aws

実装ガイド

AWS での Instance Scheduler

AWS での Instance Scheduler: 実装ガイド

Table of Contents

AWS インスタンスの開始と停止を自動化する

AWS での Instance Scheduler ソリューションは、Amazon Elastic Compute Cloud (Amazon EC2) および Amazon Relational Database Service (Amazon RDS) インスタンスなど、さまざまな AWS のサービスの起動と停止を自動化します。

このソリューションは、使用されていないリソースを停止し、キャパシティーが必要なときにリソースを起動することで、運用コストを削減するのに役立ちます。例えば、企業は AWS での Instance Scheduler を使用して、毎日営業時間外にインスタンスを自動的に停止できます。すべてのインスタンスをフル活用している場合は、このソリューションで通常の営業時間中にのみ必要なインスタンスに対して最大 70% のコスト削減を実現できます (毎週の使用率を 168 時間から 50 時間に削減)。

AWS での Instance Scheduler は、Amazon Web Services (AWS) リソースタグと AWS Lambda を使用して、独自に定義されたスケジュールに従って、複数の AWS リージョンとアカウントのインスタンスを自動的に停止および再起動します。このソリューションでは、停止した Amazon EC2 インスタンスにハイバネーション (休止) を使用することもできます。

この実装ガイドでは、AWS での Instance Scheduler ソリューションの概要、そのリファレンスアーキテクチャとコンポーネント、デプロイを計画する際の考慮事項、AWS クラウドにソリューションをデプロイするための設定手順について説明します。

このガイドは、環境に AWS での Instance Scheduler を実装したい IT インフラストラクチャアーキテクト、管理者、DevOps プロフェッショナルを対象としています。

このナビゲーションテーブルを使用すると、次の質問に対する回答をすばやく見つけることができます。

質問内容	参照先
このソリューションの実行に必要なコストを確認する。米国東部 (バージニア北部) リージョンでこのソリューションを実行するための推定コストは、13.15 USD / 月です。	コスト
このソリューションのセキュリティ上の考慮事項を理解する。	AWS Well-Architected の設計に関する考慮事項、セキュリティ

質問内容	参照先
スケジュールを設定する。	スケジューラ設定テーブル
どの AWS リージョンでこのソリューションをサポートしているか知りたい場合。	サポートしている AWS リージョン
このソリューションに含まれている AWS CloudFormation テンプレートを表示またはダウンロードして、このソリューションのインフラストラクチャリソース (スタック) を自動的にデプロイする。	AWS CloudFormation テンプレート
ソースコードにアクセスし、オプションで AWS Cloud Development Kit (AWS CDK) を使用してソリューションをデプロイする。	GitHub リポジトリ

機能とメリット

AWS での Instance Schedule ソリューションには、次の機能があります。

クロスアカウントインスタンスのスケジューリング

このソリューションには、セカンダリアカウントでインスタンスを起動および停止するために必要な AWS Identity and Access Management (IAM) ロールを作成するテンプレートが含まれています。詳細については、「クロスアカウントインスタンスのスケジューリング」セクションを参照してください。

自動化されたタグ付け

AWS での Instance Scheduler では、起動または停止するすべてのインスタンスにタグを自動的に追加できます。ソリューションには、タグに変数情報を追加できるマクロも含まれています。

Scheduler CLI を使用したスケジュールまたは期間の設定

このソリューションには、スケジュールと期間を設定するためのコマンドを提供するコマンドラインインターフェイス (CLI) が含まれています。CLI を使用すると、特定のスケジュールで削減できるコストを見積もることができます。詳細については、「Scheduler CLI」セクションを参照してください。

Infrastructure as Code (IaC) を使用したスケジュールの管理

このソリューションでは、Infrastructure as Code (IaC) を使用してスケジュールを管理するために使用できる AWS CloudFormation カスタムリソースを提供します。詳細については、「Infrastructure as Code (IaC) を使用してスケジュールを管理する」セクションを参照してください。

Systems Manager メンテナンスウィンドウとの統合

Amazon EC2 インスタンスの場合、AWS での Instance Scheduler は、それらのインスタンスと同じリージョンで定義されている AWS Systems Manager メンテナンスウィンドウと統合して、メンテナンスウィンドウに従ってインスタンスを起動および停止することができます。

ユースケース

業務時間中にのみインスタンスを実行する

すべてのインスタンスをフル活用している場合は、このソリューションで通常の営業時間中にのみ必要なインスタンスに対して最大 76% のコスト削減を実現できます (毎週の使用率は 168 時間から 40 時間に削減)。詳細については、「サンプルスケジュール」を参照してください。

業務時間後にインスタンスを停止する

開発インスタンスが業務時間終了後から再び必要になるまでオフになっていることを確認したい場合は、このソリューションを使用して開始期間なしで終了期間を設定できます。詳細については、「サンプルスケジュール」を参照してください。

概念と定義

このセクションでは、主要な概念について説明し、このソリューション固有の用語を定義します。

スケジュール

インスタンスがバインドされる 1 つ以上の期間のグループ。

期間

開始時間と停止時間によって定義される実行期間。

インスタンス

スケジュール可能なサポート対象リソース。例えば、Amazon EC2 インスタンスまたは Amazon RDS クラスターの Amazon EC2 と Amazon RDS など。

通常の業務時間

平日の 9:00 から 17:00 (午前 9 時 ~ 午後 5 時) 東部標準時

AWS 用語の一般的なリファレンスについては、「AWS 用語集」を参照してください。

Cost

AWS での Instance Scheduler を実行する際に使用される AWS サービスのコストは、お客様の負担となります。デプロイサイズに合わせてコストがどのように増加するかを理解することは、実装の計画と最適化に役立ちます。

コストの増加要因

Instance Scheduler のコストは、いくつかの要因に基づいて増加します。

スケジュールターゲットの数: 管理されている一意の account-region-service の組み合わせの数。各ターゲットには、スケジューリング間隔ごとに個別の Lambda 呼び出しが必要です。

ターゲットあたりのリソース: 各ターゲット内のリソース (EC2 インスタンス、RDS データベースなど) の数は、Lambda の実行時間と期間コストに影響します。

運用メトリクスの複雑さ: オプションの CloudWatch メトリクスのコストは、デプロイ全体で追跡される一意のインスタンスタイプとアクティブなスケジュールの数に応じて増加します。

スケジュールの頻度: ソリューションは、設定した頻度 (デフォルトは 5 分) に基づいて実行されます。チェックの頻度が高いほど、Lambda 呼び出しは 1 日 24 回 (1 時間間隔) から 1 日 288 回 (5 分間隔) に増加します。

スケジューリングターゲットの計算

スケジューリングターゲットは、少なくとも 1 つのアクティブマネージドインスタンスを含む account-region-service の一意の組み合わせです。同じ account-region-service の組み合わせ内の複数のインスタンスは、単一のスケジューリングターゲットとしてカウントされます。

計算例

- アカウント A、us-east-1、5 EC2 インスタンス = 1 スケジューリングターゲット
- アカウント A、us-east-1、3 RDS データベース = 1 スケジューリングターゲット
- アカウント A、us-east-1、2 Auto Scaling グループ = 1 スケジューリングターゲット

- アカウント A、us-west-2、2 EC2 インスタンス = 1 スケジューリングターゲット
- アカウント B、us-east-1、10 EC2 インスタンス = 1 スケジューリングターゲット

合計: 5 つのスケジューリングターゲット

つまり、このソリューションはスケジュール間隔ごとに 5 つの個別の Lambda 関数を呼び出し、これらの account-region-service の組み合わせ全体ですべてのリソースを管理します。

> **Note**
>
> ターゲットはスケジューリングの対象とすることができますが、少なくとも 1 つのリソースがそのターゲットでスケジューリング用にタグ付けされるまで、コスト計算では「アクティブ」とは見なされません。
> コストを最適化するために、Instance Scheduler はすべての Amazon RDS 関連サービスを 1 回の呼び出しにグループ化します。したがって、Amazon RDS、Amazon Aurora、Amazon Neptune、Amazon DocDB スケジューリングはすべてコスト計算のために 1 つの「RDS」サービスとしてカウントされます。

コスト最適化戦略

1. Lambda の料金が低いリージョンにデプロイする
2. 単一のターゲットスケールの制限によって増やす必要がある場合を除き、デフォルトの 512 MB Lambda メモリ設定を使用する
3. アクティブな使用における一意のスケジュールとインスタンスタイプの数を最小限に抑える
4. 要件に基づいてスケジューリング頻度を調整する
5. 使用する予定がない場合は、運用メトリクスダッシュボードを無効にする

このソリューションの各 AWS サービスの料金ウェブページを参照してください。

AWS Cost Explorer を使用して予算を作成することをお勧めします。これはコスト管理に役立ちます。料金は変更されることがあります。

リファレンス料金の例 (毎月)

次の例は、さまざまなデプロイサイズでコストがどのように変化するかを示しています。これらをリファレンスポイントとして使用して、特定のデプロイのコストを見積もります。

> **Note**
>
> すべての参照料金は、ソリューションで使用されるプライマリサービスのコストの概算です。

小規模デプロイ (~ 月額約 9 USD)

以下は一般的な開発または小規模な本番稼働用デプロイの例です。

- アクティブなターゲット 5
- マネージドリソース 20
- アクティブなスケジュール 3
- インスタンスタイプ 2
- スケジューリングの間隔 5 分
- Lambda 関数 512 MB、平均ランタイム 5 秒

AWS のサービス	月額コスト [USD]
AWS Lambda	~ 2.00 USD
AWS KMS	~ 1.50 USD
CloudWatch Logs	~ 0.30 USD
CloudWatch メトリクス	~ 5.30 USD
Amazon DynamoDB	~ 0.05 USD
合計:	~ 9.15 USD

中規模デプロイ (~ 月額 161 USD)

以下は中規模のエンタープライズデプロイの例です。

- アクティブなターゲット 250

- マネージドリソース 1000
- アクティブなスケジュール 15
- インスタンスタイプ 15
- スケジューリングの間隔 5 分
- Lambda 関数 512 MB、平均ランタイム 5 秒
- EC2 メンテナンスウィンドウ 5

AWS のサービス	月額コスト [USD]
AWS Lambda	~ 95.00 USD
Amazon DynamoDB	~ 1.00 USD
CloudWatch Logs	~ 10.00 USD
CloudWatch メトリクス	~ 40.00 USD
AWS KMS	~ 15.00 USD
合計:	~ 161.00 USD

大規模なデプロイ (~ 月額 630 USD)

以下は大規模なエンタープライズデプロイの例です。

- アクティブなターゲット 1000
- マネージドリソース 5000
- アクティブなスケジュール 500
- インスタンスタイプ 50
- スケジューリングの間隔 5 分
- Lambda 関数 512 MB、平均ランタイム 5 秒
- EC2 メンテナンスウィンドウ 100

AWS のサービス	月額コスト [USD]
AWS Lambda	~ 380.00 USD
Amazon DynamoDB	~ 5.00 USD
CloudWatch Logs	~ 50.00 USD
CloudWatch メトリクス	~ 140.00 USD
AWS KMS	~ 55.00 USD
合計:	~ 630.00 USD

デプロイのコスト見積もり

特定のデプロイのコストを見積もるには

1. マネージドリソース (EC2 インスタンス、RDS データベースなど) の合計をカウントする
2. 管理するアカウントとリージョンの数を決定する
3. 必要なスケジューリング頻度を考慮する
4. 運用メトリクスが必要かどうかを判断する
5. 上記の参考例を使用して、予想されるコストを補間します。

クォータ

スケーリングの制限事項

Instance Scheduler は、2 つのプライマリ軸でスケールして、大規模なエンタープライズデプロイを管理します。

垂直スケーリング (ターゲットあたりのリソース)

垂直スケーリングは、単一のスケジューリングリクエスト Lambda 関数が単一のスケジューリングターゲット (account/region/service の組み合わせ) 内で効率的に処理できるリソースの数によって制限されます。

Instance Scheduler は、1 つのスケジューリングターゲットで 1,000 の EC2、100 の ASG、および 100 の RDS DBS/クラスターを処理できるように設計されていますが、クロスリージョンレイテンシーによって制限される場合があります。

最適なパフォーマンスを確保するために、スケジューリングリクエスト Lambda の実行時間をモニタリングすることをお勧めします (「運用上のインサイトダッシュボード」を参照してください)。平均ランタイムは 90 秒未満にし、最大ピーク時間は 4 分以下にすることをお勧めします。

水平スケーリング (ターゲット数)

水平スケーリングは、管理されているアクティブなスケジューリングターゲットの数によって制限されます。アクティブなターゲットは、アクティブにタグ付けされたリソースが少なくとも 1 つある account/region/service の組み合わせです。Instance Scheduler は、より多くのアカウントやリージョンにデプロイできますが、アクティブにタグ付けされたリソースを持つターゲットのみがパフォーマンスに影響します。

デフォルトの Lambda 同時実行クォータが 1000 の場合、アクティブなターゲットを 1000 個同時に実行できます。Lambda は追加の実行を自動的にキューに入れ、この制限を超えてスケールできるようにします。最適なパフォーマンスを得るには、累積遅延を 3 分未満にしておくことをお勧めします。

例えば、15 秒の平均ランタイムと Lambda 同時実行数の制限が 1000 の場合、累積遅延を 3 分 に保ちながら、最大 12,000 のアクティブなターゲットを管理できます (3 分 ÷ 15 秒 × 1000 = 12,000 ターゲット)。

大規模なデプロイでは、AWS サポートに Lambda 同時実行クォータの引き上げをリクエストできます。

追加の考慮事項

AWS リソースタグ: AWS リソースには通常、リソースあたり 50 個のタグの制限があります。Instance Scheduler では、ソリューションを操作するために 6 つの情報タグとコントロールタグが必要です。リソースに、Instance Scheduler タグと既存のタグ付け戦略の両方に対応するのに十分なタグ容量があることを確認します。

Lambda 実行制限: 各スケジュールリクエストハンドラー Lambda 関数の実行タイムアウトは 5 分です。

DynamoDB スケーリング: このソリューションは Amazon DynamoDB テーブルにオンデマンドスケーリングを使用し、ワークロードに基づいて容量を自動的に調整します。

API レート制限: AWS サービス API スロットリングは、非常に大規模なデプロイで発生する可能性があります。このソリューションには、一時的なスロットリングを処理する再試行ロジックが含まれていますが、過剰なスロットリングはソリューションの上限スケーリング制限を減らす可能性があります。

AWS のサービスクォータ

AWS サービスのサービスクォータ

サービスクォータ (制限とも呼ばれます) は、AWS アカウントのサービスリソースまたはオペレーションの最大数です。このソリューションに実装されている各サービスに十分なクォータがあることを確認してください。詳細については、「AWS のサービスクォータ」を参照してください。

AWS CloudFormation のクォータ

ご使用の AWS アカウントには AWS CloudFormation のクォータがあり、このソリューションでスタックを起動する際に注意する必要があります。これらのクォータを理解することで、このソリューションを正常にデプロイできなくなるような制限エラーを回避できます。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation のクォータ」を参照してください。

AWS Lambda のクォータ

ご使用のアカウントでは、デフォルトの AWS Lambda 同時実行クォータは 1,000 になります。大規模なデプロイでは、Lambda 同時実行のために他のワークロードと競合しないように、Instance Scheduler を専用アカウントにデプロイすることをお勧めします。この値は調整可能です。詳細については、「AWS Lambda 入門ガイド」を参照してください。

サポートしている AWS リージョン

AWS GovCloud (米国) リージョンや一部のオプトインリージョン (デフォルトで無効になっているリージョン) など、任意の AWS リージョン に Instance Scheduler をデプロイできます。ソリューションをデプロイしたら、アカウントの任意のリージョンでタグ付けされた EC2 と RDS DB のインスタンスに適切な起動または停止のアクションを適用するようにソリューションを設定できます。クロスアカウントのインスタンススケジューリングを使用すると、ソリューションはすべてのアカウントのすべての設定済みリージョンのインスタンスにアクションを適用します。

> **Important**
>
> Lambda 関数が単一のリージョンで実行されている場合でも、AWS での Instance Scheduler のアクションは、アカウントのすべての AWS リージョンで適切にタグ付けされたインスタンスに影響します。

ソリューションを複数デプロイして使用すると、多数のインスタンス、または多くのアカウントとリージョンのインスタンスをスケジュールできます。複数のスケジューラをデプロイする場合は、スタックごとに異なるタグ名を使用し、デプロイごとに重複しないリージョンのセットを設定します。

各デプロイでは、スケジュールすべきリソースを識別するタグキーについて、アカウント内のすべての設定済みリージョンですべてのインスタンスをチェックします。複数のデプロイでリージョンが重複している場合は、各インスタンスは複数のデプロイによって確認されます。

> **Note**
>
> AWS での Instance Scheduler は、ソリューションスタックが標準の AWS リージョンにデプロイされている場合でも、任意のオプトインリージョン内のインスタンスをターゲットにしてスケジューリングできます。

アカウント ID または AWS Organization ID を使用したクロスアカウントのインスタンススケジューリング

このソリューションには、AWS Identity and Access Management (IAM) ロールとその他の必要なリソースを作成して、ソリューションがセカンダリアカウントでスケジューリングを起動できるようにするテンプレート (instance-scheduler-on-aws-remote.template) が含まれています。スタックを起動する前に、リモートテンプレートのアクセス許可を確認および変更できます。

アカウント ID を使用したクロスアカウントスケジューリングを有効にする

自動化された開始/停止スケジュールをセカンダリアカウントのリソースに適用するには、以下の手順を行います。

1. AWS マネジメントコンソールにサインインし、ボタンを選択して、プライマリアカウントに instance-scheduler-on-aws AWS CloudFormation テンプレートを起動します。

2. 次に、該当する各セカンダリアカウントでリモートテンプレート (instance-scheduler-remote) を起動します。各リモートスタックが起動されると、クロスアカウントロールの Amazon リソースネーム (ARN) が作成されます。
3. プライマリのソリューションスタックを Provide Organization Id OR List of Remote Account Ids パラメーターのアカウント ID で更新して、ソリューションがセカンダリアカウントのインスタンスで起動と停止のアクションを実行できるようにします。

AWS Organization ID を使用したクロスアカウントスケジューリングの有効化

自動化された開始/停止スケジュールをセカンダリアカウントのリソースに適用するには、以下の手順を行います。

1. AWS マネジメントコンソールにサインインし、ボタンを選択して、プライマリアカウントに instance-scheduler-on-aws AWS CloudFormation テンプレートを起動します。
2. CloudFormation パラメータの Using AWS Organizations? を Yes に設定し、Provide Organization ID または List of Remote Account IDs の CloudFormation パラメータに組織 ID を入力します。
3. プライマリアカウントにスタックをデプロイした後、プライマリアカウントのソリューションと同じリージョンの該当する各セカンダリアカウントで、リモートテンプレート (`instance-scheduler-on-aws-remote`) を起動します。各リモートスタックが正常に起動すると、プライマリアカウントにそれ以上変更を加えることなく、プライマリソリューションアカウントがアカウント ID で更新されます。

AWS Systems Manager Parameter Store でアカウント ID を管理する

AWS Systems Manager Parameter Store を使用してリモートアカウント ID を保存します。リモートアカウント ID は、すべての項目がアカウント ID であるリストパラメータとして、またはリモートアカウント ID をカンマで区切ったリストを含む文字列パラメータとして保存できます。パラメータのフォーマットは、{param:_name_} です。このフォーマットの name は、Parameter Store のパラメータ名前になります。

この機能を活用するには、AWS での Instance Scheduler のハブスタックをパラメータストアと同じアカウントで起動する必要があります。

スケジューリングをサポートしているサービス

AWS での Instance Scheduler は現在、次のサービスのスケジューリングをサポートしています。

- Amazon EC2
- Amazon EC2 Auto Scaling グループ
- Amazon RDS
- Amazon Aurora クラスター
- Amazon DocumentDB
- Amazon Neptune

インスタンスのシャットダウン動作

Amazon EC2

このソリューションは、EC2 インスタンスを自動的に停止するように設計されており、インスタンスのシャットダウン動作が終了ではなく停止に設定されていることを前提としています。Amazon EC2 インスタンスを終了した後で再起動できないことに注意してください。

EC2 インスタンスは、デフォルトでシャットダウン時に終了ではなく停止するように設定されていますが、この動作を変更することができます。そのため、AWS での Instance Scheduler を使用して制御するインスタンスが停止のシャットダウン動作に設定されていることを確認します。設定されていない場合、インスタンスは終了します。

Amazon RDS、Amazon Neptune、Amazon DocumentDB

このソリューションは、RDS、Neptune、DocDB の各インスタンスを削除するのではなく、自動的に停止するように設計されています。Create RDS Instance Snapshot AWS CloudFormation テンプレートパラメータを使用して、ソリューションがインスタンスを停止する前に RDS DB インスタンスのスナップショットを作成できます。スナップショットは、次回インスタンスが停止されて新しいスナップショットが作成されるまで保持されます。

> **Note**
>
> スナップショットは Amazon Aurora クラスターでは使用できません。Schedule Aurora Clusters テンプレートパラメータを使用して、Aurora クラスターの一部である、または Aurora データベースを管理する RDS DB インスタンスを起動および停止できます。初期設

定時に定義したタグキーと、そのクラスターをスケジュールするタグ値としてスケジュール名を使用して、クラスター (個々のインスタンスではない) にタグ付けする必要があります。

RDS DB インスタンスの起動と停止の制限に関する詳細については、Amazon RDS ユーザーガイドの「一時的に Amazon RDS DB インスタンスを停止する」を参照してください。

RDS DB インスタンスが停止すると、キャッシュがクリアされるため、インスタンスの再起動時にパフォーマンスが低下する可能性があります。

Amazon RDS のメンテナンスウィンドウ

すべての RDS DB インスタンスには、システムの変更が適用される週 1 回のメンテナンスウィンドウがあります。メンテナンスウィンドウ中、Amazon RDS は 7 日以上停止したインスタンスを自動的に起動し、メンテナンスを適用します。Amazon RDS はメンテナンスイベントが完了してもインスタンスを停止しません。

ソリューションでは、RDS DB インスタンスの希望するメンテナンスウィンドウを実行期間としてスケジュールに追加するかどうかを指定できます。ソリューションでは、メンテナンスウィンドウの最初にインスタンスを起動しますが、他の実行期間でインスタンスを実行するように指定されておらず、メンテナンスイベントが完了した場合はメンテナンスウィンドウの終了時にインスタンスを停止します。

メンテナンスウィンドウの終了までにメンテナンスイベントが完了しなかった場合、インスタンスはメンテナンスイベントが完了した後のスケジュール間隔まで実行されます。Amazon RDS のメンテナンスウィンドウに関する詳細は、Amazon RDS ユーザーガイドの「DB インスタンスのメンテナンス」を参照してください。

Amazon EC2 Auto Scaling グループ

このソリューションは、スケジュールされたスケーリングアクションを使用することで Amazon EC2 Auto Scaling グループが自動的に停止するように設計されています。スケジュールされたスケーリングアクションは、Auto Scaling グループ (ASG) に設定されます。スケジュールされたスケーリングアクションが ASG を停止すると、ASG が自動的に再開されるまで、その最小キャパシティ、希望するキャパシティ、最大キャパシティが `0` に設定されます。これにより、最小キャパシティ、希望するキャパシティ、最大キャパシティが元の値に戻ります。

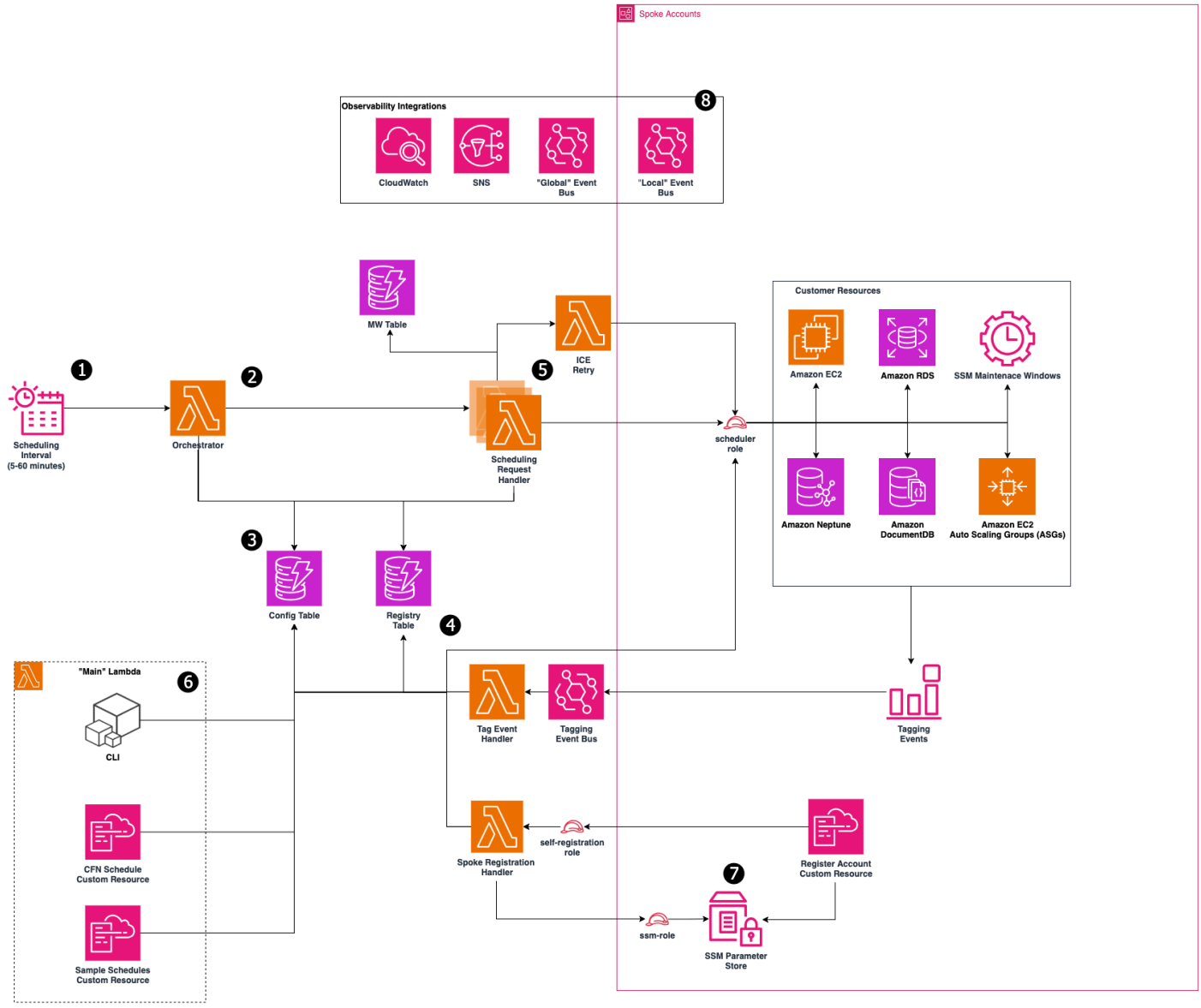
アーキテクチャ

このセクションでは、リファレンス実装アーキテクチャ図、AWS Well-Architected の設計上の考慮事項、セキュリティコンポーネント、スケジューラ設定、およびこのソリューションで使用される AWS サービスについて説明します。

アーキテクチャ図

このソリューションをデプロイすると、AWS アカウント に次のコンポーネントがデプロイされます。

AWS クラウド上の Instance Scheduler



1. Amazon EventBridge ルールは、設定可能な間隔 (デフォルトは 5 分ごと) でオーケストレーション Lambda 関数をトリガーします。
2. EventBridge ルールは、DynamoDB 設定テーブルをクエリしてアクティブなスケジューリングターゲットを識別する AWS Lambda オーケストレーション関数を呼び出します。次に、オーケストレーターはアクティブなターゲットごとに並列スケジューリング Lambda 関数を呼び出します。
3. スケジュール定義と期間は Amazon DynamoDB 設定テーブルに保存されます。このテーブルでは、インスタンスの起動と停止のタイミングを制御するために、任意の数のスケジュールと期間を定義できます。

4. DynamoDB レジストリテーブルは、すべてのマネージドリソースを自動的に追跡します。リソースがスケジューリング用にタグ付けされると、AWS タグ付けイベントに応じてこのテーブルに登録されます。
5. 各スケジューリング Lambda 関数は、タグ付けされたリソースを記述し、現在の時刻に対してスケジュールを評価し、適切な開始または停止アクションを実行します。
   a. EC2 インスタンスの場合、容量不足のために開始オペレーションが失敗した場合、ソリューションは、開始オペレーションを再試行する前にインスタンスを代替インスタンスタイプにサイズ変更するよう設定できます。
6. スケジュール管理は、DynamoDB コンソール、スケジューラ CLI ツール、または AWS CloudFormation Custom リソースから利用できます。このソリューションは、いくつかのサンプルスケジュールを事前設定してデプロイします。
7. クロスアカウントデプロイでは、スポークアカウントが自動的にハブアカウントに自己登録するハブスポークアーキテクチャを使用します。スポークスタックはハブスタックと同じリージョンにデプロイする必要があり、ハブスタックまたは同じ AWS Organization のメンバーによって事前に承認されている必要があります。
8. このソリューションは、ハブアカウント (グローバルイベント) とスポークアカウント (リージョンごとのローカルイベント) の EventBridge バスに、スケジューリングイベントと登録イベントを発行します。

> **Note**
>
> AWS CloudFormation のリソースは、(AWS CDK) のコンストラクトで作成されています。

このソリューションで使用されるすべての Lambda 関数は、リソースの権限要件として AWS IAM を利用し、Amazon Simple Notification Service (Amazon SNS トピック) と DynamoDB テーブルの暗号化には AWS KMS を活用します。

ソリューションはスケジューリング間隔を実行するたびに、適切にタグ付けされた各インスタンスの現在の状態を、関連するスケジュールのターゲットの状態 (インスタンスタグのスケジュール内の 1 つ以上の期間によって定義される) を確認します。その後、スケジュール間隔によって、開始または停止のアクションが必要に応じて適切に適用されます。

例えば、Lambda 関数が金曜日の午前 9 時 (東部標準時) に呼び出され、停止している EC2 または RDS DB のインスタンスに Schedule=office-hours タグが付与されていることが確認されると、Amazon DynamoDB で office-hours スケジュールの設定詳細が確認されます。office-hours スケ

ジュールに、月曜日から金曜日の午前 9 時 (東部標準時) から午後 5 時 (東部標準時) までインスタンスを実行することを示す期間が含まれている場合、Lambda 関数はそのインスタンスを起動します。

また、Lambda 関数はリソースに関する情報を記録し、オプションの Amazon CloudWatch Custom ダッシュボードに表示します。たとえば、スケジュールごとにタグ付けされたインスタンスの数、それらのインスタンスのサイズ、およびインスタンスが現在実行中かそれとも停止状態かなどの情報が表示されれます。このカスタムダッシュボードの詳細については、「Operational insights ダッシュボード」を参照してください。

> **Note**
>
> Amazon EC2 インスタンスを停止することは、Amazon EC2 インスタンスを終了することとは異なります。Amazon EC2 インスタンスは、デフォルトでシャットダウン時に終了するのではなく停止するように設定されていますが、この動作を変更することもできます。このソリューションを使用する前に、インスタンスが適切に停止または終了するように設定されていることを確認してください。

AWS Well-Architected の設計に関する考慮事項

このソリューションは、AWS Well-Architected フレームワークのベストプラクティスを使用して設計されています。そのため、ユーザーは信頼性、セキュリティ、効率性、およびコスト効果の高いクラウドワークロードを設計し運用することができます。

このセクションでは、このソリューションを構築する際に AWS Well-Architected フレームワークの設計原則とベストプラクティスがどのように適用されたかを説明します。

運用上の優秀性

このセクションでは、オペレーショナルエクセレンスの柱に関する原則とベストプラクティスを用いてこのソリューションをどのように設計したかを説明します。

- このソリューションはメトリクスを Amazon CloudWatch にプッシュして、そのコンポーネント (インフラストラクチャや Lambda 関数など) にオブザーバビリティを提供します。
- AWS X-Ray は Lambda 関数をトレースします。
- エラー報告には Amazon SNS を使用します。

セキュリティ

このセクションでは、このソリューションを設計する際に、セキュリティの柱の原則とベストプラクティスをどのように適用したかについて説明します。

- すべてのサービス間通信は、IAM ロールを使用します。
- すべてのマルチアカウント通信は、IAM ロールを使用します。
- ソリューションで使用されるすべてのロールは、最小特権アクセスに従います。つまり、サービスが正しく機能するために必要な最小限のアクセス許可のみが含まれます。
- DynamoDB テーブルを含むすべてのデータストレージは、保存時に暗号化されます。

信頼性

このセクションでは、信頼性の柱に関する原則とベストプラクティスを用いてこのソリューションをどのように設計したかを説明します。

- このソリューションでは、可能な限りサーバーレスの AWS サービス (Lambda や DynamoDB など) を使用して、高可用性とサービス障害からの回復を確保しています。
- データ処理では Lambda 関数を使用します。このソリューションはデータを DynamoDB に保存するため、デフォルトでは複数のアベイラビリティーゾーンに保持されます。

パフォーマンス効率

このセクションでは、パフォーマンス効率の柱に関する原則とベストプラクティスを用いてこのソリューションをどのように設計したかを説明します。

- このソリューションはサーバーレスアーキテクチャを使用しています。
- このソリューションで使用される AWS のサービス (Lambda や DynamoDB など) をサポートする任意の AWS リージョン でこのソリューションを起動できます。詳細は、「サポートしている AWS リージョン」を参照してください。
- このソリューションは毎日自動的にテストされて、デプロイされます。ソリューションアーキテクトと対象分野の専門家が、実験と改善が必要な分野についてこのソリューションをレビューします。

コスト最適化

このセクションでは、このソリューションを設計する際に、コスト最適化の柱の原則とベストプラクティスをどのように適用したかを説明します。

- このソリューションはサーバーレスアーキテクチャを使用しており、ユーザーは使用した分のみを支払います。
- コンピューティングレイヤーのデフォルトは Lambda で、従量課金制モデルを使用しています。

持続可能性

このセクションでは、このソリューションを設計する際に、持続可能性の柱の原則とベストプラクティスをどのように適用したかを説明します。

- このソリューションは、マネージドサービスとサーバーレスサービスを使用して、バックエンドサービスの環境への影響を最小限に抑えます。
- このソリューションのサーバーレス設計は、継続的に運用されているオンプレミスサーバーのフットプリントと比較して、二酸化炭素排出量を削減することを目的としています。

スケジューラ設定テーブル

AWS での Instance Scheduler がデプロイされると、グローバル設定を含む Amazon DynamoDB テーブルが作成されます。

グローバル設定に含まれているのは、設定テーブル内の type 属性に config の値を持つ項目です。スケジュールと期間の設定は、それぞれ type 属性に schedule と period の値を持つ項目です。DynamoDB コンソールまたはこのソリューションのコマンドラインインターフェイスを使用して、設定テーブルのスケジュールと期間を追加、更新、削除できます。ただし、type が config である項目はすべてソリューションが管理するため、編集しないでください。

Scheduler CLI

このソリューションには、スケジュールと期間を設定するためのコマンドを提供する CLI が含まれています。CLI を使用すると、特定のスケジュールで削減できるコストを見積もることができます。Scheduler CLI によって提供されるコストの見積もりは、概算のみを目的としています。Scheduler CLI の設定と使用の詳細については、「Scheduler CLI」を参照してください。

このソリューションで使用している AWS のサービス

AWS のサービス	説明
AWS Lambda	コア。ソリューションは、インスタンスをスケジュールし、カスタムリソース機能を使用して CloudFormation スタックの更新を管理するためのすべてのロジックを含む Lambda 関数をデプロイします。
Amazon DynamoDB	コア。ソリューションでは、スケジュール設定、状態情報、インスタンスで最後に実行されたアクションを格納する DynamoDB テーブルと、スケジューリング用の Systems Manager メンテナンスウィンドウを格納するテーブルを作成します。
Amazon CloudWatch	コア。ソリューションには、デバッグログと情報ログが格納されます。
AWS IAM	コア。ソリューションは IAM を使用して、インスタンスをスケジューリングするためのアクセス許可を取得します。
Amazon SNS	コア。ソリューションでは、SNS トピックを作成して、ユーザーにエラーメッセージを送信し、サブスクライブして、エラーが発生した場合のトラブルシューティングを行います。
* AWS KMS*	コア。ソリューションでは AWS KMS キーを作成して、SNS トピックを暗号化します。
Amazon EventBridge	コア。ソリューションでは EventBridge ソリューションを作成し、EventBridge でスケジュールされるルールを作成して、一定の間隔で AWS Lambda を呼び出します。

AWS のサービス	説明
AWS Systems Manager	サポート。アプリケーションレベルのリソースの監視と、リソースの操作とコストデータの可視化を提供します。
(Amazon EC2)	スケジューリング。ソリューションは、EC2 インスタンスを起動および停止するために使用されます。インスタンスは、ソリューションで設定された特定のタグのキー/値によって識別されます。
Amazon RDS	スケジューリング。ソリューションで RDS DB インスタンスのステータスを Available または Stopped に変更するのに使用します。インスタンスは、ソリューションで設定された特定のタグのキー/値によって識別されます。
Amazon Aurora:	スケジューリング。ソリューションで Aurora クラスターのステータスを Available または Stopped に変更するのに使用します。クラスターは、ソリューションで設定された特定のタグのキー/値によって識別されます。
Amazon Neptune	スケジューリング。ソリューションで Neptune インスタンスのステータスを Available または Stopped に変更するのに使用します。インスタンスは、ソリューションで設定された特定のタグのキー/値によって識別されます。
– Amazon DocumentDB	スケジューリング。ソリューションで DocumentDB インスタンスのステータスを Available または Stopped に変更するのに使用します。インスタンスは、ソリューションで設定された特定のタグのキー/値によって識別されます。

AWS のサービス	説明
Amazon EC2 Auto Scaling グループ	スケジューリング。ソリューションで EC2 Auto Scaling グループのスケジュールされたスケーリングルールを管理するのに使用します。これらのルールは、設定されたスケジュールに従って Auto Scaling グループを開始/停止します。グループは、ソリューションで設定された特定のタグのキー/値によって識別されます。

セキュリティ

AWS インフラストラクチャでシステムを構築すると、お客様と AWS の間でセキュリティ上の責任が分担されます。この責任共有モデルにより、ホストオペレーティングシステムと仮想化レイヤーからサービスが運用されている施設の物理的なセキュリティに至るまでのコンポーネントを AWS が運用、管理、制御するため、お客様の運用上の負担を軽減するのに役立ちます。AWS セキュリティの詳細については、「AWS クラウドセキュリティ」を参照してください。

AWS KMS

ソリューションでは、AWS マネージドのカスタマーマネージドキーが作成されます。これは、SNS トピックと DynamoDB テーブルのサーバーサイドの暗号化を設定するために使用されます。

AWS IAM

ソリューションの Lambda 関数には、ハブアカウントリソースへのアクセス許可と、Systems Manager パラメータを取得/入力するためのアクセス許可、および CloudWatch ロググループ、AWS KMS キーの暗号化/復号化、SNS へのメッセージ公開のためのアクセス許可が必要です。さらに、Instance Scheduler からスケジューリングロールをすべてのマネージドアカウントで作成し、EC2 の開始/停止、RDS、Auto Scaling リソース、DB インスタンス、インスタンス属性の変更、およびそれらのリソースのタグ更新に必要なアクセス許可を提供する必要があります。必要なすべてのアクセス許可は、ソリューションテンプレートの一部として作成された Lambda サービスロールに対して、ソリューションが提供します。

デプロイ時に、Instance Scheduler によって、デプロイされたハブテンプレート内の特定のスケジューリング Lambda によってのみ引き受けることができるスケジューラロールとともに、Lambda 関数ごとにスコープダウンされた IAM ロールもデプロイされます。これらのスケジュールロールに

は、{namespace}-Scheduler-Role、および {namespace}-ASG-Scheduling-Role のパターンで名前が付けられます。

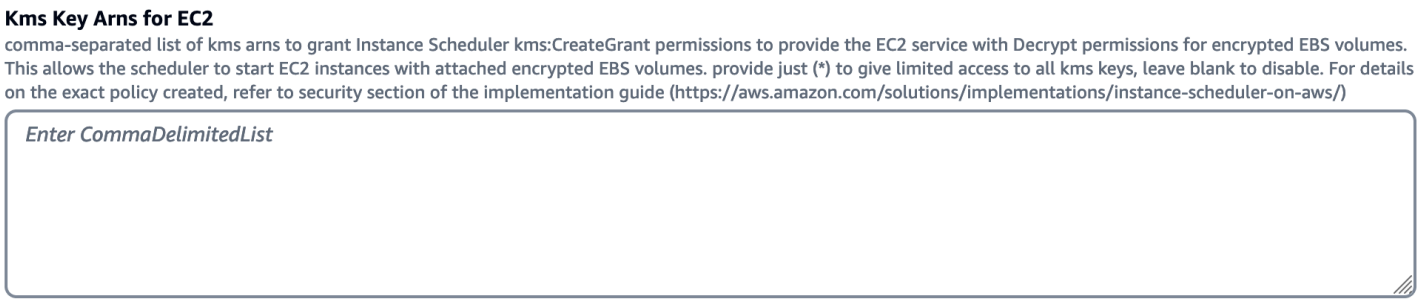
各サービスロールに付与されるアクセス許可の詳細については、「CloudFormation テンプレート」を参照してください。

暗号化された EC2 EBS ボリューム

AWS KMS で暗号化した EBS ボリュームにアタッチされる EC2 インスタンスをスケジューリングする場合は、関連する AWS KMS キーを使用するためのアクセス許可を Instance Scheduler に付与する必要があります。これにより、Amazon EC2 は、アタッチされた EBS ボリュームを関数の実行中に復号することが可能になります。このアクセス許可は、キーを使用する EC2 インスタンスと同じアカウントのスケジューリングロールに付与する必要があります。

AWS KMS キーを Instance Scheduler で使用するためのアクセス許可を付与するには、AWS KMS キーの ARN を、キーを使用している EC2 インスタンスと同じアカウントの、Instance Scheduler スタック (ハブまたはスポーク) に追加します。

EC2 の KMS キーの ARN



これにより、次のポリシーが自動的に生成され、そのアカウントのスケジューリングロールに追加されます。

```
{

  "Version": "2012-10-17",
  "Statement": [
     {
       "Condition": {
           "StringLike": {
              "kms:ViaService": "ec2.*.amazonaws.com"
        },
       "Null": {
```

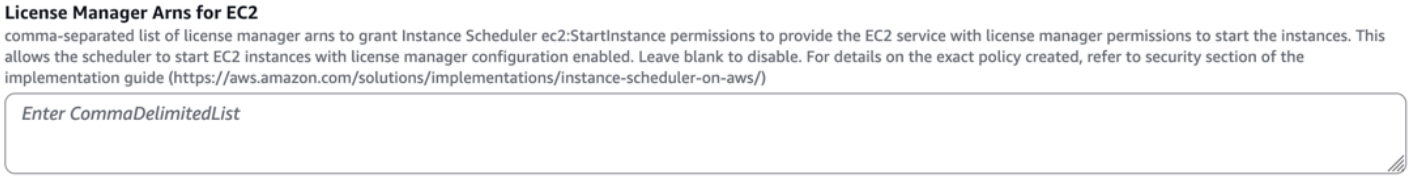
```
                "kms:EncryptionContextKeys": "false",
                "kms:GrantOperations": "false"
            },
            "ForAllValues:StringEquals": {
                "kms:EncryptionContextKeys": [
                    "aws:ebs:id"
                ],
                "kms:GrantOperations": [
                    "Decrypt"
                ]
            },
            "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            },
            "Action": "kms:CreateGrant",
            "Resource": [
                "Your-KMS-ARNs-Here"
            ],
            "Effect": "Allow"
        }
    ]
}
```

EC2 License Manager

AWS License Manager で管理されている EC2 インスタンスをスケジュールする場合は、関連するライセンス設定を使用するためのアクセス許可を Instance Scheduler に付与する必要があります。これにより、ライセンスコンプライアンスを維持しながら、ソリューションでインスタンスを適切に起動および停止できます。このアクセス許可は、License Manager を使用する EC2 インスタンスと同じアカウントのスケジューリングロールに付与する必要があります。

AWS License Manager を Instance Scheduler で使用するためのアクセス許可を付与するには、License Manager 構成 ARN を、License Manager を使用している EC2 インスタンスと同じアカウントで Instance Scheduler スタック (ハブまたはスポーク) に追加します。

EC2 の License Manager 構成 ARN



これにより、次のポリシーが自動的に生成され、そのアカウントのスケジューリングロールに追加されます。

```
{

  "Version": "2012-10-17",
  "Statement": [
     {
       "Action": "ec2:StartInstances",
       "Resource": [
            "Your-License-Manager-ARNs-Here"
       ],
       "Effect": "Allow"
     }
  ]
}
```

License Manager のアクセス許可の詳細については、「AWS License Manager ユーザーガイド」の「AWS License Manager の Identity and Access Management」を参照してください。

はじめに

このガイドには、ソリューションをすばやくデプロイするための簡単な概要と手順が含まれています。このソリューションは、AWS CloudFormation テンプレートとスタックを使用してデプロイを自動化します。CloudFormation テンプレートは、このソリューションに含まれる AWS リソースとそのプロパティを指定します。CloudFormation スタックは、テンプレートに記述されているリソースをプロビジョニングします。

デプロイプロセスの概要

> **Important**
>
> このソリューションには、匿名化された運用メトリクスを AWS に送信するオプションが含まれています。AWS ではこのデータを使用して、ユーザーがこのソリューション、関連サービスおよび製品をどのように使用しているかをよりよく理解し、提供するサービスや製品の改善に役立てます。このアンケートで収集されたデータは AWS が所有します。データ収集にはプライバシー通知が適用されます。

この機能を無効にするには、テンプレートをダウンロードして、AWS CloudFormation の Mapping セクションを変更し、AWS CloudFormation コンソールを使用してアップデートされたテンプレートをアップロードして、ソリューションをデプロイします。

このセクションのステップバイステップの手順に従って、ソリューションを設定してアカウントにデプロイします。

デプロイ時間: 約 5 ~ 10 分 (設定を含まない)。

ステップ 1: Instance Scheduler スタックを起動する

1. AWS アカウントで AWS CloudFormation テンプレートを起動します。
2. 必須パラメータの値を入力します。
3. 他のテンプレートパラメータを確認して、必要に応じて調整します。

ステップ 2 (オプション): セカンダリアカウントでリモートスタックを起動する

1. AWS アカウントで AWS CloudFormation テンプレートを起動します。

2. 必須パラメータの値を入力します。

AWS CloudFormation テンプレート

このソリューションは、AWS CloudFormation テンプレートとスタックを使用してデプロイを自動化します。CloudFormation テンプレートは、このソリューションに含まれる AWS リソースとそのプロパティを指定します。CloudFormation スタックは、テンプレートに記述されているリソースをプロビジョニングします。

このソリューションの CloudFormation テンプレートは、デプロイする前にダウンロードできます。

View template

instance-scheduler-on-aws.template - このテンプレートを使用して、ソリューションとすべての関連コンポーネントを起動します。デフォルト設定では、AWS Lambda 関数、Amazon DynamoDB テーブル、Amazon CloudWatch イベント、CloudWatch カスタムメトリクスがデプロイされますが、特定のニーズに基づいてテンプレートをカスタマイズすることもできます。

View template

instance-scheduler-on-aws-remote.template - このテンプレートを使用して、スポークアカウントのインスタンスをスケジュールするためにソリューションで使用するクロスアカウントロールを起動します。AWS Organizations を使用するデプロイの場合、テンプレートをデプロイするとスポークアカウントもハブとともに登録されるため、手動設定は必要ありません。

> **Note**
>
> このソリューションを以前にデプロイしている場合は、更新手順について「ソリューションを更新する」を参照してください。

ステップ 1: Instance Scheduler ハブスタックを起動する

このセクションのステップバイステップの手順に従って、ソリューションをアカウントにデプロイします。

デプロイ時間: 約 5 分

Launch solution

1. AWS マネジメントコンソールにサインインし、* instance-scheduler-on-aws.template* AWS CloudFormation テンプレートを起動するボタンを選択します。
2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーでリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前に使用する文字の制限に関する詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。
5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションでは、次のデフォルト値を使用します。

パラメータ	デフォルト	説明
	インフラストラクチャ	
Namespace	`default`	複数のソリューションのデプロイを区別するために使用される一意の識別子を指定します (スペースなし)。例: 開発者。
AWS Organizations を使用する	No	AWS Organizations を使用してスポークアカウントの登録を自動化します。
Organization ID/Remote Account IDs	<オプション入力>	AWS Organizations を使用している場合、このフィールドは必須です。`o-xxxxyyy` などの組織 ID を指定します。それ以外の場合は、スケジューリングのために登録できる信頼できるスポー

パラメータ	デフォルト	説明
		クアカウント ID のカンマ区切りリストを指定します (最大 40)。例: 1111111111,2222222222
Schedule tag key	`Schedule`	リソースのスケジュールを決定するためにソリューションで読み取るタグキー。リソースの値ではスケジュールの名前を指定します。デフォルト値を変更する場合は、すべての必要なインスタンスに一貫して正しく適用しやすい名前を割り当ててください。注記: タグキーは大文字と小文字を区別します。
データおよびログを保持する	`Enabled`	ソリューションで使用される DynamoDB テーブルの削除保護を有効にします。これにより、このスタックを削除するときにテーブルが保持されます。このスタックを削除するときにテーブルを削除するには、まずこのパラメータを無効にします。
グローバル設定		
Enable scheduling	`Yes`	No に設定すると、すべてのスケジューリングオペレーションが停止します。

パラメータ	デフォルト	説明
Default time zone	UTC	タイムゾーンを指定しないスケジュールのデフォルトの IANA (International assigned Numbers Authority) タイムゾーン識別子。有効なタイムゾーン識別子のリストについては、tz データベースタイムゾーンのリストの TZ 識別子列を参照してください。
Scheduling interval (minutes)	5	スケジューラの実行間隔を分単位で表します。間隔を短くすると、精度と応答性が向上しますが、コストも増加します。本番稼働用のデプロイでは、安定したオペレーションに最低 5 分かかります。短い値は小規模なテスト専用です。
Enable EC2 SSM maintenance windows	No	スケジュールで 1 つ以上の Systems Manger メンテナンスウィンドウ名を指定できるようにします。AWS での Instance Scheduler ではその後、そのスケジュールでタグ付けされたインスタンスが、関連するメンテナンスウィンドウの少なくとも 10 分前に開始されるようにします。

パラメータ	デフォルト	説明
Create RDS instance snapshots on stop	No	RDS DB インスタンスを停止する前にスナップショットを作成するかどうかを選択します。注記: スナップショットは Amazon Aurora クラスターでは使用できません。
ASG action name prefix	`IS-`	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に名前を付けるときにソリューションが使用するプレフィックス。このプレフィックスが付いたアクションは、必要に応じてソリューションによって追加および削除されます。
ASG scheduled tag key	`scheduled`	廃止。このパラメータは移行のみを目的としており、編集しないでください。
Hub-Account Scheduling		
Region(s)	<オプション入力>	インスタンスがスケジュールされるリージョンのリスト。例: `us-east-1` 、`us-west-1` 。注: このパラメータを空白のままにすると、このソリューションは現在のリージョンを使用します。

パラメータ	デフォルト	説明
KMS Key ARNs for EC2	<オプション入力>	暗号化された EBS ボリュームの復号アクセス許可を EC2 サービスに付与するための、kms:CreateGrant アクセス権限を AWS での Instance Scheduler に付与する KMS ARN のカンマ区切りリスト。これにより、スケジューラーは暗号化された EBS ボリュームがアタッチされた EC2 インスタンスを起動できます。すべての KMS キーへの制限付きアクセスを付与するには (*) を指定します。無効にするには空白のままにします。作成されたポリシーの詳細については、「暗号化された EC2 EBS ボリューム」を参照してください。
EC2 の License Manager ARN	<オプション入力>	License Manager によって管理される EC2 インスタンスを開始するアクセス許可を Instance Scheduler に付与する License Manager 構成 ARN のカンマ区切りリスト。無効にするには空白のままにします。詳細については、「EC2 License Manager」を参照してください。
モニタリング:		

パラメータ	デフォルト	説明
情報タグ付けを有効にする	Yes	有効にすると、Instance Scheduler は、最後に実行されたスケジューリングアクションと発生したエラーを示す情報タグをマネージドリソースに書き込みます。詳細については、「情報タグ」を参照してください。
Enable CloudWatch Debug Logs	No	CloudWatch のログでデバッグレベルのログ記録を有効にします。
Log retention period (days)	30	CloudWatch ログのログ保持期間 (日数)。
Operational Monitoring	Enabled	operational insights ダッシュボードを CloudWatch にデプロイし、ソリューションのオペレーションに関するカスタムメトリクスデータを収集します。必要に応じて、ダッシュボードを無効にすると関連コストを削減できます。
	その他：	
SchedulingRequestHandler メモリサイズ (MB)	512	リソースをスケジュールする AWS Lambda 関数のメモリサイズ。メモリ使用量が多い場合やタイムアウトが頻発する場合に増やします。

パラメータ	デフォルト	説明
オーケストレーターのメモリサイズ (MB)	512	オーケストレーター Lambda 関数のメモリサイズ。メモリ使用量が多い場合やタイムアウトが頻発する場合に増やします。

6. [次へ] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [確認および作成] ページで、設定を確認して確定します。テンプレートによって IAM のリソースが作成されることを確認するボックスをチェックします。
9. [送信] を選択してスタックをデプロイします。

AWS CloudFormation コンソールの [ステータス] 列でスタックのステータスを確認できます。約 5 分で CREATE_COMPLETE ステータスが表示されます。

ステップ 2 (オプション): セカンダリアカウントでリモートスタックを起動する

> **Important**
>
> リモートスタックは、ハブスタックと同じリージョンにデプロイする必要があります。

この自動化された AWS CloudFormation テンプレートは、ハブスタックが他のアカウントのインスタンスをスケジュールできるようにするセカンダリアカウントのアクセス許可を設定します。プライマリ/ハブスタックがハブアカウントに正常にインストールされた後にのみ、リモートテンプレートをインストールしてください。

Launch solution

1. 該当するセカンダリアカウントの AWS マネジメントコンソールにサインインし、instance-scheduler-on-aws-remote AWS CloudFormation テンプレートを起動するボタンを選択します。

2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーでリージョンセレクターを使用します。ハブスタックが AWS Organizations を使用するように設定されている場合は、リモートテンプレートをハブスタックと同じリージョンにデプロイします。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、リモートスタックに名前を割り当てます。
5. [パラメータ] で、テンプレートのパラメータを確認し、変更します。
6. AWS Organizations オプションが有効で、ハブスタックが同様に設定されている場合は、スケジューリングを開始するためにメインスタックをさらに変更する必要はありません。
7. AWS Organization オプションが No に設定されている場合は、ハブスタックを新しいアカウント ID で更新する必要があります。

パラメータ	デフォルト	説明
	インフラストラクチャ	
Namespace	`default`	複数のソリューションのデプロイを区別するために使用される一意の識別子。ハブスタックと同じ値に設定する必要があります。
AWS Organizations を使用する	`No`	AWS Organizations を使用してスポークアカウントの登録を自動化します。ハブスタックと同じ値に設定する必要があります。
Hub Account ID	<必須入力>	このアカウントのリソースをスケジュールする AWS での Instance Scheduler ハブスタックのアカウント ID。
Schedule tag key	`Schedule`	リソースのスケジュールを決定するためにソリューション

パラメータ	デフォルト	説明
		で読み取るタグキー。ハブスタックと同じ値に設定する必要があります。
メンバーアカウントスケジューリング		
Region(s)	<オプション入力>	インスタンスがスケジュールされるリージョンのリスト。例えば、`us-east-1` 、`us-west-1` 。(ハブと同じリージョンリストである必要はありません)。このパラメータを空白のままにすると、このソリューションは現在のリージョンを使用します。
KMS Key ARNs for EC2	<オプション入力>	暗号化された EBS ボリュームの復号アクセス許可を EC2 サービスに付与するための、kms:CreateGrant アクセス権限をソリューションに付与する KMS ARN のカンマ区切りリスト。これにより、スケジューラーは暗号化された EBS ボリュームがアタッチされた EC2 インスタンスを起動できます。すべての KMS キーへの制限付きアクセスを付与するには (*) を指定します。無効にするには空白のままにします。詳細については、「暗号化された EC2 EBS ボリューム」を参照してください。

パラメータ	デフォルト	説明
EC2 の License Manager ARN	<オプション入力>	License Manager によって管理される EC2 インスタンスを開始するアクセス許可を Instance Scheduler に付与する License Manager 構成 ARN のカンマ区切りリスト。無効にするには空白のままにします。詳細については、「EC2 License Manager」を参照してください。

1. [次へ] を選択します。
2. [Options(オプション)] ページで、[Next(次へ)] を選択します。
3. [確認および作成] ページで、設定を確認して確定します。テンプレートによって IAM リソースが作成されることを確認するボックスをチェックします。
4. [送信] を選択してスタックをデプロイします。

AWS CloudFormation コンソールの [ステータス] 列でスタックのステータスを確認できます。約 5 分で CREATE_COMPLETE のステータスが表示されます。

ソリューションを設定する

ソリューションがデプロイされたので、スケジューラーのスケジュールの設定とインスタンスのタグ付けを開始できます。これらの操作を行う方法の詳細については、「スケジュールを設定する」と「スケジューリング用にインスタンスにタグを付ける」を参照してください。

オペレーターガイド

このガイドはこのソリューションのユーザーとオペレーターを対象としており、スケジュールの設定とソリューションのモニタリングの各方法の詳細が記載されています。

スケジュールを設定する

Important

スケジュールの設定ミスにより、インスタンスが継続的に実行され、予期しないコストが発生する可能性があります。リソースにスケジュールを適用する前に、以下を確認してください。

- リソースタグのスケジュール名は、設定テーブルで定義されたスケジュールと完全に一致します。スペルミスまたは存在しないスケジュール名は UnknownSchedule エラーになり、スケジューラによってインスタンスが停止されることはありません。リソースの IS-Error タグをチェックして、この条件を特定します。
- stop_new_instances が false に設定されている場合、最初にタグ付けされた時点でスケジュール期間外に実行中のインスタンスは、次にスケジュールされた停止移行まで停止されません。これにより、インスタンスの実行時間が予想よりも長くなる可能性があります。
- retain_running が true に設定されている場合、実行期間が始まる前に手動で開始されたインスタンスは、その期間の終了時に停止されません。これは設計によるものですが、モニタリングを行わないとインスタンスが無期限に稼働し続ける可能性があります。
- enforced: false (デフォルト) を使用する場合、スケジューラは実行期間中に手動で停止されたインスタンスを再起動せず、最初の停止移行後に実行期間外に手動で起動されたインスタンスを停止しません。

情報タグ付け (デフォルトで有効) を有効にし、リソースの IS-Error および IS-LastAction タグを定期的に確認して、スケジューリングが期待どおりに動作していることを確認することをお勧めします。

ソリューションが正常にデプロイされたら、スケジュールの設定を開始できます。AWS での Instance Scheduler では、以下で説明している 2 つのスケジュール管理方法をサポートしています。

> **Note**
>
> ソリューションは任意のスケジュールの数をサポートでき、そのスケジュールにより制御されるインスタンスをいつ実行するかを定義する 1 つ以上の期間を含めることができます。詳細については、「スケジュール」と「期間」のセクションを参照してください。

Infrastructure as Code の使用 (推奨)

AWS での Instance Scheduler には、Infrastructure as Code (IaC) を使用してスケジュールと期間を管理できる AWS CloudFormation カスタムリソースが用意されています。

IaC を使用してスケジュールを管理する方法については、「Infrastructure as Code (IaC) を使用したスケジュールの管理」を参照してください。

Amazon DynamoDB コンソールと AWS CLI での Instance Scheduler の使用

> **Important**
>
> IaC を使用してスケジュールを管理するカスタムリソースを使用した場合は、DynamoDB コンソールまたは Scheduler CLI を使用してそれらのスケジュールや期間を削除または変更しないでください。これを行うと、CloudFormation に保存されたパラメータとテーブル内の値との間に競合が発生します。また、DynamoDB コンソールまたは Scheduler CLI を使用して作成されたスケジュールには、CloudFormation が管理する期間を使用しないでください。

AWS での Instance Scheduler ハブスタックをデプロイすると、ソリューションは複数のサンプル期間とスケジュールを含む Amazon DynamoDB テーブルを作成し、これらを参考にして独自のカスタム期間とスケジュールを作成できます。DynamoDB でスケジュールを作成するには、設定テーブル (ConfigTable) でスケジュールのいずれかを変更するか、新しいスケジュールを作成します。CLI を使用してスケジュールを作成するには、まず Scheduler CLI をインストールし、次に利用可能なコマンドを使用します。

> **Note**
>
> IaC、DynamoDB、InstanceScheduler CLI を使用して複数のサンプルスケジュールを作成する方法の例については、「サンプルスケジュール」セクションを参照してください。

このセクションには、ソリューションを使用、モニタリング、更新する方法に加え、トラブルシューティングとサポートの情報に関する指示とリファレンスが記載されています。

スケジューリング用にインスタンスにタグを付ける

AWS CloudFormation テンプレートをデプロイすると、ソリューションのカスタムタグの名前 (タグキー) が定義されます。AWS での Instance Scheduler が Amazon EC2 または Amazon RDS のインスタンスを認識するには、そのインスタンスのタグキーがこのカスタムタグキーと一致する必要があります。そのため、すべての該当するインスタンスに一貫して正しくタグを適用することが重要です。このソリューションを使用している間も、インスタンスに対して既存のタグ付けに関するベストプラクティスを引き続き使用できます。詳細については、「Amazon EC2 リソースのタグ付け」および「Amazon RDS リソースのタグ付け」を参照してください。

AWS マネジメントコンソールで、タグエディタを使用して、複数のリソースのタグを一度に適用または変更します。また、そのコンソールで、手動でタグを適用および変更することもできます。

リソースにタグを付けるとすぐに、IS-ManagedBy タグが Instance Scheduler によってリソースに適用され、リソースがスケジューラによって管理されていることを示します。このタグを検索して、リソースがスケジューリングに正しく登録されたことを確認できます。

タグ値の設定

インスタンスにタグを適用する場合は、初期設定時に定義したタグキー (デフォルトでは、タグキーは Schedule) を使用し、タグ値をインスタンスに適用するスケジュールの名前に設定します。タグキーを変更したい場合は、ソリューションパラメータを更新することで変更できます。

> **Note**
>
> Amazon RDS インスタンスの場合、タグ値は長さが 1～256 の Unicode 文字です。プレフィックスに aws: を付けることはできません。文字列には、一連の Unicode 文字、数字、空白、「_」、「.」、「/」、「=」、「」、「-」 (Java 正規表現: "^([\\p{L}\\p{Z}\\p{N}_.:/=\

\-]*)$") のみ使用できます。詳細については、「Amazon RDS リソースのタグ付け」を参照してください。

暗号化された EBS ボリュームを使用する EC2 インスタンス

EC2 DB インスタンスに、カスタマーマネージド KMS キーで暗号化された EBS ボリュームがある場合、それらのインスタンスを起動できるように、Instance Scheduler ロールに KMS:CreateGrant のアクセス許可を与える必要があります。詳細については、「暗号化された EC2 EBS ボリューム」を参照してください。

License Manager で管理される EC2 インスタンス

EC2 インスタンスが AWS License Manager で管理されている場合は、インスタンススケジューラロールに適切な License Manager アクセス許可を付与して、それらのインスタンスを起動および停止できるようにする必要があります。詳細については、「EC2 License Manager」を参照してください。

スケジュールのリファレンス

スケジュールでは、そのスケジュールでタグ付けされたインスタンスを実行するタイミングを指定します。各スケジュールには一意の名前が必要です。この名前は、タグ付けされたリソースに適用するスケジュールを識別するタグ値として使用されます。

期間

各スケジュールには、インスタンスを実行する時間を定義する期間を少なくとも 1 つ含める必要があります。スケジュールには複数の期間を含めることができます。スケジュールで 1 つ以上の期間が使用されている場合、AWS での Instance Scheduler は、少なくとも 1 つの期間が true であれば、適切な開始アクションを適用します。詳細については、「期間のリファレンス」を参照してください。

タイムゾーン

スケジュールのタイムゾーンを指定することもできます。タイムゾーンを指定しない場合は、ソリューションを起動するときに指定したデフォルトのタイムゾーンがスケジュールで使用されます。許容されるタイムゾーン値のリストについては、TZ データベースのタイムゾーンのリストの TZ 列を参照してください。

新しいインスタンスの停止フィールド

stop_new_instances フィールドは、インスタンススケジューラが現在実行期間外に実行されている場合、スケジューリング用に最初にタグ付けされたときにインスタンスを停止するかどうかを制御します。デフォルトでは、このフィールドは true に設定されます。

true に設定すると、スケジュールされた実行期間外の実行中のインスタンスにタグを付けると、Instance Scheduler はインスタンスを直ちに停止します。false に設定すると、Instance Scheduler は次のスケジュールされた停止時間までインスタンスを実行したままにします。

hibernate フィールド

hibernate フィールドでは、Amazon EC2 インスタンスを停止させる際にハイバネーション (休止) を使用できます。このフィールドが true に設定されている場合、EC2 インスタンスではハイバネーションをサポートする Amazon マシンイメージ (AMI) を使用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「Linux AMI」を参照してください。休止状態に入ると、インスタンスメモリ (RAM) に置かれていた内容が、Amazon Elastic Block Store (Amazon EBS) のルートボリュームに保存されます。このフィールドを true に設定すると、ソリューションによってインスタンスが停止したときに、インスタンスは停止するのではなく休止状態になります。

ハイバネーションを使用するようにソリューションを設定しても、インスタンスがハイバネーションに設定がされていない、またはハイバネーションの前提条件を満たしていない場合、ソリューションは警告をログに記録し、インスタンスはハイバネーションなしで停止します。詳細については、Amazon EC2 ユーザーガイドの「Amazon EC2 インスタンスの休止」を参照してください。

enforced フィールド

スケジュールには、インスタンスが実行期間外に手動で起動されたり、実行期間中に手動で停止されたりすることを防ぐための enforced フィールドが含まれています。このフィールドを true に設定している状態でユーザーが実行期間外に手動でインスタンスを起動すると、ソリューションはインスタンスを停止します。このフィールドが true に設定されていると、実行期間中に手動で停止されるインスタンスも再起動されます。

retain_running フィールド

retain_running フィールドは、期間の開始前にインスタンスを手動で起動した場合に、ソリューションが実行期間の終了時にインスタンスを停止しないようにします。例えば、午前 9 時から午後 5 時まで実行する期間のインスタンスを午前 9 時より前に手動で起動した場合、ソリューションは午後 5 時にインスタンスを停止しません。

Systems Manager メンテナンスウィンドウのフィールド (EC2 インスタンスのみに適用)

ssm-maintenance-window フィールドでは、AWS Systems Manager のメンテナンスウィンドウを実行期間として自動的にスケジュールに追加できます。Amazon EC2 インスタンスと同じアカウントと AWS リージョンに存在するメンテナンスウィンドウの名前を指定すると、ソリューションはメンテナンスウィンドウの開始の少なくとも 10 分前にインスタンスを起動し、他の実行期間でインスタンスを実行するように指定されていない場合は、メンテナンスウィンドウの終了時にインスタンスを停止します。

SSM メンテナンスウィンドウが作成され、SSM メンテナンスウィンドウの名前でスケジュールが設定されると、次回にスケジュールされている Lambda の実行時に変更が反映されます。例えば、スケジューラー Lambda の実行頻度を 5 分に設定した場合、メンテナンスウィンドウの変更は次の 5 分間隔で Lambda によって反映されます。

AWS での Instance Scheduler では、メンテナンスウィンドウの開始の少なくとも 10 分前にインスタンスが起動されるようにします。Scheduling Interval AWS CloudFormation パラメータに設定した値によっては、インスタンスが少なくとも 10 分早く起動するように、インスタンスがメンテナンスウィンドウの開始の 10 分以上前に起動される場合があります。たとえば、Scheduling Interval を 30 分に設定した場合、スケジューラーはメンテナンスウィンドウが開始する 10～40 分前にインスタンスを起動します。

> **Note**
>
> この機能を使用するには、ソリューションハブスタックの Enable EC2 SSM Maintenance Windows CloudFormation パラメータを `yes` に設定する必要があります。

詳細については、AWS Systems Manager ユーザーガイドの「AWS Systems Manager Maintenance Windows」を参照してください。

インスタンスタイプ

Amazon EC2 インスタンスの場合のみ、スケジュールの各期間にオプションとしてインスタンスタイプを指定できます。期間にインスタンスタイプを指定すると、ソリューションはリクエストされたインスタンスタイプに合わせて EC2 インスタンスのサイズを自動的に変更します。

インスタンスタイプを指定するには、<period-name>@<instance-type> 構文を使用します。例えば、`weekends@t2.nano`。Amazon EC2 インスタンスと Amazon RDS のインスタンスをスケ

ジュールする期間のインスタンスタイプを指定した場合、そのインスタンスタイプは Amazon RDS インスタンスでは無視されることに注意してください。

実行中のインスタンスのインスタンスタイプが、その期間に指定されたインスタンスタイプと異なる場合、ソリューションは実行中のインスタンスを停止し、指定されたインスタンスタイプでインスタンスを再起動します。詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイドの「Amazon EC2 インスタンスタイプの変更」を参照してください。

スケジュールの定義

Amazon DynamoDB の AWS での Instance Scheduler の設定テーブルには、スケジュールの定義が含まれています。スケジュールの定義には次のフィールドを含めることができます。

フィールド	説明
`description`	スケジュールの説明 (オプション)
`hibernate`	Amazon Linux を実行する Amazon EC2 インスタンスを休止するかどうかを選択します。このフィールドを true に設定すると、スケジューラはインスタンスを停止したときにインスタンスを休止状態にします。インスタンスのハイバネーションをオンにして、ハイバネーションの前提条件を満たしている必要があることに注意してください。
`enforced`	スケジュールを強制するかどうかを選択します。このフィールドが true に設定されている場合は、スケジューラは実行中のインスタンスを実行期間外に手動で起動すると停止し、実行期間中に手動で停止した場合はインスタンスを起動します。
`name`	スケジュールを識別するために使用される名前。この名前は一意にする必要があり、英数字、ハイフン (-)、アンダースコア (_) 以外は使用できません。

フィールド	説明
`periods`	このスケジュールで使用される期間の名前。期間の name フィールドに表示されているとおりに名前を入力します。 <period-name>@<instance-type> 構文を使用して、期間のインスタンスタイプを指定することもできます。例えば、`weekdays@t2.large` 。
`retain_running`	実行期間の開始前にインスタンスを手動で起動した場合、ソリューションが実行期間の終了時にインスタンスを停止しないようにするかどうかを選択します。
`use_maintenance_window`	実行期間として Amazon RDS メンテナンスウィンドウを Amazon RDS インスタンススケジュールに含めるか、もしくは実行期間として AWS Systems Manager メンテナンスウィンドウを Amazon EC2 インスタンススケジュールに含めることを選択します。このフィールドはデフォルトで有効になっており、値を「false」に設定することで無効にできます。
`ssm_maintenance_window`	このスケジュールの追加実行期間として AWS Systems Manager のメンテナンスウィンドウを追加するかどうかを選択します。スケジュールされた EC2 インスタンスと同じアカウント/リージョンのウィンドウ名と照合される、メンテナンスウィンドウ名の StringSet を受け入れます。 注記: この機能は EC2 インスタンスにのみ適用されます。

フィールド	説明
stop_new_instances	インスタンスが実行期間外に実行されている場合に、最初にタグ付けされたときにインスタンスを停止するかどうかを選択します。デフォルトでは、このフィールドは true に設定されます。
timezone	スケジュールが使用するタイムゾーン。タイムゾーンを指定しない場合は、デフォルトのタイムゾーン (UTC) が使用されます。許容されるタイムゾーン値のリストについては、TZ データベースのタイムゾーンのリストの TZ 列を参照してください。
use_metrics	スケジュールレベルで CloudWatch メトリクスをオンにするかどうかを選択します。このフィールドは、デプロイ時に指定した CloudWatch メトリクスの設定を上書きします。 注記: この機能を有効にすると、スケジュールまたはスケジュールされたサービスごとに 0.90 USD / 月の料金が発生します。

期間のリファレンス

期間には、インスタンスを実行する特定の時間、日、月を設定できる条件が含まれています。期間には複数の条件を含めることができますが、AWS での Instance Scheduler で適切な開始アクションまたは停止アクションを適用するには、すべての条件が true である必要があります。

開始時刻と停止時刻

begintime フィールドと endtime フィールドは、AWS での Instance Scheduler がインスタンスをいつ開始し、いつ停止するのかを定義します。開始時刻のみを指定する場合は、インスタンスを手動で停止する必要があります。weekdays フィールドに値を指定した場合は、ソリューションはその値を使用してインスタンスを停止するタイミングを決定することに注意してください。例え

ば、begintime を午前 9 時に指定し、endtime なしで weekdays の値を月曜日から金曜日までにすると、インスタンスは、隣接する期間をスケジュールしていない限り、金曜日の午後 11 時 59 分に停止されます。

同様に、停止時刻のみを指定する場合は、インスタンスを手動で起動する必要があります。どちらの時間も指定しない場合、このソリューションでは、曜日、その月の日数、または月のルールを使用して、必要に応じて、各日の開始/終了時にインスタンスを開始および停止します。

期間の begintime と endtime の値は、スケジュールで指定されたタイムゾーンである必要があります。スケジュールでタイムゾーンを指定しない場合、ソリューションの起動時に指定されたタイムゾーンを使用します。

スケジュールに複数の期間が含まれる場合は、必ず期間の begintime と endtime の両方を指定することをお勧めします。

指定した開始時間より前にインスタンスを起動した場合は、そのインスタンスは実行期間の終了まで実行されます。例えば、ユーザーがインスタンスを毎日午前 9 時に開始し、そのインスタンスを午後 5 時に停止する期間を定義できます。

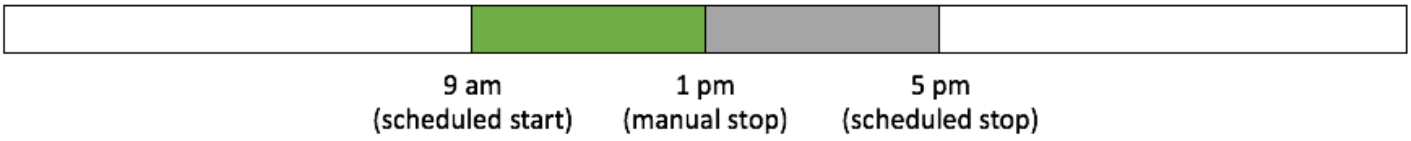
9 時から 5 時でスケジュールされた開始と停止

午前 5 時にそのインスタンスを手動で開始した場合は、ソリューションは午後 5 時にインスタンスを停止します。retain_running フィールドを使用すると、ソリューションは午後 5 時にインスタンスを停止しません。

午前 5 時にスケジュールされた停止

指定した停止時間より前にインスタンスを停止した場合は、そのインスタンスは次の実行期間の開始まで実行されません。前述の例えから引き続き、ユーザーが水曜日の午後 1 時にインスタンスを停止した場合は、このソリューションは木曜日の午前 9 時までそのインスタンスを起動しません。

タイムラインは、スケジュールされた開始時刻は午前 9 時、手動停止時刻は午後 1 時、スケジュールされた停止時刻は午後 5 時であることを示しています。



隣接期間

スケジュールに隣接する 2 つの実行期間が含まれている場合、このソリューションは実行インスタンスを停止しません。例えば、ある期間の `endtime` が午後 11:59 で、別の期間の `begintime` が翌日の午前 0 時である場合は、インスタンスを停止する `weekdays`、`monthdays`、または `months` のルールがないとソリューションはインスタンスの実行を停止しません。

月曜日の午前 9 時から金曜日の午後 5 時までインスタンスを実行するスケジュールを実装するには、次の 3 つの期間が必要です。1 つ目は、月曜日の午前 9 時から午後 11 時 59 分まで該当インスタンスを実行する期間です。2 つ目は、火曜日の午前 0 時から木曜日の午後 11 時 59 分までインスタンスを実行する期間です。3 つ目は、金曜日の午前 0 時から金曜日の午後 5 時までインスタンスを実行する期間です。詳細については、「サンプルスケジュール」を参照してください。

曜日

`weekdays` フィールドは、インスタンスを実行する曜日を定義します。曜日のリスト、曜日の範囲、その月の第 n^th^ 曜日 (その月で n 回目の該当曜日)、またはその月の最後曜日を指定できます。このソリューションでは、省略形の曜日名 (Mon) と数字 (0) がサポートされています。

月の日数

`monthdays` フィールドは、インスタンスを実行する月の日数を定義します。日のリスト、日の範囲、その月の n^th^ 日ごと、月の最終日、または特定の日付に最も近い平日を指定できます。

月

`months` フィールドは、インスタンスを実行する月を定義します。月のリスト、月の範囲、またはその年のある月を起点に n^th^ 月毎を指定できます。このソリューションでは、省略形の月名 (Jan) と数字 (1) がサポートされています。

期間の定義

Amazon DynamoDB の AWS での Instance Scheduler 設定テーブルには、期間の定義が含まれています。期間の定義には、次のフィールドを含めることができます。一部のフィールドでは、Cron 非標準文字がサポートされています。

> **Important**
>
> begintime、endtime、weekdays、months、または monthdays のうち、少なくとも 1 つ指定する必要があります。

フィールド	説明
`begintime`	インスタンスが起動する時刻 (HH:MM 形式)
`description`	期間の説明 (オプション)
`endtime`	インスタンスが停止する時間 (HH:MM 形式)
`months`	インスタンスを実行する月のカンマ区切りリスト、またはハイフンで区切られた月範囲を入力します。例えば、`jan, feb, mar` または `1, 2, 3` と入力して、その月にインスタンスを実行します。または、`jan-mar` または `1-3` と入力することもできます。 また、n^th^ 番目の月ごと、またはある範囲の n^th^ 番目の月ごとに実行するようにインスタンスをスケジュールすることもできます。例えば、`Jan/3` または `1/3` と入力して、1 月から 3 か月ごとにインスタンスを実行します。1 月から 7 月まで 1 か月おきに実行するには、`Jan-Jul/2` と入力します。
`monthdays`	インスタンスを実行する月の日のカンマ区切りリスト、またはハイフンで区切られた日付範囲を入力します。例えば、`1, 2, 3` または `1-3`

フィールド	説明
	と入力して、月の最初の 3 日間にインスタンスを実行します。複数の範囲を入力することもできます。例えば、1-3、7-9 と入力して、1^st^ から 3^rd^、7^th^ から 9^th^ までのインスタンスを実行させます。 また、月の n^th^ 日ごと、またはある範囲の月の n^th^ 日ごとにインスタンスを実行するようにインスタンスをスケジュールすることもできます。例えば、1^st^ 日目から 7 日ごとにインスタンスを実行するには、1/7 と入力します。1^st^ 日から 15^th^ 日までの間で 1 日おきにインスタンスを実行するには、1-15/2 と入力します。 月の最終日にインスタンスを実行するには、L と入力します。指定した日付に最も近い平日にインスタンスを実行するには、日付と W を入力します。例えば、15^th^ 日に最も近い平日にインスタンスを実行するには、15W と入力します。
name	期間を識別するために使用する名前。この名前は一意にする必要があり、英数字、ハイフン (-)、アンダースコア (_) 以外は使用できません。

フィールド	説明
weekdays	インスタンスを実行する曜日のカンマ区切りリスト、または曜日の範囲を入力します。例えば、0, 1, 2 または 0-2 と入力して、月曜日から水曜日にインスタンスを実行します。複数の範囲を入力することもできます。例えば、木曜日を除いて毎日インスタンスを実行するには、0-2, 4-6 と入力します。 また、その月の曜日の n^th^ 回ごとにインスタンスを実行するようにスケジュールすることもできます。例えば、Mon#1 または 0#1 と入力して、月の最初の月曜日にインスタンスを実行します。 1 日と L を入力して、その月の曜日の最後の発生時にインスタンスを実行します。例えば、friL または 4L と入力して、月の最終金曜日にインスタンスを実行します。

期間に複数の条件が含まれている場合は、AWS での Instance Scheduler が適切なアクションを適用するには、すべての条件が true である必要があることに注意してください。例えば、値が Mon#1 の weekdays フィールドと値が Jan/3 の months フィールドを含む期間では、四半期の最初の月曜日にアクションが適用されます。

サンプルスケジュール

AWS での Instance Scheduler を使用すると、Amazon Elastic Compute Cloud (Amazon EC2) と Amazon Relational Database Service (Amazon RDS) のインスタンスを自動的に開始および停止できます。次のセクションでは、多くの一般的なユースケースに適応できるスケジュールの例をいくつか紹介します。

標準の午前 9 時 ～ 午後 5 時までの労働時間

このスケジュールは、ロンドンで、平日の午前 9 時から午後 5 時までインスタンスを実行する方法を示しています。

期間

この期間では、平日 (月 ~ 金) の午前 9 時にインスタンスを開始し、午後 5 時にインスタンスを停止します。

フィールド	タイプ	値
begintime	String	09:00
endtime	String	16:59
name	String	weekdays-9-5
weekdays	StringSet	mon-fri

スケジュール

スケジュール名では、使用するインスタンスとタイムゾーンに適用する必要があるタグ値を提供します。

フィールド	タイプ	値
name	String	london-working-hours
periods	StringSet	weekdays-9-5
timezone	String	Europe/London

インスタンスのタグ

このスケジュールをインスタンスに適用するには、Schedule=london-working-hours タグをインスタンスに追加する必要があります。AWS CloudFormation の Instance Scheduler tag name パラメータでデフォルトのタグ名を変更すると、タグが変わります。例えば、タグ名として Sked と入力した場合、タグは Sked=london-working-hours になります。詳細については、Amazon Elastic Compute Cloud ユーザーガイドの「Amazon EC2 リソースのタグ付け」を参照してください。

Scheduler CLI

Instance Scheduler CLI を使用して上記のスケジュールを設定するには、次のコマンドを使用します。

```
scheduler-cli create-period --stack <stackname> --name weekdays-9-5 --weekdays mon-fri
 --begintime 9:00 --endtime 16:59

scheduler-cli create-schedule --stack <stackname> --name london-working-hours --periods
 weekdays-9-5 --timezone Europe/London

Europe/London
```

カスタムリソース

次の CloudFormation テンプレートは、スケジュールのカスタムリソースを使用して前述のスケジュールを作成します。

このテンプレートをデプロイするには、ServiceInstanceScheduleServiceToken ARN を指定する必要があります。この ARN は、以前にデプロイした Instance Scheduler のハブスタックを選択してから、[出力] を選択すると、AWS CloudFormation コンソールに表示されます。

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  ServiceInstanceScheduleServiceTokenARN:
    Type: String
    Description: (Required) service token arn taken from InstanceScheduler outputs
Metadata:
  'AWS::CloudFormation::Designer': {}
Resources:
  LondonWorkingWeek:
    Type: 'Custom::ServiceInstanceSchedule'
    Properties:
      NoStackPrefix: 'True'
      Name: london-working-hours
      Description: run instances from 9am to 5pm in London on weekdays
      ServiceToken: !Ref ServiceInstanceScheduleServiceTokenARN
      Timezone: Europe/London
      Periods:
     - Description: 9am to 5pm on weekdays
        BeginTime: '09:00'
        EndTime: '16:59'
```

```
        WeekDays: mon-fri
```

午後 5 時以降にインスタンスを停止する

インスタンスは日中いつでも自由に起動できます。このスケジュールにより、毎日午後 5 時 (ET) に停止コマンドが自動的に送信されます。

期間

この期間では、毎日午後 5 時にインスタンスを停止します。

フィールド	タイプ	値
endtime	String	16:59
name	String	stop-at-5

スケジュール

スケジュール名では、使用するインスタンスとタイムゾーンに適用する必要があるタグ値を提供します。

フィールド		値
name	String	stop-at-5-new-york
periods	StringSet	stop-at-5
timezone	String	America/New York

インスタンスのタグ

このスケジュールをインスタンスに適用するには、Schedule=stop-at-5-new-york タグをインスタンスに追加する必要があります。AWS CloudFormation の Instance Scheduler tag name パラメータでデフォルトのタグ名を変更した場合、タグが変わります。例えば、タグ名として Sked と入力した場合、タグは Sked=stop-at-5-new-york になります。詳細については、Amazon Elastic Compute Cloud ユーザーガイドの「Amazon EC2 リソースのタグ付け」を参照してください。

Scheduler CLI

Instance Scheduler CLI を使用して上記のスケジュールを設定するには、次のコマンドを使用します。

```
scheduler-cli create-period --stack <stackname> --name stop-at-5 --endtime 16:59

scheduler-cli create-schedule --stack <stackname> --name stop-at-5-new-york --periods
 stop-at-5 --timezone America/New_York
```

カスタムリソース

次の CloudFormation テンプレートは、スケジュールのカスタムリソースを使用して前述のスケジュールを作成します。

このテンプレートをデプロイするには、ServiceInstanceScheduleServiceToken ARN を指定する必要があります。この ARN は、以前にデプロイした Instance Scheduler のハブスタックをクリックしてから、[出力] を選択すると、AWS CloudFormation コンソールに表示されます。

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  ServiceInstanceScheduleServiceTokenARN:
    Type: String
    Description: (Required) service token arn taken from InstanceScheduler outputs
Metadata:
  'AWS::CloudFormation::Designer': {}
Resources:
  StopAfter5:
    Type: 'Custom::ServiceInstanceSchedule'
    Properties:
      NoStackPrefix: 'True'
      Name: stop-at-5-new-york
      Description: stop instances at 5pm ET every day
      ServiceToken: !Ref ServiceInstanceScheduleServiceTokenARN
      Timezone: America/New_York
      Periods:
      - Description: stop at 5pm
        EndTime: '16:59'
```

週末にインスタンスの停止

このスケジュールは、月曜日の午前 9 時 (ET) から金曜日の午後 5 時 (ET) までのインスタンスの実行方法を示しています。月曜日と金曜日は終日ではないため、このスケジュールには、月曜日、火曜日から木曜日、金曜日の 3 つの期間が含まれます。

期間

1 つ目の期間は、タグ付けされたインスタンスを月曜日の午前 9 時に開始し、深夜に停止します。この期間には、次のフィールドと値が含まれます。

フィールド	タイプ	値
begintime	String	09:00
endtime	String	23:59
name	String	mon-start-9am
weekdays	StringSet	mon

2 つ目の期間は、火曜日から木曜日までにタグ付けされたインスタンスを毎日実行します。この期間には、次のフィールドと値が含まれます。

フィールド		値
name	String	tue-thu-full-day
weekdays	StringSet	tue-thu

3 つ目の期間は、タグ付けされたインスタンスを金曜日の午後 5 時に停止します。この期間には、次のフィールドと値が含まれます。

フィールド		値
begintime	String	00:00

フィールド		値
endtime	String	16:59
name	String	fri-stop-5pm
weekdays	StringSet	fri

スケジュール

スケジュールは、3 つの期間をタグ付けされたインスタンスのスケジュールに結合します。スケジュールには、次のフィールドと値が含まれます。

フィールド		値
name	String	mon-9am-fri-5pm
periods	StringSet	mon-start-9am,tue-thu-full-day,fri-stop-5pm
timezone	String	America/New_York

インスタンスのタグ

このスケジュールをインスタンスに適用するには、Schedule=mon-9am-fri-5pm タグをインスタンスに追加する必要があります。AWS CloudFormation の Instance Scheduler tag name パラメータでデフォルトのタグ名を変更した場合、タグが変わることにご注意ください。例えば、タグ名として Sked と入力した場合、タグは Sked=mon-9am-fri-5pm になります。詳細については、Amazon Elastic Compute Cloud ユーザーガイドの「Amazon EC2 リソースのタグ付け」を参照してください。

Scheduler CLI

Instance Scheduler CLI を使用して上記のスケジュールを設定するには、次のコマンドを使用します。

```
scheduler-cli create-period --stack <stackname> --name
mon-start-9am --weekdays mon --begintime 9:00 --endtime 23:59
```

```
scheduler-cli create-period --stack <stackname> --name
tue-thu-full-day --weekdays tue-thu
scheduler-cli create-period --stack <stackname> --namefri-stop-5pm --weekdays fri --
begintime 0:00 --endtime 17:00

scheduler-cli create-schedule --stack <stackname> --name
mon-9am-fri-5pm --periods
mon-start-9am,tue-thu-full-day,fri-stop-5pm -timezone
America/New_York
```

カスタムリソース

次の CloudFormation テンプレートは、スケジュールのカスタムリソースを使用して前述のスケジュールを作成します。

このテンプレートをデプロイするには、ServiceInstanceScheduleServiceToken ARN を指定する必要があります。この ARN は、以前にデプロイした Instance Scheduler のハブスタックを選択してから、[出力] を選択すると、AWS CloudFormation コンソールに表示されます。

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  ServiceInstanceScheduleServiceTokenARN:
    Type: String
    Description: (Required) service token arn taken from InstanceScheduler outputs
Metadata:
  'AWS::CloudFormation::Designer': {}
Resources:
  StopOnWeekends:
    Type: 'Custom::ServiceInstanceSchedule'
    Properties:
      NoStackPrefix: 'True'
      Name: mon-9am-fri-5pm
      Description: start instances at 9am on monday and stop them at 5pm on friday
      ServiceToken: !Ref ServiceInstanceScheduleServiceTokenARN
      Timezone: America/New_York
      Periods:
      - Description: 9am monday start
        BeginTime: '09:00'
        EndTime: '23:59'
        WeekDays: mon
      - Description: all day tuesday-thursday
        WeekDays: tue-thu
      - Description: 5pm friday stop
```

```
        BeginTime: '00:00'
        EndTime: '16:59'
        WeekDays: fri
```

ソリューションリソース

次のリソースは、AWS での Instance Scheduler スタックの一部として作成されます。

リソース名	型	説明
Main	`AWS::Lambda::Function`	AWS Lambda 関数のインスタンススケジューラ。
Scheduler Config Helper	`Custom::ServiceSetup`	グローバル構成設定を Amazon DynamoDB に保存します。
Scheduler Invoke Permission	`AWS::Lambda::Permission`	Amazon CloudWatch イベントが Instance Scheduler の AWS Lambda 関数を呼び出すことを許可します。
Scheduler Logs	`AWS::Logs::LogGroup`	Instance Scheduler 用の Amazon CloudWatch ロググループ
Scheduler Policy	`AWS::IAM::Policy`	スケジューラがアクションの開始と停止の実行、Amazon EC2 インスタンス属性の変更、タグの設定、スケジューラリソースへのアクセスを許可するポリシー。
Scheduler Rule	`AWS::Events::Rule`	スケジューラの Lambda 関数を呼び出す Amazon EventBridge のイベントルール。
Configuration Metrics Event Rule	`AWS::Events::Rule`	設定記述の匿名化されたメトリクスの関数を定期的に呼び

リソース名	型	説明
		出す Amazon EventBridge イベントルール。匿名化されたメトリクスが無効になっている場合は、無効になります。
State Table	`AWS::DynamamoDB::Table`	インスタンスの最後の目的の状態を保存する DynamoDB テーブル。
Config Table	`AWS::DynamamoDB::Table`	グローバル設定、スケジュール、および期間のデータを保存する DynamoDB テーブル。
Instance Scheduler SNS Topic	`AWS::SNS::Topic`	サブスクライブした E メールアドレスに警告メッセージとエラーメッセージを送信します。

Scheduler CLI

AWS での Instance Scheduler ソリューションのコマンドラインインターフェイス (CLI) を使用すると、スケジュールと期間を設定し、特定のスケジュールのコスト削減を見積もることができます。

前提条件

このソリューションの CLI には、Python 3.8 以降と boto3 の最新バージョンが必要です。

認証情報

Scheduler CLI を使用するには、AWS CLI の認証情報が必要です。詳細については、AWS CLI ユーザーガイドの「AWS CLI での設定と認証情報ファイル設定」を参照してください。

認証情報には、次のアクセス権限が必要です。

- `lambda:InvokeFunction` - スケジューラスタックで InstanceSchedulerMain 関数を呼び出し、コマンドラインからスケジューラ設定データベースのスケジュールと期間の情報を更新します。

- `cloudformation:DescribeStackResource` – スタックから AWS Lambda 関数の物理リソース ID を取得し、CLI リクエストを処理します。

Scheduler CLI とレスポンスによって行われたリクエストは、`AdminCliRequestHandler-yyyymmdd` ログストリームに記録されます。

> **Note**
>
> profile-name 引数を使用してプロファイルを指定する場合、指定するプロファイルにこれらのアクセス許可が必要です。profile-name 引数の詳細については、「共通引数」を参照してください。

Scheduler CLI をインストールする

1. Scheduler CLI パッケージ (instance_scheduler_cli.zip) をダウンロードして、ご自分のコンピューターのディレクトリに配置します。

   > **Important**
   >
   > ファイルをご自分のコンピューターのディレクトリに配置せずにそのディレクトリからインストールすると、インストールは失敗します。

2. zip アーカイブをご自分のコンピューターのディレクトリ (instance_scheduler_cli) に解凍します。
3. 解凍した CLI パッケージを配置したのと同じディレクトリから、scheduler-cli をご自分の環境にインストールします。

   > **Note**
   >
   > Scheduler-CLI には、Python 3.8 以降と pip および boto3 の最新バージョンが必要です。ローカルマシンにこれらすべてがインストールされていない場合は、Scheduler-CLI をインストールする前に pip の公式ドキュメントでインストール手順を確認してください。

   ```
   pip install --no-index --find-links=instance_scheduler_cli instance_scheduler_cli
   ```

4. インストールが成功したことを確認します。

```
scheduler-cli --help
```

> **Note**
>
> 必要に応じて CLI の sdist を使用し、上記と同じ手順でインストールすることもできます。

コマンド構造

Scheduler CLI は、コマンドラインでマルチパート構造を使用します。次のパートでは、Scheduler CLI の Python スクリプトを指定します。Scheduler CLI には、期間とスケジュールで実行するオペレーションを指定するコマンドがあります。オペレーションの特定の引数は、コマンドラインで任意の順序で指定できます。

```
scheduler-cli <command> <arguments>
```

共通引数

Scheduler CLI では、すべてのコマンドで使用できる次の引数がサポートされています。

引数	説明
`--stack [replaceable]<stackname>`	スケジューラースタックの名前 重要: この引数はすべてのコマンドで必須です。
`--region [replaceable]<regionname>`	スケジューラースタックをデプロイした AWS リージョンの名前 注記: デフォルトの設定ファイルと認証情報ファイルがこのソリューションのスタックと同じリージョンにインストールされていない場合は、この引数を使用する必要があります。
`--profile-name [replaceable]<profilename>`	コマンドの実行に使用するプロファイルの名前。プロファイル名が指定されていない場合

引数	説明
	は、デフォルトのプロファイルが使用されます。
`--query`	コマンド出力を制御する JMESPath 式。出力制御の詳細については、AWS CLI ユーザーガイドの「AWS コマンドラインインターフェイスでのコマンド出力の制御」を参照してください。
`--help`	Scheduler CLI の有効なコマンドと引数を表示します。特定のコマンドとともに使用すると、そのコマンドの有効なサブコマンドと引数が表示されます。
`--version`	Scheduler CLI のバージョン番号を表示します。

使用できるコマンド

- create-period
- create-schedule
- delete-period
- delete-schedule
- describe-periods
- describe-schedules
- describe-schedule-usage
- update-period
- update-schedule
- help

create-period

説明

期間を作成します。期間には、begintime、endtime、weekdays、months、または monthdays のうち少なくとも 1 つが含まれている必要があります。

引数

--name

- 期間の名前

  タイプ: 文字列

  必須: はい

--description

- 期間の名前

  タイプ: 文字列

  必須: いいえ

--begintime

- 実行期間が開始される時刻。begintime と endtime を指定しない場合、実行期間は 00:00 - 23:59 です。

  タイプ: 文字列

  制約: H:MM または HH:MM の形式

  必須: いいえ

--endtime

- 実行期間が停止する時間。begintime と endtime を指定しない場合、実行期間は 00:00 - 23:59 です。

  タイプ: 文字列

  制約: H:MM または HH:MM の形式

  必須: いいえ

--weekdays

- 期間の曜日

  タイプ: 文字列

  制約: 短縮された曜日名 (mon) または数字 (0) のカンマ区切りリスト。「-」を使用して、範囲を指定します。「/」を使用して、n^th^ 番目の曜日ごとに指定します。

  必須: いいえ

--months

- 期間の月数

  タイプ: 文字列

  制約: 短縮された月名 (jan) または数字 (1) のカンマ区切りリスト。「-」を使用して、範囲を指定します。「/」を使用して、n^th^ 番目の月ごとに指定します。

  必須: いいえ

--monthdays

- 期間の日数

  タイプ: 文字列

  制約: 短縮された月名 (jan) または数字 (1) のカンマ区切りリスト。「-」を使用して、範囲を指定します。「/」を使用して、月の n^th^ 日ごとに指定します。

  必須: いいえ

例

```
$ scheduler-cli create-period --name "weekdays" --begintime 09:00 --endtime 18:00 --
weekdays mon-fri --stack Scheduler
{
   "Period": {
      "Name": "weekdays",
      "Endtime": "18:00",
      "Type": "period",
      "Begintime": "09:00",
      "Weekdays": [
         "mon-fri"
```

```
      ]
    }
     }
```

create-schedule

説明

スケジュールを作成します。

引数

`--name`

- スケジュールの名前

  タイプ: 文字列

  必須: はい

`--description`

- スケジュールの説明

  タイプ: 文字列

  必須: いいえ

`--enforced`

- インスタンスにスケジュールされた状態を適用します。

  必須: いいえ

`--use-metrics`

- Amazon CloudWatch メトリクスを収集します。

  必須: いいえ

`--periods`

- スケジュールの実行期間のリスト。複数の期間が指定されている場合、いずれかの期間が `true` と評価されると、このソリューションはインスタンスを起動します。

  タイプ: 文字列

制約: 期間のカンマ区切りリスト。<period-name>@[replaceable]<instance type> を使用して、期間のインスタンスタイプを指定します。例えば、weekdays@t2.large。

必須: はい

--retain-running

- インスタンスが期間の開始前に手動で起動された場合、実行期間の終了時にこのソリューションによって停止されるのを防ぎます。

必須: いいえ

--ssm-maintenance-window

- 実行期間として AWS Systems Manager のメンテナンスウィンドウを Amazon EC2 インスタンスのスケジュールに追加します。

タイプ: 文字列

必須: いいえ

--do-not-stop-new-instances

- インスタンスが実行期間外で実行されている場合、最初にタグ付けされたインスタンスを停止しないでください。

必須: いいえ

--timezone

- スケジュールが使用するタイムゾーン

型: 文字列の配列

必須: いいえ (この引数を使用しない場合、主要なソリューションスタックのデフォルトのタイムゾーンが使用されます)

--use-maintenance-window

- 実行期間として Amazon RDS メンテナンスウィンドウを Amazon RDS インスタンススケジュールに追加するか、もしくは実行期間として AWS Systems Manager メンテナンスウィンドウを Amazon EC2 インスタンススケジュールに追加します。

タイプ: true/false

必須: いいえ (デフォルトは true)

例

```
$ scheduler-cli create-schedule --name LondonOfficeHours --periods weekdays,weekends --
timezone Europe/London --stack Scheduler
{
   "Schedule": {
      "Enforced": false,
      "Name": "LondonOfficeHours",
      "StopNewInstances": true,
      "Periods": [
         "weekends",
         "weekdays"
      ],
      "Timezone": "Europe/London",
      "Type": "schedule"
   }
}
```

delete-period

`--name`

- 該当する期間の名前

  タイプ: 文字列

  必須: はい

> **Important**
> 期間が既存のスケジュールで使用されている場合は、削除する前にその期間をスケジュールから削除する必要があります。

例

```
$ scheduler-cli delete-period --name weekdays --stack Scheduler
{
   "Period": "weekdays"
}
```

delete-schedule

説明

既存のスケジュールを削除します。

引数

`--name`

- 該当するスケジュールの名前

  タイプ: 文字列

  必須: はい

例

```
$ scheduler-cli delete-schedule --name LondonOfficeHours --stack Scheduler
{
   "Schedule": "LondonOfficeHours"
}
```

describe-periods

説明

Instance Scheduler スタックに設定された期間を一覧表示します。

引数

`--name`

- 説明する特定の期間の名前

  タイプ: 文字列

  必須: いいえ

例

```
$ scheduler-cli describe-periods --stack Scheduler
```

```
{
   "Periods": [
      {
         "Name": "first-monday-in-quarter",
         "Months": [
            "jan/3"
         ],
         "Type": "period",
         "Weekdays": [
            "mon#1"
         ],
         "Description": "Every first Monday of each quarter"
      },
      {
         "Description": "Office hours",
         "Weekdays": [
            "mon-fri"
         ],
         "Begintime": "09:00",
         "Endtime": "17:00",
         "Type": "period",
         "Name": "office-hours"
      },

      {
         "Name": "weekdays",
         "Endtime": "18:00",
         "Type": "period",
         "Weekdays": [
            "mon-fri"
         ],
         "Begintime": "09:00"
      },
      {
         "Name": "weekends",
         "Type": "period",
         "Weekdays": [
            "sat-sun"
         ],
         "Description": "Days in weekend"
      }
   ]
}
```

describe-schedules

説明

Instance Scheduler スタックに設定されたスケジュールを一覧表示します。

引数

`--name`

- 説明する特定のスケジュールの名前

  タイプ: 文字列

  必須: いいえ

例

```
$ scheduler-cli describe-schedules --stack Scheduler

{
   "Schedules": [
      {
         "OverrideStatus": "running",
         "Type": "schedule",
         "Name": "Running",
         "UseMetrics": false
      },
      {
         "Timezone": "UTC",
         "Type": "schedule",
         "Periods": [
            "working-days@t2.micro",
            "weekends@t2.nano"
         ],
         "Name": "scale-up-down"
      },
      {
         "Timezone": "US/Pacific",
         "Type": "schedule",
         "Periods": [
            "office-hours"
```

```
         ],
         "Name": "seattle-office-hours"
      },
      {
         "OverrideStatus": "stopped",
         "Type": "schedule",
         "Name": "stopped",
         "UseMetrics": true
      }
   ]
}
```

describe-schedule-usage

説明

スケジュール内で実行されているすべての期間を一覧表示し、インスタンスの請求時間を計算します。このコマンドを使用して、スケジュールを作成または更新した後の削減額と実行期間を計算するスケジュールをシミュレートします。

引数

`--name`

- 該当するスケジュールの名前

  タイプ: 文字列

  必須: はい

`--startdate`

- 計算に使用する期間の開始日。デフォルトの日付は現在の日付です。

  タイプ: 文字列

  必須: いいえ

`--enddate`

- 計算に使用する期間の終了日。デフォルトの日付は現在の日付です。

  タイプ: 文字列

  必須: いいえ

例

```
$ scheduler-cli describe-schedule-usage --stack InstanceScheduler --name seattle-
office-hours
{
   "Usage": {
      "2017-12-04": {
         "BillingHours": 8,
         "RunningPeriods": {
            "Office-hours": {
               "Begin": "12/04/17 09:00:00",
               "End": "12/04/17 17:00:00",
               "BillingHours": 8,
               "BillingSeconds": 28800
            }
         },
         "BillingSeconds": 28800
      }
   },
   "Schedule": "seattle-office-hours"
```

update-period

説明

既存の期間を更新します。

引数

update-period コマンドは、create-period コマンドと同じ引数をサポートします。引数の詳細については、create period コマンドを参照してください。

> **Important**
>
> 引数を指定しない場合、その引数は期間から削除されます。

update-schedule

説明

既存のスケジュールを更新します。

引数

update-schedule コマンドは、create-schedule コマンドと同じ引数をサポートします。引数の詳細については、create schedule コマンドを参照してください。

> **Important**
>
> 引数を指定しない場合、その引数はスケジュールから削除されます。

help

説明

Scheduler CLI の有効なコマンドと引数のリストを表示します。

例

```
$ scheduler-cli --help
usage: scheduler-cli [-h] [--version]
                     {create-period,create-schedule,delete-period,delete-
schedule,describe-periods,describe-schedule-usage,describe-schedules,update-
period,update-schedule}
                     ...

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit

subcommands:
  Valid subcommands

  {create-period,create-schedule,delete-period,delete-schedule,describe-
periods,describe-schedule-usage,describe-schedules,update-period,update-schedule}
                        Commands help
    create-period       Creates a period
    create-schedule     Creates a schedule
    delete-period       Deletes a period
    delete-schedule     Deletes a schedule
    describe-periods    Describes configured periods
    describe-schedule-usage
```

```
                        Calculates periods and billing hours in which
                        instances are running
    describe-schedules  Described configured schedules
    update-period       Updates a period
    update-schedule     Updates a schedule
```

特定のコマンドと使用すると、--help 引数はそのコマンドの有効なサブコマンドと引数を表示します。

特定のコマンド例

```
$ scheduler-cli describe-schedules --help
usage: scheduler-cli describe-schedules [-h] [--name NAME] [--query QUERY]
                                        [--region REGION] --stack STACK

optional arguments:
  -h, --help         show this help message and exit
  --name NAME        Name of the schedule
  --query QUERY      JMESPath query to transform or filter the result
  --region REGION    Region in which the Instance Scheduler stack is
                        deployed
  --stack STACK, -s STACK
                     Name of the Instance Scheduler stack
```

グローバル構成設定の更新

AWS CloudFormation で Instance Scheduler のハブテンプレートを初めてデプロイしたときに、パラメータ入力として多数のグローバル設定が選択されました。これらのグローバル設定パラメータは、CloudFormation コンソール内でいつでも更新できます。

Instance Scheduler のグローバル設定を更新するには、ハブデプロイを含むアカウント/リージョンにログインし、AWS CloudFormation コンソールに移動します。Instance Scheduler のハブスタックを検索し、 [更新] -> [既存のテンプレートを使用する] を選択します。変更するグローバル設定パラメータを更新し、 [次へ] -> [次へ] -> [送信] を選択して、関連するソリューションリソースの CloudFormation 更新を実行します。

Infrastructure as Code (IaC) を使用したスケジュールの管理

> **Important**
>
> ハブスタックのデプロイが完了したら、別のテンプレートを使用してスケジュールをデプロイします。

AWS での Instance Scheduler には、AWS CloudFormation を通じてスケジュールを設定および管理するために使用できるカスタムリソース (`ServiceInstanceSchedule`) が用意されています。カスタムリソースは、Amazon DynamoDB の Instance Scheduler 設定テーブルと同じデータに PascalCase キーを使用します (例については以下のテンプレートを参照してください)。スケジュールのフィールドの詳細については、「スケジュールの定義」を参照してください。期間のフィールドの詳細については、「期間の定義」を参照してください。

カスタムリソースを使用してスケジュールを作成すると、そのスケジュールの名前はデフォルトでカスタムリソースの論理リソース名になります。別の名前を指定するには、カスタムリソースの Name プロパティを使用します。また、このソリューションはデフォルトでプレフィックスとしてスケジュール名にスタック名を追加します。スタック名をプレフィックスとして追加しない場合は、`NoStackPrefix` プロパティを使用します。

Name プロパティと `NoStackPrefix` プロパティを使用する場合は、必ず一意のスケジュール名を選択していることを確認してください。同じ名前のスケジュールが既に存在する場合は、リソースは作成または更新されません。

IaC を使用してスケジュール管理を始めるには、次のサンプルテンプレートをコピーして貼り付け、好きなだけスケジュールをカスタマイズしてください。ファイルを .template ファイル (例: `my-schedules.template`) として保存し、AWS CloudFormation を使用して新しいテンプレートをデプロイします。完成したスケジュールテンプレートの例については、「サンプルスケジュール」を参照してください。

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  ServiceInstanceScheduleServiceTokenARN:
    Type: String
    Description: (Required) service token arn taken from InstanceScheduler outputs
Metadata:
  'AWS::CloudFormation::Designer': {}
Resources:
```

```
  SampleSchedule1:
    Type: 'Custom::ServiceInstanceSchedule'
    Properties:
      ServiceToken: !Ref ServiceInstanceScheduleServiceTokenARN #do not edit this line
      NoStackPrefix: 'False'
      Name: my-renamed-sample-schedule
      Description: a full sample template for creating cfn schedules showing all
 possible values
      Timezone: America/New_York
      Enforced: 'True'
      Hibernate: 'True'
      RetainRunning: 'True'
      StopNewInstances: 'True'
      UseMaintenanceWindow: 'True'
      SsmMaintenanceWindow: 'my_window_name'
      Periods:
      - Description: run from 9-5 on the first 3 days of March
        BeginTime: '9:00'
        EndTime: '17:00'
        InstanceType: 't2.micro'
        MonthDays: '1-3'
        Months: '3'
      - Description: run from 2pm-5pm on the weekends
        BeginTime: '14:00'
        EndTime: '17:00'
        InstanceType: 't2.micro'
        WeekDays: 'Sat-Sun'

  SampleSchedule2:
    Type: 'Custom::ServiceInstanceSchedule'
    Properties:
      ServiceToken: !Ref ServiceInstanceScheduleServiceTokenARN #do not edit this line
      NoStackPrefix: 'True'
      Description: a sample template for creating simple cfn schedules
      Timezone: Europe/Amsterdam
      Periods:
      - Description: stop at 5pm every day
        EndTime: '17:00'
```

テンプレートをデプロイする場合は、AWS での Instance Scheduler のデプロイ用に ServiceTokenARN を指定する必要があります。この ARN は、デプロイされた Instance Scheduler スタックに移動し、[出力] を選択して、`ServiceInstanceScheduleServiceToken.` を探すことで CloudFormation 内で確認できます。

> **Important**
>
> DynamoDB コンソールまたは Scheduler CLI を使用して、カスタムリソースを使用して設定されたスケジュールと期間を削除または変更しないでください。そうすると、スタックに保存されたパラメータとテーブル内の値の間に競合が発生します。また、DynamoDB コンソールまたは Scheduler CLI を使用して作成されたスケジュールには、カスタムリソースを使用して設定された期間を使用しないでください。

メインの Instance Scheduler スタックを削除する前に、カスタムリソースを使用して作成されたスケジュールと期間を含むすべての追加スタックを削除する必要があります。カスタムリソーススタックには、メインスタックの DynamoDB テーブルへの依存関係が含まれているためです。

DynamoDB の設定テーブルで、カスタムリソースで設定されたスケジュールと期間は configured_in_stack 属性で識別できます。その属性には、項目の作成に使用されたスタックの Amazon リソースネームが含まれます。

EC2 容量不足エラーの処理

容量不足が原因で Instance Scheduler がインスタンスの起動に失敗した場合、デフォルトの動作は、起動に失敗したイベントを発行し (EventBridge イベントを参照)、次のスケジューリング間隔で再試行することです。または、Instance Scheduler を設定して、開始オペレーションを再試行する前にインスタンスのサイズを別のインスタンスタイプに変更することもできます。この機能は、容量に制約のある環境でのインスタンスの可用性を向上させるのに役立ちます。

設定

EC2 インスタンスの代替インスタンスタイプを有効にするには、以下のようなインスタンスタイプのカンマ区切りリストに記載された優先順に `IS-PreferredInstanceTypes` タグをインスタンスに追加します (最初のタイプが最も優先されます)。

```
IS-PreferredInstanceTypes: t3.medium,t3.large,m5.large
```

仕組み

代替インスタンスタイプリストは優先順に提供され、最初のタイプが最も優先されます。Instance Scheduler が EC2 インスタンスを起動しようとすると、次のようになります。

1. インスタンスが現在最も望ましいサイズでない場合、開始する前に最も望ましいサイズへの変更を試みます。
2. 開始オペレーションが成功した場合、それ以上の代替操作は試行されません。
3. 容量不足が原因で開始オペレーションが失敗した場合:
   a. リスト内の次の代替インスタンスタイプにサイズ変更を試みます
   b. 開始オペレーションを再試行します
   c. それでも失敗した場合、次の代替タイプを試行します
   d. 成功するか、すべての代替案が使い果たされるまで続行します

要件と制限

インスタンスの互換性: 代替インスタンスタイプは、インスタンスの現在の設定 (AMI、サブネット、セキュリティグループなど) と互換性がある必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスタイプを変更する」を参照してください。

タグ形式: `IS-PreferredInstanceTypes` タグ値は、有効な EC2 インスタンスタイプのカンマ区切りリストである必要があります。

例

最初に `t3.small` として設定されたインスタンスの場合、以下のように設定できます。

```
Schedule: office-hours
IS-PreferredInstanceTypes: t3.small,t3.medium,t3.large,m5.large
```

容量の問題が原因で `t3.small` インスタンスの起動に失敗した場合、Instance Scheduler は、成功するかすべてのオプションが使い果たされるまで、インスタンスのサイズ変更と起動を `t3.medium` として、次いで、`t3.large`、`m5.large` の順に試みます。

EC2 Auto Scaling グループのスケジューリング

AWS での Instance Scheduler では、スケジュールされたスケーリングアクションを使用した EC2 Auto Scaling グループ (ASG) のスケジューリングをサポートしています。これは EC2/RDS スケジューリングの実装とは異なり、このセクションで詳しく説明します。

スケジュールされたスケーリングアクションの詳細については、「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。

ASG スケジューリングの概要

「スケジューリング用にインスタンスにタグを付ける」の説明に従ってスケジュールタグを適用することで ASG をスケジュールできます。

ASG の実行/停止の定義

Auto Scaling グループを設定すると、ユーザーはその ASG の最小容量、希望容量、最大容量を指定します。Instance Scheduler では、これらの値を ASG の min-desired-max または MDM として参照します。

ASG の実行状態は、`IS-MinDesiredMax` コントロールタグを使用して定義されます。このタグには、目的の MDM 値を 形式 `min,desired,max` (例: `1,3,5`) で含める必要があります。

ASG がスケジューリング用に最初にタグ付けされたときに `IS-MinDesiredMax` タグが指定されていない場合、タグ付け時に ASG の現在のサイズから自動的に生成されます。

すべての ASG で、MDM を 0-0-0 として停止状態が定義されます。

ASG の開始/停止動作

Instance Scheduler が ASG を開始または停止すると、ASG の容量設定を変更します。

ASG の開始: `IS-MinDesiredMax` タグで定義された値 (または ASG が最初にタグ付けされた時点から自動的に生成された値) に最小容量、希望容量、最大容量を設定します。

ASG の停止: 最小容量、希望容量、最大容量を 0-0-0 に設定します。これにより、ASG 内のすべてのインスタンスが終了します。

制限事項

AWS での Instance Scheduler を ASG サービスと互換性のあるスケジュールされたスケーリングルールに変換することで、ASG スケジューリングが実行されます。この変換は、複雑な cron 式を使用しない単純な単一期間スケジュールに最適です。

以下のスケジュール機能は、ASG スケジューリングではサポートされていません。

- enforced や retain running などの高度なスケジュールフラグ。
- 期間の N 番目の平日、最も近い平日、および最後の平日の式。
- 期間がすぐ隣接する、または重複する複数期間スケジュール。

- 複数期間スケジュールのスケジュールされたスケーリングアクションを設定する場合、AWS での Instance Scheduler では、別の重複または隣接する期間によって通常そのアクションがスキップされる場合でも、期間の開始/終了を ASG のアクションの開始/停止に直接変換します。

ソリューションのモニタリング

ロギングと通知

Instance Scheduler は、CloudWatch Logs Insights クエリ用に最適化された構造化ログ記録を使用します。このソリューションは、タグ付けされた各インスタンスの処理情報、インスタンスの期間評価の結果、その期間中のインスタンスの望ましい状態、適用されたアクション、デバッグメッセージを記録します。

ログは 2 つのロググループで Amazon CloudWatch Logs に書き込まれます。

`{stackName}-{namespace}-administrative-logs`

リソースの登録と登録解除、カスタムリソースオペレーション、CLI リクエスト、およびその他の管理アクティビティのログ。

`{stackName}-{namespace}-scheduling-logs`

オーケストレーションやリクエストハンドラーの実行などのスケジューリングオペレーションのログ。

警告とエラーログは、ソリューションが作成した Amazon SNS トピックにも送信され、そこからサブスクライブされたメールアドレスにメッセージが送信されるよう設定することもできます。Amazon SNS のトピックの名前は、このソリューションスタックの [出力] タブで確認できます。

情報タグ

情報タグ付けが有効になっている場合 (デフォルト)、Instance Scheduler はマネージドリソースに直接タグを書き込み、ソリューションのスケジューリングアクティビティを一目で把握できるようにします。これらのタグは AWS Resource Groups タグ付け API を使用して適用され、スケジューラがリソースを処理するたびに更新されます。

この機能は、ハブスタックの情報タグ付けを有効にするパラメータを使用して有効または無効にできます。詳細については、「グローバル構成設定の更新」を参照してください。

情報タグキー

マネージドリソースには、次のタグが書き込まれます。

タグキー	説明
IS-ManagedBy	このリソースを管理する Instance Scheduler ハブスタックの ARN。リソースがスケジューリングのために最初に登録され、その後のスケジューリングアクションごとに適用されます。
IS-LastAction	リソースに対して最後に実行されたスケジューリングアクションと UTC タイムスタンプ。例えば、Started 2025-06-15 09:00:00 UTC や Stopped 2025-06-15 17:00:00 UTC などです。このタグは、スケジューラがリソースをアクティブに開始または停止する場合にのみ更新されます (リソースを評価し、アクションが必要ではないと判断した場合は更新されません)。
IS-Error	リソースの処理中にスケジューラでエラーが発生した場合、このタグにはエラーコードと UTC タイムスタンプが含まれます。例: StartFailed 2025-06-15 09:00:05 UTC 。このタグは、次に成功したスケジューリングアクションで自動的にクリアされます。
IS-ErrorMessage	人が読み取り可能なエラーの説明。このタグは、IS-Error が存在する場合にのみ付与され、それと同時に削除されます。

エラーコード

IS-Error タグには、次のエラーコードが表示される場合があります。

エラーコード	説明
UnknownSchedule	リソースのスケジュールタグで指定されたスケジュール名が、設定テーブルに定義されているいずれのスケジュールとも一致しません。

エラーコード	説明
UnsupportedResource	リソースタイプがスケジューリングに対応していません (別の RDS インスタンスのリードレプリカなど)。
IncompatibleSchedule	リソースに割り当てられたスケジュールは、リソースタイプと互換性がありません (サポートされていない cron 式を使用する ASG スケジュールなど)。
StartFailed	スケジューラはリソースの起動を試みましたが、オペレーションは失敗しました。
StopFailed	スケジューラはリソースの停止を試みましたが、オペレーションは失敗しました。
ConfigurationFailed	スケジューラは Auto Scaling グループでスケジュールされたスケーリングルールを設定しようとしましたが、オペレーションは失敗しました。
UnknownError	リソースの処理中に予期しないエラーが発生しました。

タグの動作

- リソースがスケジューリングのために最初に登録されると、IS-ManagedBy タグはすぐに適用されます。
- リソースの登録が解除されると (スケジュールタグが削除されると)、すべての情報タグ (IS-ManagedBy、IS-LastAction、IS-Error、IS-ErrorMessage) がリソースから削除されます。
- 同じエラーが持続し、リソースに既存のタグがまだ存在する場合、エラータグはスケジューリング間隔ごとに書き換えられません。エラーコードが変更された場合にのみ更新されます。
- すべてのタグ値は、AWS のタグ付け制限に準拠するために 256 文字に切り捨てられます。

タグガバナンスに関する考慮事項

> **Important**
>
> Instance Scheduler は、通常のオペレーションの一環として、マネージドリソースで上記のタグを作成および更新します。組織が AWS Config ルール、タグポリシー、サービスコントロールポリシー、または自動修復を通じてタグガバナンスを強制する場合は、変更管理コントロールが次のタグキーを許可するように設定されていることを確認します。
>
> - `IS-ManagedBy`
> - `IS-LastAction`
> - `IS-Error`
> - `IS-ErrorMessage`
> - `IS-PreferredInstanceTypes` (代替インスタンスタイプを使用する場合)
> - `IS-MinDesiredMax` (Auto Scaling グループをスケジュールする場合)
>
> ガバナンスポリシーでこれらのタグに対応できない場合は、ハブスタックの情報タグ付けを有効にするパラメータを `No` に設定して、情報タグ付けを無効にします。これにより、リソース登録の確認に使用される `IS-ManagedBy` タグも無効になることに注意してください。

コントロールタグ

Instance Scheduler は、情報タグに加えて、特定の機能に次のコントロールタグを使用します。

タグキー	説明
`IS-PreferredInstanceTypes`	容量不足が原因でインスタンスの起動が失敗した場合に試行する代替 EC2 インスタンスタイプのカンマ区切りリスト。詳細については、「EC2 容量不足エラーの処理」を参照してください。
`IS-MinDesiredMax`	`min,desired,max` 形式の Auto Scaling グループの最小容量値、希望容量値、最大容量値。詳細については、「EC2 Auto Scaling グループのスケジューリング」を参照してください。

タグ容量

> **Important**
>
> AWS リソースには通常、リソースあたり 50 個のタグの制限があります。Instance Scheduler は、リソースに対して最大 6 つのタグ (4 つの情報タグと最大 2 つのコントロールタグ) を使用できます。リソースに、Instance Scheduler タグおよび既存のタグ付け戦略に対応するのに十分なタグ容量があることを確認します。
> リソースが 50 タグの制限またはそれに近い場合、情報タグの書き込みが失敗する可能性があります。スケジューラはこれらの失敗を口グに記録しますが、引き続きオペレーションをスケジュールします。タグ付けの問題が疑われる場合は、CloudWatch Logs を確認してください。

CloudWatch Logs Insights クエリの例

Instance Scheduler の構造化口グ記録形式により、CloudWatch Logs Insights を使用した効率的なクエリが可能になります。Logs Insights を使用して口グデータを検索、分析、視覚化し、運用上の問題のトラブルシューティングやスケジューリングアクティビティのモニタリングを行うことができます。

Instance Scheduler には、CloudWatch Logs コンソールの [保存されたクエリー] セクションからアクセスできる事前形式の口グクエリが用意されています。

`SchedulingHistory`

開始および停止オペレーションなど、リソースに対して実行されるスケジューリングアクションをクエリします。

`RegistrationEvents`

リソースの登録イベントと登録解除イベントをクエリします。

`Errors`

エラーログをクエリして、ソリューションに関する問題をトラブルシューティングします。

CloudWatch Logs Insights の詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs Insights で口グデータを分析する」を参照してください。

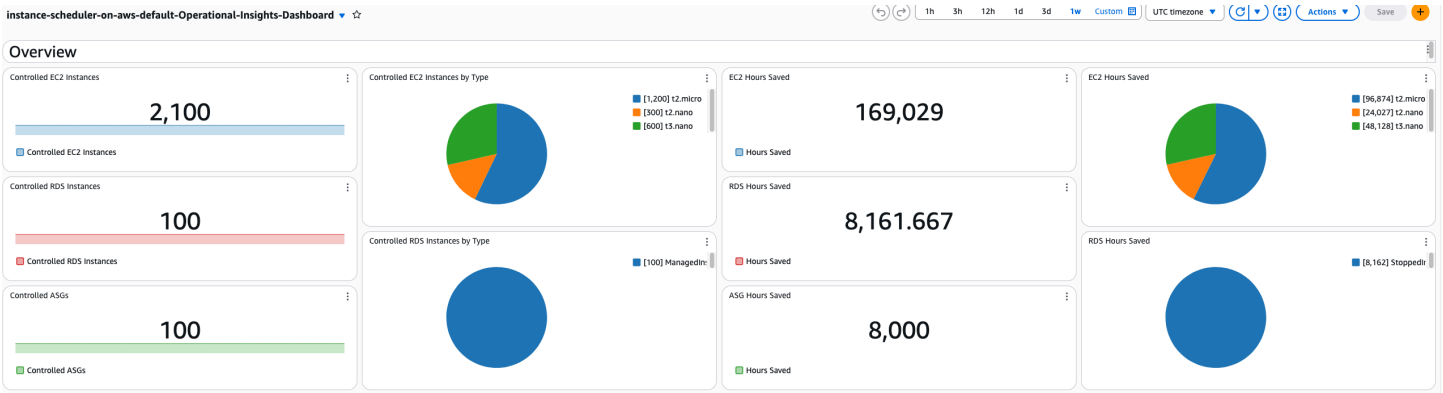
Operational Insights ダッシュボード

Operational Insights ダッシュボードは、スケジュールされたインスタンス管理によるソリューションのパフォーマンスとコスト削減を可視化します。

このダッシュボードを使用するには、ハブスタックパラメータで、[Operational Monitoring] が [有効] に設定されていることを確認します。CloudWatch に移動し、ナビゲーションメニューから [ダッシュボード] を選択します。ダッシュボード名は *{stack-name}-Operational-Insights-Dashboard* になります。

ダッシュボードには、マネージドインスタンス数、保存された実行時間、Lambda 関数のパフォーマンスメトリクスが表示されます。

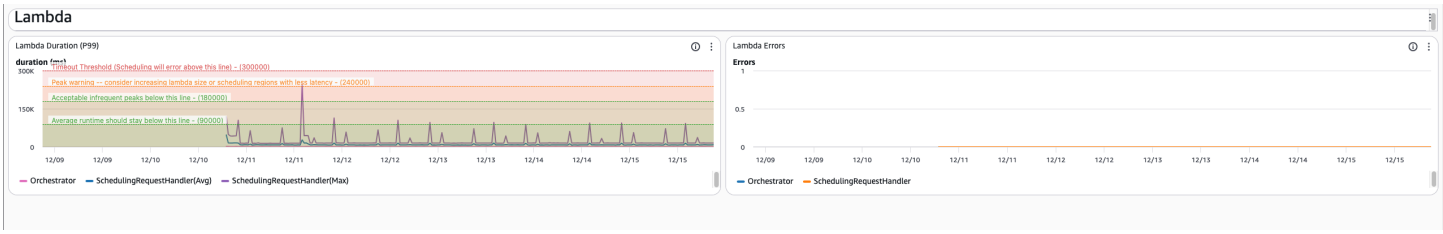
Operational Insights ダッシュボードの概要



> **Note**
>
> これらのグラフの情報は、ソリューションハブスタックで設定されているスケジューリング間隔によって異なります。ソリューションのスケジューリング間隔を更新すると、スケジューリング間隔に対する直近の更新後のスケジューリングメトリクスのみがダッシュボードに表示されます。

Lambda の実行時間をモニタリングして、最適なパフォーマンスを提供します (「クォータ」を参照)。実行時間がタイムアウトしきい値に一貫して近づいている場合は、Lambda サイズプロパティを増やすか、Instance Scheduler をマネージドリージョンへのレイテンシーが低いリージョンにデプロイすることを検討してください。

期間とエラー数を示す Lambda メトリクス



この機能に関連する追加コスト

この運用ダッシュボードは、ソリューションで収集したカスタム CloudWatch メトリクスを利用するため、追加コストが発生します。この機能をオフにするには、ソリューションハブスタックで [Operational Monitoring] を無効にします。この機能には、デプロイのサイズに基づいて、1 か月あたり 3.00 USD と追加のスケーリングコストがかかります。コストは以下のとおりです。

カスタム CloudWatch ダッシュボード	3 USD
インスタンスタイプごとのメトリックス	0.90 USD / インスタンスタイプ*
API の使用	アクティブなターゲットあたり最大 0.10 USD**

*これらのコストはサービスカテゴリ (EC2/RDS) ごとに追跡され、スケジューリングに実際に使用されたインスタンスタイプに対してのみ追跡されます。

*

EventBridge イベントのモニタリング

Instance Scheduler は、スケジューリングイベントと登録イベントを EventBridge イベントバスに発行して、ソリューションオペレーションを可視化し、他の AWS サービスとの統合を可能にします。

イベントタイプ

このソリューションは、主に次の 2 つのカテゴリのイベントを発行します。

スケジュールイベント: Instance Scheduler がマネージドリソースを開始、停止、または設定するアクションを実行したときに発行されます。これらのイベントには、インスタンス、スケジュール、実行されたアクションに関する詳細が含まれます。これらのイベントには、インスタンス、スケジュール、実行されたアクションに関する詳細が含まれます。

登録イベント: リソースがタグ付けオペレーションに基づいてスケジューリングのために登録または登録解除されたときに発行されます。

イベントの送信先

IS-LocalEvents イベントバス: IS-LocalEvents イベントバスは、各メンバーアカウント (ハブアカウントを含む) の各マネージドリージョンにデプロイされます。各バスは、そのリージョン内のスケジュールアクションとリソース登録のイベントを受け取ります。

IS-GlobalEvents イベントバス: ハブアカウントの IS-GlobalEvents イベントバスは、すべての IS-LocalEvents イベントバスに送信された各イベントのコピーを受け取り、すべてのアカウントとリージョンを一元的にモニタリングできます。

EBS EventBridge イベント

以下の EventBridge ルールを作成できます。

- インフラストラクチャ全体のスケジューリングオペレーションをモニタリングする
- インスタンスの起動または停止時に通知をトリガーする
- 他の AWS サービスと統合してワークフローを自動化する
- コンプライアンスのモニタリングとアラートの実装

イベントの構造

すべてのイベントは標準の EventBridge 形式を使用します。次の例は、各イベントタイプの構造を示しています。

スケジュールイベント:

```
{
  "Source": "instance-scheduler",
  "DetailType": "Scheduling Action",
  "Resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0"],
  "Detail": {
    "account": "123456789012",
    "region": "us-east-1",
    "service": "ec2",
    "resource_id": "i-1234567890abcdef0",
    "requested_action": "Start",
    "action_taken": "Started",
```

```
    "schedule": "office-hours"
  }
}
```

登録イベント:

```
{
  "Source": "instance-scheduler",
  "DetailType": "Resource Registered",
  "Resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0"],
  "Detail": {
    "account": "123456789012",
    "region": "us-east-1",
    "service": "ec2",
    "resource_id": "i-1234567890abcdef0",
    "schedule": "office-hours"
  }
}
```

各イベントには、次のキーフィールドが含まれます。

- Source - イベントソースを「インスタンススケジューラ」として識別する
- DetailType - イベントカテゴリを指定する。インスタンスオペレーションの場合は「スケジュールアクション」、イベントのタグ付けの場合は「リソース登録済み」
- Resources – 影響を受けた AWS リソースの ARN を含む配列。
- Detail - リクエストされたアクションと実際の結果の両方について、アカウント ID、リージョン、サービスタイプ (EC2/RDS)、リソース ID、スケジュール、およびイベントをスケジュールするためのイベントペイロードが含まれる

イベントをスケジュールするために指定できる requested_action 値:

- Start: インスタンスを起動するスケジューラ
- Stop: インスタンスを停止するスケジューラ
- Configure: インスタンスを設定するスケジューラ

イベントをスケジュールするために指定できる action_taken 値:

- Started: インスタンスが開始された

- `Stopped`: インスタンスが停止した
- `Hibernated`: インスタンスが休止した
- `Configured`: インスタンス設定が変更された
- `Error`: スケジューリングオペレーション中にエラーが発生した

EventBridge ルールの作成

Instance Scheduler イベントをモニタリングするには:

1. ご利用の AWS アカウント で EventBridge コンソール に移動する
2. `IS-GlobalEvents` イベントバス (集中モニタリング用) または `IS-LocalEvents` イベントバス (ローカルモニタリング用) をターゲットとする新しいルールを作成する
3. Instance Scheduler イベントと一致するイベントパターンを定義する
4. SNS トピック、Lambda 関数、CloudWatch Logs などのターゲットを設定する

Eventbridge の詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon EventBridge とは」を参照してください。

トラブルシューティング

このセクションでは、ソリューションをデプロイして使用するためのトラブルシューティングの手順を説明します。

既知の問題解決には、既知のエラーを軽減する手順が記載されています。これらの手順で問題が解決しない場合は、「サポートに問い合わせる」に、このソリューションに関するサポートケースを開く方法が記載されています。

既知の問題解決

問題: リモートアカウントでインスタンスがスケジュールされていない (v1.4-v3.0)

リモートアカウントでインスタンスがスケジュールされていないことに気付いた場合。

解像度

セカンダリアカウント ID でハブスタックを更新するか、次のタスクを実行します。

1. プライマリアカウントで、CloudWatch コンソールに移動します。
2. ナビゲーションペインで、[ログ] > [ロググループ] の順に選択します。
3. *<STACK_NAME>*-logs という名前のロググループを選択します。
4. アカウント ID (リモートアカウント) のログストリームを検索します。
5. 例えば、アカウント ID と同じ名前のログストリームがない場合は、DynamoDB コンソールに移動して、 *<STACK_NAME>*-<ConfigTable>-<RANDOM> という名前のテーブルを選択します。
6. 検索する項目を選択し、[実行] を選択します。
7. アイテムタイプで [設定] を選択します。
8. remote_account_ids 属性にアカウント ID があるかどうか確認します。
9. アカウント ID がこの属性に表示されていないか確認します。
10. このソリューションが AWS Organizations に設定されている場合は、リモートアカウントでリモートテンプレートをアンインストールしてから再インストールします。
11. このソリューションがリモートアカウント ID を使用するように設定されている場合は、インスタンスをスケジュールし、リモートテンプレートをデプロイするアカウント ID のリスト

で、CloudFormation パラメータの Provide Organization Id OR List of Remote Account IDs を更新します。

問題: スケジュールされていないインスタンス (v3.1 以降)

インスタンスがスケジュールされていないことに気付いた場合。

解像度

1. リソースに IS-ManagedBy タグが適用されていることを確認します。
2. タグが存在しない場合は、Schedule タグを削除して再作成し、登録を再度トリガーします。
3. タグがまだ適用されていない場合は、リージョンでスケジューリングが有効になっていることを確認します。
   a. リージョンのハブ/スポークスタック設定を確認するか、または
   b. リソースと同じリージョンの EventBridge コンソールに移動し、デフォルトのイベントバスに IS-Tagging というプレフィックスが付いたイベントルールがあることを確認します。
4. リージョンが有効になっていない場合は、Instance Scheduler スタックを更新してリージョン CloudFormation パラメータにリージョンを含めます。
5. 問題が解決しない場合は、ソリューション管理ログでハブ登録エラーを確認します。
6. アカウントからソリューションハブアカウントにイベントが転送されないようにするポリシーが組織に存在していないことを確認します。

問題: 暗号化された EC2 インスタンスが起動しない

Instance Scheduler は、暗号化された EBS ボリュームを持つ EC2 インスタンスが起動中であることを報告しているが、実際には起動していません。

解像度

暗号化された EBS ボリュームを持つ EC2 インスタンスをスケジュールするために必要なアクセス許可を Instance Scheduler に付与する方法については、「暗号化された EC2 EBS ボリューム」を参照してください。

問題: 情報タグ付けによる予期しない API コスト

AWS Resource Groups タグ付け API コール、AWS Config 評価、または関連する修復アクションによる予想外に高いコスト。

解像度

Instance Scheduler は、各スケジューリング間隔でマネージドリソースに情報タグを書き込みます。環境が AWS Config ルール、タグポリシー、または自動修復を通じてタグガバナンスを強制する場合は、Instance Scheduler のタグキーが許可されていることを確認します。タグキーと設定ガイダンスの完全なリストについては、「タグガバナンスに関する考慮事項」を参照してください。

ガバナンスポリシーを更新できないは、ハブスタックの情報タグ付けを有効にするパラメータを No に設定して、情報タグ付けを無効にします。

問題: [RDS スナップショットの作成] が有効である場合に RDS インスタンスが停止しない

`rds:CreateDBSnapshot` のアクセス許可がないために `StopDBInstance` 操作を呼び出した場合に、RDS インスタンスが停止されず、ソリューションのスケジューラログが (AccessDenied) エラーを報告します。

解像度

ソリューションを v3.0.5 以降に更新するか、スケジュールされた各アカウントのスケジューラロールに `rds:CreateDBSnapshot` のアクセス許可を追加します。

AWS サポートに問い合わせる

AWS ビジネスサポート+、AWS エンタープライズサポート、または Unified Operations をご利用の場合は、AWS サポートセンターを利用して、このソリューションに関するエキスパートのサポートを受けることができます。次のセクションで、その方法を説明します。

ケースを作成する

1. サポートセンターにサインインします。
2. [ケースを作成] を選択します。

どのようなサポートをご希望ですか?

1. [技術] を選択します。
2. [サービス] で、[ソリューション] を選択します。
3. [カテゴリ] で、AWS での Instance Scheduler (Linux または Windows) を選択します。
4. [重要度] で、ユースケースに最も適したオプションを選択します。
5. [サービス]、[カテゴリ]、[重要度] を入力すると、インターフェイスに一般的なトラブルシューティングの質問へのリンクが表示されます。これらのリンクを使用しても問題を解決できない場合は、[次のステップ: 追加情報] を選択してください。

追加情報

1. [件名] に、質問または問題を要約したテキストを入力します。
2. [説明] で、この製品の名前と使用しているバージョン (例: `Instance Scheduler on AWS vX.Y.Z`) を含めて、問題を詳しく説明します。
3. [ファイルを添付] を選択します。
4. リクエストを処理するためにサポートに必要な情報を添付します。

ケースの迅速な解決にご協力ください

1. 必要な情報を記入します。
2. [次のステップ: 今すぐ解決またはお問い合わせ] を選択します。

今すぐ解決またはお問い合わせ

1. [今すぐ解決] で解決策を確認します。
2. これらの解決策で問題を解決できない場合は、[お問い合わせ] を選択し、必要な情報を入力して [送信] を選択します。

ソリューションを更新する

Instance Scheduler は、AWS CloudFormation を使用してインプレースで安全に更新できるように設計されています。これを行う一般的な手順は次のとおりです。

1. AWS CloudFormation コンソールにサインインし、ハブスタックがインストールされているアカウント/リージョンで `instance-scheduler-on-aws` を選択してから [スタックの更新] を選択します。
2. [直接更新を実行] を選択します。
3. [既存のテンプレートを置換] を選択します。
4. [テンプレートを指定] で、以下を実行します。
   - [Amazon S3 URL] を選択します。
   - 最新のテンプレートのリンクをコピーします。
   - [Amazon S3 URL] ボックスにリンクを貼り付けます。
   - 正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。[次へ] をもう一度選択します。
5. [パラメータ] で、テンプレートのパラメータを確認し、必要に応じて変更します (必要なパラメータの更新については、以下の重大な変更点のリストを参照してください)。各パラメータの詳細については、「ステップ 1. Instance Scheduler スタックを起動する」を参照してください。
6. [次へ] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [レビュー] ページで、設定を確認して確定します。テンプレートが AWS Identity and Access Management (IAM) リソースを作成することを承認するボックスを必ずオンにします。
9. [変更セットの表示] を選択して、変更を確認します。
10. [スタックの更新] を選択してスタックをデプロイします。

AWS CloudFormation コンソールの [ステータス] 列でスタックのステータスを確認できます。数分後に UPDATE_COMPLETE ステータスが表示されます。

各スポークアカウントの aws-instance-scheduler-remote スタックに対して上記の手順を繰り返します。

特定のバージョンの下位互換性のない変更点

ソリューションを更新する場合、重要なデータが失われたり、スケジューリングが中断されたりすることなく、任意の古いバージョンから新しいバージョンに直接アップグレードできます。各メジャーバージョンの動作および下位互換性のない変更のリストについては、以下を参照してください。

完全な変更ログは、ソリューションの GitHub ページで確認できます。

v1.5.0

バージョン 1.5.0 では、クロスアカウントスケジューリングロール ARN のリストを提供する必要がなくなり、AWS Organization を通じてそれらを自動的に管理できるようになりました。AWS Organizations を使用しない場合は、代わりにスポークアカウント ID のリストを指定でき、Instance Scheduler がスケジューリングロールを管理します。

v1.5.0 以降に更新する場合は、以下を行う必要があります。

1. 次のパラメーターを更新しながら、通常の更新手順に従ってハブスタックのテンプレートを更新します。
   a. このソリューションの一意の名前空間を選択します。
   b. [Use AWS Organizations] で、今後スポークアカウントの登録を管理するかどうかを選択します。
      i. [Yes] を選択した場合は、組織 ID / リモートアカウント ID を AWS Organization の ID に置き換えます。
      ii. [No] を選択した場合は、OrganizationID/RemoteAccountIDs をスポークアカウントのアカウント ID のカンマ区切りリストに置き換えます。
2. 次のパラメーターを更新しながら、通常の更新手順に従ってすべてのリモートスタックを更新します。
   a. Namespace - ハブアカウントに選択したものと同じ。
   b. AWS Organizations を使用する - ハブアカウントと同じ。
   c. Hub Account ID - ハブアカウントのアカウント ID (以前と同じのため変更なし)。

v3.0.0

v3.0.0 EC2 Auto Scaling グループのサポートを追加し、ソリューションのコア Lambda 関数を個別の関数に分割し、各関数に専用の責任を割り当てることで、各関数のセキュリティ分離を強化しま

す。このリリースでは、スケジューリングオペレーションに関するインサイトを向上させるために、「SchedulingDecision」ログを含めるようにスケジューリングログの動作も更新されています。

v3.0.0 には、以前のバージョンと比較して以下の下位互換性のない変更点が含まれています。

- 1.5.x の「CloudWatch メトリクス」機能は Operational Insights ダッシュボードに置き換えられました。
- CloudWatch のスケジュールごとのメトリクスは、Schedule/Service/MetricName から Schedule/Service/SchedulingInterval/MetricName に移動されました。
- 既存のメトリクスはすべて残りますが、新しいメトリクスが新しい名前空間の下にまとめられ、ソリューションダッシュボードで使用できるようになります。
- EC2 DB インスタンスの暗号化された EBS ボリュームで使用する KMS キー ARN を、対応するアカウントのハブ/スポーク CloudFormation スタックに提供することが必須になりました (詳細については、「暗号化された EC2 EBS ボリューム」を参照)。
  - 暗号化された EBS ボリュームで EC2 をスケジュールする場合は、使用されている KMS キー ARN をハブ/スポークスタックパラメータにコピーする必要があります。
- スケジュールされたサービスの CloudFormation パラメータは、サポートされているサービスごとに個々のパラメータに分割されています。
  - すべてのサービスはデフォルトで有効になり、個別に無効にできます。
- Instance Scheduler 3.0 には、古いバージョンの Instance Scheduler CLI との下位互換性がありません。
  - CLI コマンドを引き続き使用するには、Instance Scheduler CLI の最新バージョンに更新する必要があります。

上記に加えて、メンテナンスウィンドウテーブルのスキーマが更新されており、更新の一環として置き換えられます。これにより、v3.x への更新後最初の数分間、EC2 メンテナンスウィンドウの追跡がリセットされます。まれに、現在メンテナンスウィンドウ内のインスタンスが更新直後に通常より早く停止することがあります。このデータが再生成された後、スケジューリングオペレーションは通常どおり続行されます。

v3.1.0

v3.1.0 は、AWS タグ付けイベントを使用してリソースがスケジューリング用にタグ付けされるタイミングを追跡するように、ソリューションのコアインフラストラクチャをリファクタリングします。組織のアクセス許可により、これらのタグ付けイベントをメンバーアカウントから中央ハブアカウントに送信できることを確認してください。

v3.1.0 以降に更新した場合

- スポークアカウントは、ハブアカウントとは独立してスケジュールされたリージョンを宣言するようになります。各スポークスタックは、そのアカウントでスケジュールするリージョンを、リージョン (複数可) パラメータを使用して指定する必要があります。
- 合計アカウント数が 40 を超えるデプロイでは、AWS Organizations モードが必要になりました。40 を超えるアカウントがあり、Organizations モードを使用していない場合は、更新中に有効にする必要があります。
- スケジュールする AWS License Manager で EC2 インスタンスを管理している場合は、License Manager 構成 ARN をハブ/スポーク CloudFormation スタックの License Manager 構成 ARN パラメータに追加します。詳細については、「EC2 License Manager」を参照してください。
- このソリューションでは、スケジューリング用にタグ付けされてスケジューラによって管理されていることを示すと、IS-ManagedBy タグがリソースに自動的に適用されます。
- (v3.2.0 で復元) スケジュールされたインスタンスのサイズ変更 (スケジュールで `period-name@size` を定義) は v3.1.0 で一時的に削除されましたが、v3.2.0 以降で再実装されています。「インスタンスタイプ」を参照してください。
- SSM パラメータ (ハブスタックのアカウントパラメータに渡される `{param: ssm-param-name}`) を使用したメンバーアカウントの一覧表示はサポートされなくなりました。すべての信頼されたアカウントは、デプロイ時にハブスタックに渡される必要があります
- Instance Scheduler では、スケジューリング中にリソースに最大 6 つの一意のタグが必要です。組織のタグ付け戦略の残りの部分と組み合わせる場合は、リソースに十分なタグ付け容量があることを確認してください。
- スケジュールごとのメトリクスが CloudWatch から削除されました。
- ソリューションログは、個別の管理ロググループとスケジューリングロググループに再パッケージ化され、CloudWatch Log Insights によるクエリ用に最適化されました。詳細については、「ソリューションのモニタリング」を参照してください。
- 開始タグと停止タグは、CloudFormation パラメータを使用して設定できなくなりました。このソリューションでは、スケジューリングアクションを追跡するために、より豊富な情報を含む固定タグ名を使用するようになりました。

> **Important**
>
> Instance Scheduler は、通常のオペレーション中にマネージドリソースに最大 6 つの一意のタグを書き込みます。タグガバナンスポリシー (AWS Config ルール、タグポリシー、自動修復など) が、これらのタグを許可するように設定されていることを確認します。タグの完全

なリストとガバナンスに関する重要な考慮事項については、「情報タグ」を参照してください。

ソリューションをアンインストールする

> **Important**
>
> ソリューションをアンインストールする場合は、このソリューション自体をアンインストールする前に、必ずすべてのカスタムスケジュールスタックをアンインストールしてください。

AWS での Instance Scheduler ソリューションは、AWS マネジメントコンソールから、または AWS コマンドラインインターフェイスを使用してアンインストールできます。このソリューションをアンインストールするには、AWS Cloud Formation のハブスタックと、インストールされているすべてのリモートスタックを削除してください。その後、スケジューリングの目的でインスタンスに適用していたスケジューリングタグをすべて削除できます。

> **Note**
>
> ソリューションのハブスタックで Protect DynamoDB Tables が有効になっている場合、CloudFormation はソリューションの DynamoDB テーブルと KMS キーを削除せずに保持します。これらのリソースを削除する場合は、ハブスタックを削除する前に、このプロパティが [無効] に設定されていることを確認してください。または、ハブスタックが削除された後に、手動で削除することもできます。

AWS マネジメントコンソールの使用

1. AWS CloudFormation コンソールにサインインします。
2. [スタック] ページで、このソリューションのインストールスタックを選択します。
3. [削除] を選択します。

AWS コマンドラインインターフェイスの使用

AWS コマンドラインインターフェイス (AWS CLI) が環境で使用可能かどうかを判断します。インストール手順については、「AWS CLI ユーザーガイド」の「AWS コマンドラインインターフェイスとは」を参照してください。AWS CLI が使用可能なことを確認したら、次のコマンドを実行します。

```
$ aws cloudformation delete-stack --stack-name
     <installation-stack-name>
```

デベロッパーガイド

このセクションでは、ソリューションのソースコードを提示し、ここに追加されるセクションの一覧と、各サブトピックへのリンクを示します。

ソースコード

GitHub リポジトリにアクセスして、このソリューションのソースファイルをダウンロードし、カスタマイズを他のユーザーと共有できます。

AWS での Instance Scheduler テンプレートは AWS CDK を使用して生成されます。詳細については、README.md ファイルのリンク先を参照してください。

参照資料

このセクションには、データ収集、関連リソースへのポインタ、このソリューションに貢献したビルダーのリストに関する情報が含まれています。

データ収集

このソリューションは、このソリューションの使用に関するオペレーションメトリクスを AWS (「データ」) に送信します。AWS ではこのデータを使用して、ユーザーがこのソリューション、関連サービスおよび製品をどのように使用しているかをよりよく理解し、提供するサービスや製品の改善に役立てます。AWS によるこのデータの収集には、AWS プライバシー通知が適用されます。

関連リソース

Resource Scheduler は AWS での Instance Scheduler と似ていますが、その実装は次の点で異なります。

AWS での Instance Scheduler では Lambda 関数を使用して、その設定に保存されているスケジュールを頻繁に評価し、インスタンスが望ましい状態にあるかどうかを確認します。Resource Scheduler のクイックセットアップは、開始と停止の時間を使用して、SSM ランブックを使用して開始と停止のアクションを実行します。これは、現在の時刻が開始時刻と等しいとき、または現在の時刻が開始時刻を過ぎているときに 1 回発生します。

AWS での Instance Scheduler では、現在 EC2、RDS、Aurora クラスターのスケジューリングが可能です。Resource Scheduler は EC2 インスタンスをスケジュールまたは起動および停止するだけです。

Resource Scheduler を使用して EC2 インスタンスを識別し、特定の時間に起動/停止します。

インスタンスを起動 / 停止するためにアカウントを定期的にスキャンする必要がある場合は、AWS での Instance Scheduler を使用してください。

この表は、シナリオに基づいてどのソリューションが良いかを示しています。

シナリオ	Resource Scheduler	AWS での Instance Scheduler
Amazon Neptune インスタンスをスケジュールする	いいえ	はい

シナリオ	Resource Scheduler	AWS での Instance Scheduler
Amazon DocumentDB インスタンスをスケジュールする	いいえ	はい
Auto Scaling グループのインスタンスをスケジュールする	いいえ	はい
EC2 のインスタンスをスケジュールする	可能	はい
RDS のインスタンスをスケジュールする	いいえ	はい
Aurora のクラスターをスケジュールする	いいえ	はい
単一のアカウント (ハブアカウント) でスケジュールを管理する	いいえ	はい
個々のアカウントでスケジュールを管理する	はい	いいえ
カレンダー統合を変更する	はい	いいえ
開始と停止のアクションのみ	はい	いいえ
インスタンスを定期的にモニタリングして、インスタンスの現在の状態に基づいて開始と停止をする	いいえ	はい

寄稿者

- Arie Leeuwesteijn
- Mahmoud ElZayet
- Ruald Andreae

- Nikhil Reddy
- Caleb Pearson
- Jason DiDomenico
- Max Granat
- Pratyush Das
- Amanda Jones
- Kevin Hargita
- Beomseok Lee
- Abe Wubshet

リビジョン

公開日: 2020 年 10 月

ソフトウェアの主な変更点と更新点を確認するには、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。この改訂履歴には、各バージョンの改良点と修正点が明確に記録されています。

注意

お客様は、本書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤー、またはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または黙示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は AWS 契約によって規定されています。また、本文書は、AWS とお客様との間の契約に属するものではなく、また、当該契約が本文書によって修正されることもありません。

AWS での Instance Scheduler は、Apache ライセンスバージョン 2.0 の条件に基づいてライセンスされています。