報告したときに、Auto Scaling グループがそのインスタンスを置き換えるよう ELB を指定します。このコマンドは、ヘルスチェックの猶予期間 (600 秒) も定義します。猶予期間は、新しく起動したインスタンスが早期に終了するのを防ぐのに役立ちます。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/943f017f100becff \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」のElastic Load Balancing と Amazon Auto Scaling」を参照してください。 EC2 Auto Scaling

例 3: プレイスメントグループを指定し、起動テンプレートの最新バージョンを使用する方法

この例では、単一のアベイラビリティーゾーン内のプレイスメントグループ内でインスタンスを起動します。これは、HPCワークロードを持つ低レイテンシーのグループに役立ちます。この例では、グループの最小サイズ、最大サイズ、希望する容量も指定しています。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \
    --min-size 1 \
    --max-size 5 \
    --desired-capacity 3 \
    --placement-group my-placement-group \
    --vpc-zone-identifier "subnet-6194ea3b"
```

このコマンドでは何も出力されません。

詳細については、「Linux インスタンス用 Amazon ユーザーガイド」の「プレイスメントグループ」を参照してください。 EC2

例 4: 単一のインスタンスの Auto Scaling グループを指定し、特定のバージョンの起動テンプレートを使用する方法

この例では、単一のインスタンスが強制的に実行されるように、最小容量と最大容量を 1 に設定した Auto Scaling グループを作成します。コマンドは、既存の の ID がENI指定されている起動テンプレートの v1 も指定します。eth0 ENIの既存の を指定する起動テンプレートを使用する場合は、リクエストでサブネット ID を指定せずに、ネットワークインターフェイスに一致する Auto Scaling グループのアベイラビリティーゾーンを指定する必要があります。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
 \
    --min-size 1 \
    --max-size 1 \
    --availability-zones us-west-2a
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループ」を参照してください。 EC2 Auto Scaling

例 5: 別の終了ポリシーを指定する方法

この例では、起動構成を使用して Auto Scaling グループを作成し、最も古いインスタンスを最初に終了するように終了ポリシーを構成します。またこのコマンドは、`Role` キーと `WebServer` 値を使用して、グループとインスタンスにタグを適用します。

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-configuration-name my-lc \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "OldestInstance" \
    --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling 終了ポリシーの使用」を参照してください。 EC2 Auto Scaling

例 6: 起動ライフサイクルフックを指定する方法

この例では、インスタンス起動時のカスタムアクションをサポートするライフサイクルフックで Auto Scaling グループを設定します。

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

`config.json` ファイルの内容。

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
```

```
        "LifecycleHookName": "my-launch-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
        "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-
queue",
        "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",
        "NotificationMetadata": "SQS message metadata",
        "HeartbeatTimeout": 4800,
        "DefaultResult": "ABANDON"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": [{
        "ResourceType": "auto-scaling-group",
        "ResourceId": "my-asg",
        "PropagateAtLaunch": true,
        "Value": "test",
        "Key": "environment"
    }]
}
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

例 7: 終了ライフサイクルフックを指定する方法

次の例では、インスタンス終了時のカスタムアクションをサポートするライフサイクルフックで Auto Scaling グループを設定します。

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

config.json の内容:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
```

```
        "LifecycleHookName": "my-termination-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
        "HeartbeatTimeout": 120,
        "DefaultResult": "CONTINUE"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067"
    ],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

例 8: カスタム終了ポリシーを指定する方法

この例では、スケールイン時にどのインスタンスを安全に終了できるかを Amazon EC2 Auto Scaling に指示するカスタム Lambda 関数終了ポリシーを指定する Auto Scaling グループを作成します。 Auto Scaling

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "arn:aws:lambda:us-
west-2:123456789012:function:HelloFunction:prod" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Lambda を使用したカスタム終了ポリシーの作成」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスCreateAutoScalingGroup」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
 <vpcZoneId>

                Where:
```

```
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
```

```
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスCreateAutoScalingGroup」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
```

```
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- API 詳細については、Kotlin リファレンスのCreateAutoScalingGroup「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
            'LaunchTemplate' => [
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスCreateAutoScalingGroup」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した名前と属性で Auto Scaling グループを作成します。デフォルトの希望する容量は最小サイズになります。したがって、この Auto Scaling グループは、指定した 2 つのアベイラビリティーゾーンのそれぞれに 1 つずつ、合計 2 つのインスタンスを起動します。

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- API 詳細については、「 コマンドレットリファレンスCreateAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
 created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
 template.
                                     The launch template specifies the
 configuration of
                                     instances that are created by auto scaling
 activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
```

```
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
 group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )

            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
 exists. "
                    "Please use a different name or update the existing group.",
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch
 configurations. "
                    "Consider deleting unused resources or request a limit
 increase. "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
ec2-auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスCreateAutoScalingGroup」の「」を参照してください。

Rust

SDK Rust の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
 {
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- API 詳細については、AWS SDKRust APIリファレンスのCreateAutoScalingGroup「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`CreateLaunchConfiguration` で を使用する CLI

以下のコード例は、`CreateLaunchConfiguration` の使用方法を示しています。

CLI

AWS CLI

例 1: 起動設定を作成するには

この例では、シンプルな起動構成を作成します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「起動設定の作成」を参照してください。

例 2: セキュリティグループ、キーペア、ブートラッピングスクリプトを使用して起動構成を作成する方法

この例では、セキュリティグループ、キーペア、ユーザーデータに含まれるブートラッピングスクリプトを使用して起動構成を作成します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --security-groups sg-eb2af88example \
    --key-name my-key-pair \
    --user-data file://myuserdata.txt
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「起動設定の作成」を参照してください。

例 3: IAMロールを使用して起動設定を作成するには

この例では、 IAMロールのインスタンスプロファイル名を使用して起動設定を作成します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --iam-instance-profile my-autoscaling-role
```

このコマンドでは何も出力されません。

詳細については、IAM「Amazon Auto Scaling ユーザーガイド」の「Amazon EC2インスタンスで実行されるアプリケーションのロール」を参照してください。 EC2 Auto Scaling

例 4: 起動構成で詳細モニタリングを有効にする方法

この例では、EC2詳細モニタリングを有効にした起動設定を作成し、1 分 CloudWatch 間隔でEC2メトリクスを に送信します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --instance-monitoring Enabled=true
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling インスタンスのモニタリングの設定」を参照してください。 EC2 Auto Scaling

例 5: スポットインスタンスを起動する起動構成を作成する方法

この例では、スポットインスタンスを唯一の購入オプションとして使用する起動構成を作成します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --spot-price "0.50"
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「スポットインスタンスのリクエスト」を参照してください。 EC2 Auto Scaling

例 6: EC2インスタンスを使用して起動設定を作成するには

この例では、既存のインスタンスの属性に基づいて起動構成を作成します。これにより、プレイスメントテナンシーと、 `--placement-tenancy` および `--no-associate-public-ip-address`オプションを含めることでパブリック IP アドレスが設定されるかどうかが上書きされます。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc-from-instance \
    --instance-id i-0123a456700123456 \
    --instance-type m5.large \
    --no-associate-public-ip-address \
    --placement-tenancy dedicated
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」のEC2「インスタンスを使用した起動設定の作成」を参照してください。

例 7: Amazon EBSボリュームのブロックデバイスマッピングを使用して起動設定を作成するには

この例では、デバイス名とEBSgp3ボリュームサイズが 20 の Amazon `/dev/sdh`ボリュームのブロックデバイスマッピングを使用して起動設定を作成します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdh","Ebs":
{"VolumeSize":20,"VolumeType":"gp3"}}]'
```

このコマンドでは何も出力されません。

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

JSON形式のパラメータ値を引用するための構文については、AWS 「 コマンドラインインターフェイスユーザーガイド」の「 の文字列での引用符の使用 AWS CLI」を参照してください。

例 8: インスタンスストアボリュームのブロックデバイスマッピングを使用して起動構成を作成する方法

この例では、デバイス名「/dev/sdc」のインスタンスストアボリュームとして ephemeral1 で起動構成を作成します。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/
sdc","VirtualName":"ephemeral1"}]'
```

このコマンドでは何も出力されません。

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

JSON形式のパラメータ値を引用するための構文については、AWS 「 コマンドラインインターフェイスユーザーガイド」の「 の文字列での引用符の使用 AWS CLI」を参照してください。

例 9: 起動構成を作成し、起動時にブロックデバイスがアタッチされないようにする方法

この例では、 のブロックデバイスマッピングで指定されたブロックデバイスを抑制する起動設定を作成します AMI (例: /dev/sdf) 。

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdf","NoDevice":""}]'
```

このコマンドでは何も出力されません。

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

JSON形式のパラメータ値を引用するための構文については、AWS 「 コマンドラインインターフェイスユーザーガイド」の「 の文字列での引用符の使用 AWS CLI」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスCreateLaunchConfiguration」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、「my-lc」という名前の起動設定を作成します。この起動設定を使用するAuto Scaling グループによって起動されるEC2インスタンスは、指定されたインスタンスタイプ、、AMIセキュリティグループ、IAMロールを使用します。

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- API 詳細については、「 コマンドレットリファレンスCreateLaunchConfiguration」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`CreateOrUpdateTags` で を使用する CLI

以下のコード例は、`CreateOrUpdateTags` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループのタグを作成または変更するには

この例では、指定された Auto Scaling グループに 2 つのタグを追加します。

```
aws autoscaling create-or-update-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-
```

```
asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスのタグ付け」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスCreateOrUpdateTags」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループに単一のタグを追加します。タグキーはmyTag「」で、タグ値はmyTagValue「」です。Auto Scaling は、Auto Scaling グループによって起動された後続のEC2インスタンスにこのタグを伝播します。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

例 2: PowerShell バージョン 2 では、New-Object を使用して Tag パラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- API 詳細については、「 コマンドレットリファレンスCreateOrUpdateTags」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **DeleteAutoScalingGroup**で を使用する AWS SDK CLI

以下のコード例は、DeleteAutoScalingGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ
- レジリエントなサービスの構築と管理

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Auto Scaling グループの最小サイズをゼロに更新し、グループ内のすべてのインスタンスを終了して、グループを削除します。

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
```

```
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
```

```
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDeleteAutoScalingGroup」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDeleteAutoScalingGroup」の「」を参照してください。

CLI

AWS CLI

例 1: 指定された Auto Scaling グループを削除する方法

この例では、指定された Auto Scaling グループを削除します。

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling インフラストラクチャの削除」を参照してください。 EC2 Auto Scaling

例 2: 指定された Auto Scaling グループを強制的に削除する方法

グループ内のインスタンスが終了するのを待たずに Auto Scaling グループを削除するには、`--force-delete` オプションを使用します。

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --force-delete
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling インフラストラクチャの削除」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDeleteAutoScalingGroup」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDeleteAutoScalingGroup」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API 詳細については、Kotlin リファレンスのDeleteAutoScalingGroup「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDeleteAutoScalingGroup」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループに実行中のインスタンスがない場合、そのグループを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

例 3: この例では、指定した Auto Scaling グループを削除し、そのグループに含まれる実行中のインスタンスをすべて終了します。

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- API 詳細については、「 コマンドレットリファレンスDeleteAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Auto Scaling グループの最小サイズをゼロに更新し、グループ内のすべてのインスタンスを終了して、グループを削除します。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
```

```
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

        :param group_name: The name of the group to delete.
```

```
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDeleteAutoScalingGroup」の「」を参照してください。

Rust

SDK Rust の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
 Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- API 詳細については、AWS SDKRust APIリファレンスのDeleteAutoScalingGroup「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeleteLaunchConfiguration` で を使用する CLI

以下のコード例は、`DeleteLaunchConfiguration` の使用方法を示しています。

CLI

AWS CLI

起動設定を削除するには

この例では、指定された起動構成を削除します。

```
aws autoscaling delete-launch-configuration \
    --launch-configuration-name my-launch-config
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling インフラストラクチャの削除」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDeleteLaunchConfiguration」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した起動設定が Auto Scaling グループにアタッチされていない場合、そのグループを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- API 詳細については、「 コマンドレットリファレンスDeleteLaunchConfiguration」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeleteLifecycleHook` で を使用する CLI

以下のコード例は、`DeleteLifecycleHook` の使用方法を示しています。

CLI

AWS CLI

ライフサイクルフックを削除するには

指定されたライフサイクルフックを削除します。

```
aws autoscaling delete-lifecycle-hook \
    --lifecycle-hook-name my-lifecycle-hook \
    --auto-scaling-group-name my-asg
```

このコマンドでは何も出力されません。

- API 詳細については、AWS CLI 「 コマンドリファレンスDeleteLifecycleHook」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定したライフサイクルフックを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- API 詳細については、「 コマンドレットリファレンスDeleteLifecycleHook」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeleteNotificationConfiguration` で を使用する CLI

以下のコード例は、`DeleteNotificationConfiguration` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling 通知を削除するには

この例では、指定された Auto Scaling グループから指定された通知を削除します。

```
aws autoscaling delete-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「通知設定の削除」を参照してください。

- API 詳細については、AWS CLI「 コマンドリファレンスDeleteNotificationConfiguration」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した通知アクションを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- API 詳細については、「 コマンドレットリファレンスDeleteNotificationConfiguration」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeletePolicy` で を使用する CLI

以下のコード例は、`DeletePolicy` の使用方法を示しています。

CLI

AWS CLI

スケーリングポリシーを削除するには

この例では、指定されたスケーリングポリシーを削除します。

```
aws autoscaling delete-policy \
    --auto-scaling-group-name my-asg \
    --policy-name alb1000-target-tracking-scaling-policy
```

このコマンドでは何も出力されません。

- API 詳細については、AWS CLI 「 コマンドリファレンスDeletePolicy」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定したポリシーを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- API 詳細については、「 コマンドレットリファレンスDeletePolicy」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

DeleteScheduledAction で を使用する CLI

以下のコード例は、DeleteScheduledAction の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループからスケジュールされたアクションを削除するには

この例では、指定された Auto Scaling グループから指定されたスケジュール済みの通知を削除します。

```
aws autoscaling delete-scheduled-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action
```

このコマンドでは何も出力されません。

- API 詳細については、AWS CLI 「 コマンドリファレンスDeleteScheduledAction」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定したスケジュールされたアクションを削除します。操作を続行する前に確認画面が表示されます。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- API 詳細については、「 コマンドレットリファレンスDeleteScheduledAction」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeleteTags` で を使用する CLI

以下のコード例は、`DeleteTags` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループからタグを削除するには

この例では、指定された Auto Scaling グループから指定されたタグを削除します。

```
aws autoscaling delete-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスのタグ付け」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDeleteTags」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループから指定したタグを削除します。操作を続行する前に確認画面が表示されます。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

例 2: Force パラメータを指定すると、操作を続行する前に確認を求めるプロンプトが表示されません。

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

例 3: PowerShell バージョン 2 では、New-Object を使用してタグパラメータのタグを作成する必要があります。

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- API 詳細については、「 コマンドレットリファレンスDeleteTags」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeAccountLimits` で を使用する CLI

以下のコード例は、`DescribeAccountLimits` の使用方法を示しています。

CLI

AWS CLI

Amazon EC2 Auto Scaling アカウントの制限を記述するには

この例では、 AWS アカウントの Amazon EC2 Auto Scaling 制限について説明します。

```
aws autoscaling describe-account-limits
```

出力:

```
{
    "NumberOfLaunchConfigurations": 5,
    "MaxNumberOfLaunchConfigurations": 100,
    "NumberOfAutoScalingGroups": 3,
    "MaxNumberOfAutoScalingGroups": 20
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling サービスクォータ」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeAccountLimits」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、 AWS アカウントの Auto Scaling リソース制限について説明します。

```
Get-ASAccountLimit
```

出力:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- API 詳細については、「 コマンドレットリファレンスDescribeAccountLimits」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeAdjustmentTypes` で を使用する CLI

以下のコード例は、`DescribeAdjustmentTypes` の使用方法を示しています。

CLI

AWS CLI

使用可能なスケーリング調整タイプを記述するには

この例では、使用可能な調整タイプについて記述します。

```
aws autoscaling describe-adjustment-types
```

出力:

```
{
    "AdjustmentTypes": [
        {
            "AdjustmentType": "ChangeInCapacity"
        },
        {
            "AdjustmentType": "ExactCapacity"
        },
        {
            "AdjustmentType": "PercentChangeInCapacity"
        }
    ]
```

```
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「スケーリング調整タイプAuto ScalingEC2」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeAdjustmentTypes」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされている調整タイプを記述します。

```
Get-ASAdjustmentType
```

出力:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- API 詳細については、「 コマンドレットリファレンスDescribeAdjustmentTypes」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **DescribeAutoScalingGroups**で を使用する AWS SDK CLI

以下のコード例は、DescribeAutoScalingGroups の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

- レジリエントなサービスの構築と管理

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };
```

```
        var response = await
_amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
```

```
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

CLI

AWS CLI

例 1: 指定された Auto Scaling グループを記述する方法

この例では、指定された Auto Scaling グループを記述します。

```
aws autoscaling describe-auto-scaling-groups \
    --auto-scaling-group-names my-asg
```

出力:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-1234567890abcde12"
            },
            "MinSize": 0,
            "MaxSize": 1,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
```

```
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-06905f55584de02da",
                    "InstanceType": "t2.micro",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "ProtectedFromScaleIn": false,
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcde12"
                    }
                }
            ],
            "CreatedTime": "2023-10-28T02:39:22.152Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN":"arn",
            "TrafficSources": []
        }
    ]
}
```

例 2: 指定された最初の 100 個の Auto Scaling グループを記述する方法

この例では、指定された複数の Auto Scaling グループを記述します。最大 100 個のグループ名を指定できます。

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 100 \
    --auto-scaling-group-names "group1" "group2" "group3" "group4"
```

出力例については、例 1 を参照してください。

例 3: 指定されたリージョンの Auto Scaling グループを記述する方法

この例では、指定されたリージョンの Auto Scaling グループを最大 75 グループまで記述します。

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 75 \
    --region us-east-1
```

出力例については、例 1 を参照してください。

例 4: 指定された数の Auto Scaling グループを記述する方法

特定の数の Auto Scaling グループを返すには、`--max-items` オプションを使用します。

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 1
```

出力例については、例 1 を参照してください。

出力に `NextToken` フィールドが含まれている場合は、さらに多くのグループがあることを示しています。追加のグループを取得するには、次のように、以降の呼び出しで `--starting-token` オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-auto-scaling-groups \
    --starting-token Z3M3LMPEXAMPLE
```

出力例については、例 1 を参照してください。

例 5: 起動構成を使用している Auto Scaling グループを記述する方法

この例では、 `--query` オプションを使用して、起動構成を使用する Auto Scaling グループを記述します。

```
aws autoscaling describe-auto-scaling-groups \
    --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

出力:

```
[
```

```
    {
        "AutoScalingGroupName": "my-asg",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
        "LaunchConfigurationName": "my-lc",
        "MinSize": 0,
        "MaxSize": 1,
        "DesiredCapacity": 1,
        "DefaultCooldown": 300,
        "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
        ],
        "LoadBalancerNames": [],
        "TargetGroupARNs": [],
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": 0,
        "Instances": [
            {
                "InstanceId": "i-088c57934a6449037",
                "InstanceType": "t2.micro",
                "AvailabilityZone": "us-west-2c",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-lc",
                "ProtectedFromScaleIn": false
            }
        ],
        "CreatedTime": "2023-10-28T02:39:22.152Z",
        "SuspendedProcesses": [],
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN":"arn",
        "TrafficSources": []
    }
]
```

詳細については、AWS 「 コマンドラインインターフェイスユーザーガイド」の「フィルター AWS CLI出力」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>
```

```
                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- API 詳細については、Kotlin リファレンスのDescribeAutoScalingGroups「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling グループの名前を一覧表示します。

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

出力:

```
AutoScalingGroupName
--------------------
my-asg-1
```

```
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

例 2: この例では、指定した Auto Scaling グループを記述します。

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

出力:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

例 3: この例では、指定した 2 つの Auto Scaling グループを記述します。

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

例 4: この例では、指定した Auto Scaling グループの Auto Scaling インスタンスを記述します。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

例 5: この例では、すべての Auto Scaling グループを記述します。

```
Get-ASAutoScalingGroup
```

例 6: この例では、指定した Auto Scaling グループの を記述 LaunchTemplate します。この例では、[インスタンスの購入オプション] が [起動テンプレートに準拠する] に設定されていることを前提としています。このオプションが「購入オプションとインスタンスタイプを組み合わせる」に設定されている場合、 LaunchTemplate MixedInstancesPolicy「.LaunchTemplate」プロパティを使用してアクセスできます。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

出力:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- API 詳細については、「 コマンドレットリファレンスDescribeAutoScalingGroups」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
```

```
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.

        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found,
otherwise None.
        :raises ClientError: If there is an error describing the Auto Scaling
group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
        else:
            return groups[0] if len(groups) > 0 else None
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDescribeAutoScalingGroups」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API 詳細については、AWS SDKRust APIリファレンスのDescribeAutoScalingGroups「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **DescribeAutoScalingInstances**で を使用する AWS SDK CLI

以下のコード例は、DescribeAutoScalingInstances の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
```

```
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
_amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDescribeAutoScalingInstances」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDescribeAutoScalingInstances」の「」を参照してください。

CLI

AWS CLI

例 1: 1 つまたは複数のインスタンスを記述する方法

この例では、指定されたインスタンスを記述します。

```
aws autoscaling describe-auto-scaling-instances \
    --instance-ids i-06905f55584de02da
```

出力:

```
{
    "AutoScalingInstances": [
```

```
        {
            "InstanceId": "i-06905f55584de02da",
            "InstanceType": "t2.micro",
            "AutoScalingGroupName": "my-asg",
            "AvailabilityZone": "us-west-2b",
            "LifecycleState": "InService",
            "HealthStatus": "HEALTHY",
            "ProtectedFromScaleIn": false,
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-1234567890abcde12",
                "LaunchTemplateName": "my-launch-template",
                "Version": "1"
            }
        }
    ]
}
```

例 2: 1 つまたは複数のインスタンスを記述する方法

この例では、--max-items オプションを使用して、この呼び出しで返されるインスタンスの数を指定します。

```
aws autoscaling describe-auto-scaling-instances \
    --max-items 1
```

出力に NextToken フィールドが含まれている場合は、さらに多くのインスタンスがあることを示しています。追加のインスタンスを取得するには、次のように、以降の呼び出しで --starting-token オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-auto-scaling-instances \
    --starting-token Z3M3LMPEXAMPLE
```

出力例については、例 1 を参照してください。

例 3: 起動構成を使用するインスタンスを記述する方法

この例では、 --query オプションを使用して、起動構成を使用するインスタンスを記述します。

```
aws autoscaling describe-auto-scaling-instances \
```

```
    --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

出力:

```
[
    {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AutoScalingGroupName": "my-asg",
        "AvailabilityZone": "us-west-2c",
        "LifecycleState": "InService",
        "HealthStatus": "HEALTHY",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
    }
]
```

詳細については、AWS 「 コマンドラインインターフェイスユーザーガイド」の「フィルター AWS CLI出力」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeAutoScalingInstances」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                   .builder()
                   .instanceIds(id)
                   .build();
```

```
            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDescribeAutoScalingInstances」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
```

```
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- API 詳細については、Kotlin リファレンスのDescribeAutoScalingInstances「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function describeAutoScalingInstances($instanceIds)
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDescribeAutoScalingInstances」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling インスタンスIDsの を一覧表示します。

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

出力:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

例 2: この例では、指定した Auto Scaling インスタンスを記述します。

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

出力:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

例 3: この例では、指定した 2 つの Auto Scaling インスタンスを記述します。

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

例 4: この例では、指定した Auto Scaling グループの Auto Scaling インスタンスを記述します。

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

例 5: この例では、すべての Auto Scaling インスタンスを記述します。

```
Get-ASAutoScalingInstance
```

- API 詳細については、「 コマンドレットリファレンスDescribeAutoScalingInstances」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str,
 Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDescribeAutoScalingInstances」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
```

```
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }
```

- API 詳細については、AWS SDKRust APIリファレンスのDescribeAutoScalingInstances「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeAutoScalingNotificationTypes` で を使用する CLI

以下のコード例は、`DescribeAutoScalingNotificationTypes` の使用方法を示しています。

CLI

AWS CLI

使用可能な通知タイプを記述するには

この例では、使用可能な通知タイプについて記述します。

```
aws autoscaling describe-auto-scaling-notification-types
```

出力:

```
{
    "AutoScalingNotificationTypes": [
```

```
        "autoscaling:EC2_INSTANCE_LAUNCH",
        "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
        "autoscaling:EC2_INSTANCE_TERMINATE",
        "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
        "autoscaling:TEST_NOTIFICATION"
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループがスケーリングされたときの Amazon SNS通知の取得」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeAutoScalingNotificationTypes」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされている通知タイプを一覧表示します。

```
Get-ASAutoScalingNotificationType
```

出力:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- API 詳細については、「 コマンドレットリファレンスDescribeAutoScalingNotificationTypes」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeLaunchConfigurations` で を使用する CLI

以下のコード例は、`DescribeLaunchConfigurations` の使用方法を示しています。

CLI

AWS CLI

例 1: 指定した起動設定を記述するには

この例では、指定された起動構成について記述します。

```
aws autoscaling describe-launch-configurations \
    --launch-configuration-names my-launch-config
```

出力:

```
{
    "LaunchConfigurations": [
        {
            "LaunchConfigurationName": "my-launch-config",
            "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
            "ImageId": "ami-0528a5175983e7f28",
            "KeyName": "my-key-pair-uswest2",
            "SecurityGroups": [
                "sg-05eaec502fcdadc2e"
            ],
            "ClassicLinkVPCSecurityGroups": [],
            "UserData": "",
            "InstanceType": "t2.micro",
            "KernelId": "",
            "RamdiskId": "",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "SnapshotId": "snap-06c1606ba5ca274b1",
                        "VolumeSize": 8,
                        "VolumeType": "gp2",
                        "DeleteOnTermination": true,
                        "Encrypted": false
                    }
                }
            ],
            "InstanceMonitoring": {
```

```
                "Enabled": true
            },
            "CreatedTime": "2020-10-28T02:39:22.321Z",
            "EbsOptimized": false,
            "AssociatePublicIpAddress": true,
            "MetadataOptions": {
                "HttpTokens": "required",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "disabled"
            }
        }
    ]
}
```

例 2: 指定された数の起動構成を記述する方法

特定の数の起動構成を返すには、 `--max-items` オプションを使用します。

```
aws autoscaling describe-launch-configurations \
    --max-items 1
```

出力に NextToken フィールドが含まれている場合、起動構成がさらに増えます。追加の起動構成を取得するには、次のように、以降の呼び出しで `--starting-token` オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-launch-configurations \
    --starting-token Z3M3LMPEXAMPLE
```

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeLaunchConfigurations」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、起動設定の名前を一覧表示します。

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

出力:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

例 2: この例では、指定した起動設定を記述します。

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

出力:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

例 3: この例では、指定した 2 つの起動設定を記述します。

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

例 4: この例では、すべての起動設定を記述します。

```
Get-ASLaunchConfiguration
```

- API 詳細については、「 コマンドレットリファレンスDescribeLaunchConfigurations」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeLifecycleHookTypes` で を使用する CLI

以下のコード例は、`DescribeLifecycleHookTypes` の使用方法を示しています。

CLI

AWS CLI

使用可能なライフサイクルフックの種類を記述するには

この例では、使用可能なライフサイクルフックのタイプを記述します。

```
aws autoscaling describe-lifecycle-hook-types
```

出力:

```
{
    "LifecycleHookTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCHING",
        "autoscaling:EC2_INSTANCE_TERMINATING"
    ]
}
```

- API 詳細については、AWS CLI「 コマンドリファレンスDescribeLifecycleHookTypes」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例は、Auto Scaling でサポートされているライフサイクルフックの種類を一覧表示します。

```
Get-ASLifecycleHookType
```

出力:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- API 詳細については、「 コマンドレットリファレンスDescribeLifecycleHookTypes」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeLifecycleHooks` で を使用する CLI

以下のコード例は、`DescribeLifecycleHooks` の使用方法を示しています。

CLI

AWS CLI

ライフサイクルフックを記述するには

指定された Auto Scaling グループのライフサイクルフックを記述します。

```
aws autoscaling describe-lifecycle-hooks \
    --auto-scaling-group-name my-asg
```

出力:

```
{
```

```
    "LifecycleHooks": [
        {
            "GlobalTimeout": 3000,
            "HeartbeatTimeout": 30,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-hook",
            "DefaultResult": "ABANDON",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"
        },
        {
            "GlobalTimeout": 6000,
            "HeartbeatTimeout": 60,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-termination-hook",
            "DefaultResult": "CONTINUE",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"
        }
    ]
}
```

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeLifecycleHooks」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したライフサイクルフックを記述します。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

出力:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
```

```
RoleARN                : arn:aws:iam::123456789012:role/my-iam-role
```

例 2: この例では、指定した Auto Scaling グループのすべてのライフサイクルフックを記述します。

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

例 3: この例では、すべての Auto Scaling グループのすべてのライフサイクルフックを記述します。

```
Get-ASLifecycleHook
```

- API 詳細については、「 コマンドレットリファレンスDescribeLifecycleHooks」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeLoadBalancers` で を使用する CLI

以下のコード例は、`DescribeLoadBalancers` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループの Classic Load Balancer を記述するには

指定された Auto Scaling グループのクラシックロードバランサーを記述します。

```
aws autoscaling describe-load-balancers \
    --auto-scaling-group-name my-asg
```

出力:

```
{
    "LoadBalancers": [
        {
```

```
            "State": "Added",
            "LoadBalancerName": "my-load-balancer"
        }
    ]
}
```

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeLoadBalancers」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループのロードバランサーを記述します。

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

出力:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- API 詳細については、「 コマンドレットリファレンスDescribeLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeMetricCollectionTypes` で を使用する CLI

以下のコード例は、`DescribeMetricCollectionTypes` の使用方法を示しています。

CLI

AWS CLI

使用可能なメトリクスコレクションタイプを記述するには

この例では、使用可能なメトリクスコレクションタイプについて記述します。

```
aws autoscaling describe-metric-collection-types
```

出力:

```
{
    "Metrics": [
        {
            "Metric": "GroupMinSize"
        },
        {
            "Metric": "GroupMaxSize"
        },
        {
            "Metric": "GroupDesiredCapacity"
        },
        {
            "Metric": "GroupInServiceInstances"
        },
        {
            "Metric": "GroupInServiceCapacity"
        },
        {
            "Metric": "GroupPendingInstances"
        },
        {
            "Metric": "GroupPendingCapacity"
        },
        {
            "Metric": "GroupTerminatingInstances"
        },
        {
            "Metric": "GroupTerminatingCapacity"
        },
        {
            "Metric": "GroupStandbyInstances"
        },
        {
            "Metric": "GroupStandbyCapacity"
        },
        {
            "Metric": "GroupTotalInstances"
```

```
        },
        {
            "Metric": "GroupTotalCapacity"
        }
    ],
    "Granularities": [
        {
            "Granularity": "1Minute"
        }
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループのメトリクス」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeMetricCollectionTypes」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされているメトリクスコレクションタイプを一覧表示します。

```
(Get-ASMetricCollectionType).Metrics
```

出力:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

例 2: この例では、対応する粒度を一覧表示します。

```
(Get-ASMetricCollectionType).Granularities
```

出力:

```
Granularity
-----------
1Minute
```

- API 詳細については、「 コマンドレットリファレンスDescribeMetricCollectionTypes」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

DescribeNotificationConfigurations で を使用する CLI

以下のコード例は、DescribeNotificationConfigurations の使用方法を示しています。

CLI

AWS CLI

例 1: 指定したグループの通知設定を記述するには

この例では、指定された Auto Scaling グループの通知構成を記述します。

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg
```

出力:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
```

```
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループがスケーリングされたときの Amazon SNS通知の取得」を参照してください。 EC2 Auto Scaling

例 1: 指定された数の通知構成を記述する方法

特定の数の通知構成を返すには、 max-items パラメータを使用します。

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-auto-scaling-group \
    --max-items 1
```

出力:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

出力に NextToken フィールドが含まれている場合、通知構成がさらに増えます。追加の通知構成を取得するには、次のように、以降の呼び出しで starting-token オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループがスケーリングされたときの Amazon SNS通知の取得」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeNotificationConfigurations」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループに関連付けられている通知アクションを記述します。

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

出力:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

例 2: この例では、すべての Auto Scaling グループに関連付けられている通知アクションを記述します。

```
Get-ASNotificationConfiguration
```

- API 詳細については、「 コマンドレットリファレンスDescribeNotificationConfigurations」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribePolicies` で を使用する CLI

以下のコード例は、`DescribePolicies` の使用方法を示しています。

CLI

AWS CLI

例 1: 指定した Auto Scaling グループのスケーリングポリシーを記述するには

この例では、指定された Auto Scaling グループのスケーリングポリシーを記述します。

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg
```

出力:

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "alb1000-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ALBRequestCountPerTarget",
```

```
                    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
                },
                "TargetValue": 1000.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        },
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "cpu40-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 40.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
```

```
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「動的スケーリング」を参照してください。

例 2: 指定された名前のスケーリングポリシーを記述する方法

特定のスケーリングポリシーを返すには、 `--policy-names` オプションを使用します。

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --policy-names cpu40-target-tracking-scaling-policy
```

出力例については、例 1 を参照してください。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「動的スケーリング」を参照してください。

例 3: スケーリングポリシーの数を記述する方法

特定の数のポリシーを返すには、`--max-items` オプションを使用します。

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

出力例については、例 1 を参照してください。

出力に `NextToken` フィールドが含まれている場合は、後続の呼び出しで `--starting-token` オプションを使用してこのフィールドの値を使用し、追加のポリシーを取得します。

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-
token Z3M3LMPEXAMPLE
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「動的スケーリング」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribePolicies」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループのすべてのポリシーを記述します。

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

出力:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

例 2: この例では、指定した Auto Scaling グループの指定したポリシーを記述します。

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

例 3: この例では、すべての Auto Scaling グループのすべてのポリシーを記述します。

```
Get-ASPolicy
```

- API 詳細については、「 コマンドレットリファレンスDescribePolicies」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **DescribeScalingActivities**AWS SDKで を使用する CLI

以下のコード例は、DescribeScalingActivities の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Amazon.AutoScaling.Model.Activity>>
 DescribeScalingActivitiesAsync(
        string groupName)
    {
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        return response.Activities;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDescribeScalingActivities」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
```

```
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDescribeScalingActivities」の「」を参照してください。

CLI

AWS CLI

例 1: 指定されたグループのスケーリングアクティビティを記述する方法

この例では、指定された Auto Scaling グループのスケーリングアクティビティを記述します。

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
```

```
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループのスケーリングアクティビティを検証する」を参照してください。 EC2 Auto Scaling

例 2: 削除されたグループのスケーリングアクティビティを記述する方法

Auto Scaling グループが削除された後にスケーリングアクティビティを説明するには、`--include-deleted-groups` オプションを追加します。

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg \
    --include-deleted-groups
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "Description": "Launching a new EC2 instance.  Status Reason: Your
 Spot request price of 0.001 is lower than the minimum required Spot request
 fulfillment price of 0.0031. Launching EC2 instance failed.",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of
 AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired
 capacity from 0 to 3.  At 2021-01-13T20:47:27Z an instance was started in
 response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 3.",
```

```
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than
 the minimum required Spot request fulfillment price of 0.0031. Launching EC2
 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling のトラブルシューティング」を参照してください。 EC2 Auto Scaling

例 3: 指定された数のスケーリングアクティビティを記述する方法

特定の数のアクティビティを返すには、--max-items オプションを使用します。

```
aws autoscaling describe-scaling-activities \
    --max-items 1
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
```

```
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

出力に NextToken フィールドが含まれている場合は、さらに多くのアクティビティがあることを示しています。追加のアクティビティを取得するには、次のように、以降の呼び出しで --starting-token オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-scaling-activities \
    --starting-token Z3M3LMPEXAMPLE
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループのスケーリングアクティビティを検証する」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeScalingActivities」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                   .autoScalingGroupName(groupName)
                   .maxRecords(10)
                   .build();

           DescribeScalingActivitiesResponse response = autoScalingClient
```

```
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
 activity.activityId());
                System.out.println("The activity details are " +
 activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDescribeScalingActivities」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
```

```
        }
    }
}
```

- API 詳細については、Kotlin リファレンスのDescribeScalingActivities「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function describeScalingActivities($autoScalingGroupName)
    {
        return $this->autoScalingClient->describeScalingActivities([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDescribeScalingActivities」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: こので例は、指定した Auto Scaling グループの過去 6 週間のスケーリングアクティビティを記述します。

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

出力:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

例 2: この例では、指定したスケーリングアクティビティを記述します。

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

例 3: この例では、すべての Auto Scaling グループの過去 6 週間のスケーリングアクティビティを記述します。

```
Get-ASScalingActivity
```

- API 詳細については、「 コマンドレットリファレンスDescribeScalingActivities」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
 Any]]:
        """
        Gets information about scaling activities for the group. Scaling
 activities
        are things like instances stopping or starting in response to user
 requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for
 the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling
 activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
```

```
                "describe_scaling_activities"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupName=group_name)
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
        else:
            return activities
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDescribeScalingActivities」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
```

```
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- API 詳細については、AWS SDKRust APIリファレンスのDescribeScalingActivities「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeScalingProcessTypes` で を使用する CLI

以下のコード例は、DescribeScalingProcessTypes の使用方法を示しています。

CLI

AWS CLI

使用可能なプロセスタイプを記述するには

この例では、使用可能なプロセスタイプを記述します。

```
aws autoscaling describe-scaling-process-types
```

出力:

```
{
    "Processes": [
        {
            "ProcessName": "AZRebalance"
        },
        {
            "ProcessName": "AddToLoadBalancer"
        },
        {
            "ProcessName": "AlarmNotification"
        },
        {
            "ProcessName": "HealthCheck"
        },
        {
            "ProcessName": "InstanceRefresh"
        },
        {
            "ProcessName": "Launch"
        },
        {
            "ProcessName": "ReplaceUnhealthy"
        },
        {
            "ProcessName": "ScheduledActions"
        },
        {
            "ProcessName": "Terminate"
        }
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「スケーリングプロセスの一時停止と再開」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeScalingProcessTypes」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされているプロセスタイプを一覧表示します。

```
Get-ASScalingProcessType
```

出力:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- API 詳細については、「 コマンドレットリファレンスDescribeScalingProcessTypes」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeScheduledActions` で を使用する CLI

以下のコード例は、`DescribeScheduledActions` の使用方法を示しています。

CLI

AWS CLI

例 1: スケジュールされたすべてのアクションを記述するには

この例では、スケジュール済みのすべてのアクションを記述します。

```
aws autoscaling describe-scheduled-actions
```

出力:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

例 2: 指定されたグループのスケジュール済みのアクションを記述する方法

特定の Auto Scaling グループのスケジュール済みのアクションを説明するには、 `--auto-scaling-group-name` オプションを使用します。

```
aws autoscaling describe-scheduled-actions \
```

```
    --auto-scaling-group-name my-asg
```

出力:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

例 3: 指定されたスケジュール済みのアクションを記述する方法

特定のスケジュール済みのアクションを説明するには、--scheduled-action-names オプションを使用します。

```
aws autoscaling describe-scheduled-actions \
    --scheduled-action-names my-recurring-action
```

出力:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
```

```
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

例 4: 指定された開始時刻でスケジュール済みのアクションを記述する方法

特定の時間に開始されるスケジュール済みのアクションを説明するには、`--start-time` オプションを使用します。

```
aws autoscaling describe-scheduled-actions \
    --start-time "2023-12-01T04:00:00Z"
```

出力:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
```

```
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

例 5: 指定された時間に終了するスケジュール済みのアクションを記述する方法

特定の時間に終了するスケジュール済みのアクションを説明するには、--end-time オプションを使用します。

```
aws autoscaling describe-scheduled-actions \
    --end-time "2023-12-01T04:00:00Z"
```

出力:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

例 6: 指定された数のスケーリングアクティビティを記述する方法

特定の数のアクティビティを返すには、--max-items オプションを使用します。

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

出力:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

出力に NextToken フィールドが含まれている場合は、さらに多くのスケジュール済みのアクションがあることを示しています。追加のスケジュール済みのアクションを取得するには、次のように、以降の呼び出しで --starting-token オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeScheduledActions」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループのスケジュールされたスケーリングアクティビティを記述します。

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

出力:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

例 2: この例では、指定したスケジュールされたスケーリングアクションを記述します。

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

例 3: この例では、指定した時刻までに開始されるスケジュールされたスケーリングアクションを記述します。

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

例 4: この例では、指定した時刻までに終了するスケジュールされたスケーリングアクションを記述します。

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

例 5: この例では、すべての Auto Scaling グループのスケジュールされたスケーリングアクションを記述します。

```
Get-ASScheduledAction
```

- API 詳細については、「 コマンドレットリファレンスDescribeScheduledActions」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

DescribeTags で を使用する CLI

以下のコード例は、DescribeTags の使用方法を示しています。

CLI

AWS CLI

すべてのタグを記述するには

この例では、すべてのタグを記述します。

```
aws autoscaling describe-tags
```

出力:

```
{
    "Tags": [
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "Research",
            "Key": "Dept"
        },
        {
```

```
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "WebServer",
            "Key": "Role"
        }
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスのタグ付け」を参照してください。 EC2 Auto Scaling

例 2: 指定されたグループのタグを記述する方法

特定の Auto Scaling グループのタグを説明するには、`--filters` オプションを使用します。

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスのタグ付け」を参照してください。 EC2 Auto Scaling

例 3: 指定された数のタグを記述する方法

特定の数のタグを返すには、`--max-items` オプションを使用します。

```
aws autoscaling describe-tags \
    --max-items 1
```

出力に `NextToken` フィールドが含まれている場合は、さらに多くのタグがあることを示しています。追加のタグを取得するには、次のように、以降の呼び出しで `--starting-token` オプションを使用してこのフィールドの値を使用します。

```
aws autoscaling describe-tags \
    --filters Name=auto-scaling-group,Values=my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスのタグ付け」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeTags」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、キー値がmyTag「」またはmyTag「2」のタグを記述します。フィルター名に使用できる値は、auto-scaling-group「」、「キー」、「値」、propagate-at-launch「」です。この例で使用される構文には、 PowerShell バージョン 3 以降が必要です。

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

出力:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

例 2: PowerShell バージョン 2 では、New-Object を使用して Filter パラメータのフィルターを作成する必要があります。

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

例 3: この例では、すべての Auto Scaling グループのすべてのタグを記述します。

```
Get-ASTag
```

- API 詳細については、「 コマンドレットリファレンスDescribeTags」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeTerminationPolicyTypes` で を使用する CLI

以下のコード例は、`DescribeTerminationPolicyTypes` の使用方法を示しています。

CLI

AWS CLI

使用可能な終了ポリシータイプを記述するには

この例では、使用可能な終了ポリシータイプを記述します。

```
aws autoscaling describe-termination-policy-types
```

出力:

```
{
    "TerminationPolicyTypes": [
        "AllocationStrategy",
        "ClosestToNextInstanceHour",
        "Default",
        "NewestInstance",
        "OldestInstance",
        "OldestLaunchConfiguration",
        "OldestLaunchTemplate"
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「スケールイン中に終了するAuto Scaling インスタンスの制御」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDescribeTerminationPolicyTypes」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、Auto Scaling でサポートされている終了ポリシーを一覧表示します。

```
Get-ASTerminationPolicyType
```

出力:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- API 詳細については、「 コマンドレットリファレンスDescribeTerminationPolicyTypes」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DetachInstances` で を使用する CLI

以下のコード例は、`DetachInstances` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループからインスタンスをデタッチするには

この例では、指定されたインスタンスを指定された Auto Scaling グループからデタッチします。

```
aws autoscaling detach-instances \
    --instance-ids i-030017cfa84b20135 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",
            "AutoScalingGroupName": "my-asg",
            "Description": "Detaching EC2 instance: i-030017cfa84b20135",
            "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was
 detached in response to a user request, shrinking the capacity from 2 to 1.",
            "StartTime": "2020-04-12T15:02:16.179Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- API 詳細については、AWS CLI 「 コマンドリファレンスDetachInstances」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループから指定したインスタンスをデタッチし、希望する容量を減らして、Auto Scaling が代替インスタンスを起動しないようにします。

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

出力:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
```

```
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

例 2: この例では、希望する容量を減らすことなく、指定したインスタンスを指定した Auto Scaling グループからデタッチします。Auto Scaling が代替インスタンスを起動します。

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

出力:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- API 詳細については、「 コマンドレットリファレンスDetachInstances」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DetachLoadBalancers` で を使用する CLI

以下のコード例は、`DetachLoadBalancers` の使用方法を示しています。

CLI

AWS CLI

Classic Load Balancer を Auto Scaling グループからデタッチするには

この例では、指定されたクラシックロードバランサーを指定された Auto Scaling グループからデタッチします。

```
aws autoscaling detach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループにロードバランサーをアタッチする」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDetachLoadBalancers」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループから指定したロードバランサーをデタッチします。

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- API 詳細については、「 コマンドレットリファレンスDetachLoadBalancers」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **DisableMetricsCollection**で を使用する AWS SDK CLI

以下のコード例は、DisableMetricsCollection の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスDisableMetricsCollection」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
            autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

    }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスDisableMetricsCollection」の「」を参照してください。

CLI

AWS CLI

Auto Scaling グループのメトリクス収集を無効にする方法

この例では、指定された Auto Scaling グループの GroupDesiredCapacity メトリクスの収集を無効にします。

```
aws autoscaling disable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスの CloudWatch メトリクスのモニタリング」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスDisableMetricsCollection」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスDisableMetricsCollection」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- API 詳細については、Kotlin リファレンスのDisableMetricsCollection「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスDisableMetricsCollection」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定したメトリクスのモニタリングを無効にします。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

例 2: この例では、指定した Auto Scaling グループのすべてのメトリクスのモニタリングを無効にします。

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- API 詳細については、「 コマンドレットリファレンスDisableMetricsCollection」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.

        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
 collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
 '{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
```

```
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスDisableMetricsCollection」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- API 詳細については、AWS SDKRust APIリファレンスのDisableMetricsCollection「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **EnableMetricsCollection**で を使用する AWS SDK CLI

以下のコード例は、EnableMetricsCollection の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>
        {
            "GroupMaxSize",
        };

    var collectionRequest = new EnableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
        Metrics = listMetrics,
        Granularity = "1Minute",
    };
```

```
        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスEnableMetricsCollection」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
```

```
                    << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスEnableMetricsCollection」の「」を参照してください。

CLI

AWS CLI

例 1: Auto Scaling グループのメトリクス収集を有効にする方法

この例では、指定された Auto Scaling グループのデータの収集を有効にします。

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --granularity "1Minute"
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスの CloudWatch メトリクスのモニタリング」を参照してください。 EC2 Auto Scaling

例 2: Auto Scaling グループの指定されたメトリックスのデータを収集する方法

特定のメトリックスのデータを収集するには、`--metrics` オプションを使用します。

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity --granularity "1Minute"
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループとインスタンスの CloudWatch メトリクスのモニタリング」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスEnableMetricsCollection」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスEnableMetricsCollection」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- API 詳細については、Kotlin リファレンスのEnableMetricsCollection「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
```

```
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスEnableMetricsCollection」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定したメトリクスのモニタリングを有効にします。

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

例 2: この例では、指定した Auto Scaling グループのすべてのメトリクスのモニタリングを有効にします。

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- API 詳細については、「 コマンドレットリファレンスEnableMetricsCollection」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
```

```
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,
Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
Granularity="1Minute"
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
```

```
            elif error_code == "InvalidParameterCombination":
                logger.error(
                    f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- API 詳細については、「 for AWS SDK Python (Boto3) APIリファレンスEnableMetricsCollection」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- API 詳細については、AWS SDKRust APIリファレンスのEnableMetricsCollection「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`EnterStandby` で を使用する CLI

以下のコード例は、`EnterStandby` の使用方法を示しています。

CLI

AWS CLI

インスタンスをスタンバイモードに移行するには

この例では、指定されたインスタンスをスタンバイモードにします。これは、現在稼働中のインスタンスの更新またはトラブルシューティングに役立ちます。

```
aws autoscaling enter-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was
 moved to standby in response to a user request, shrinking the capacity from 1 to
 0.",
            "StartTime": "2020-10-31T20:31:00.949Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
```

```
        }
    ]
}
```

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling インスタンスのライフサイクル」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスEnterStandby」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したインスタンスをスタンバイモードにし、希望する容量を減らして、Auto Scaling が代替インスタンスを起動しないようにします。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

出力:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

例 2: この例では、希望する容量を減らすことなく、指定したインスタンスをスタンバイモードにします。Auto Scaling が代替インスタンスを起動します。

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

出力:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- API 詳細については、「 コマンドレットリファレンスEnterStandby」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`ExecutePolicy` で を使用する CLI

以下のコード例は、`ExecutePolicy` の使用方法を示しています。

CLI

AWS CLI

スケーリングポリシーを実行するには

この例では、指定された Auto Scaling グループに「`my-step-scale-out-policy`」という名前のスケーリングポリシーを実行します。

```
aws autoscaling execute-policy \
    --auto-scaling-group-name my-asg \
    --policy-name  my-step-scale-out-policy \
    --metric-value 95 \
    --breach-threshold 80
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「ステップポリシーと簡易スケーリングポリシー」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスExecutePolicy」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定したポリシーを実行します。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

例 2: この例では、クールダウン期間が完了するまで待機した後、指定した Auto Scaling グループの指定したポリシーを実行します。

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- API 詳細については、「 コマンドレットリファレンスExecutePolicy」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

ExitStandby で を使用する CLI

以下のコード例は、ExitStandby の使用方法を示しています。

CLI

AWS CLI

インスタンスをスタンバイモードから解除するには

指定されたインスタンスのスタンバイモードを解除します。

```
aws autoscaling exit-standby \
```

```
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was
 moved out of standby in response to a user request, increasing the capacity from
 0 to 1.",
            "StartTime": "2020-10-31T20:32:50.222Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループからインスタンスを一時的に削除する」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスExitStandby」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したインスタンスをスタンバイモードから解除します。

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

出力:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
```

```
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                        request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- API 詳細については、「 コマンドレットリファレンスExitStandby」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

PutLifecycleHook で を使用する CLI

以下のコード例は、PutLifecycleHook の使用方法を示しています。

CLI

AWS CLI

例 1: ライフサイクルフックを作成するには

この例では、新しく起動した任意のインスタンスで呼び出すライフサイクルフックを作成します。タイムアウトは 4800 秒です。これは、ユーザーデータスクリプトが完了するまでインスタンスを待機状態に保つ場合や、 を使用して AWS Lambda 関数を呼び出す場合に便利です EventBridge。

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --heartbeat-timeout 4800
```

このコマンドでは何も出力されません。同じ名前のライフサイクルフックがすでに存在する場合、新しいライフサイクルフックによって上書きされます。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

例 2: インスタンスの状態遷移を通知する Amazon SNS E メールメッセージを送信するには

この例では、インスタンスの起動時に通知を受信するために使用する Amazon SNSトピックとIAMロールを含むライフサイクルフックを作成します。

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

例 3: Amazon SQSキューにメッセージを発行するには

この例では、メタデータを含むメッセージを指定された Amazon SQSキューに発行するライフサイクルフックを作成します。

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \
    --role-arn arn:aws:iam::123456789012:role/my-notification-role \
    --notification-metadata "SQS message metadata"
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスPutLifecycleHook」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したライフサイクルフックを、指定した Auto Scaling グループに追加します。

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- API 詳細については、「 コマンドレットリファレンスPutLifecycleHook」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

PutNotificationConfiguration で を使用する CLI

以下のコード例は、PutNotificationConfiguration の使用方法を示しています。

CLI

AWS CLI

通知を追加するには

この例では、指定された Auto Scaling グループに指定された通知を追加します。

```
aws autoscaling put-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --notification-type autoscaling:TEST_NOTIFICATION
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループがスケーリングされたときの Amazon SNS通知の取得」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスPutNotificationConfiguration」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、EC2インスタンスの起動時に指定されたSNSトピックに通知を送信するように、指定された Auto Scaling グループを設定します。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

例 2: この例では、インスタンスを起動または終了するときに、指定したSNSトピックに通知を送信するように、指定した Auto Scaling グループを設定しますEC2。

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- API 詳細については、「 コマンドレットリファレンスPutNotificationConfiguration」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`PutScalingPolicy` で を使用する CLI

以下のコード例は、`PutScalingPolicy` の使用方法を示しています。

CLI

AWS CLI

Auto Scaling グループにターゲット追跡スケーリングポリシーを追加するには

次の put-scaling-policy 例では、指定された Auto Scaling グループにターゲットトラッキングスケーリングポリシーを適用します。出力には、ユーザーに代わって作成された 2 つの CloudWatch アラームの ARNsと名前が含まれます。同じ名前のスケーリングポリシーが既に存在する場合は、新しいスケーリングポリシーで上書きされます。

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

config.json の内容:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
          "PredefinedMetricType": "ALBRequestCountPerTarget",
          "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
     }
}
```

出力:

```
{
     "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
     "Alarms": [
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
             "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
         },
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
             "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
         }
     ]
```

```
}
```

その他の例については、「Amazon Auto Scaling AWS ユーザーガイド」の「コマンドラインインターフェイス (AWS CLI) のスケーリングポリシーの例」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスPutScalingPolicy」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループに指定したポリシーを追加します。指定された調整タイプによって、 ScalingAdjustment パラメータの解釈方法が決まります。'ChangeInCapacity' の場合、正の値は指定されたインスタンス数だけ容量を増やし、負の値は指定されたインスタンス数だけ容量を減らします。

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

出力:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- API 詳細については、「 コマンドレットリファレンスPutScalingPolicy」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`PutScheduledUpdateGroupAction` で を使用する CLI

以下のコード例は、`PutScheduledUpdateGroupAction` の使用方法を示しています。

CLI

AWS CLI

例 1: スケジュールされたアクションを Auto Scaling グループに追加するには

この例では、指定された Auto Scaling グループから指定されたスケジュール済みの通知を追加します。

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action \
    --start-time "2023-05-12T08:00:00Z" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

このコマンドでは何も出力されません。同じ名前のスケジュールされたアクションが既に存在する場合、新しいスケジュールされたアクションで上書きされます。

その他の例については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

例 2: 定期的なスケジュールを指定する方法

この例では、毎年 1 月、6 月、12 月の 1 日の 00:30 に実行する定期スケジュールでスケーリングするスケジュール済みのアクションを作成します。

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-recurring-action \
    --recurrence "30 0 1 1,6,12 *" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

このコマンドでは何も出力されません。同じ名前のスケジュールされたアクションが既に存在する場合、新しいスケジュールされたアクションで上書きされます。

その他の例については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。

- API 詳細については、AWS CLI 「 コマンドリファレンスPutScheduledUpdateGroupAction」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、1 回限りのスケジュールされたアクションを作成または更新して、指定した開始時刻に希望する容量を変更します。

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- API 詳細については、「 コマンドレットリファレンスPutScheduledUpdateGroupAction」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`RecordLifecycleActionHeartbeat` で を使用する CLI

以下のコード例は、`RecordLifecycleActionHeartbeat` の使用方法を示しています。

CLI

AWS CLI

ライフサイクルアクションのハートビートを記録するには

この例では、ライフサイクルアクションのハートビートを記録して、インスタンスを保留状態に保ちます。

```
aws autoscaling record-lifecycle-action-heartbeat \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスRecordLifecycleActionHeartbeat」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したライフサイクルアクションのハートビートを記録します。これにより、カスタムアクションが完了するまでインスタンスは保留中の状態になります。

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- API 詳細については、「 コマンドレットリファレンスRecordLifecycleActionHeartbeat」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`ResumeProcesses` で を使用する CLI

以下のコード例は、`ResumeProcesses` の使用方法を示しています。

CLI

AWS CLI

中断されたプロセスを再開するには

この例では、指定された Auto Scaling グループの指定された一時停止スケーリングプロセスを再開します。

```
aws autoscaling resume-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「スケーリングプロセスの停止と再開」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスResumeProcesses」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定した Auto Scaling プロセスを再開します。

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

例 2: この例では、指定した Auto Scaling グループの中断されたすべての Auto Scaling プロセスを再開します。

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- API 詳細については、「 コマンドレットリファレンスResumeProcesses」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **SetDesiredCapacity**AWS SDKで を使用する CLI

以下のコード例は、SetDesiredCapacity の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスSetDesiredCapacity」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスSetDesiredCapacity」の「」を参照してください。

CLI

AWS CLI

Auto Scaling グループに希望する容量を設定する方法

この例では、指定された Auto Scaling グループの希望する容量を設定します。

```
aws autoscaling set-desired-capacity \
    --auto-scaling-group-name my-asg \
    --desired-capacity 2 \
    --honor-cooldown
```

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。

- API 詳細については、AWS CLI 「 コマンドリファレンスSetDesiredCapacity」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスSetDesiredCapacity」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- API 詳細については、Kotlin リファレンスのSetDesiredCapacity「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
```

```
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスSetDesiredCapacity」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループのサイズを設定します。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

例 2: この例では、指定した Auto Scaling グループのサイズを設定し、クールダウン期間が完了するまで待ってから、新しいサイズにスケールします。

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- API 詳細については、「 コマンドレットリファレンスSetDesiredCapacity」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
```

```
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
keep the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto
Scaling group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting
to set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスSetDesiredCapacity」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity})").as_str(),
               &err,
           ));
       }
       Ok(())
   }
```

- API 詳細については、AWS SDKRust APIリファレンスのSetDesiredCapacity「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SetInstanceHealth で を使用する CLI

以下のコード例は、SetInstanceHealth の使用方法を示しています。

CLI

AWS CLI

インスタンスのヘルスステータスを設定するには

指定されたインスタンスのヘルスステータスを Unhealthy に設定します。

```
aws autoscaling set-instance-health \
    --instance-id i-061c63c5eb45f0416 \
    --health-status Unhealthy
```

このコマンドでは何も出力されません。

- API 詳細については、AWS CLI 「 コマンドリファレンスSetInstanceHealth」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したインスタンスのステータスを「Unhealthy」に設定し、使用を中止します。Auto Scaling はインスタンスを終了して置き換えます。

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

例 2: この例では、指定したインスタンスのステータスを「Healthy」に設定し、そのまま稼働させます。Auto Scaling グループのヘルスチェック猶予期間は無視されます。

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- API 詳細については、「 コマンドレットリファレンスSetInstanceHealth」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SetInstanceProtection で を使用する CLI

以下のコード例は、SetInstanceProtection の使用方法を示しています。

CLI

AWS CLI

例 1: インスタンスに対するインスタンスの保護設定を有効にするには

この例では、指定されたインスタンスのインスタンス保護を有効にします。

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg --protected-from-scale-in
```

このコマンドでは何も出力されません。

例 2: インスタンスに対するインスタンスの保護設定を無効にする方法

この例では、指定されたインスタンスのインスタンス保護を無効にします。

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --no-protected-from-scale-in
```

このコマンドでは何も出力されません。

- API 詳細については、AWS CLI「 コマンドリファレンスSetInstanceProtection」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したインスタンスのインスタンス保護を有効にします。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

例 2: この例では、指定したインスタンスのインスタンス保護を無効にします。

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- API 詳細については、「 コマンドレットリファレンスSetInstanceProtection」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SuspendProcesses で を使用する CLI

以下のコード例は、SuspendProcesses の使用方法を示しています。

CLI

AWS CLI

Auto Scaling プロセスを中断するには

この例では、指定された Auto Scaling グループの指定されたスケーリングプロセスを一時停止します。

```
aws autoscaling suspend-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「スケーリングプロセスの停止と再開」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスSuspendProcesses」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの指定した Auto Scaling プロセスを中断します。

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

例 2: この例では、指定した Auto Scaling グループのすべての Auto Scaling プロセスを中断します。

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- API 詳細については、「 コマンドレットリファレンスSuspendProcesses」の「」を参照してください。 AWS Tools for PowerShell

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **TerminateInstanceInAutoScalingGroup**で を使用する AWS SDK CLI

以下のコード例は、TerminateInstanceInAutoScalingGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ
- レジリエントなサービスの構築と管理

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。

CLI

AWS CLI

Auto Scaling グループのインスタンスを終了する方法

この例では、グループのサイズを更新せずに、指定された Auto Scaling グループの指定されたインスタンスを終了します。Amazon EC2 Auto Scaling は、指定されたインスタンスの終了後、代替インスタンスを起動します。

```
aws autoscaling terminate-instance-in-auto-scaling-group \
    --instance-id i-061c63c5eb45f0416 \
    --no-should-decrement-desired-capacity
```

出力:

```
{
    "Activities": [
        {
            "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
            "Cause": "",
            "StartTime": "2020-10-31T20:34:25.680Z",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- API 詳細については、AWS CLI 「 コマンドリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
       try {
           TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                   .instanceId(instanceId)
                   .shouldDecrementDesiredCapacity(false)
                   .build();

           autoScalingClient.terminateInstanceInAutoScalingGroup(request);
           System.out.println("You have terminated instance " + instanceId);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- API 詳細については、Kotlin リファレンスのTerminateInstanceInAutoScalingGroup「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' =>
 $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
```

```
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
 $attempts < 5) {
                error_log("Cannot terminate an instance while it is still
 pending. Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定したインスタンスを終了し、Auto Scaling グループの希望する容量を減らして、Auto Scaling が代替インスタンスを起動しないようにします。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

出力:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
```

```
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

例 2: この例では、Auto Scaling グループの希望する容量を減らすことなく、指定したインスタンスを終了します。Auto Scaling が代替インスタンスを起動します。

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

出力:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- API 詳細については、「 コマンドレットリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
 capacity
                                  of the group. When passing True for this
 parameter,
                                  you can stop an instance without having a
 replacement
                                  instance start when the desired capacity
 threshold is
                                  crossed.
        :return: A dictionary containing details of the scaling activity that
 occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
 self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
 ShouldDecrementDesiredCapacity=decrease_capacity
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
```

```
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting
to terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスTerminateInstanceInAutoScalingGroup」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
```

```
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }
```

```
        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }
```

- API 詳細については、AWS SDKRust APIリファレンス のTerminateInstanceInAutoScalingGroup「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **UpdateAutoScalingGroup**AWS SDKで を使用する CLI

以下のコード例は、UpdateAutoScalingGroup の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 基本を学ぶ
- レジリエントなサービスの構築と管理

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
```

```
            return true;
        }
        else
        {
            return false;
        }
    }
```

- API 詳細については、「 AWS SDK for .NET APIリファレンスUpdateAutoScalingGroup」の「」を参照してください。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
        }
```

- API 詳細については、「 AWS SDK for C++ APIリファレンスUpdateAutoScalingGroup」の「」を参照してください。

CLI

AWS CLI

例 1: Auto Scaling グループのサイズ制限を更新する方法

この例では、最小サイズが 2、最大サイズが 10 で、指定された Auto Scaling グループを更新します。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --min-size 2 \
    --max-size 10
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループの容量制限の設定」を参照してください。 EC2 Auto Scaling

例 2: Elastic Load Balancing ヘルスチェックを追加し、使用するアベイラビリティーゾーンとサブネットを指定する方法

この例では、指定された Auto Scaling グループを更新して、Elastic Load Balancing のヘルスチェックを追加します。このコマンドは、IDs複数のアベイラビリティーゾーンのサブネットのリスト`--vpc-zone-identifier`で の値も更新します。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」のElastic Load Balancing と Amazon Auto Scaling」を参照してください。 EC2 Auto Scaling

例 3: プレイスメントグループと終了ポリシーを更新する方法

この例では、プレイスメントグループと終了ポリシーを更新します。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --placement-group my-placement-group \
    --termination-policies "OldestInstance"
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「Auto Scaling グループ」を参照してください。 EC2 Auto Scaling

例 4: 起動テンプレートの最新バージョンを使用する方法

この例では、最新の起動テンプレートバージョンを使用するように、指定された Auto Scaling グループを更新します。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「起動テンプレート」を参照してください。

例 5: 特定のバージョンの起動テンプレートを使用する方法

この例では、最新バージョンやデフォルトバージョンではなく、指定された起動テンプレートのバージョンを使用するように、指定された Auto Scaling グループを更新します。

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

このコマンドでは何も出力されません。

詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「起動テンプレート」を参照してください。

例 6: 混合インスタンスポリシーを定義し、容量のリバランスを有効にする方法

この例では、指定された Auto Scaling グループを更新して、混合インスタンスポリシーを使用し、容量のリバランスを有効にします。この構造により、スポット容量とオンデマンド容量でグループを指定し、アーキテクチャごとに異なる起動テンプレートを使用できます。

```
aws autoscaling update-auto-scaling-group \
    --cli-input-json file://~/config.json
```

config.json の内容:

```
{
    "AutoScalingGroupName": "my-asg",
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c6g.large",
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateName": "my-launch-template-for-arm",
                        "Version": "$Latest"
                    }
                },
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
```

```
    }
}
```

このコマンドでは何も出力されません。

詳細については、「Amazon Auto Scaling ユーザーガイド」の「複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ」を参照してください。 EC2 Auto Scaling

- API 詳細については、AWS CLI 「 コマンドリファレンスUpdateAutoScalingGroup」の「」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
                   .build();

           autoScalingClient.updateAutoScalingGroup(groupRequest);
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
```

```
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
 " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンスUpdateAutoScalingGroup」の「」を参照してください。

Kotlin

SDK Kotlin 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
```

```
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- API 詳細については、Kotlin リファレンスのUpdateAutoScalingGroup「」の「」を参照してください。 AWS SDK API

PHP

PHP 用の SDK

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
```

```
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- API 詳細については、「 AWS SDK for PHP APIリファレンスUpdateAutoScalingGroup」の「」を参照してください。

PowerShell

のツール PowerShell

例 1: この例では、指定した Auto Scaling グループの最小サイズと最大サイズを更新します。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

例 2: この例では、指定した Auto Scaling グループのデフォルトのクールダウン期間を更新します。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

例 3: この例では、指定した Auto Scaling グループのアベイラビリティーゾーンを更新します。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

例 4: この例では、指定した Auto Scaling グループを更新して、Elastic Load Balancing のヘルスチェックを使用します。

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- API 詳細については、「 コマンドレットリファレンスUpdateAutoScalingGroup」の「」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling
 group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
```

```
                    "The Auto Scaling group '%s' is currently in use and cannot
 be modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting
 to update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 詳細については、「 for AWS SDKPython (Boto3) APIリファレンスUpdateAutoScalingGroup」の「」を参照してください。

Rust

SDK Rust 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
 Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
```

```
}
```

- API 詳細については、AWS SDKRust APIリファレンスのUpdateAutoScalingGroup「」の「」を参照してください。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用した Auto Scaling のシナリオ AWS SDKs

次のコード例は、 を使用して Auto Scaling で一般的なシナリオを実装する方法を示しています AWS SDKs。これらのシナリオは、Auto Scaling 内で複数の機能を呼び出すか、他の AWS のサービスと組み合わせて、特定のタスクを実行する方法を示します。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- を使用して回復力のあるサービスを構築および管理します。 AWS SDK

を使用して回復力のあるサービスを構築および管理します。 AWS SDK

次のコード例は、本、映画、曲のレコメンデーションを返すロードバランシングウェブサービスの作成方法を示しています。この例では、障害に対するサービスの対応方法と、障害発生時の耐障害性を高めるためにサービスを再構築する方法を示しています。

- Amazon EC2 Auto Scaling グループを使用して、起動テンプレートに基づいて Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを作成し、インスタンスの数を指定された範囲内に維持します。
- Elastic Load Balancing を使用してHTTPリクエストを処理し、配信します。
- Auto Scaling グループ内のインスタンスの状態を監視し、正常なインスタンスにのみリクエストを転送します。
- 各EC2インスタンスで Python ウェブサーバーを実行してHTTPリクエストを処理します。ウェブサーバーはレコメンデーションとヘルスチェックを返します。

- Amazon DynamoDB テーブルを使用してレコメンデーションサービスをシミュレートできます。
- AWS Systems Manager パラメータを更新して、リクエストとヘルスチェックに対するウェブサーバーの応答を制御します。

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
```

```
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }
```

```
    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");
```

```
        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
```

```
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
```

```
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
```

```
                    "\nFor this example to work, the default security group for
your default VPC must\n"
                    + "allows access from this computer. You can either add it
automatically from this\n"
                    + "example or add it yourself using the AWS Management
Console.\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
```

```
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
```

```
                        $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
                        "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
```

```
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
```

```
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");
```

```
        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
```

```
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Auto Scaling アクションと Amazon EC2アクションをラップするクラスを作成します。

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
```

```
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
```

```
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
```

```
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
```

```
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
```

```
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
```

```
   /// the instance is started. This script installs the Python packages and
starts a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
```

```
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
 zones.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
```

```
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
```

```
   public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
   {
       try
       {
           var subnets = new List<Subnet>();
           var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
               new DescribeSubnetsRequest()
               {
                   Filters = new List<Amazon.EC2.Model.Filter>()
                   {
                       new("vpc-id", new List<string>() { vpcId }),
                       new("availability-zone", availabilityZones),
                       new("default-for-az", new List<string>() { "true" })
                   }
               });

           // Get the entire list using the paginator.
           await foreach (var subnet in subnetPaginator.Subnets)
           {
               subnets.Add(subnet);
           }

           return subnets;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
           if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
           {
               _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
           }

           throw;
       }
       catch (Exception ex)
       {
           _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
           throw;
       }
   }

   /// <summary>
```

```
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
```

```
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
```

```
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
```

```
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);
```

```
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
```

```
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
```

```
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
```

```
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
```

```
                    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
```

```
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
```

```
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }
```

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
```

```
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
```

```
    {
        var createResponse = await
_amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
```

```
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
```

```
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
```

```
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
```

```
                    new DeleteTargetGroupRequest() { TargetGroupArn =
 targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
```

```
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
```

```
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
```

```
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- API 詳細については、「 AWS SDK for .NET APIリファレンス」の以下のトピックを参照してください。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups

- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
```

```
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
```

```
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
```

```
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
```

```
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();
```

```
                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
```

```
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
```

```
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);
```

```
        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
```

```
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }
```

```
   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode =
response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
                               new
InputStreamReader(response.getEntity().getContent()));
```

```
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
balancer targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
```

```
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Auto Scaling アクションと Amazon EC2アクションをラップするクラスを作成します。

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
```

```
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
```

```
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                                // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
```

```
    * Detaches a role from an instance profile, detaches policies from the role,
    * and deletes all the resources.
    */
   public void deleteInstanceProfile(String roleName, String profileName) {
       try {
           software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                   .builder()
                   .instanceProfileName(profileName)
                   .build();

           GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
           String name = response.instanceProfile().instanceProfileName();
           System.out.println(name);

           RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .roleName(roleName)
                   .build();

           getIAMClient().removeRoleFromInstanceProfile(profileRequest);
           DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .build();

           getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
           System.out.println("Deleted instance profile " + profileName);

           DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           // List attached role policies.
           ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                   .listAttachedRolePolicies(role -> role.roleName(roleName));
           List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
           for (AttachedPolicy attachedPolicy : attachedPolicies) {
               DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
```

```
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
```

```
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
```

```
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
```

```
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
```

```
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }
```

```
   // Gets data about the profile associated with an instance.
   public String getInstanceProfile(String instanceId) {
       Filter filter = Filter.builder()
               .name("instance-id")
               .values(instanceId)
               .build();

       DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
               .builder()
               .filters(filter)
               .build();

       DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
               .describeIamInstanceProfileAssociations(associationsRequest);
       return response.iamInstanceProfileAssociations().get(0).associationId();
   }

   public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
       ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
       ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
       for (Policy policy : listPoliciesResponse.policies()) {
           if (policy.policyName().equals(policyName)) {
               // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                       .builder()
                       .policyArn(policy.arn())
                       .build();
               ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                       .listEntitiesForPolicy(listEntitiesRequest);
               if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                       || !listEntitiesResponse.policyRoles().isEmpty()) {
                   // Detach the policy from any entities it is attached to.
                   DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
```

```
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
```

```
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
```

```
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
```

```
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }
```

```
        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
```

```
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
```

```
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
```

```
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
```

```
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
```

```
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
 enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Systems Manager のアクションをラップするクラスを作成します。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
```

```
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API 詳細については、「 AWS SDK for Java 2.x APIリファレンス」の以下のトピックを参照してください。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones

- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
```

```
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

すべてのリソースをデプロイするための手順を作成します。

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";
```

```
import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
```

```
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
```

```
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
```

```
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
```

```
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
```

```
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
```

```
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
```

```
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
```

```
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
```

```
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
```

```
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
```

```
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
```

```
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

デモを実行するための手順を作成します。

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
```

```
import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
```

```
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
```

```
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
```

```
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
```

```
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
```

```
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
```

```
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
```

```
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
```

```
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
```

```
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

すべてのリソースを破棄するための手順を作成します。

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
```

```
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
```

```
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
```

```
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
```

```
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
```

```
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
```

```
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
```

```
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
```

```
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
```

```
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
```

```
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});
```

```
      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
```

```
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
```

```
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- API 詳細については、「 AWS SDK for JavaScript APIリファレンス」の以下のトピックを参照してください。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
```

```
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
```

```
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
```

```
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
```

```
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
```

```
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
```

```
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
```

```
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
 created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
```

```
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../workflows/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
```

```
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Auto Scaling アクションと Amazon EC2アクションをラップするクラスを作成します。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.
```

```
        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
```

```
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
```

```
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
```

```
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
```

```
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
```

```
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
```

```
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")
```

```
    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
```

```
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template
```

```
    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
```

```
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
```

```
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
```

```
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
```

```
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
```

```
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
```

```
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
```

```
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open
```

```
    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Elastic Load Balancing のアクションをラップするクラスを作成します。

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
 specifies how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
 times and
        lower thresholds. In production, you might want to decrease the
 sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
```

```
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
```

```
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
```

```
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
```

```
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
```

```
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
```

```
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
```

```
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
                    "Ensure the load balancer exists, is in the correct AWS
region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

DynamoDB を使用してレコメンデーションサービスをシミュレートするクラスを作成します。

```
class RecommendationService:
```

```
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
```

```
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
```

```
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Systems Manager のアクションをラップするクラスを作成します。

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name
```

```
    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- API 詳細については、 AWS SDK for Python (Boto3) APIリファレンスの以下のトピックを参照してください。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいでのこのサービスの使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

Amazon EC2 Auto Scaling の問題のトラブルシューティング

Amazon EC2 Auto Scaling には、問題のトラブルシューティングに役立つ具体的でわかりやすいエラーが用意されています。エラーメッセージは規模の拡大や縮小の説明から取得できます。

トピック

スケーリングアクティビティからのエラーメッセージを取得する

スケーリングアクティビティの説明からエラーメッセージを取得するには、 describe-scaling-activities コマンドを使用します。スケーリングについては、6 週間前からの記録が保存されています。このためのリストでは、最新のスケーリングを最上位に、スケーリングが開始時刻順に並べられます。

> **Note**
>
> スケーリングアクティビティは、Auto Scaling グループのアクティビティタブの Amazon EC2 Auto Scaling コンソールのアクティビティ履歴にも表示されます。

特定の Auto Scaling グループのスケーリングを表示するには、次のコマンドを使用します。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

以下は、StatusCode で現在のスケーリングのステータスを示し、StatusMessage でエラーメッセージを示している応答の例です。

```
{
    "Activities": [
        {
            "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e.  Status
 Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with
 token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed
 with ABANDON Result",
            "Cause": "At 2021-01-11T00:35:52Z a user request created an
 AutoScalingGroup changing the desired capacity from 0 to 1.  At 2021-01-11T00:35:53Z
 an instance was started in response to a difference between desired and actual
 capacity, increasing the capacity from 0 to 1.",
            "StartTime": "2021-01-11T00:35:55.542Z",
            "EndTime": "2021-01-11T01:06:31Z",
            "StatusCode": "Cancelled",
            "StatusMessage": "Instance failed to complete user's Lifecycle Action:
 Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned:
 Lifecycle Action Completed with ABANDON Result",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
      ...
    ]
}
```

出力のフィールドの説明については、「Amazon EC2 Auto Scaling APIリファレンス」の「アクティビティ」を参照してください。

削除されたグループのスケーリングアクティビティを表示するには

Auto Scaling グループが削除された後にスケーリングアクティビティを表示するには、次のように describe-scaling-activities コマンドに `--include-deleted-groups`オプションを追加します。

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg --include-
deleted-groups
```

下記は、削除されたグループのスケーリングに関する応答の例です。

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance.  Status Reason: Your Spot
 request price of 0.001 is lower than the minimum required Spot request fulfillment
 price of 0.0031. Launching EC2 instance failed.",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup
 constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3.
  At 2021-01-13T20:47:27Z an instance was started in response to a difference between
 desired and actual capacity, increasing the capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than the
 minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance
 failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
      ...
    ]
}
```

スケーリングアクティビティをオフにする

スケーリングポリシーやスケジュールされたアクションに影響を受けずに問題を調査する必要がある場合、次のオプションがあります。

- AlarmNotification プロセスと ScheduledActions プロセスを中断することで、すべての動的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グループの希望するキャパシティが変更されるのを防ぎます。詳細については、「Amazon EC2 Auto Scaling プロセスの停止と再開」を参照してください。

- 個々の動的スケーリングポリシーを無効にして、負荷の変化に応じてグループの希望するキャパシティを変更しないようにします。詳細については、「Auto Scaling グループのスケーリングポリシーを無効化する」を参照してください。
- 個々のターゲット追跡スケーリングポリシーを更新して、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 (キャパシティを追加) のみを行います。この方法では、グループの希望するキャパシティが縮小しないようになりますが、負荷が増加するとキャパシティを増やすことができます。詳細については、「Amazon EC2 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。
- 予測スケーリングポリシーを予測のみモードに更新します。予測のみモードでは、予測スケーリングは予測を生成し続けますが、自動的にキャパシティを増やすことはありません。詳細については、「Auto Scaling グループの予測スケーリングポリシーを作成する」を参照してください。

その他のトラブルシューティングリソース

以下のページでは、Amazon EC2 Auto Scaling の問題のトラブルシューティングに関する追加情報を提供します。

- Auto Scaling グループのスケーリングアクティビティを検証する
- Amazon EC2 Auto Scaling コンソールでモニタリンググラフを表示する
- Auto Scaling グループでのインスタンスのヘルスチェック
- ライフサイクルフックの考慮事項と制限
- Auto Scaling グループでライフサイクルアクションを完了する
- Amazon VPC を使用して Auto Scaling インスタンスのネットワーク接続を提供する
- Auto Scaling グループからインスタンスを一時的に削除する
- Auto Scaling グループのスケーリングポリシーを無効化する
- Amazon EC2 Auto Scaling プロセスの停止と再開
- スケールイン中に終了する Auto Scaling インスタンスを制御する
- Auto Scaling インフラストラクチャを削除する
- Auto Scaling リソースとグループのクォータ

以下の AWS リソースも役立ちます。

- AWS ナレッジセンターの Amazon EC2 Auto Scaling トピック

- AWS re:Post に関する Amazon EC2 Auto Scaling の質問
- コンピューティングブログの Amazon EC2 Auto Scaling AWS の投稿
- AWS CloudFormation ユーザーガイド CloudFormation のトラブルシューティング

トラブルシューティングには、エキスパート、またはヘルパーコミュニティによる反復的な照会と検出が必要になることがよくあります。このセクションの提案を試した後も問題が解決しない場合は、AWS Support（「」の「 サポート AWS Management Console」、「 サポートセンター」) に問い合わせるか、Amazon EC2 Auto Scaling タグを使用して AWS re:Post に質問をしてください。

Amazon EC2 Auto Scaling のトラブルシューティング: EC2インスタンスの起動失敗

このページでは、起動に失敗したEC2インスタンス、考えられる原因、および問題を解決するために実行できる手順について説明します。

エラーメッセージを取得するには、「スケーリングアクティビティからのエラーメッセージを取得する」を参照してください。

EC2 インスタンスの起動に失敗すると、次のエラーメッセージが 1 つ以上表示されることがあります。

起動に関する問題

- リクエストされた設定は現在サポートされていません。
- セキュリティグループ <セキュリティグループ名> は存在しません。EC2 インスタンスの起動に失敗しました。
- EC2 instance> に関連付けられたキーペア <key pair。EC2 インスタンスの起動に失敗しました。
- リクエストされたインスタンスタイプ (<インスタンスタイプ>) は、リクエストされたアベイラビリティーゾーン (<インスタンスのアベイラビリティーゾーン>) ではサポートされません。
- 設定したポットリクエスト料金 0.015 は、スポットリクエストに最低限必要な料金 0.0735 を下回っています...。
- デバイス名 <device name> が無効です/無効なデバイス名をアップロードしようとしています。EC2 インスタンスの起動に失敗しました。
- パラメータの値 (<インスタンスストレージデバイスに関連付けられた名前>) virtualName が無効です。EC2 インスタンスの起動に失敗しました。
- EBS ブロックデバイスマッピングは、インスタンスストア ではサポートされていませんAMIs。

- プレイスメントグループは、タイプが '<instance type>' のインスタンスでは使用できません。EC2 インスタンスの起動に失敗しました。
- Client.InternalError: 起動時のクライアントエラー。
- 現在、お客様がリクエストしたアベイラビリティゾーン内には、<instance type> の十分なキャパシティーがありません..。EC2 インスタンスの起動に失敗しました。
- リクエストされた予約には、このリクエストに十分な互換性と利用可能なキャパシティがありません。EC2 インスタンスの起動に失敗しました。
- キャパシティブロックの予約 <予約 ID> はまだアクティブではありません。EC2 インスタンスの起動に失敗しました。
- リクエストに適合した、使用可能なスポットキャパシティーはありません。EC2 インスタンスの起動に失敗しました。
- <インスタンス数> 個のインスタンスがすでに実行中です。EC2 インスタンスの起動に失敗しました。

リクエストされた設定は現在サポートされていません。

原因: 起動テンプレートまたは起動設定の一部のオプションは、インスタンスタイプと互換性がない、またはリクエストされた AWS リージョンまたはアベイラビリティーゾーンでインスタンス設定がサポートされていない可能性があります。

解決策: 別のインスタンス設定を試す。要件を満たすインスタンスタイプを検索するには、「Amazon ユーザーガイド」の「Amazon EC2インスタンスタイプの検索」を参照してください。 EC2

この問題の詳細な解決方法については、下記を確認してください。

- インスタンスタイプでAMIサポートされている を選択していることを確認します。例えば、インスタンスタイプがインテル Xeon プロセッサではなく Arm ベースの AWS Graviton プロセッサを使用している場合、Arm 互換の が必要ですAMI。互換性のあるインスタンスタイプの選択の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプを変更するための互換性」を参照してください。
- リクエストしたリージョンとアベイラビリティーゾーンの中で、インスタンスタイプが利用可能であるかをテストします。最新世代のインスタンスタイプには、特定のリージョンまたはアベイラビリティーゾーンではまだ利用できないものがあります。古い世代のインスタンスタイプについては、新しいリージョンとアベイラビリティーゾーンで利用できない場合があります。ロケーション (リージョンまたはアベイラビリティーゾーン) で提供されるインスタンスタイプを検索するには、

describe-instance-type-offerings コマンドを使用します。詳細については、「Amazon ユーザーガイド」の「Amazon EC2インスタンスタイプの検索」を参照してください。 EC2

- ハードウェア専有インスタンスまたは Dedicated Hosts を使用する場合は、ハードウェア専有インスタンスまたは Dedicated Hosts としてサポートされているインスタンスタイプを選択する必要があります。

セキュリティグループ <セキュリティグループ名> は存在しません。EC2 インスタンスの起動に失敗しました。

原因: 起動テンプレートもしくは起動設定で指定されたセキュリティグループが、削除されている可能性があります。

解決策:

1. describe-security-groups コマンドを使用して、アカウントに関連付けられているセキュリティグループのリストを取得します。
2. リストから、使用するセキュリティグループを選択します。代わりにセキュリティグループを作成するには、create-security-group コマンドを使用します。
3. 新しい起動テンプレート、または起動設定を作成します。
4. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

EC2 instance> に関連付けられたキーペア <key pair。EC2 インスタンスの起動に失敗しました。

原因: インスタンスの起動時に使用されたキーペアが削除された可能性があります。

解決策:

1. describe-key-pairs コマンドを使用して、利用可能なキーペアのリストを取得します。
2. リストから、使用するキーペアを選択します。代わりにキーペアを作成するには、create-key-pair コマンドを使用します。
3. 新しい起動テンプレート、または起動設定を作成します。
4. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

リクエストされたインスタンスタイプ (<インスタンスタイプ>) は、リクエストされたアベイラビリティーゾーン (<インスタンスのアベイラビリティーゾーン>) ではサポートされません。

エラーメッセージ: リクエストされたインスタンスタイプ (<instance type>) は、リクエストされたアベイラビリティーゾーン (<instance Availability Zone>) でサポートされていません。EC2 インスタンスの起動に失敗しました。

原因: Auto Scaling グループで指定されているアベイラビリティーゾーンでは、選択したインスタンスタイプがサポートされていません。

解決策:

1. describe-instance-type-offerings コマンドを使用するか、インスタンスタイプページのネットワークペインでアベイラビリティーゾーンの値を確認して、Amazon EC2コンソールから選択したインスタンスタイプをサポートするアベイラビリティーゾーンを確認します。
2. update-auto-scaling-group コマンドを使用して、Auto Scaling グループの設定でサポートされていないゾーンのサブネットを更新または削除します。詳細については、「アベイラビリティーゾーンを追加する」を参照してください。

設定したポットリクエスト料金 0.015 は、スポットリクエストに最低限必要な料金 0.0735 を下回っています...。

原因: リクエストしたスポットの上限価格が、選択したインスタンスタイプでのスポット料金を下回っています。

解決策: より高いスポット上限価格 (オンデマンド料金が利用できます) を指定して、新しいリクエストを送信します。以前は、支払うスポット料金は入札に基づいていました。現在は、その時点のスポット料金が適用されます。上限価格を高く設定することで、Amazon EC2 スポットサービスが必要な容量を起動して維持する可能性が高くなります。

デバイス名 <device name> が無効です/無効なデバイス名をアップロードしようとしています。EC2 インスタンスの起動に失敗しました。

原因 1: 起動テンプレートまたは起動設定のブロックデバイスマッピングに、利用が不可能な、または現在サポートされていないブロックデバイス名が含まれている可能性があります。

解決策:

1. ご使用の具体的なインスタンス設定で、使用が可能なデバイス名を確認します。デバイスの名前付けの詳細については、「Amazon EC2ユーザーガイド」の「Linux インスタンスのデバイス名」を参照してください。
2. Auto Scaling グループの一部ではない Amazon EC2インスタンスを手動で作成し、問題を調査します。ブロックデバイスの命名設定が Amazon マシンイメージ (AMI) の名前と競合する場合、インスタンスは起動時に失敗します。詳細については、「Amazon EC2ユーザーガイド」の「ブロックデバイスマッピング」を参照してください。
3. インスタンスが正常に起動したことを確認したら、describe-volumes コマンドを使用して、どのようにボリュームがインスタンスに公開されているかを確認します。
4. ボリュームの説明にリストされているデバイス名を使用して、新しい起動テンプレートもしくは起動設定を作成します。
5. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

パラメータの値 (<インスタンスストレージデバイスに関連付けられた名前>) virtualName が無効です。EC2 インスタンスの起動に失敗しました。

原因: ブロックデバイスに関連付けられている仮想名の指定形式が正しくありません。

解決策:

1. `virtualName` パラメータでデバイス名を指定して、新しい起動テンプレートか起動設定を作成します。デバイス名の形式の詳細については、「Amazon EC2ユーザーガイド」の「Linux インスタンスでのデバイスの名前付け」を参照してください。
2. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

EBS ブロックデバイスマッピングは、インスタンスストア ではサポートされていませんAMIs。

原因: 起動テンプレートあるいは起動設定で指定されたブロックデバイスマッピングが、お使いのインスタンスではサポートされていません。

解決策:

1. お使いのインスタンスタイプでサポートされるブロックデバイスマッピングを指定して、新しい起動テンプレートか起動設定を作成します。詳細については、「Amazon EC2ユーザーガイド」の「ブロックデバイスマッピング」を参照してください。
2. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

プレイスメントグループは、タイプが '<instance type>' のインスタンスでは使用できません。EC2 インスタンスの起動に失敗しました。

原因: お使いのクラスタープレイスメントグループに無効なインスタンスタイプが含まれています。

解決策:

1. プレイスメントグループでサポートされている有効なインスタンスタイプについては、「Amazon EC2ユーザーガイド」の「プレイスメントグループ」を参照してください。
2. 新しいプレイスメントグループを作成するには、「プレイスメントグループ」で説明している手順に従います。
3. または、サポートされるインスタンスタイプを使用する、新しい起動テンプレートもしくは起動設定を作成します。
4. update-auto-scaling-group コマンドを使用して、新しいプレイスメントグループ、起動テンプレート、または起動設定で Auto Scaling グループを更新します。

Client.InternalError: 起動時のクライアントエラー。

問題: Amazon EC2 Auto Scaling は、暗号化されたEBSボリュームを持つインスタンスを起動しようとしますが、サービスにリンクされたロールは、暗号化に使用される AWS KMS カスタマーマネージドキーにアクセスできません。詳細については、「暗号化されたボリュームで使用するために必要な AWS KMS キーポリシー」を参照してください。

原因 1: 適切なサービスにリンクされたロールがカスタマーマネージドキーを使用するための、アクセス許可を付与できるキーポリシーが必要です。

解決策 1: 次のように、サービスにリンクされたロールがカスタマーマネージドキーを使用することを許可します。

1. この Auto Scaling グループで使用するサービスにリンクされたロールを決定します。

2. カスタマーマネージドキーのキーポリシーを更新して、サービスにリンクされたロールでカスタマーマネージドキーが使用できるように許可します。
3. サービスにリンクされたロールを使用するため、Auto Scaling グループを更新します。

サービスにリンクされたロールにカスタマーマネージドキーの使用を許可するキーポリシーの例については、「例 1: カスタマー管理キーへのアクセスを許可するキーポリシーセクション」を参照してください。

原因 2: カスタマーマネージドキーと Auto Scaling グループが異なる AWS アカウントにある場合は、カスタマーマネージドキーへのクロスアカウントアクセスを設定して、カスタマーマネージドキーを使用するアクセス許可を適切なサービスにリンクされたロールに付与する必要があります。

解決策 2: 次のように、外部アカウントのサービスにリンクされたロールによる、ローカルアカウント内のカスタマーマネージドキーの使用を許可します。

1. カスタマーマネージドキーのキーポリシーを更新して、Auto Scaling グループアカウントが、そのカスタマーマネージドキーにアクセスできるようにします。
2. 許可を作成できる Auto Scaling グループアカウントで、IAM ユーザーまたはロールを定義します。
3. この Auto Scaling グループで使用するサービスにリンクされたロールを決定します。
4. 適切なサービスにリンクされたロールを付与対象プリンシパルとして、カスタマー管理キーに対するグラントを作成します。
5. サービスにリンクされたロールを使用するため、Auto Scaling グループを更新します。

詳細については、「例 2: カスタマー管理キーへのクロスアカウントアクセスを許可するキーポリシーセクション」を参照してください。

解決策 3: Auto Scaling グループと同じ AWS アカウントにあるカスタマーマネージドキーを使用する

1. スナップショットをコピーし、Auto Scaling グループと同じアカウントに属する別のカスタマーマネージドキーを使用して再暗号化を行います。
2. サービスにリンクされたロールに新しいカスタマーマネージドキーの使用を許可します。解決策 1 のステップを参照してください。

現在、お客様がリクエストしたアベイラビリティゾーン内には、<instance type> の十分なキャパシティーがありません..。EC2 インスタンスの起動に失敗しました。

エラーメッセージ: We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>) (現在、リクエストされたアベイラビリティーゾーン (<requested Availability Zone>) に、<instance type> の十分なキャパシティーがありません) 追加のキャパシティーをプロビジョニングする作業を進めます。現時点では、リクエストでアベイラビリティーゾーンを指定しないか、<このインスタンスタイプを現在サポートしているアベイラビリティーゾーンのリスト> から選択することによって、<インスタンスタイプ> のキャパシティーを取得できます。EC2 インスタンスの起動に失敗しました。

原因: 現時点では、リクエストされたインスタンスタイプとアベイラビリティーゾーンの組み合わせはサポートされていません。

解決策: この問題を解決するには、以下の手順を試行します。

- Amazon EC2 Auto Scaling が他の有効なアベイラビリティーゾーンでこのインスタンスタイプの容量を見つけるまで数分待ちます。
- Auto Scaling グループを他のアベイラビリティーゾーンに拡張してください。詳細については、「アベイラビリティーゾーンを追加する」を参照してください。
- ベストプラクティスとして、さまざまなインスタンスタイプのセットを使用することで、特定の 1 つのインスタンスタイプに依存することを防ぎます。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。

リクエストされた予約には、このリクエストに十分な互換性と利用可能なキャパシティがありません。EC2 インスタンスの起動に失敗しました。

原因 1: `targeted` オンデマンドキャパシティ予約で起動できるインスタンスの数の上限に達しました。

解決策 1: `targeted` オンデマンドキャパシティ予約で起動できるインスタンスの数を増やすか、キャパシティ予約グループを使用して、リザーブドキャパシティ以外のインスタンスが通常のオンデマンドキャパシティとして起動されるようにします。詳細については、「キャパシティ予約を使用して特定のアベイラビリティーゾーンでキャパシティを予約する 」を参照してください。

原因 2: 1 つのキャパシティブロックで起動できるインスタンスの数の上限に達しました。

キャパシティブロックでは、最初に購入したキャパシティの量によって制約されます。予想よりも起動回数が多くなり、使用可能なすべてのキャパシティを使い果たすと、起動できなくなります。インスタンスの終了には、完全に終了する前に、長いクリーンアッププロセスが必要です。この間は再利用できません。これにより、起動できない可能性もあります。詳細については、「使用アイテム Capacity Blocks 機械学習ワークロード用の」を参照してください。

解決策 2: この問題を解決するには、以下の手順を試行します。

- リクエストはそのままにしておきます。キャパシティブロックインスタンスが終了中の場合は、インスタンスが終了し、キャパシティが再び利用可能になるまで数分待つ必要があります。Amazon EC2 Auto Scaling は、キャパシティーが利用可能になるまで、起動リクエストを自動的に作成し続けます。
- このエラーが頻繁に発生しないように、ピークワークロードに対応するのに十分なキャパシティを購入してください。

キャパシティブロックの予約 <予約 ID> はまだアクティブではありません。EC2 インスタンスの起動に失敗しました。

原因: 指定されたキャパシティブロックがまだアクティブではありません。

解決策: キャパシティブロックの推奨アプローチに従い、スケジュールされたスケーリングを使用します。これにより、予約がアクティブな場合にのみ Auto Scaling グループの希望するキャパシティを増やし、予約が終了する前に減らすことができます。

リクエストに適合した、使用可能なスポットキャパシティーはありません。EC2 インスタンスの起動に失敗しました。

原因: 現時点では、このスポットインスタンスのリクエストを満たすのに十分な空きキャパシティーがありません。

解決策: この問題を解決するには、以下の手順を試行します。

- キャパシティーは頻繁に変化するので、数分間待ってください。Amazon EC2 Auto Scaling は、キャパシティーが利用可能になるまで、起動リクエストを自動的に作成し続けます。
- Auto Scaling グループを他のアベイラビリティーゾーンに拡張してください。詳細については、「アベイラビリティーゾーンを追加する」を参照してください。

- ベストプラクティスとして、さまざまなインスタンスタイプのセットを使用することで、特定の 1 つのインスタンスタイプに依存することを防ぎます。詳細については、「複数のインスタンスタイプと購入オプションを使用する Auto Scaling グループ」を参照してください。

<インスタンス数> 個のインスタンスがすでに実行中です。EC2 インスタンスの起動に失敗しました。

原因: 1 つのリージョンで起動できるインスタンスの合計数には制限があります。 AWS アカウントを作成すると、リージョンごとに実行できるインスタンスの数にデフォルトの制限が設定されます。

解決策: この問題を解決するには、以下の手順を試行します。

- 現在の制限がニーズに見合っていない場合は、リージョンごとにクォータの引き上げをリクエストできます。詳細については、「Amazon ユーザーガイド」の「Amazon EC2 Service Quotas」を参照してください。 EC2
- この新しいリクエストでは、インスタンス数を減らして送信します (これは後で増やすことができます)。

Amazon EC2 Auto Scaling のトラブルシューティング: AMI問題

このページでは、 に関連する問題AMIs、考えられる原因、および問題を解決するために実行できる手順について説明します。

エラーメッセージを取得するには、「スケーリングアクティビティからのエラーメッセージを取得する」を参照してください。

の問題によりEC2インスタンスの起動に失敗するとAMI、次のエラーメッセージが 1 つ以上表示されることがあります。

AMI 問題点

- AMI> の AMI ID <ID は存在しません。EC2 インスタンスの起動に失敗しました。
- AMI <AMI ID> は保留中であり、実行できません。EC2 インスタンスの起動に失敗しました。
- デバイス名 <device name> が無効です。EC2 インスタンスの起動に失敗しました。
- 指定されたインスタンスタイプのアーキテクチャ「arm64」が、指定された のアーキテクチャ「x86_64」と一致しませんAMI。EC2 インスタンスの起動に失敗しました。

- AMI 「 < AMI ID>」は無効になっており、実行できません。EC2 インスタンスの起動に失敗しました。

> **Important**
>
> AWS は、 アクセスAMI許可を変更することで、 を別の AWS アカウントとAMIプライベートに共有することをサポートしています。を共有せずにプライベートAMIにすると、新しいインスタンスを起動するときに認可エラーが発生する可能性があります。プライベート の共有の詳細についてはAMIs、「Amazon EC2ユーザーガイド」の「特定の AWS アカウントAMI と を共有する」を参照してください。

AMI> の AMI ID <ID は存在しません。EC2 インスタンスの起動に失敗しました。

- 原因: 起動テンプレートまたは起動設定の作成後に が削除されたAMI可能性があります。
- 解決策:
  1. 有効な を使用して、新しい起動テンプレートまたは起動設定を作成しますAMI。
  2. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

AMI <AMI ID> は保留中であり、実行できません。EC2 インスタンスの起動に失敗しました。

原因: を作成したばかりで AMI (実行中のインスタンスのスナップショットまたはその他の方法で) 、まだ使用できない場合があります。

解決策: が使用可能AMIになるまで待ってから、起動テンプレートまたは起動設定を作成する必要があります。

デバイス名 <device name> が無効です。EC2 インスタンスの起動に失敗しました。

原因: EBSボリュームをEC2インスタンスにアタッチするときは、ボリュームに有効なデバイス名を指定する必要があります。選択した は、このデバイス名をサポートAMIしている必要があります。

解決策:

1. 新しい起動テンプレートまたは起動設定を作成し、 に正しいデバイス名を指定しますAMI。推奨される命名規則は、 の仮想化タイプによって異なりますAMI。詳細については、「Amazon EC2 ユーザーガイド」の「デバイス名」を参照してください。
2. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

指定されたインスタンスタイプのアーキテクチャ「arm64」が、指定された のアーキテクチャ「x86_64」と一致しませんAMI。EC2 インスタンスの起動に失敗しました。

原因 1: のアーキテクチャAMIと起動テンプレートまたは起動設定で使用されるインスタンスタイプが同じでない場合、Amazon EC2 Auto Scaling が互換性のないインスタンス設定を使用してインスタンスを起動しようとするとエラーが発生します。

解決策 1:

1. describe-images コマンドAMIを使用するか、Amazon EC2コンソールから、Amazon マシンイメージ (AMIs) ページの詳細ペインのアーキテクチャ値を確認して、 のアーキテクチャを確認します。
2. コマンドAMIを使用するdescribe-instance-typesか、インスタンスタイプ画面のアーキテクチャ列を確認して、Amazon EC2コンソールから、 と同じアーキテクチャを持つインスタンスタイプを見つけます。 互換性のあるインスタンスタイプの選択の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプを変更するための互換性」を参照してください。
3. と同じアーキテクチャのインスタンスタイプを使用して、新しい起動テンプレートまたは起動設定を作成しますAMI。
4. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

原因 2: Amazon EC2 Auto Scaling は、Auto Scaling グループの混合インスタンスポリシーで指定されたインスタンスタイプを起動しようとしますが、インスタンスタイプに起動テンプレートでAMI指定されたアーキテクチャと同じアーキテクチャがありません。

解決策 1:混合インスタンスポリシーには、アーキテクチャの異なるインスタンスタイプを含めないでください。

1. describe-images コマンドAMIを使用するか、Amazon EC2コンソールから、Amazon マシンイメージ (AMIs) ページの詳細ペインのアーキテクチャ値を確認して、 のアーキテクチャを確認します。
2. describe-instance-types コマンドを使用するか、インスタンスタイプ画面のアーキテクチャ列を確認して Amazon EC2コンソールから、混合インスタンスポリシーに含める各インスタンスタイプのアーキテクチャを確認します。互換性のあるインスタンスタイプの選択の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプを変更するための互換性」を参照してください。
3. update-auto-scaling-group コマンドを使用して、互換性のないインスタンスタイプを Auto Scaling グループで更新または削除します。

解決策 2: 同じ Auto Scaling グループで Arm (Graviton2) インスタンスと x86_64 (Intel) インスタンスの両方を起動するには、Arm 互換 AMIと Intel x86 互換 でサポートされている起動テンプレートAMIをそれぞれ使用して、混合インスタンスポリシーのインスタンスタイプと一致する必要があります。

1. describe-images コマンドを使用するか、Amazon マシンイメージ (AMIs) ページの詳細ペインのアーキテクチャ値を確認して、Amazon EC2コンソールからAMI既存の起動テンプレートののアーキテクチャを確認します。
2. 使用する予定の他のアーキテクチャAMIと一致する を使用して、新しい起動テンプレートを作成します。
3. Auto Scaling グループを更新して既存の起動テンプレートを上書きし、 update-auto-scaling-group コマンドを使用して互換性のあるインスタンスタイプごとに新しい起動テンプレートを指定します。詳細については、「インスタンスタイプに異なる起動テンプレートを使用する」を参照してください。

AMI 「 < AMI ID>」は無効になっており、実行できません。EC2 インスタンスの起動に失敗しました。

原因: 無効AMIになっている からインスタンスを起動しようとしています。詳細については、「Amazon EC2ユーザーガイド」の「 の無効化AMI」を参照してください。

解決策:

1. 新しい起動テンプレートまたは起動設定を作成し、無効にされていない AMI を指定します。
2. update-auto-scaling-group コマンドを使用して、新しい起動テンプレートまたは起動設定で Auto Scaling グループを更新します。

Amazon EC2 Auto Scaling のトラブルシューティング: ロードバランサーの問題

このページでは、Auto Scaling グループに関連付けられているロードバランサーが原因で発生する問題、考えられる原因、問題を解決するために実行できるステップに関する情報を提供します。

エラーメッセージを取得するには、「スケーリングアクティビティからのエラーメッセージを取得する」を参照してください。

Auto Scaling グループに関連付けられたロードバランサーの問題によりEC2インスタンスの起動に失敗すると、次のエラーメッセージが 1 つ以上表示されることがあります。

ロードバランサーに関する問題

- 1 つ以上のターゲットグループが見つかりませんでした。ロードバランサーの設定の確認が失敗しました。
- Load Balancer <ご使用のロードバランサー> が見つかりません。ロードバランサーの設定の確認が失敗しました。
- < ACTIVELoad Balancer名> という名前のロードバランサーはありません。ロードバランサーの設定の更新が失敗しました。
- EC2 インスタンス <インスタンス ID> が にありませんVPC。ロードバランサーの設定の更新が失敗しました。

> **Note**
>
> Reachability Analyzer を使用して、Auto Scaling グループのインスタンスにロードバランサー経由でアクセスできるかどうかを確認することで、接続の問題をトラブルシューティングできます。Reachability Analyzer によって自動的に検出されるさまざまなネットワーク設定ミスの問題については、「Reachability Analyzer ユーザーガイド」の「Reachability Analyzer explanation codes」(Reachability Analyzer の説明コード) を参照してください。

1 つ以上のターゲットグループが見つかりませんでした。ロードバランサーの設定の確認が失敗しました。

問題: Auto Scaling グループがインスタンスを起動すると、Amazon EC2 Auto Scaling は Auto Scaling グループに関連付けられている Elastic Load Balancing リソースが存在することを検証しよ

うとします。ターゲットグループが見つからない場合、スケーリングアクティビティが失敗し、`One or more target groups not found. Validating load balancer configuration failed.` というエラーが表示されます。

原因 1: Auto Scaling グループにアタッチされているターゲットグループが削除されています。

解決策 1: ターゲットグループなしで新しい Auto Scaling グループを作成するか、Amazon EC2 Auto Scaling コンソールまたは detach-load-balancer-target-groups コマンドを使用して、未使用のターゲットグループを Auto Scaling グループから削除できます。 Auto Scaling

原因 2: ターゲットグループが存在するが、Auto Scaling グループの作成ARN時にターゲットグループを指定しようとして問題が発生しました。リソースが正しい順序で作成されていません。

解決策 2: 新しい Auto Scaling グループを作成して、最後にターゲットグループを指定します。

Load Balancer <ご使用のロードバランサー> が見つかりません。ロードバランサーの設定の確認が失敗しました。

問題: Auto Scaling グループがインスタンスを起動すると、Amazon EC2 Auto Scaling は Auto Scaling グループに関連付けられている Elastic Load Balancing リソースが存在することを検証しようとします。Classic Load Balancer が見つからない場合、スケーリングアクティビティは失敗し、`Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed.` というエラーが表示されます。

原因 1: この Classic Load Balancer は削除されました。

解決策 1: ロードバランサーなしで新しい Auto Scaling グループを作成するか、Amazon EC2 Auto Scaling コンソールまたは detach-load-balancers コマンドを使用して、未使用のロードバランサーを Auto Scaling グループから削除できます。 Auto Scaling

原因 2: Classic Load Balancer は存在しますが、Auto Scaling グループの作成時に、ロードバランサーの名前を指定しようとして問題が発生しました。リソースが正しい順序で作成されていません。

解決策 2: 新しい Auto Scaling グループを作成する際は、最後にロードバランサーの名前を指定します。

< ACTIVELoad Balancer名> という名前のロードバランサーはありません。ロードバランサーの設定の更新が失敗しました。

原因: 指定されたロードバランサーが削除された可能性があります。

解決策: 新しいロードバランサーを作成してから新たに Auto Scaling グループを作成する、またはロードバランサーを指定せずに新しい Auto Scaling グループを作成します。

EC2 インスタンス <インスタンス ID> が にありませんVPC。ロードバランサーの設定の更新が失敗しました。

原因: 指定されたインスタンスが に存在しませんVPC。

解決策: インスタンスに関連付けられているロードバランサーを削除するか、新しい Auto Scaling グループを作成します。

Amazon EC2 Auto Scaling のトラブルシューティング: テンプレートの起動

次の情報を使用して、Auto Scaling グループの起動テンプレートを指定しようとする際に発生する可能性のある一般的な問題を診断して修正します。

インスタンスを起動できない

すでに指定された起動テンプレートを使用してインスタンスを起動できない場合は、一般的なトラブルシューティングについて以下を確認してください: Amazon EC2 Auto Scaling のトラブルシューティング: EC2インスタンスの起動失敗。

完全な形式の有効な起動テンプレートを使用する必要があります (無効な値)

問題: Auto Scaling グループの起動テンプレートを指定しようとすると、`You must use a valid fully-formed launch template` というエラーが表示されます。起動テンプレートの値は、起動テンプレートを使用している Auto Scaling グループが作成または更新された場合にのみ検証されるため、このエラーが発生する可能性があります。

原因 1: `You must use a valid fully-formed launch template` エラーが発生した場合、Amazon EC2 Auto Scaling が起動テンプレートが無効であると見なす原因となる問題があります。これは一般的なエラーで、いくつかの原因が考えられます。

解決策 1: 次のステップを実行して、トラブルシューティングを行います。

1. エラーメッセージの 2 つ目の部分に注目して、詳細を確認してください。`You must use a valid fully-formed launch template` のエラーに続いて、対処する必要のある問題を特定する、より具体的なエラーメッセージを参照してください。
2. 原因が見つからない場合は、「run-instances」コマンドで起動テンプレートをテストします。次の例のように、`--dry-run` オプションを使用します。これにより問題を再現し、その原因に関するインサイトを提供します。

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

3. 値が有効でない場合は、指定された適切なリソースが存在することを確認してください。例えば、Amazon EC2キーペアを指定する場合、リソースはアカウントと Auto Scaling グループを作成または更新するリージョンに存在する必要があります。
4. 期待した情報がない場合は、設定を確認し、必要に応じて起動テンプレートを調整します。
5. 変更を加えた後、`--dry-run` オプションで「run-instances」コマンドを再実行し、起動テンプレートが有効な値を使用していることを検証します。

詳細については、「Auto Scaling グループの起動テンプレートを作成する」を参照してください。

起動テンプレートを使用する権限がありません (許可が不十分)。

問題: Auto Scaling グループの起動テンプレートを指定しようとすると、`You are not authorized to use launch template` というエラーが表示されます。

原因 1: 起動テンプレートを使用しようとしたときに、使用しているIAM認証情報に十分なアクセス許可がない場合、起動テンプレートを使用する権限がないというエラーが表示されます。

解決策 1: この問題を解決するには、以下の手順を試行します。

- リクエストの作成に使用しているIAM認証情報に、 EC2APIアクションを含む必要な`ec2:RunInstances`アクションを呼び出すアクセス許可があることを確認します。何らかのタグを起動テンプレートで指定している場合は、`ec2:CreateTags` アクションを使用するためのアクセス許可も必要です。
- または、リクエストの作成に使用しているIAM認証情報に`AmazonEC2FullAccess`ポリシーが割り当てられていることを確認します。この AWS 管理ポリシーは、Amazon EC2 Auto Scaling、Elastic Load Balancing など CloudWatch、すべての Amazon EC2リソースおよび関連サービスへのフルアクセスを許可します。

IAM ポリシーの例など、起動テンプレートを使用するために必要なアクセス許可の詳細については、「Amazon EC2ユーザーガイド」の「アクセスIAM許可を使用して起動テンプレートへのアクセスを制御する」を参照してください。その他のIAMポリシーの例については、「」を参照してくださいAuto Scaling グループで Amazon EC2 起動テンプレートの使用を制御する。

原因 2: インスタンスプロファイルを指定する起動テンプレートを使用しようとする場合は、インスタンスプロファイルに関連付けられたIAMロールを渡すIAMアクセス許可が必要です。

解決策 2: リクエストの作成に使用しているIAM認証情報に、指定されたロールを Amazon EC2 Auto Scaling サービスに渡すための正しい`iam:PassRole`アクセス許可があることを確認します。ポリシーの詳細と例についてはIAM、「」を参照してくださいAmazon IAM インスタンスで実行されるアプリケーションの EC2 ロール。インスタンスプロファイルに関連するトラブルシューティングのトピックについては、「 IAMユーザーガイド」の「Amazon EC2と のトラブルシューティングIAM」を参照してください。

原因 3: 別の AMIで を指定する起動テンプレートを使用しようとしたときに AWS アカウント、 AMI がプライベートであり、 AWS アカウント 使用している と共有されていない場合、起動テンプレートを使用する権限がないというエラーが表示されます。

解決策 3: のアクセス許可に、使用しているアカウントAMIが含まれていることを確認します。詳細については、「Amazon EC2ユーザーガイド」の「特定の AMIと を共有する AWS アカウント」を参照してください。

関連情報

このサービスを利用する際に役立つ関連リソースは以下の通りです。

リソース	説明
「Amazon EC2 Auto Scaling API Reference」	各 API オペレーションのドキュメントには、リクエストパラメータと XML レスポンスが示され、言語固有の SDK リファレンストピックへのリンクが掲載されています。
「autoscaling」（「AWS CLI Command Reference」）	Auto Scaling グループの操作に使用できる AWS CLI コマンドの説明。
AWS Tools for PowerShell コマンドレットリファレンス	AWS Tools for PowerShell では、PowerShell のコマンドラインから AWS リソースのオペレーションのスクリプトを作成できます。
AWS CloudFormation で Auto Scaling グループを作成する	AWS::AutoScaling::AutoScalingGroup リソースを使用すると、手動操作なしで Auto Scaling グループを構築、モデル化、管理できます。
「Amazon EC2 Auto Scaling エンドポイントとクォータ」（「AWS 全般のリファレンス」）	Amazon EC2 Auto Scaling のリージョンとエンドポイントに関する情報
製品ページ	Amazon EC2 Auto Scaling に関する情報の主要なウェブページ。
AWS re:Post	AWS マネージドの質疑応答 (Q & A) サービス。技術的な質問に対して、エキスパートによるレビューを経た、クラウドソーシングされた回答を提供します。
「Amazon EC2 ユーザーガイド」の AMI の作成に関するページ	インスタンスからカスタム Amazon マシンイメージ (AMI) を作成する方法について説明します。

リソース	説明
「Amazon EC2 ユーザーガイド」の Linux インスタンスに接続する方法に関するページ	起動する Linux インスタンスに接続する方法について説明します。
「Amazon EC2 ユーザーガイド」の Windows インスタンスに接続する方法に関するページ	起動する Windows インスタンスに接続する方法について説明します。
「AWS の予想請求額をモニタリングする請求アラームの作成」(「Amazon CloudWatch ユーザーガイド」)	CloudWatch を使用して予想請求額をモニターリングする方法について説明します。
アプリケーション Auto Scaling ユーザーガイドhttps://docs.aws.amazon.com/autoscaling/application/userguide/	Amazon EC2 以外の Amazon Web Services のスケーラブルなリソースに対して自動スケーリングを設定する方法について説明します。

AWS の詳細を参照するために、以下の一般的なリソースを利用できます。

- クラスとワークショップ – AWS のスキルを磨き、実践的な経験を得るために役立つセルフペースラボに加えて、ロールベースのコースと特別コースへのリンクです。
- AWS デベロッパーセンター – チュートリアルの検索、ツールのダウンロード、AWS デベロッパーイベントの確認を行います。
- AWS デベロッパーツール - AWS アプリケーションを開発および管理するためのデベロッパーツール、SDK、IDE ツールキット、およびコマンドラインツールへのリンクです。
- ご利用開始のためのリソースセンター — AWS アカウント をセットアップする方法、AWS コミュニティに参加する方法、最初のアプリケーションを起動する方法を説明します。
- ハンズオンチュートリアル - ステップ バイ ステップのチュートリアルに従って、最初のアプリケーションを AWS で起動します。
- AWS ホワイトペーパー – アーキテクチャ、セキュリティ、エコノミクスなどのトピックについて、AWS のソリューションアーキテクトや他の技術エキスパートが記述した AWS の技術ホワイトペーパーの包括的なリストへのリンクです。
- AWS Support センター – AWS Support のケースを作成して管理するためのハブです。フォーラム、技術上のよくある質問、サービスヘルスステータス、AWS Trusted Advisor など、他の役立つリソースへのリンクも含まれています。

- AWS Support – AWS Support に関する情報のメインウェブページです。クラウド内でのアプリケーションの構築および実行を支援するために 1 対 1 での迅速な対応を行うサポートチャネルとして機能します。
- お問い合わせ - AWS の請求、アカウント、イベント、不正使用、その他の問題などに関するお問い合わせの受付窓口です。
- AWS サイトの利用規約 – 当社の著作権、商標、お客様のアカウント、ライセンス、サイトへのアクセス、その他のトピックに関する詳細情報。

ドキュメント履歴

次の表は、2018 年 7 月以降の Amazon EC2 Auto Scaling ドキュメントへの重要な追加項目を示しています。このドキュメントの更新に関する通知については、RSS フィードをサブスクライブできます。

変更	説明	日付
高解像度メトリクス	ターゲット追跡では、1 分よりも低い間隔で公開される秒レベルのデータポイントを持つ高解像度の CloudWatch メトリクスがサポートされるようになりました。詳細については、「高解像度メトリクスを使用してターゲット追跡ポリシーを作成し、応答を高速化する」を参照してください。	2024 年 11 月 22 日
Security IAM の更新	AutoScalingServiceRolePolicy 管理ポリシーは、Resource Groups に追加のアクセス許可を付与するようになりましたresource-groups:ListGroupResources 。	2024 年 11 月 20 日
パフォーマンス保護	Auto Scaling グループに属性ベースのインスタンスタイプの選択を使用する場合、パフォーマンス保護を有効にして、選択したインスタンスタイプが指定されたパフォーマンスベースラインと類似しているか、それを超えるようにできるようになりました。詳細については、「属性ベース	2024 年 11 月 20 日

	のインスタンスタイプの選択を使用して混合インスタンスグループを作成する」を参照してください。	
キャパシティ予約の設定	キャパシティ予約へのインスタンスの起動を優先できるようになりました。詳細については、「キャパシティーの予約」を参照してください。	2024 年 11 月 20 日
ゾーンシフト	ゾーンシフトを使用して、アベイラビリティーゾーンのアプリケーション障害から回復できるようになりました。詳細については、Auto Scaling グループのゾーンシフト」を参照してください。	2024 年 11 月 18 日
アベイラビリティーゾーンのディストリビューション	Auto Scaling グループのアベイラビリティーゾーンディストリビューションを選択できるようになりました。詳細については、Auto Scaling グループのアベイラビリティーゾーンのディストリビューション」を参照してください。	2024 年 11 月 7 日
Security IAM の更新	AutoScalingServiceRolePolicy 管理ポリシーは、Amazon EC2 (`ec2:GetSecurityGroupsForVpc` および) に追加のアクセス許可を付与するようになりましたec2:GetInstanceTypesFromInstanceRequirements 。	2024 年 2 月 29 日

ウォームプールの休止は、追加の でサポートされています AWS リージョン	Word(米国東部) と AWS GovCloud （ 米国 AWS GovCloud 西部) の 2 つの追加リージョンで、ウォームプール内のインスタンスを休止できるようになりました。ウォームプールの詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling のウォームプール」を参照してください。 EC2 Auto Scaling	2024 年 2 月 26 日
ウォームプールの休止は、追加の でサポートされています AWS リージョン	欧州 (チューリッヒ) と中東 (UAE) の 2 つの追加リージョンで、ウォームプール内のインスタンスを休止できるようになりました。ウォームプールの詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling のウォームプール」を参照してください。 EC2 Auto Scaling	2024 年 2 月 21 日

クロスアカウントパラメータの使用のサポート	Amazon EC2 Auto Scaling AWS アカウント で別の から共有された AWS Systems Manager パラメータを使用できるようになりました。詳細については、「Amazon AMI Auto Scaling ユーザーガイド」の「起動テンプレートで IDsWord の代わりに AWS Systems Manager パラメータを使用する」を参照してください。 EC2 Auto Scaling	2024 年 2 月 21 日
新しいスポット料金保護オプション	属性ベースのインスタンスタイプの選択を使用する場合、スポットインスタンス料金保護のしきい値をオンデマンド料金の割合 (%) として定義できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「料金保護」を参照してください。	2024 年 1 月 29 日

インスタンスメンテナンスポリシー	インスタンスメンテナンスポリシーを使用して、インスタンスの置き換えが発生する要因となるイベント中 (インスタンスの更新など) のインスタンスの起動を、既存のインスタンスが終了する前または後で定義できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「インスタンスメンテナンスポリシー」を参照してください。	2023 年 11 月 15 日
機械学習用のキャパシティブロック	起動テンプレートの作成時にキャパシティブロックの予約 ID を指定することで、キャパシティブロック内にインスタンスを起動できるようになりました。キャパシティブロックを使用すると、短期間の機械学習 (ML) ワークロードをサポートするために、将来の日付で GPU インスタンスを予約できます。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「機械学習ワークロードにキャパシティブロックを使用する」を参照してください。EC2 Auto Scaling	2023 年 10 月 31 日

新しいインスタンスの更新機能	インスタンスの更新を設定してステータスを失敗に設定し、指定した CloudWatch アラームが ALARM状態になったことを検出したときにオプションでロールバックできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「ロールバックによる変更の取り消し」を参照してください。	2023 年 7 月 31 日
ガイドの変更点	キャパシティ予約へのオンデマンドインスタンスの起動に関する新しいトピックがガイドに追加されました。詳細については、「Amazon EC2 Auto Scaling Word Auto Scaling ユーザーガイド」の「オンデマンドキャパシティ予約を使用して特定のアベイラビリティーゾーンのキャパシティを予約する」を参照してください。	2023 年 7 月 28 日

ガイドの変更点	起動設定から起動テンプレートへの AWS CloudFormation スタックの移行に関する新しいトピックがガイドに追加されました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイドAWS CloudFormation 」の「スタックを起動設定から起動テンプレートに移行する」を参照してください。	2023 年 4 月 18 日

新しい API オペレーションのサポート	このリリースでは、、AttachTrafficSources 、 DetachTrafficSources の 3 つの新しい API オペレーションが追加されましたDescribeTrafficSources 。また、新しいフィールドである TrafficSources が DescribeAutoScalingGroups オペレーションの結果に追加されました。新しいアクティビティステータスである WaitingForConnectionDraining が DescribeScalingActivities オペレーションの結果に追加されました。Amazon EC2 Auto Scaling は、、、および DescribeAutoScalingGroups オペレーションの CreateAutoScalingGroup HealthCheckType フィールドVPC_LATTICE に対してUpdateAutoScalingGroup 、新しい値 もサポートしています。詳細については、「Amazon EC2 Auto Scaling API リファレンス」を参照してください。	2023 年 3 月 31 日

Amazon VPC Lattice のサポート	これは、Amazon VPC Auto Scaling 用の EC2 Lattice の一般提供リリースです。詳細については、「Amazon VPC Auto Scaling ユーザーガイド」の「Word Lattice ターゲットグループを使用して Auto Scaling グループにトラフィックをルーティングする」を参照してください。 EC2 Auto Scaling	2023 年 3 月 31 日
ガイドの変更点	Elastic Load Balancing の使用 AWS CLI 例を含むセクションに、新しい例と更新された例が追加されました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」のAWS Command Line Interface 「 (AWS CLI) を使用した Elastic Load Balancing の使用例」を参照してください。	2023 年 3 月 31 日
追加での予測スケーリングのサポート AWS リージョン	中東 (UAE) および AWS GovCloud (米国東部) リージョンで予測スケーリングポリシーを作成できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の予測スケーリング」を参照してください。 EC2 Auto Scaling	2023 年 3 月 16 日

新しいインスタンスの更新機能	スタンバイ状態のインスタンスを終了または無視し、スケールインから保護されているインスタンスが交換可能になるのを待たず、置き換えまたは無視することができます。失敗したインスタンスの更新による変更をロールバックすることもできます。この更新の一環として、ドキュメントが拡張され、インスタンス更新のロールバック、インスタンス更新のキャンセル、インスタンス更新の設定可能なパラメータのデフォルト値の説明に関するトピックが含まれるようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの更新に基づく Auto Scaling インスタンスの置き換え」を参照してください。 EC2 Auto Scaling	2023 年 2 月 10 日
AMI ID の AWS Systems Manager パラメータの使用のサポート	起動テンプレートで AMI ID の代わりに Systems Manager パラメータを使用できるようになりました。詳細については、「Amazon AMI Auto Scaling ユーザーガイド」の「起動テンプレートで IDsWord の代わりに AWS Systems Manager パラメータを使用する」を参照してください。 EC2 Auto Scaling	2023 年 1 月 19 日

予測スケーリングの推奨事項	Amazon EC2 Auto Scaling コンソールから適切な予測スケーリングポリシーを評価して選択するための推奨事項を取得できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「予測スケーリングポリシーを評価する」を参照してください。	2023 年 1 月 18 日
予測スケーリングによる予測	予測スケーリングによる予測の生成は、更新頻度が毎日から 6 時間ごとに変更されました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の予測スケーリング」を参照してください。 EC2 Auto Scaling	2023 年 1 月 6 日

for CloudWatch メトリクス数学のサポート	ターゲット追跡スケーリングポリシーの作成時に Metric Math を使用できるようになりました。Metric Math を使用すると、multiple 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。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Metric Math を使用して Amazon Word Auto Scaling のターゲット追跡スケーリングポリシーを作成する」を参照してください。 EC2 Auto Scaling	2022 年 12 月 8 日
IAM サービスにリンクされたロールのアクセス許可の更新	このAutoScalingServiceRolePolicy ポリシーは、Amazon EC2 Auto Scaling に追加のアクセス許可を付与するようになりました。詳細については、AWS「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の マネージドポリシー」を参照してください。 EC2 Auto Scaling	2022 年 12 月 6 日

新しいスポット配分戦略	価格とキャパシティを最適化した配分戦略を使用して、中断される可能性が最も低く、価格も可能な限り低いスポットプールからスポットインスタンスをリクエストできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「配分戦略」を参照してください。	2022 年 11 月 10 日
アジアパシフィック (ジャカルタ) リージョンでの予測スケーリングのサポート	アジアパシフィック (ジャカルタ) リージョンで予測スケーリングポリシーを作成できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の予測スケーリング」を参照してください。 EC2 Auto Scaling	2022 年 10 月 13 日
コンソールでの予測スケーリングのためのカスタムメトリクスに対するサポート	Amazon EC2 Auto Scaling コンソールから予測スケーリングポリシーを作成するときに、カスタムメトリクスを使用できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の予測スケーリング」を参照してください。 EC2 Auto Scaling	2022 年 10 月 13 日

予測スケーリングメトリクスのCloudWatch モニタリング	CloudWatch を使用して予測スケーリングのモニタリングデータにアクセスできるようになりました。これにより、Metric Math を使用して、予測データの精度を表示する新しい時系列を作成できます。詳細については、「Amazon CloudWatch Auto Scaling ユーザーガイド」の「Word による予測スケーリングメトリクスのモニタリング」を参照してください。 EC2 Auto Scaling	2022 年 7 月 7 日
アジアパシフィック (大阪) リージョンでの予測スケーリングのサポート	アジアパシフィック (大阪) リージョンで予測スケーリングポリシーを作成できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の予測スケーリング」を参照してください。 EC2 Auto Scaling	2022 年 7 月 6 日

ウォームプール休止状態をサポートするリージョンの追加	アフリカ (ケープタウン)、アジアパシフィック (ジャカルタ)、アジアパシフィック (大阪)、欧州 (ミラノ) の 4 つのリージョンでウォームプール内のインスタンスを休止できるようになりました。ウォームプールの詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling のウォームプール」を参照してください。 EC2 Auto Scaling	2022 年 7 月 5 日
ヘルスチェックに対する更新	ヘルスチェックを実行するときに、Amazon EC2 Auto Scaling は、一時的な問題やヘルスチェックの設定ミスが原因で発生する可能性のあるダウンタイムを最小限に抑えるのに役立ちます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling がダウンタイムを最小限に抑える方法」を参照してください。 EC2 Auto Scaling	2022 年 5 月 21 日

インスタンスのデフォルトウォームアップ	インスタンスのデフォルトウォームアップを有効にすることによって、Auto Scaling グループのすべてのウォームアップ設定とクールダウン設定を統合し、継続的にスケールするスケーリングポリシーのパフォーマンスを最適化できるようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「Auto Scaling グループのデフォルトのインスタンスウォームアップを設定する」を参照してください。EC2 Auto Scaling	2022 年 4 月 19 日
ガイドの変更点	他の AWS サービスとの統合に関する新しい章がガイドに追加されました。詳細については、AWS 「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling と統合された のサービス」を参照してください。EC2 Auto Scaling	2022 年 3 月 29 日

IAM サービスにリンクされたロールのアクセス許可の更新	このAutoScalingServiceRolePolicy ポリシーは、Amazon EC2 Auto Scaling に追加の読み取りアクセス許可を付与するようになりました。詳細については、AWS 「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の マネージドポリシー」を参照してください。 EC2 Auto Scaling	2022 年 3 月 28 日
インスタンスメタデータがターゲットライフサイクルステータスを提供	インスタンスメタデータから Auto Scaling インスタンスのターゲットライフサイクル状態を取得できます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「インスタンスメタデータを使用してターゲットライフサイクル状態を取得する」を参照してください。	2022 年 3 月 24 日

新しいウォームプール機能をサポート	ウォームプール内のインスタンスを休止して、メモリコンテンツ (RAM) を削除せずにインスタンスを停止できるようになりました。また、後で必要になるインスタンスキャパシティを常に終了する代わりに、スケールイン時にインスタンスをウォームプールに戻すこともできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling のウォームプール」を参照してください。 EC2 Auto Scaling	2022 年 2 月 24 日
ガイドの変更点	Amazon EC2 Auto Scaling コンソールが更新され、スキップマッチングが有効で、必要な設定が指定されてインスタンスの更新を開始するのに役立つ追加オプションが追加されました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「インスタンスの更新を開始またはキャンセルする (コンソール)」を参照してください。	2022 年 2 月 3 日

予測スケーリングポリシーのカスタムメトリクス	予測スケーリングポリシーを作成する際に、カスタムメトリクスの使用を選択できるようになりました。メトリクスの数式を使用して、ポリシーに含めるメトリクスをさらにカスタマイズすることもできます。詳細については、「Advanced predictive scaling policy configurations using custom metrics」(カスタムメトリクスを使用した高度な予測スケーリングポリシーの設定) を参照してください。	2021 年 11 月 24 日
新しいオンデマンド配分戦略	混合インスタンスポリシーを使用する Auto Scaling グループを作成する際に、価格に基づくオンデマンドインスタンスの起動 (最低価格のインスタンスタイプが最初に起動) を選択できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「配分戦略」を参照してください。	2021 年 10 月 27 日

属性ベースのインスタンスタイプの選択	Amazon EC2 Auto Scaling では、属性ベースのインスタンスタイプの選択のサポートが追加されました。インスタンスタイプを手動で選択する代わりに、インスタンスの要件を vCPU、メモリ、ストレージなどの属性のセットとして表現できます。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「属性ベースのインスタンスタイプの選択を使用した Auto Scaling グループの作成」を参照してください。 EC2 Auto Scaling	2021 年 10 月 27 日
タグによるグループのフィルタリングのサポート	`describe-auto-scaling-groups` コマンドを使用して Auto Scaling グループに関する情報を取得する際に、タグフィルターを使用して Auto Scaling グループをフィルタリングできるようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「タグを使用して Auto Scaling グループをフィルタリングする」を参照してください。 EC2 Auto Scaling	2021 年 10 月 14 日

ガイドの変更点	Amazon EC2 Auto Scaling コンソールが更新され、カスタム終了ポリシーの作成が容易になりました AWS Lambda。コンソールのドキュメントも、それに応じて改訂されました。詳細については、「Using different termination policies (console)」(異なる終了ポリシーを使用する (コンソール)) を参照してください。	2021 年 10 月 14 日
起動設定を起動テンプレートへコピーするためのSupport	Amazon EC2 Auto Scaling コンソールから、 AWS リージョンのすべての起動設定を新しい起動テンプレートにコピーできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「起動設定を起動テンプレートにコピーする」を参照してください。	2021 年 8 月 9 日

インスタンスの更新機能を拡張	希望する設定を`start-instance-refresh` コマンドに追加することで、起動テンプレートの新しいバージョンなど、アップデートを含むことができるようになりました。スキップマッチングを有効にすることで、すでに希望の設定があるインスタンスの置き換えをスキップすることもできます。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの更新に基づく Auto Scaling インスタンスの置き換え」を参照してください。 EC2 Auto Scaling	2021 年 8 月 5 日
カスタム終了ポリシーの Support	でカスタム終了ポリシーを作成できるようになりました AWS Lambda。詳細については、「Lambda を使用したカスタム終了ポリシーの作成」を参照してください。それに伴い、終了 ポリシーを指定するためのドキュメントが更新されました。	2021 年 7 月 29 日

ガイドの変更点	Amazon EC2 Auto Scaling コンソールが更新され、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を作成するのに役立つ追加機能が追加されました。それに応じてスケジュールされたスケーリングのドキュメントが改訂されました。	2021 年 6 月 3 日
起動設定の gp3 ボリューム	起動設定のブロックデバイスマッピングで gp3 ボリュームを指定することができるようになりました。	2021 年 6 月 2 日
予測スケーリングのSupport	予測スケーリングを使用して、スケーリングポリシーを使用して Amazon EC2 Auto Scaling グループをプロアクティブにスケーリングできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling の予測スケーリング」を参照してください。EC2 Auto Scaling この更新により、AutoScalingService RolePolicy 管理ポリシーに API `cloudwatch:GetMetricData` アクションを呼び出すアクセス許可が含まれるようになりました。	2021 年 5 月 19 日

ガイドの変更点	GitHub からライフサイクルフックのサンプルテンプレートにアクセスできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling ライフサイクルフック」を参照してください。 EC2 Auto Scaling	2021 年 4 月 9 日
ウォームプールのSupport	Auto Scaling グループにウォームプールを追加することで、初回の起動時間の長いアプリケーションのパフォーマンス（コールドスタートを最小限にする）とコスト（インスタンスキャパシティーのオーバー-プロビジョニングを停止）のバランスをとることができるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling のウォームプール」を参照してください。 EC2 Auto Scaling	2021 年 4 月 8 日

チェックポイントのSupport	インスタンスの更新にチェックポイントを追加して、インスタンスを段階的に置き換え、特定のポイントでインスタンスの検証を実行できるようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの更新にチェックポイントを追加する」を参照してください。 EC2 Auto Scaling	2021 年 3 月 18 日
ガイドの変更点	Amazon EventBridge Auto Scaling イベントとライフサイクルフックで EC2 を使用するためのドキュメントが改善されました。詳細については、Amazon EC2 Auto Scaling ユーザーガイドの「Amazon Word Auto Scaling を EventBridge で使用する」および「チュートリアル: Lambda 関数を呼び出すライフサイクルフックを設定する」を参照してください。 EC2 Auto Scaling	2021 年 3 月 18 日

「ローカルタイムゾーンの Support」	`--time-zone` オプションをput-scheduled-update-group-actionコマンドに追加することで、ローカルタイムゾーンで定期的に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。タイムゾーンで夏時間 (DST) が観測された場合、定期的なアクションは夏時間に合わせて自動的に調整されます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「スケジュールされたスケーリング」を参照してください。	2021 年 3 月 9 日
混合インスタンスポリシーの機能を拡張	混合インスタンスポリシーを使用する場合に、スポットキャパシティのインスタンスタイプを優先できるようになりました。Amazon EC2 Auto Scaling はベストエフォートベースで優先順位を達成しようとしますが、最初に容量を最適化します。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ」を参照してください。 EC2 Auto Scaling	2021 年 3 月 8 日

削除されたグループのアクティビティをスケーリング	`--include-deleted-groups` オプションを describe-scaling-activities コマンドを追加することによって、削除された Auto Scaling グループのスケーリングアクティビティを表示できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling のトラブルシューティング」を参照してください。	2021 年 2 月 23 日
コンソールの改善	Amazon EC2 Auto Scaling コンソールから Application Load Balancer または Network Load Balancer を作成してアタッチできるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「新しい Application Load Balancer または Network Load Balancer (コンソール) を作成してアタッチする」を参照してください。	2020 年 11 月 24 日
「複数のネットワークインターフェイス」	複数のネットワークインターフェイスを指定する Auto Scaling グループの起動テンプレートを設定できるようになりました。詳細については、VPC のネットワークインターフェイス」を参照してください。	2020 年 11 月 23 日

複数の起動テンプレート	Auto Scaling グループで複数の起動テンプレートを使用できるようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスタイプに別の起動テンプレートを指定する」を参照してください。 EC2 Auto Scaling	2020 年 11 月 19 日
Gateway Load Balancer	Gateway Load Balancer を Auto Scaling グループにアタッチして、Amazon EC2 Auto Scaling によって起動されたアプライアンスインスタンスがロードバランサーから自動的に登録および登録解除されるようにするためのガイドを更新しました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の Elastic Load Balancing タイプ」および「Auto Scaling グループにロードバランサーをアタッチする」を参照してください。 Auto Scaling EC2 Auto Scaling	2020 年 11 月 10 日

最大インスタンス有効期間	インスタンスの最大有効期間を 1 日 (86400 秒) に減らすことができるようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの最大有効期間に基づく Auto Scaling インスタンスの置き換え」を参照してください。 EC2 Auto Scaling	2020 年 11 月 9 日
キャパシティーの再調整	Amazon EC2 が再調整に関するレコメンデーションを発行したときに、代替スポットインスタンスを起動するように Auto Scaling グループを設定できます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling 容量の再調整」を参照してください。 EC2 Auto Scaling	2020 年 11 月 4 日
「インスタンス メタデータ サービス バージョン 2」	起動設定を使用するときに、インスタンスメタデータのリクエストに、セッション志向な方法であるインスタンス メタデータサービス バージョン 2の使用を要求することができます。詳細については、「Amazon EC2 Auto Scaling Word Auto Scaling ユーザーガイド」の「インスタンスメタデータオプションの設定」を参照してください。	2020 年 7 月 28 日

ガイドの変更点	「Amazon Word Auto Scaling ユーザーガイド」の「スケールイン中に Auto Scaling インスタンスが終了する制御」、Auto Scaling インスタンスとグループのモニタリング」、「起動テンプレート」、「起動設定」セクションのさまざまな改善点と新しいコンソール手順。 EC2 Auto Scaling	2020 年 7 月 28 日
インスタンスの更新	設定を変更したときに、インスタンスの更新をスタートして、Auto Scaling グループ内のすべてのインスタンスを更新します。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの更新に基づく Auto Scaling インスタンスの置き換え」を参照してください。 EC2 Auto Scaling	2020 年 6 月 16 日

ガイドの変更点	「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの最大有効期間に基づく Auto Scaling インスタンスの置き換えAuto Scaling 」、「複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ」、「Amazon SQS に基づくスケーリング」、Auto Scaling グループとインスタンスのタグ付け」セクションのさまざまな改善点。 EC2 Auto Scaling	2020 年 5 月 6 日
ガイドの変更点	IAM ドキュメントに対するさまざまな改善。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「起動テンプレートのサポート」および「Amazon Word Auto Scaling アイデンティティベースのポリシーの例」を参照してください。 EC2 Auto Scaling EC2 Auto Scaling	2020 年 3 月 4 日

スケーリングポリシーの無効化	スケーリングポリシーを無効化して、再有効化することができるようになりました。この機能を使用すると、設定の詳細を保持しながら、スケーリングポリシーを一時的に無効化して、後でポリシーを再有効化することができます。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「Auto Scaling グループのスケーリングポリシーの無効化」を参照してください。 EC2 Auto Scaling	2020 年 2 月 18 日
通知機能の追加	Amazon EC2 Auto Scaling は、セキュリティグループまたは起動テンプレートがないために Auto Scaling グループ AWS Health Dashboard をスケールアウトできない場合に、 にイベントを送信するようになりました。詳細については、AWS Health Dashboard「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon EC2 Auto Scaling の通知」を参照してください。	2020 年 2 月 12 日

ガイドの変更点	「Amazon EC2 Auto Scaling と IAM の連携方法」、「Amazon EC2 Auto Scaling アイデンティティベースのポリシーの例」、「暗号化されたボリュームで使用するために必要な CMK キーポリシー」、「Amazon Word Auto Scaling ユーザーガイド」のAuto Scaling インスタンスとグループのモニタリング」セクションのさまざまな改善と修正。 EC2 Auto Scaling	2020 年 2 月 10 日
ガイドの変更点	インスタンスの重み付けを使用する Auto Scaling グループのドキュメントが改善されました。「キャパシティーユニット」を使用して希望するキャパシティーを測定するときに、スケーリングポリシーを使用する方法について説明します。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「スケーリングポリシーの仕組み」および「スケーリング調整タイプ」を参照してください。 EC2 Auto Scaling	2020 年 2 月 6 日

新しい「セキュリティ」章	Amazon EC2 Auto Scaling ユーザーガイドの新しいセキュリティの章は、Amazon EC2 Auto Scaling を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。この更新の一環として、「Amazon EC2 Auto Scaling リソースへのアクセスの制御」というユーザーガイドの章が、Amazon EC2 Auto Scaling の Identity and Access Management という新しい便利なセクションに置き換えられました。	2020 年 2 月 4 日
インスタンスタイプに関する推奨事項	AWS Compute Optimizer は、パフォーマンスの向上、コストの削減、またはその両方に役立つ Amazon EC2 インスタンスのレコメンデーションを提供します。詳細については、「Amazon EC2 Auto Scaling Word Auto Scaling ユーザーガイド」の「インスタンスタイプのレコメンデーションの取得」を参照してください。	2019 年 12 月 3 日

Dedicated Hosts およびホストリソースグループ	ホストリソースグループを指定する起動テンプレートの作成方法を示すガイドを更新しました。これにより、Dedicated Hosts AMIで使用するBYOLを指定する起動テンプレートを使用して Auto Scaling グループを作成できます。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「Auto Scaling グループの起動テンプレートの作成」を参照してください。 EC2 Auto Scaling	2019 年 12 月 3 日
Amazon VPC エンドポイントのサポート	VPC と Amazon EC2 Auto Scaling の間にプライベート接続を確立できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling とインターフェイス Word VPCエンドポイント」を参照してください。 EC2 Auto Scaling	2019 年 11 月 22 日

最大インスタンス有効期間	インスタンスが稼働できる最大時間を指定することにより、インスタンスを自動的に置き換えることができるようになりました。いずれかのインスタンスがこの制限に近づいている場合、Amazon EC2 Auto Scaling はそれらを徐々に置き換えます。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「インスタンスの最大有効期間に基づく Auto Scaling インスタンスの置き換え」を参照してください。 EC2 Auto Scaling	2019 年 11 月 19 日
インスタンスの重み付け	複数のインスタンスタイプを含む Auto Scaling グループの場合、オプションで、各インスタンス タイプからグループのキャパシティに割り当てるキャパシティの単位数を定義できるようになりました。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」の「Amazon Word Auto Scaling のインスタンスの重み付け」を参照してください。 EC2 Auto Scaling	2019 年 11 月 19 日

インスタンスタイプの最小数	スポット、オンデマンド、リザーブドインスタンスのグループに、追加のインスタンスタイプを指定する必要がなくなりました。すべての Auto Scaling グループで、インスタンスタイプの最小値が 1 つになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ」を参照してください。 EC2 Auto Scaling	2019 年 9 月 16 日
新しいスポット配分戦略のサポート	Amazon EC2 Auto Scaling は、利用可能なスポット容量に基づいて最適に選択されたスポットインスタンスプールを使用してリクエストを満たす新しいスポット配分戦略「容量最適化」をサポートするようになりました。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ」を参照してください。 EC2 Auto Scaling	2019 年 8 月 12 日

ガイドの変更点	サービスにリンクされたロールと、暗号化されたボリュームで使用するために必要なCMK キーポリシーの Amazon EC2 Auto Scaling ドキュメントが改善されました。	2019 年 8 月 1 日
タグ付け強化のサポート	Amazon EC2 Auto Scaling は、インスタンスを起動するのと同じ EC2 呼び出しの一部として Amazon API インスタンスにタグを追加するようになりました。詳細については、「Auto Scaling Group とインスタンスのタグ付け」を参照してください。	2019 年 7 月 26 日
ガイドの変更点	スケーリングプロセスの一時停止と再開トピックの Amazon EC2 Auto Scaling ドキュメントが改善されました。 https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-suspend-resume-processes.htmlカスタマー管理ポリシーの例を更新し、ユーザーが特定のカスタムサフィックスサービスにリンクされたロールのみを Amazon EC2 Auto Scaling に渡すことを許可するポリシーの例を追加しました。	2019 年 6 月 13 日

新しい Amazon EBS 機能のサポート	起動テンプレートトピックに新しい Amazon EBS 機能のサポートが追加されました。スナップショットからの復元中に EBS ボリュームの暗号化状態を変更します。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「Auto Scaling グループの起動テンプレートの作成」を参照してください。 EC2 Auto Scaling	2019 年 5 月 13 日
ガイドの変更点	次のセクションの Amazon EC2 Auto Scaling ドキュメントが改善されました。スケールイン中に終了する Auto Scaling インスタンスの制御、Auto Scaling グループ、複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ、Amazon EC2 Auto Scaling の動的スケーリング。	2019 年 3 月 12 日

インスタンスタイプと購入オプションの組み合わせのサポート	1 つの Auto Scaling グループ内で、購入オプション (スポット、オンデマンド、リザーブドインスタンス) およびインスタンスタイプ間でインスタンスをプロビジョンおよび自動的にスケーリングします。詳細については、「Amazon Word Auto Scaling ユーザーガイド」の「複数のインスタンスタイプと購入オプションを持つ Auto Scaling グループ」を参照してください。 EC2 Auto Scaling	2018 年 11 月 13 日
Amazon SQS に基づくスケーリングのトピックを更新しました	Amazon SQS キューからの需要の変化に応じて、カスタムメトリクスを使用して Auto Scaling グループをスケーリングする方法を説明するためにガイドを更新しました。詳細については、「Amazon SQS Auto Scaling ユーザーガイド」の「Amazon Word に基づくAuto ScalingEC2」を参照してください。	2018 年 7 月 26 日

次の表に、2018 年 7 月以前の Amazon EC2 Auto Scaling ドキュメントの重要な変更点を示します。

機能	説明	リリース日
ターゲット追跡スケーリングポリシーのサポート	わずかなステップで、アプリケーションの動的スケーリングを設定します。詳細については、「Amazon EC2 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

機能	説明	リリース日
リソースレベルのアクセス許可のサポート	IAM ポリシーを作成して、リソースレベルでアクセスを制御します。詳細については、「Amazon EC2 Auto Scaling リソースへのアクセスの制御」を参照してください。	2017 年 5 月 15 日
改善のモニタリング	Auto Scaling グループ メトリクスでは、詳細モニタリングを有効にする必要はなくなりました。コンソールの [Monitoring] タブから、グループメトリクスの収集とメトリクスグラフの表示を有効にできるようになりました。詳細については、「Amazon CloudWatch を使用した Auto Scaling グループとインスタンスのモニタリング」を参照してください。	2016 年 8 月 18 日
Application Load Balancers の Support	新しいまたは既存の Auto Scaling グループに 1 つまたは複数のターゲットグループをアタッチします。詳細については、「Auto Scaling グループにロードバランサーをアタッチする」を参照してください。	2016 年 8 月 11 日
ライフサイクルフックのイベント	Amazon EC2 Auto Scaling は、ライフサイクルフックを呼び出すときにイベントを EventBridge に送信します。詳細については、Auto Scaling グループのスケーリング時に EventBridge を取得する」を参照してください。	2016 年 2 月 24 日
インスタンスの保護	Amazon EC2 Auto Scaling がスケールイン時に終了する特定のインスタンスを選択できないようにします。詳細については、「インスタンスの保護」を参照してください。	2015 年 12 月 7 日
ステップスケーリングポリシー	超過アラームのサイズに基づいてスケールできるようになるスケーリングポリシーを作成します。詳細については、「スケーリング ポリシータイプ」を参照してください。	2015 年 7 月 6 日

機能	説明	リリース日
ロードバランサーの更新	既存の Auto Scaling グループへロードバランサーをアタッチ、またはAuto Scaling グループへロードバランサーをデタッチします。詳細については、「Auto Scaling グループにロードバランサーをアタッチする」を参照してください。	2015 年 6 月 11 日
for ClassicLink のサポート	Auto Scaling グループ内の EC2-Classic インスタンスを VPC にリンクし、プライベート IP アドレスを使用して、これらのリンクされた EC2-Classic インスタンスと VPC 内のインスタンス間の通信を有効にします。詳細については、EC2-Classic インスタンスを VPC にリンクする」を参照してください。	2015 年 1 月 19 日
ライフサイクルフック	新しく起動されたインスタンスまたは終了中のインスタンスに対してアクションを実行する間、これらのインスタンスを保留状態に維持できるようになりました。詳細については、「Amazon EC2 Auto Scaling ライフサイクルフック」を参照してください。	2014 年 7 月 30 日
インスタンスのデタッチ	Auto Scaling グループからインスタンスをデタッチします。詳細については、Auto Scaling グループから EC2 インスタンスをデタッチする」を参照してください。	2014 年 7 月 30 日
Standby 状態へのインスタンスの移行	`InService` 状態にあるインスタンスを `Standby` 状態に移行できるようになりました。詳細については、「Auto Scaling グループからのインスタンスの一時的な削除」を参照してください。	2014 年 7 月 30 日
タグの管理	AWS Management Consoleを使って、Auto Scaling グループを管理する 詳細については、「Auto Scaling Group とインスタンスのタグ付け」を参照してください。	2014 年 5 月 01 日

機能	説明	リリース日
ハードウェア専有インスタンスのサポート	起動設定を作成するときにプレイスメントテナンシー属性を指定して、ハードウェア専有インスタンスを起動できるようになりました。詳細については、「インスタンスのプレイスメント テナンシー」を参照してください。	2014 年 4 月 23 日
EC2 インスタンスからグループまたは起動設定を作成する	EC2 インスタンスを使用して Auto Scaling グループまたは起動設定を作成します。EC2 インスタンスを使用した起動設定の作成については、EC2 インスタンスを使用した起動設定の作成」を参照してください。EC2 インスタンスを使用して Auto Scaling グループを作成する方法については、EC2 インスタンスを使用して Auto Scaling グループを作成する」を参照してください。	2014 年 1 月 2 日
インスタンスのアタッチ	インスタンスを既存の Auto Scaling グループにアタッチして、EC2 インスタンスの自動スケーリングを有効にします。詳細については、Auto Scaling グループに EC2 インスタンスをアタッチする」を参照してください。	2014 年 1 月 2 日
アカウントの制限の表示	アカウントの Auto Scaling リソースの制限を表示します。詳細については、「Auto Scaling の制限」を参照してください。	2014 年 1 月 2 日
Amazon EC2 Auto Scaling のコンソールサポート	を使用して Amazon EC2 Auto Scaling にアクセスします AWS Management Console。詳細については、「Amazon EC2 Auto Scaling の開始方法」を参照してください。	2013 年 12 月 10 日
パブリック IP アドレスの割り当て	VPC で起動されたインスタンスにパブリック IP アドレスを割り当てます。詳細については、Auto Scaling インスタンスを VPC で起動する」を参照してください。	2013 年 9 月 19 日
インスタンス終了ポリシー	EC2 インスタンスを終了するときに使用する Amazon EC2 Auto Scaling のインスタンス終了ポリシーを指定します。詳細については、「スケールイン時にどの Auto Scaling インスタンスを終了するかを制御」を参照してください。	2012 年 9 月 17 日

機能	説明	リリース日
IAM ロールのサポート	EC2 インスタンスプロファイルを使用して IAM インスタンスを起動します。この機能を使用してインスタンスに IAM ロールを割り当て、アプリケーションが他の Amazon Web Services に安全にアクセスできるようにします。詳細については、IAM ロールを使用して Auto Scaling インスタンスを起動する」を参照してください。	2012 年 6 月 11 日
スポットインスタンスのサポート	起動設定でスポットインスタンスを起動します。詳細については、「Requesting Spot Instances for fault-tolerant and flexible applications」(柔軟で耐障害性を備えたアプリケーションのスポットインスタンスのリクエスト) を参照してください。	2012 年 6 月 7 日
グループとインスタンスへのタグ付け	Auto Scaling グループにタグを付け、タグの作成後に起動された EC2 インスタンスにもタグが適用されるように指定します。詳細については、「Auto Scaling Group とインスタンスのタグ付け」を参照してください。	2012 年 1 月 26 日

機能	説明	リリース日
Amazon SNS のサポート	Amazon SNS Auto Scaling が EC2 インスタンスを起動または終了するたびに通知を受け取るには、Amazon EC2 を使用します。詳細については、Auto Scaling グループのスケーリング時に SNS 通知を受け取る」を参照してください。 Amazon EC2 Auto Scaling では、次の新機能も追加されました。 • 構文を使用して反復的なスケーリングアクティビティを設定する機能。詳細については、API PutScheduledUpdateGroupAction オペレーション」を参照してください。 • 起動したインスタンスをロードバランサー (LoadBalancer) に追加せずにスケールアウトできる新しい設定。詳細については、ProcessType API データ型」を参照してください。 • インスタンスが最初に終了されるのを待たずに、関連付けられたインスタンスを持つ Auto Scaling グループを削除するように Amazon EC2 Auto Scaling に指示する DeleteAutoScalingGroup オペレーションのForceDelete フラグ。 Auto Scaling 詳細については、API DeleteAutoScalingGroup オペレーション」を参照してください。	2011 年 7 月 20 日
スケジュールされたスケーリングアクション	スケジュールに基づくスケーリングアクションのために追加されたサポート。詳細については、「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。	2010 年 12 月 2 日
Amazon VPC のサポート	Amazon VPC のサポートが追加されました。詳細については、Auto Scaling インスタンスを VPC で起動する」を参照してください。	2010 年 12 月 2 日

機能	説明	リリース日
HPC クラスターのサポート	ハイパフォーマンスコンピューティング (HPC) クラスターのサポートが追加されました。	2010 年 12 月 2 日
ヘルスチェックのサポート	Amazon EC2 Auto Scaling が管理する EC2 インスタンスで Elastic Load Balancing ヘルスチェックを使用するためのサポートが追加されました。詳細については、「Auto Scaling グループ内のインスタンスのヘルスチェック」を参照してください。	2010 年 12 月 2 日
for CloudWatch アラームのサポート	古いトリガーメカニズムを削除し、 CloudWatch アラーム機能を使用するように Amazon EC2 Auto Scaling を再設計しました。詳細については、「Amazon EC2 Auto Scaling の動的スケーリング」を参照してください。	2010 年 12 月 2 日
スケーリングの一時停止と再開	スケーリングプロセスを停止および再開するために追加されたサポート。	2010 年 12 月 2 日
IAM のサポート	IAM のサポートが追加されました。詳細については、「Amazon EC2 Auto Scaling リソースへのアクセスの制御」を参照してください。	2010 年 12 月 2 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。