AWS での Instance Scheduler

実装ガイド

2018 年 2 月

最終更新日: 2023 年 7 月

(改訂を参照)

目次

ソリューションの概要

[AWS のインスタンスの起動と停止の自動化]

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) インスタンスの開始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開始することで、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 ソリューションを使用して、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（毎週の使用率は 168 時間から 50 時間に減少）。

AWS での Instance Scheduler ソリューションは、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にわたる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 インスタンスに休止 (ハイバネーション) 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、ご使用の環境に AWS での Instance Scheduler ソリューションを実装したい 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。

行いたいこと	参照先
このソリューションを実行に必要なコストが知りたい。 (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、1 か月あたり 4.10 USD 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が知りたい。	AWS Well-Architected のセキュリティセキュリティ
スケジュールを設定したい。	アーキテクチャの詳細
どの AWS リージョンがこのソリューションをサポートしているか知りたい。	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 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したい。	CloudFormation テンプレート

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開始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、「クロスアカウントのインスタンススケジューリング」セクションを参照してください。

自動タグ付け

AWS での Instance Scheduler ソリューションでは、開始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。この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。詳細については、「自動タグ付け」を参照してください。

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。この 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

AWS での Instance Scheduler ソリューションには、Infrastructure as Code を使用してスケジュールを管理できる AWS CloudFormation のカスタムリソースが用意されています。詳細については、「Infrastructure as Code (IaC) を使用したスケジュールの管理」セクションを参照してください。

SSM 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 ソリューションは、Amazon EC2 インスタンスと同じリージョンで定義された SSM 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い、それらのインスタンスを開始および停止することができます。詳細については、「SSM メンテナンスウィンドウフィールド」を参照してください。

AWS Service Catalog AppRegistry と AWS Systems Manager Application Manager との統合

このソリューションには、CloudFormation テンプレートと基盤となるリソースを、AWS Service Catalog AppRegistry と AWS Systems Manager Application Manager の両方にアプリケーションとして登録するための Service Catalog AppRegistry リソースが含まれています。この統合により、このソリューションのリソースを一元管理できます。

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 時間に減少) 。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 Amazon EC2 と Amazon RDS のリソース。

通常の業務時間

平日の 9 時 ～ 17 時 (午前 9 時 ～ 午後 5 時)

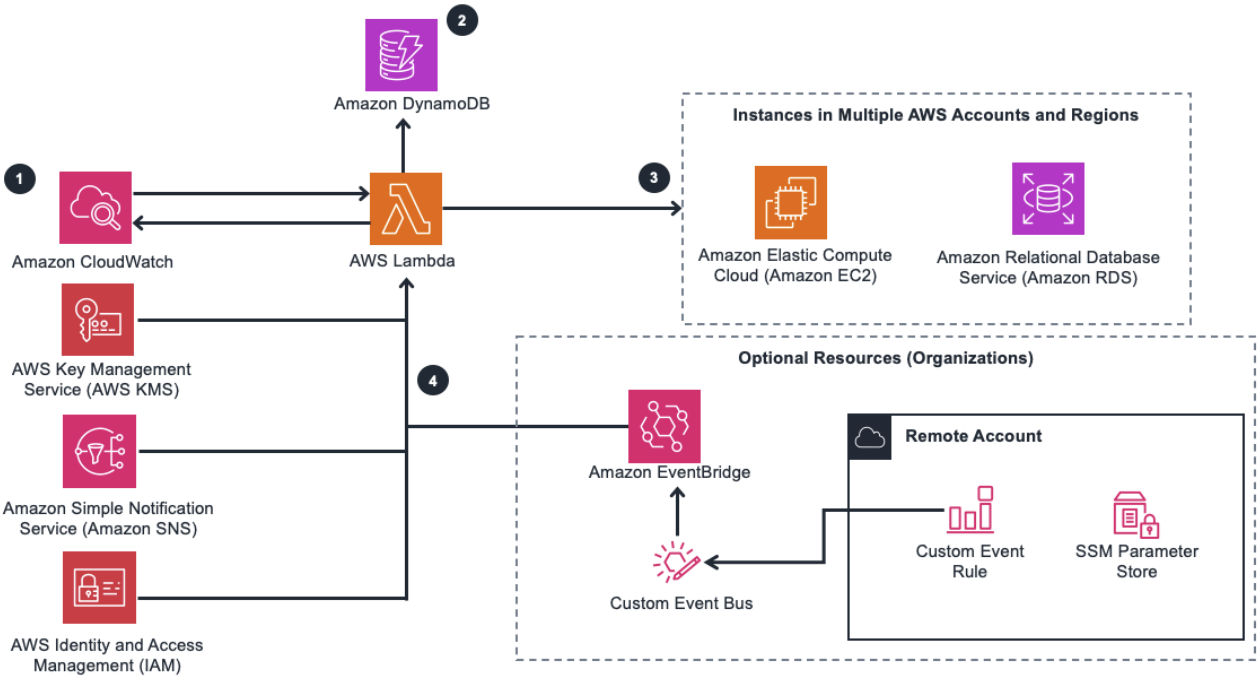
AWS 用語の一般的なリファレンスについては、AWS 一般リファレンスの「AWS 用語集」を参照してください。

アーキテクチャの概要

このセクションでは、このソリューションでデプロイされるコンポーネントのリファレンス実装アーキテクチャ図を示します。

アーキテクチャ図

このソリューションをデフォルトのパラメータを使用してデプロイすると、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 での Instance Scheduler ソリューションのアーキテクチャ

AWS での Instance Scheduler ソリューションは、ユーザーが CloudFormation パラメータの「Use AWS Organizations」を選択すると、Amazon EventBridge のイベントバスをデプロイします。

Instance Scheduler のデプロイオプションとして、Amazon EventBridge の図には、表示されているリソースに加えてデプロイされたイベントバスのみが表示されます。

CloudFormation テンプレートを使用してデプロイされたこのソリューションのコンポーネントの大まかなプロセスフローは次のとおりです。

1. CloudFormation テンプレートは、独自で定義された間隔で Amazon CloudWatch イベントをセットアップします。このイベントは、AWS Lambda の Instance Scheduler 関数を呼び出します。ユーザーは、セットアップ中に AWS のリージョンとアカウントを定義するだけでなく、AWS での Instance Scheduler ソリューションがスケジュールを該当する Amazon EC2 および Amazon RDS のインスタンスとクラスターに関連付けるために使用する**カスタムタグ**も指定します。
2. これらの値は Amazon DynamoDB に保存され、Lambda 関数が実行されるたびに取得されます。その後、カスタムタグを該当するインスタンスに適用できます。
3. Instance Scheduler の初期設定時に、該当する Amazon EC2 と Amazon RDS のインスタンスを識別するためにタグ**キー**を作成します。スケジュールを作成する時に、タグ**キー**名が、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。

   例えば、ユーザーはこのソリューションのデフォルトタグ名 (タグキー) `Schedule` を使用して、`uk-office-hours` というスケジュールを作成します。`uk-office-hours` スケジュールを使用するインスタンスを特定するには、対象インスタンスに、キーが `Schedule`、値が `uk-office-hours` のタグを追加します。
4. **(オプション)** CloudFormation パラメータの Using AWS Organizations を **Yes** にして選択して有効な組織 ID を指定すると、リモートアカウントの CloudWatch Events からのイベントを受信する追加リソースのイベントバスリソースが作成されます。リモートアカウントからのイベントは、Amazon DynamoDB のソリューション設定に追加されるアカウント ID を提供します

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

Lambda 関数では、リソースに対するアクセス許可の要件に AWS Identity Access Management (AWS IAM) 、Amazon Simple Notification Service (SNS トピック) と Dynamo DB テーブルの暗号化に AWS Key Management System (AWS KMS) を使用しています。このソリューションの Lambda 関数が実行されるたびに、適切にタグ付けされた各インスタンスの現在の状態が、関連付けられたスケジュールのターゲット状態 (インスタンスタグのスケジュールで 1 つ以上の期間で定義) に対してチェックされ、必要に応じて適切な開始アクションまたは停止アクションが適用されます。

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、停止した Amazon EC2 または Amazon RDS のインスタンスが `Schedule=office-hours` タグが付与されている場合、Amazon DynamoDB から `office-hours` というスケジュールの設定を探します。`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。

Lambda 関数は、スケジュールの名前、そのスケジュールに関連付けられているインスタンスの数、実行中のインスタンスの数も、Amazon CloudWatch オプションのカスタムメトリクスとして記録します。(「Amazon CloudWatch メトリクス」を参照)

> **注記**: Amazon EC2 インスタンスの**停止**は、Amazon EC2 インスタンスの**終了**とは異なることに注意してください。Amazon EC2 インスタンスは、デフォルトでシャットダウン時に停止し、終了しないように設定されていますが、独自に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、必要に応じてインスタンスを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に基づいて設計されました。これにより、ユーザーは信頼性が高く、安全で、効率的で、費用対効果の高いワークロードをクラウド上で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。

オペレーショナルエクセレンス

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、オペレーショナル・エクセレンス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では、メトリクスを Amazon CloudWatch に投入し、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 を使用して、Lambda 関数をトレースします。
- Amazon SNS を使用して、エラーを報告します。

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に IAM ロールを使用します。
- すべてのマルチアカウント通信に IAM ロールを使用します。
- このソリューションで使用されるすべてのロールに最小権限アクセスを適用します。つまり、サービスが正しく機能するために必要な最低限の権限しか含まれません。
- DynamoDB テーブルを含むすべてのデータストレージの保存時に暗号化を用います。

信頼性

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、信頼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、可能な限りサーバーレスの AWS サービス (AWS Lambda や Amazon 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ます。
- データ処理には Lambda 関数を使用します。このソリューションはデータを Amazon 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で保持します。

パフォーマンス効率

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ます。
- このソリューションで使用される AWS のサービス (AWS Lambda や Amazon DynamoDB など) をサポートする任意の AWS リージョンで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自動的にテストされて、毎日デプロイされます。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 AWS 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では、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。

アーキテクチャの詳細

このセクションでは、このソリューションを構成するコンポーネントと AWS のサービス、およびこれらのコンポーネントがどのように連携するのかについてのアーキテクチャの詳細について説明します。

スケジューラ設定テーブル

AWS での Instance Scheduler ソリューションでは、デプロイ時にグローバル設定を含む DynamoDB テーブルを作成します。このソリューションのデプロイ後にこれらのグローバル構成設定を変更するには、CloudFormation スタックを更新する必要があります。**DynamoDB テーブルでこれらの値を変更しないでください。**DynamoDB テーブルでこれらの値を変更すると、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。

グローバル構成設定が格納されているのは、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。Amazon 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、または削除できます。

スケジュール

スケジュールでは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスと Amazon Relational Database Service (Amazon RDS) 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。また、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 ソリューションは、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」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、この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。

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休止 (ハイバネーション) を使用できます。`hibernate` フィールドを `true` に設定する場合は、Amazon EC2 インスタンスに休止がサポートされている Amazon Machine Image (AMI) を使用するか、休止をサポートするように AMI を設定する必要があります。詳細については、***Amazon Elastic Compute Cloud Linux インスタンス用ユーザーガイド***の「オンデマンドの Linux もしくは インスタンスを休止状態にする」を参照してください。インスタンスの休止では、インスタンスメモリ (RAM) から Amazon Elastic Block Store (Amazon EBS) ルートボリュームにコンテンツを保存します。このフィールドが `true` に設定されている場合は、このソリューションがインスタンスを停止すると、インスタンスが休止状態になります。

インスタンスの休止を使用するようにこのソリューションを設定したが、インスタンスの休止が設定がされていない、または休止の前提条件を満たしていない場合、このソリューションは警告をログに記録し、インスタンスは休止ではなく停止します。詳細については、***Amazon Elastic Compute Cloud Linux インスタンス用ユーザーガイド***の「オンデマンドまたはリザーブドの Linux もしくは インスタンスを休止状態にする」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、このソリューションはそのインスタンスを停止します。このフィールドを `true` に設定している状態で実行期間中に手動でインスタンスを停止すると、このソリューションはそのインスタンスを再起動します。

retain_running フィールド

`retain_running` フィールドでは、インスタンスが期間の開始前に手動で起動されていた場合に、実行期間の終了時にそのインスタンスを停止することを防ぎます。例えば、午前 9 時から午後 5 時までの期間のインスタンスが午前 9 時より前に手動で起動された場合、このソリューションは午後 5 時にそのインスタンスを停止しません。

SSM メンテナンスウィンドウフィールド (EC2 インスタンスにのみ適用)

`ssm-maintenance-window` フィールドを使用すると、AWS Systems Manager メンテナンスウィンドウで指定されている時間帯を実行期間としてスケジュールに自動的に追加できます。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、このソリューションはメンテナンス時間が開始される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の実行が指定されていない場合、およびメンテナンスイベントが完了した場合に、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、次回予定されている AWS Lambda の実行時に変更が反映されます。例えば、AWS Lambda のスケジューラーの実行間隔を 5 分に設定した場合、メンテナンスウィンドウの変更は 1 ～ 5 分の間のどの時点でも AWS Lambda によってのみ取得されます。

このソリューションの初期設定時に指定した AWS Lambda の実行頻度によって、メンテナンスウィンドウのどのくらい前にインスタンスが起動するのか決まります。CloudFormation パラメータの

Frequency を 10 分以下に設定した場合は、このソリューションはメンテナンスウィンドウの 10 分前にインスタンスを起動します。Frequency を 10 分以上に設定すると、スケジューラは指定した実行頻度と同じ分数でインスタンスを起動します。例えば、Frequency を 30 分に設定した場合、スケジューラはメンテナンスウィンドウの 30 分前にインスタンスを起動します。

このソリューションのスタックにある CloudFormation パラメータの **Enable SSM Maintenance windows** を `Yes` に設定してスタックを更新し、このソリューションが Lambda 関数の実行時に使用される DynamoDB テーブルへの SSM メンテナンスウィンドウの読み込みを開始するように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 メンテナンスウィンドウ」を参照してください。

override_status フィールド

スケジュールには、このソリューションの開始アクションと停止アクションを一時的に上書きできる `override_status` フィールドも含まれています。このフィールドを `running` に設定した場合、このソリューションは該当するインスタンスの起動は行いますが、停止は行いません。そのインスタンスは、手動で停止するまで実行されます。フィールドを `stopped` に設定すると、このソリューションは該当するインスタンスの停止は行いますが、起動は行いません。そのインスタンスは、手動で起動するまで実行されません。

`override_status` フィールドを `running` に設定し、`enforced` フィールドを使用して実行期間外に手動でインスタンスが起動しないようすると、このソリューションはそのインスタンスを停止することに注意してください。`override_status` フィールドを `stopped` に設定し、`enforced` フィールドを使用して、実行期間中にインスタンスが手動で停止されないようにすると、このソリューションはインスタンスを再起動します。

インスタンスタイプ

Amazon EC2 インスタンスの場合のみ、スケジュール内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。期間内にインスタンスタイプを指定すると、このソリューションは該当するインスタンスタイプで Amazon EC2 インスタンスを起動します。

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: `weekends@t2.nano`) Amazon EC2 インスタンスと Amazon RDS インスタンスをスケジュールする期間に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期間に指定されたインスタンスタイプと異なる場合、この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が実行中のインスタンスの設定と互換性がある場合、指定されたインスタンスタイプでそのインスタンスを再起動します。詳細については、***Amazon EC2 Linux* インスタンス用ユーザーガイド**の「インスタンスタイプを変更する」を参照してください。

スケジュールの定義

Amazon DynamoDB 内の AWS での Instance Scheduler の設定テーブルには、スケジュール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。このフィールドが `true` に設定されている場合は、スケジューラはインスタンスを停止すると休止状態になります。インスタンスの休止をオンにして、休止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、スケジューラはインスタンスが実行期間外に手動で起動された場合は実行中のインスタンスを停止し、インスタンスが実行期間中に手動で停止され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。この名前は一意である必要があります。
`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: `weekdays@t2.large`)

フィールド	説明
`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に、実行期間の終了時にこのソリューションがインスタンスを停止しないようにする (起動したままにする) かどうかを選択します。
`ssm_maintenance_window`	AWS Systems Manager メンテナンスウィンドウを実行期間として追加するかどうかを選択します。メンテナンスウィンドウの名前を入力します。 **注記:** このフィールドを使用するには、`use_maintenance_window` パラメータも `true` に設定する必要があります。 **注記:** この機能は EC2 インスタンスにのみ適用されます。
`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_maintenance_window`	このフィールドを使用して、Amazon RDS メンテナンスウィンドウを実行期間として Amazon RDS インスタンススケジュールに追加したり、AWS Systems Manager メンテナンスウィンドウを実行期間として Amazon EC2 インスタンススケジュールに追加したりします。詳細については、「Amazon RDS メンテナンスウィンドウ」および「SSM メンテナンスウィンドウフィールド」を参照してください。
`use_metrics`	スケジュールレベルで Amazon CloudWatch メトリクスを有効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 Amazon CloudWatch メトリクス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 1 か月あたり 0.90 USD の料金が発生します。

期間

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 ソリューション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 (begintime) と停止時刻 (endtime)

`begintime` フィールドと `endtime` フィールドは、AWS での Instance Scheduler ソリューション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、このソリューションはその値を使用してその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にすると、隣接する期間をスケジュールしていない限り、インスタンスは毎日の終わりの午後 11 時 59 分に停止します。

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は、このソリューションは、曜日、月の日数、または月のルールを使用してインスタンスを起動および停止します。

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、この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、期間の `begintime` と `endtime` の両方を指定することをお勧めします。時間を指定しない場合は、このソリューションは他の期間で指定された時間を使用して、インスタンスをいつ開始し、いつ停止するのかを決定します。例えば、インスタンスを手動で停止するまで実行したい場合を考えます。ある期間で `endtime` を指定せず `begintime` を午前 9 時に指定していたとしても、AWS での Instance Scheduler ソリューションは、翌日の午前 00:00 にインスタンスを停止します。このソリューションは、その特定の日の別の期間が見つかるまで、インスタンスの起動と停止のどちらを維持するかを評価します。

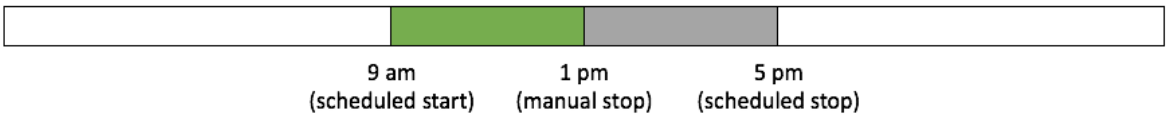
指定した開始時間より前にインスタンスを起動した場合は、そのインスタンスは実行期間の終了まで実行されます。例えば、ユーザーがインスタンスを毎日午前 9 時に開始し、そのインスタンスを午後 5 時に停止する期間を定義できます。

***9* 時から *5* 時でスケジュールされた開始と停止**

ユーザーが午前 5 時にインスタンスを手動で開始した場合は、この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、この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* 時に手動で起動

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。



午後 *5* 時に予定された停止

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` のルールがないとこの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。

weekdays フィールド

`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 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。詳細については、「ソリューションの設定」を参照してください。

monthdays フィールド

`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 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。詳細については、「ソリューションの設定」を参照してください。

months フィールド

`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 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。詳細については、「ステップ 3: スケジュールと期間の設定」を参照してください。

期間の定義

Amazon DynamoDB 内の Instance Scheduler の設定テーブルには、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。

> **重要:** `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、インスタンスを n 番目の月ごと、またはある範囲の n 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` 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 日から 3 日、および 7 日 から 9 日にインスタンスを実行します。 また、月の n 日ごと、またはある範囲の月の n 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 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 日から 15 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 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。

フィールド	説明
`name`	期間を識別するために使用する名前。この名前は一意である必要があります。
`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 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 ソリューション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。

アカウント ID または組織 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、この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開始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、プライマリアカウントで主要なソリューションテンプレート (`instance-scheduler-on-aws`) を起動します。

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-on-aws-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。このソリューションのメインスタックを 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、この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。

組織 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

自動化された起動 / 停止の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ために、AWS Organizations を使用する CloudFormation パラメータを持つプライマリアカウントでこのソリューションのメインテンプレート (`instance-scheduler-on-aws`) を起動していますか? その回答が「はい」で、かつ CloudFormation パラメーターの Provide Organization Id OR List of Remote Account Ids に組織 ID を入力したとします。メインソリューションが完全にインストールされたら、メイン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されると、メインソリューションは、メインアカウントにこれ以上変更を加えられることのないアカウント ID で更新されます。

AWS Systems Manager Parameter Store によるアカウント ID の管理

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、AWS Systems Manager Parameter Store 内のパラメータの名前になります。

この機能を使用するには、AWS での Instance Scheduler ソリューションのハブスタックを AWS Systems Manager Parameter Store と同じアカウントで起動する必要があります。

自動タグ付け

AWS での Instance Scheduler ソリューションは、開始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。**Started tags** パラメータと **Stopped tags** パラメータで、タグ名または `tagname=tagvalue` ペアのリストを指定できます。この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}`: AWS での Instance Scheduler ソリューションのスタックの名前
- `{year}`: 年 (4 桁)
- `{month}`: 月 (2 桁)
- `{day}`: 日 (2 桁)
- `{hour}`: 時間 (2 桁、24 時間形式)
- `{minute}`: 分 (2 桁)
- `{timezone}`: タイムゾーン

次の表は、さまざまな入力と結果のタグの例を示しています。

パラメータの入力例	Instance Scheduler のタグ
`ScheduleMessage=Started by scheduler {scheduler}`	`ScheduleMessage=Started by scheduler MyScheduler`
`ScheduleMessage=Started on {year}/{month}/{day}`	`ScheduleMessage=Started on 2017/07/06`
`ScheduleMessage=Started on {year}/{month}/{day} at {hour}:{minute}`	`ScheduleMessage=Started on 2017/07/06 at 09:00`
`ScheduleMessage=Started on {year}/{month}/{day} at {hour}:{minute} {timezone}`	`ScheduleMessage=Started on 2017/07/06 at 09:00 UTC`

Started tags パラメータを使用すると、スケジューラがインスタンスを停止すると、タグが自動的に削除されます。**Stopped tags** パラメータを使用すると、インスタンスの起動時にタグが自動的に削除されます。

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。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」セクション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
Amazon Elastic Compute Cloud	**コア。**ソリューションで EC2 インスタンスの起動と停止に使用されます。インスタンスは、ソリューションで設定される特定のタグのキー/値によって識別されます。
Amazon Relational Database Service	**コア。**ソリューションで RDS インスタンスのステータスを Available (使用可能) または Stopped (停止) 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る特定のタグのキー/値によって識別されます。
Amazon Aurora クラスター	**コア。**ソリューションで Aurora クラスターのステータスを Available (使用可能) または Stopped (停止中) 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る特定のタグのキー/値によって識別されます。
AWS Lambda	**コア。**このソリューションでは、インスタンスをスケジュールするためのロジックと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ロジックをすべて含む 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、状態情報、インスタンスが最後に実行したアクションの設定を格納する DynamoDB テーブルと、スケジューリング用の SSM 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。**ソリューションでは、デバッグログと情報ログを保存します。
AWS Identity Access Management	**コア。**ソリューションでは、IAM を使用して、インスタンスをスケジュールするための権限を取得します。

Amazon Simple Notification Service	**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、エラーが発生した場合にサブスクライブしてトラブルシューティングできるようにします。
AWS Key Management System	**コア。**ソリューションでは KMS キーを作成して、SNS トピックを暗号化します。
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。
AWS EventBridge	**サポート。**ソリューションでは、EventBridge イベントバスを作成して、ユーザーが AWS Organizations 機能の使用を選択した場合にリモートテンプレートからイベントを受信します。

デプロイの計画

このセクションでは、このソリューションをデプロイする前に、コスト、ネットワークセキュリティ、クォータ、およびその他の考慮事項について説明します。

コスト

AWS での Instance Scheduler ソリューションの実行中に使用した AWS サービスのコストは、お客様の負担となります。最新版の時点で、米国東部（バージニア北部）リージョンでデフォルト設定でこのソリューションを実行するためのコストは、AWS Lambda の請求額で **1 か月あたり約 9.90 USD** になります。AWS Lambda の無料利用枠の月額利用クレジットがある場合はこれより安くなります。これは、実行中の Amazon EC2 インスタンスの数とは無関係です。

オプションのカスタム Amazon CloudWatch メトリクスには、**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 1 か月あたり 0.90 USD** の追加料金が発生します。このソリューションでは、デフォルトで DynamoDB テーブルのオンデマンドスケーリングを使用して、十分な読み込みおよび書き込みキャパシティーを提供します。

このソリューションで使用している AWS の各サービスについては、料金のウェブページを参照してください。

AWS での Instance Scheduler ソリューションでは、実行サイクルごとに異なる数の Lambda 関数を実行するように設計されています。例えば、このソリューションが 1 つの AWS リージョンで Amazon EC2 インスタンスと Amazon RDS インスタンスの両方を管理するために 2 つのアカウント (このソリューションがデプロイされるアカウントとクロスアカウント) を使用している場合に、このソリューションでは 5 つの Lambda 関数が実行されるとします。Amazon CloudWatch イベントを処理するプロセスの初期起動時に、選択された頻度 (デフォルト: 5 分) に基づいて呼び出し、各 AWS サービス、AWS アカウント、AWS リージョンで、個別の Lambda 関数の実行 (2 つの AWS アカウント x 2 つの AWS サービス x 1 つの AWS リージョン) によって処理されるとします。

実行あたりのこのソリューションのコストは、このソリューションによってタグ付けおよび管理されるインスタンスの数によって異なります。Amazon EC2 および Amazon RDS インスタンスの数が増えると、Lambda 関数の実行時間も比例して増加することになります。

コスト管理を容易にするために、AWS Cost Explorer を使用して予算を作成することを推奨しています。料金は変更される可能性があります。詳細については、このソリューションで使用する AWS の各サービスの料金表ウェブページを参照してください。

コスト例

次表のコストは、次の前提に基づいています。

1. 米国東部 (バージニア北部) の AWS リージョンにデプロイ
2. Amazon EC2 インスタンスと Amazon RDS インスタンスの両方を管理
3. 追加のアカウントでインスタンスを管理
4. 1 日あたりの実行の合計数: 288 回 (AWS Lambda を 5 分毎に実行するようスケジュール)
5. 各 AWS Lambda の平均実行時間 (想定): 8 秒 (スケジュールされるインスタンス数により異なる)
6. AWS Lambda 用に選択されたメモリ: 128 MB

AWS サービス	ディメンション	コスト [USD]
AWS Lambda	288 実行 / 24 時間 実行あたり 40 秒 (AWS Lambda ごとに 8 秒) (1 回の実行につき 1 秒あたり 0.0000021 USD)	1.45 USD
Amazon CloudWatch メトリクス (有効)	スケジュールごとに 1 か月 スケジュールされたサービス	0.90 USD
Amazon DynamoDB	1,080,000 (書き込み / 月) (100 万のリクエストにつき 1.25 USD)	1.25 USD
Amazon DynamoDB	1,080,000 (読み取り / 月) (100 万回のリクエストにつき 0.5 USD)	0.50 USD
Amazon DynamoDB	< 1 GB (最初の 25 GB は無料)	0.0 USD
合計:		4.10 USD

注記: アカウントとリージョンの数が増えると、ソリューションのコストも増加します。料金は変更される可能性があります。詳細については、このソリューションで使用する AWS の各サービスの料金表ウェブページを参照してください。

ソリューションを効率的に設定するには、次の点を考慮してください。

1. AWS Lambda のコストが最も低いリージョンにソリューションをデプロイします。
2. AWS Lambda のメモリを変更しないでください (どうしても必要な場合を除き、CloudFormation パラメータは **Memory** にしておくこと) 。変更すると、ソリューションのコストが大幅に増加します。
3. 未使用のスケジュールをソリューションの設定から削除します。
4. 1 日あたりの AWS Lambda の実行回数を減らす頻度を選択します。例えば、スケジュールが数時間離れている場合は、頻度 (CloudFormation パラメータの **Frequency**) を 1 時間単位に設定します。このソリューションは、デフォルトで 5 分に設定されています。つまり、AWS

Lambda が 1 日に 288 回実行されるのに対して、1 日に 1 時間の頻度で 24 回実行さることになります。

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場合、セキュリティ上の責任はお客様と AWS の間で共有されます。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、仮想化レイヤー、サービスを運用する施設の物理的なセキュリティなど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が軽減します。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 Key Management System

このソリューションでは、AWS マネージド型のカスタムマスターキー (CMK) 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。

Amazon Identity Access Management

このソリューションによって作成された Lambda 関数には、Amazon EC2 インスタンスと Amazon RDS インスタンスの両方を開始 / 停止、インスタンス属性の変更、インスタンスのタグの更新にアクセス権限が必要です。必要なすべてのアクセス許可は、このソリューションのテンプレートの一部として作成された AWS Lambda サービスロールに対して、このソリューションが提供します。

さらに、AWS Lambda のサービスロールには、SSM get/put パラメータへのアクセス、Amazon CloudWatch ロググループへのアクセス、KMS キーの暗号化/復号化、SNS トピックへのメッセージの発行も用意されています。サービスロールに付与される各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。

サポートしている AWS リージョン

Instance Scheduler は、任意の標準 AWS リージョン、GovCloud AWS リージョン、一部のオプトインリージョンにデプロイできます。デプロイ後は、アカウントの任意のリージョンにあるタグ付きの Amazon EC2 および Amazon RDS のインスタンスに適切な開始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、この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 AWS リージョンのインスタンスにアクションを適用します。

> **重要:** Lambda 関数が 1 つのリージョンで実行されている場合でも、AWS での Instance Scheduler ソリューションのアクションは、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。

この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 AWS 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 AWS 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 AWS 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 AWS リージョンが重複している場合、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。

> **注記:** AWS での Instance Scheduler ソリューションは、すべての標準 AWS リージョンと AWS GovCloud で検証されています。オプトインリージョンの場合、AWS での Instance Scheduler ソリューションでは任意のオプトインリージョン内のインスタンスを対象としてスケジューリングできますが、CloudFormation スタック自体は現在、次のオプトインリージョンでのみデプロイできます。
>
> - ap-south-1
> - ap-east-1
> - ap-southeast-3
> - eu-south-1

- me-south-1
- me-central-1

クォータ

サービスクォータ (制限とも呼ばれます) は、AWS アカウント用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の AWS サービスのクォータ

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割り当てがあることを確認してください。詳細については、「AWS サービスクォータ」を参照してください。

次のリンクを使用して、そのサービスのページに移動してください。ページを切り替えずにドキュメント内のすべての AWS サービスのサービスクォータを表示するには、こちらの PDF にある「Service endpoints and quotas」 (サービスエンドポイントとクォータ) ページの情報を確認してください。

AWS CloudFormation のクォータ

お使いの 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

お使いの AWS アカウントには AWS Lambda があり、同時実行のクォータは 1000 です。他のワークロードが実行されていて AWS Lambda を使用しているアカウントでソリューションを使用する場合は、このクォータを適切な値に設定する必要があります。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。

ソリューションのデプロイ

デプロイに関する考慮事項

自動デプロイを開始する前に、このガイドで説明されているアーキテクチャ、設定、およびその他の考慮事項をよくお読みください。このセクションの手順に従って、AWS アカウントに AWS での Instance Scheduler ソリューションを設定してデプロイします。

デプロイ時間: 約 5 分

部分的な自動化

ユーザーは部分的な自動化を実装することもできます (例: 開始のみのアクションまたは停止のみのアクションの設定)。その一例として、あるチームで異なる時間に (手動で) インスタンスを開始する柔軟性を必要としているが、一日の終わりにインスタンスを停止しなければならない、あるいはその逆の場合が考えられます。これを行う方法の例については、「サンプルスケジュール」を参照してください。

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、Amazon Elastic Compute Cloud (Amazon 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 インスタンスが終了 (停止ではなく) してしまった場合は、再起動できないことに注意してください。

Amazon EC2 インスタンスは、デフォルトでシャットダウン時に停止し、終了しないように設定されていますが、独自にこの動作を変更することもできます。そのため、AWS での Instance Scheduler

ソリューションを使用して制御するインスタンスに停止のシャットダウン動作が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は、インスタンスは終了します。

Amazon RDS

このソリューションは、Amazon RDS 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。CloudFormation テンプレートの **Create RDS Instance Snapshot** パラメータを使用して、このソリューションがインスタンスを停止する前に Amazon RDS 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。スナップショットは Amazon Aurora クラスターでは利用できないことに注意してください。

CloudFormation テンプレートの **Schedule Aurora Clusters** パラメータを使用して、Amazon Aurora クラスターの一部である、または Amazon Aurora データベースを管理する Amazon RDS 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。

Amazon RDS インスタンスの起動と停止の制限の詳細については、*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。

Amazon RDS 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、そのインスタンスの再起動時にパフォーマンスが低下する場合があります。

Amazon RDS のメンテナンスウィンドウ

すべての Amawzon RDS インスタンスには、週ごとのメンテナンスウィンドウがあります。その時間内にシステムの変更が適用されます。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。メンテナンスイベントが完了しても、Amazon RDS はそのインスタンスを停止しないことに注意してください。

このソリューションでは、Amazon RDS インスタンスの優先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。この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、他の実行期間でインスタンスの実行が指定されていない場合、および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最後にインスタンスを停止します。

メンテナンスウィンドウが終了してもメンテナンスイベントが完了しない場合、そのインスタンスはメンテナンスイベントが完了してからスケジュール間隔まで実行されます。Amazon RDS メンテナンスウィンドウの詳細については、*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。

グローバルな構成設定

AWS での Instance Scheduler ソリューションの CloudFormation テンプレートをデプロイすると、グローバルな構成設定が DynamoDB テーブル (`ConfigTable`) に保存されます。これらの設定を変更するには、CloudFormation テンプレートを使用してソリューションスタックをアップデートします。DynamoDB テーブルでこれらの設定を変更しないでください。

パフォーマンス

AWS での Instance Scheduler ソリューションの Lambda 関数が、次の呼び出し前にスケジュールされたインスタンスをすべて処理しない場合は、このソリューションは Amazon CloudWatch Logs にエラーを記録し、Amazon Simple Notification Service (Amazon SNS) 通知をエラーの SNS トピックに送信します。次の呼び出し前にすべてのインスタンスが処理されるようにするため、Lambda 関数が実行されるデフォルトの間隔を変更するか、異なるタグ名でこのソリューションを複数デプロイして起動することができます。

デフォルトの間隔を増やすと、スケジュールの詳細度が低下する可能性があります。例えば、15 分間隔で実行するように設定された Lambda 関数は、15 分ごとにのみ起動アクションと停止アクションを実行します。

多数のインスタンスをスケジュールするには、少なくとも 5 分間隔を使用し、AWS での Instance Scheduler ソリューションの主要な Lambda 関数のメモリサイズを **Memory Size** パラメータを使用して増やすことをお勧めします。

暗号化された Amazon EBS ボリューム

Amazon EC2 インスタンスに暗号化された Amazon Elastic Block Store (Amazon EBS) ボリュームが含まれている場合は、AWS での Instance Scheduler ソリューションに、カスタマーマスターキー (CMK) を使用してインスタンスを起動および停止するアクセス許可を付与する必要があります。インスタンスが配置されているアカウントの Instance Scheduler ロール (`<stackname>-SchedulerRole-<id>`) に `kms:CreateGrant` のアクセス許可を追加します。

ログ記録と通知

AWS での Instance Scheduler ソリューションは、ログ記録に Amazon CloudWatch Logs 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。詳細については、「ソリューションリソース」を参照してください。

警告とエラーメッセージは、ソリューションが作成した SNS トピックにも発行され、サブスクライブされた E メールアドレスにメッセージが送信されます。詳細については、*Amazon SNS* 開発者ガイドの「Amazon SNS とは」を参照してください。SNS 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。

CloudFormation テンプレート

このソリューションでは、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、この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、Lambda 関数、DynamoDB テーブル、Amazon CloudWatch イベント、Amazon 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、このソリューションと関連するすべてのコンポーネントを起動します。デフォルト設定では、Lambda 関数、DynamoDB テーブル、Amazon CloudWatch イベント、Amazon 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。

> **注記:** すでにこのソリューションをデプロイしている場合は、「ソリューションのアップデート」でアップデートの手順を参照してください。

デプロイプロセスの概要

> **重要:** 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 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。このアンケートを通じて収集されたデータは AWS が所有します。データ収集には、AWS プライバシーポリシーが適用されます。
>
> 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 のマッピング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、このソリューションをデプロイします。詳細については、本書の「匿名化されたデータの収集」セクションを参照してください。

このソリューションを起動する前に、コスト、アーキテクチャ、セキュリティなど、このガイドで説明している考慮事項を確認してください。このセクションの手順に従って、このソリューションを設定して AWS アカウントにデプロイします。

デプロイ時間: 約 5 ～ 10 分 (設定は除く)

ステップ 1: Instance Scheduler スタックの起動

- AWS アカウントで CloudFormation テンプレートを起動します。
- 必須パラメータの値を入力します。
- 他のテンプレートパラメータを確認し、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の起動

- AWS アカウントで CloudFormation テンプレートを起動します。
- 必須パラメータ (**Primary account**) の値を入力します。

ステップ 3: スケジュールと期間の設定

- ご自分の環境に合わせてスケジュールと期間を作成します。

ステップ 4: インスタンスへのタグ付け

- 該当リソースにカスタムタグを付けます。

ステップ 1: Instance Scheduler スタック (ハブスタック) の起動

このセクションの手順に従って、このソリューションを設定して AWS アカウントにデプロイします。

デプロイ時間: 約 5 分

1. AWS マネジメントコンソールにサインインして、`instance-scheduler-on-aws.template` CloudFormation テンプレートを起動するボタンを選択します。

2. このテンプレートは、デフォルトで米国東部 (バージニア北部) リージョンで起動されます。別の AWS リージョンでこの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の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、この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 STS 制限」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
Instance Scheduler tag name	`Schedule`	このタグは、自動化されたアクションを受け取るインスタンスを識別するために使われます。また、カスタムの開始/停止パラメータも許可します。 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**: タグ名では、大文字と小文字が区別されます。
Service(s) to schedule	`EC2`	スケジュールするサービス。`EC2`、`RDS`、または `Both` (両方) を選択します。
Schedule Aurora Clusters	`No`	Amazon Aurora クラスターをスケジュールするかどうかを選択します。Amazon Aurora クラスターのスケジューリングをオンにするには、**Service(s)to schedule** パラメータで `RDS` または `Both` を選択する必要があります。
Create RDS instance snapshot	`Yes`	Amazon RDS 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。 **注意:** 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。
Scheduling enabled	`Yes`	スケジュールを一時的にオフにするには、`No` を選択します。
Default time zone	`UTC`	スケジュールのデフォルトタイムゾーン。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。

パラメータ	デフォルト	説明
This account	`Yes`	このアカウントのリソースを選択できるようにするには、タスクで `Yes` を選択します。 **注意：**このパラメータを `No` に設定した場合は、クロスアカウントロールを設定する必要があります。
Frequency	`5`	Lambda 関数が実行される頻度（分単位）。`1`、`2`、`5`、`10`、`15`、`30`、または `60` を選択します。
Memory Size	`128`	このソリューションの主要な Lambda 関数のメモリサイズ。デフォルトサイズを増やして、多数の Amazon EC2 および Amazon RDS のインスタンスをスケジュールします。
Namespace		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個別の識別子を指定します (スペースなし)。(例: 開発者)
Use AWS Organizations	`No`	AWS Organizations を使用してメンバーアカウントの登録を自動化します。
Organization ID/Remote Account IDs		(必須) AWS Organizations を使用している場合は、組織 ID (例: o-xxxxyyy) を指定します。それ以外の場合は、1111111111、2222222222、{param: ssm-param-name} など、スケジュールするメンバーアカウント ID のカンマ区切りリストを指定します。
Region(s)	<入力は任意>	インスタンスがスケジュールされる AWS リージョンのリスト。(例: `us-east-1`、`us-west-1`) **注記:** 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。
Enable CloudWatch Metrics	`No`	すべてのスケジュールで CloudWatch メトリクスを使用してデータを収集するかどうかを選択します。個々のスケジュールを設定するときに、このデフォルト設定を上書きできます。詳細については、「ステップ 3: スケジュールと期間の設定」を参照してください。 **重要:** 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 1 か月あたり 0.90 USD の料金が発生します。

パラメータ	デフォルト	説明
Enable CloudWatch Debug Logs	No	Amazon CloudWatch Logs でデバッグレベルのログ記録を有効にします。
Enable SSM maintenance windows	No	このソリューションで SSM メンテナンスウィンドウをロードして、Amazon EC2 インスタンスのスケジューリングに使用できるようにします。
Log retention days	30	ログの保持期間 (日数)
Started tags	< input InstanceScheduler-LastAction=Started By {scheduler} {year}/{month}/{day} {hour}:{minute}{timezone},>	起動したインスタンスに設定される key=value, key=value, ... 形式のタグキーと値のカンマ区切りリスト。無効にする場合は空白のままにします。
Stopped tags	< input InstanceScheduler-LastAction=Stopped By {scheduler} {year}/{month}/{day} {hour}:{minute}{timezone},>	停止したインスタンスに設定される key=value, key=value, ... 形式のタグキーと値のカンマ区切りリスト。無効にする場合は空白のままにします。

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. **レビュー**ページで、設定を確認します。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チェックボックスを必ずオンにします。
9. [**送信**] を選択してスタックをデプロイします。

   スタックのステータスは、AWS CloudFormation コンソールの**ステータス**列で確認できます。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の起動

> **重要:** 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 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権限を設定します。プライマリ/ハブスタックがハブアカウントに完全に、または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。

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.template` 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) リージョンで起動されます。別の AWS リージョンでこの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の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、このソリューションの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 Organizations オプションが No に設定されている場合は、プライマリ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
Namespace	<入力は任意>	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個別の識別子。ハブスタックと同じ値に設定する必要があります。
Hub Account ID	<入力が必須>	このアカウントでリソースをスケジュールすることを許可する必要がある Instance Scheduler のハブスタックのアカウント ID。
Use AWS Organizations	No	AWS Organizations を使用してメンバー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。

8. [**次へ**] を選択します。
9. **スタックオプションの設定**のページで、[**次へ**] をクリックします。
10. **レビュー**ページで、設定を確認して確定します。テンプレートによって IAM リソースが作成されることを承認するチェックボックスを必ずオンにします。
11. [**スタックの作成**] を選択してスタックをデプロイします。

    スタックのステータスは、AWS CloudFormation コンソールの**ステータス**列で確認できます。約 5 分で CREATE_COMPLETE のステータスが表示されます。

ソリューションの設定

ソリューションがデプロイされたら、ハブスタックでスケジューリングを設定できます。

ステップ 3: スケジュールと期間の設定

> **注記:** Instance Scheduler は、任意のスケジュールの数をサポートできます。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 つ以上の期間を各スケジュールに設定することができます。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 ソリューションには、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 の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 Scheduler CLI の使用

> **重要:** 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、Amazon 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、AWS 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、Amazon 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、AWS CloudFormation が管理する期間を使用しないでください。

Instance Scheduler スタックをデプロイすると、この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 DynamoDB テーブルを作成します。これらのテーブルをリファレンスとして使用することで、独自のカスタム期間とスケジュールを作成することができます。Amazon DynamoDB でスケジュールを作成するには、設定テーブル (`ConfigTable`) でスケジュールをいずれか 1 つ変更するか、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 をインストールして、次に利用可能なコマンドを使用します。

> **注記**: IaC、Amazon DynamoDB、Scheduler 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。

ステップ 4: インスタンスへのタグ付け

CloudFormation テンプレートをデプロイした際に、このソリューションのカスタムタグの名前 (タグキー) を定義しました。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。

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。

> **注記:** 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。aws: をプレフィックスとして使用す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 ボリュームを持つ EC2 インスタンス

EC2 インスタンスに、お客様管理の 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するために、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 インスタンスが起動しない」を参照してください。

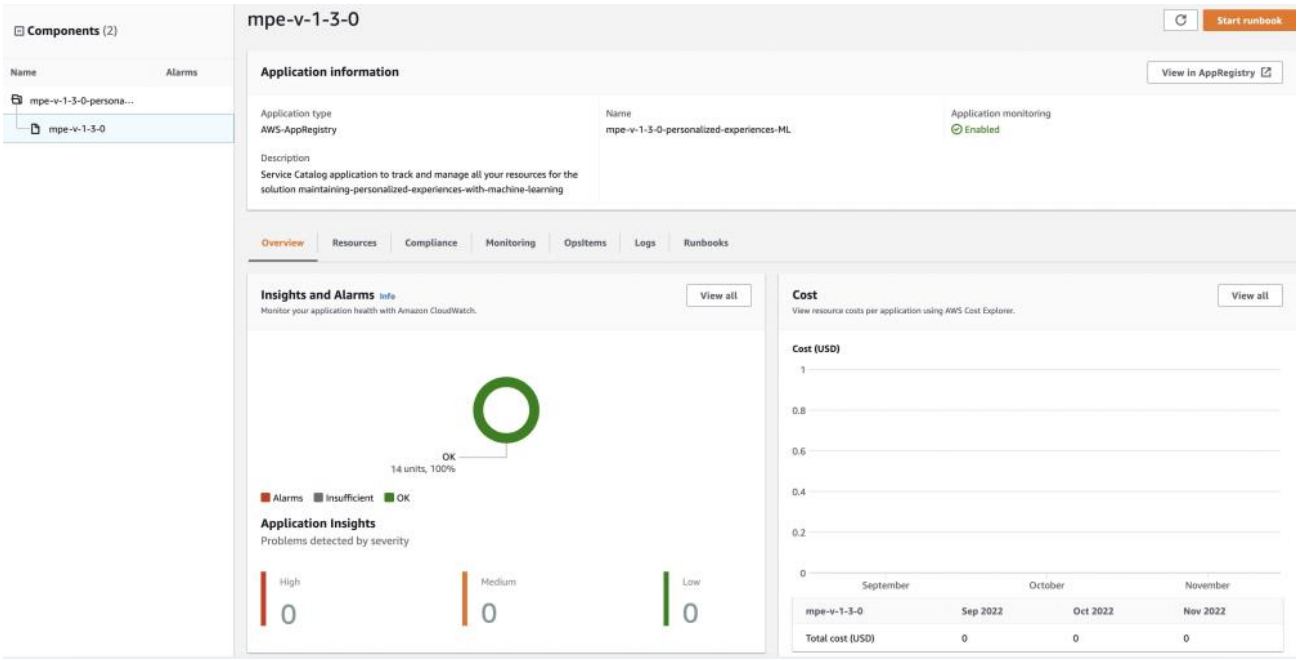
AppRegistry でこのソリューションの監視

AWS での Instance Scheduler ソリューションには、CloudFormation テンプレートと基盤となるリソースを、AWS Service Catalog AppRegistry と AWS Systems Manager Application Manager の両方にアプリケーションとして登録するための Service Catalog AppRegistry リソースが含まれています。

AWS Systems Manager Application Manager は、このソリューションとリソースをアプリケーションレベルで確認できるため、次のようなことが可能になります。

- リソース、スタックや AWS アカウント全体でデプロイされたリソースのコスト、このソリューションに関連するログを一元的に監視します。
- デプロイのステータス、CloudWatch アラーム、リソース設定、運用上の問題など、このソリューションのリソースの運用データをアプリケーションのコンテキストで表示します。

次の図は、Application Manager の AWS での Instance Scheduler ソリューションのスタックのアプリケーションビューの例を示しています。



***Application Manager* による *AWS での Instance Scheduler* ソリューション**

> **注記:** Amazon CloudWatch Application Insights、AWS Cost Explorer、このソリューションに関連するコスト配分タグをアクティブにする必要があります。デフォルトではアクティブになっていません。

Amazon CloudWatch メトリクス

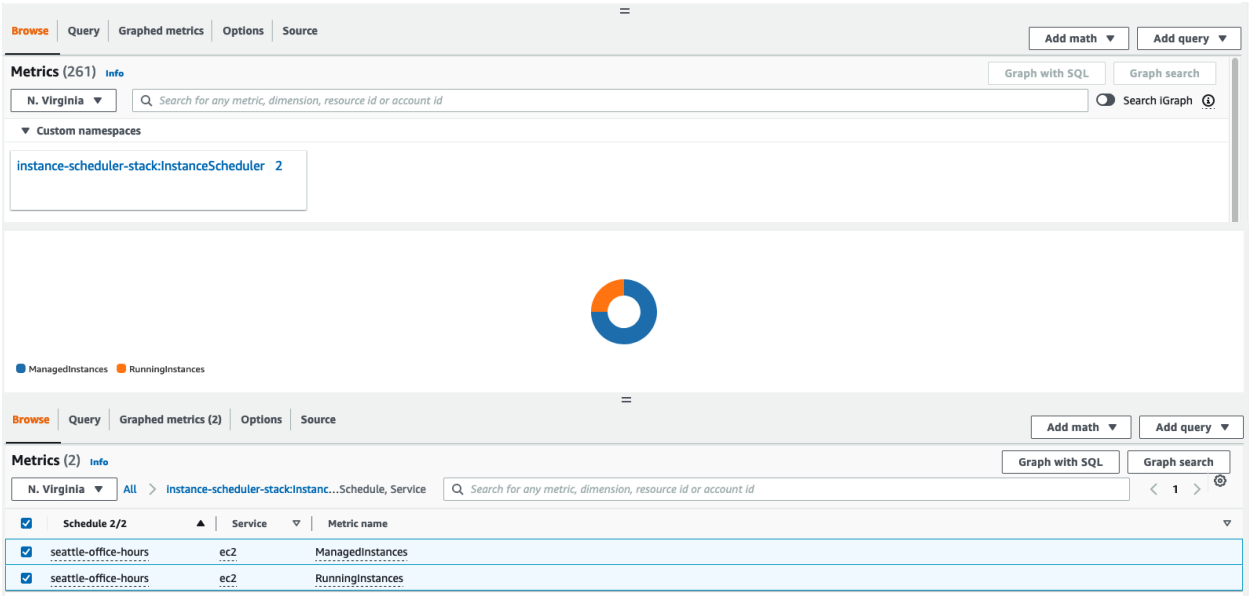
このソリューションには、新しいカスタム Amazon CloudWatch メトリクス (メトリクス名) を作成するオプションがあります。この機能をアクティブにするには、AWS CloudFormation の Enable CloudWatch Metrics パラメータ (`<stackname>:InstanceScheduler`) で Yes を選択してください。

Lambda 関数が実行されるたびに、該当する各インスタンスのメトリクスデータが更新され、適切な開始アクションまたは停止アクションが適用されます。このデータには、スケジュールの名前、そのスケジュールに関連付けられたインスタンスの数、実行中のインスタンスの数が含まれます。

AWS での Instance Scheduler ソリューションのメトリクスの表示

1. AWS マネジメントコンソールで、Amazon CloudWatch コンソールを開きます。
2. **Custom Namespaces** ドロップダウンフィールドで、`<stackname>:InstanceScheduler` を選択します。
3. スケジュールとサービスのディメンションを選択します。
4. ステータスを表示するスケジュールとサービスを選択します。

   次のサンプルのように、ページ下部に選択したスケジュールごとに個別のグラフが表示されます。値 `0` は停止したインスタンスで、値 `1.00` は実行中のインスタンスです。



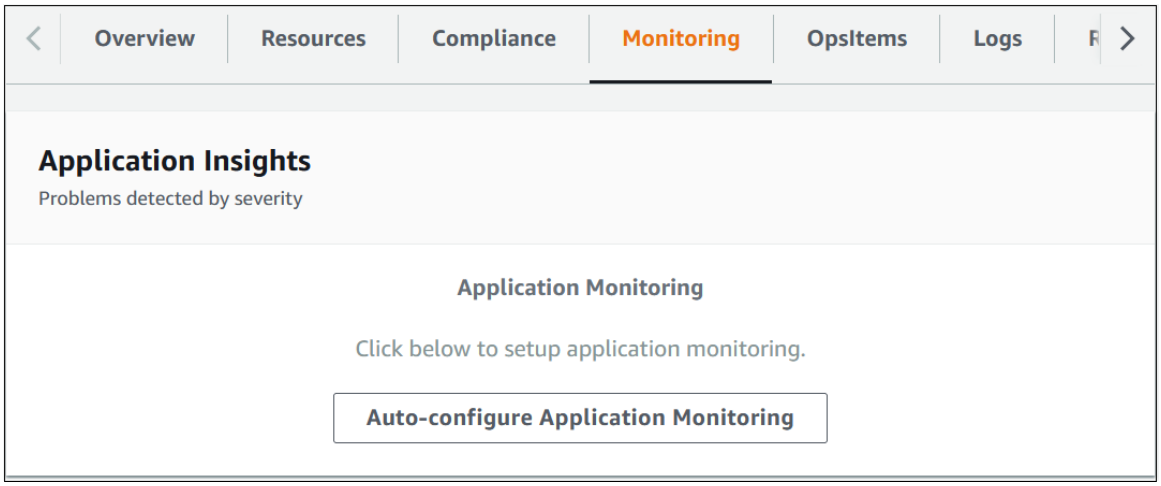
メトリクスのグラフ

Amazon CloudWatch Application Insights のアクティブ化

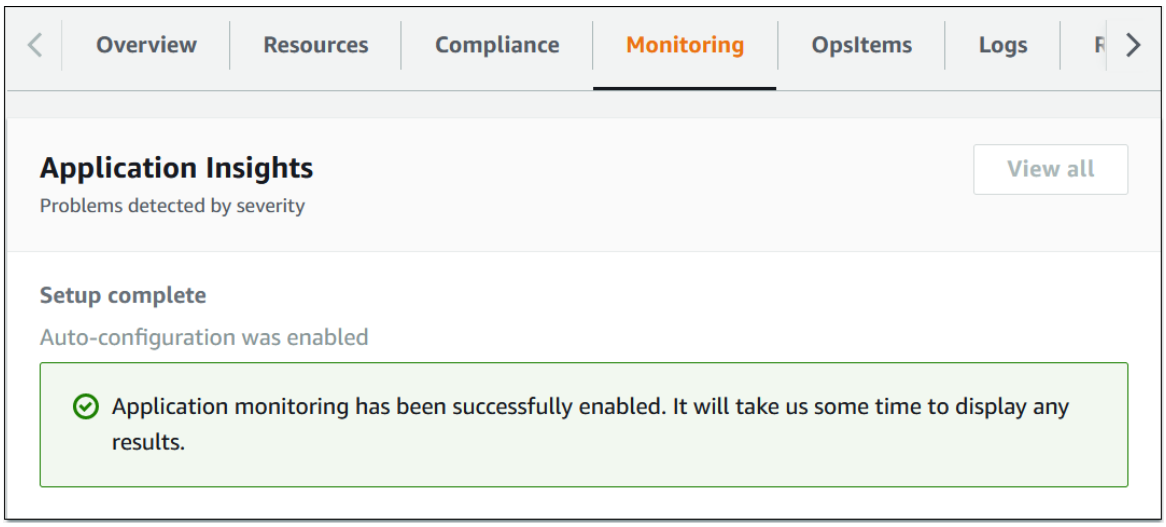
1. Systems Manager コンソールにログインします。
2. ナビゲーションペインで、[**アプリケーションマネージャー**] を選択します。
3. **Applications** で、[**AppRegistry Application**] を選択します。
4. **AppRegistry Application** で、このソリューションのアプリケーション名を検索して選択します。

次に、Application Manager を開くと、**AppRegistry Application** カテゴリにソリューション用の新しいアプリケーションが表示されます。

5. **コンポーネント**ツリーで、アクティブにするアプリケーションスタックを選択します。
6. **モニタリング**タブの**アプリケーションインサイト**で、[**アプリケーションモニタリングを自動設定**] を選択します。



アプリケーションの監視活動がアクティブになり、次のステータスボックスが表示されます。



AWS Cost Explorer のアクティブ化

最初にアクティブにする必要がある AWS Cost Explorer との統合により、アプリケーションおよびアプリケーションコンポーネントに関連するコストの概要を Application Manager コンソールで確認できます。AWS Cost Explorer では、AWS リソースのコストと使用状況を時系列で表示することで、コストを管理できます。次の手順で、このソリューション用に AWS Cost Explorer をアクティブにします。

1. AWS マネジメントコンソールにサインインします。
2. ナビゲーションペインで、[**Cost Explorer**] を選択します。
3. **AWS コスト管理へようこそ**ページで、[**Cost Explorer の起動**] を選択します。

アクティベーションプロセスが完了するまでに最大 24 時間かかる場合があります。アクティブ化されると、Cost Explorer ユーザーインターフェイスを開いて、ソリューションのコストデータをさらに分析できます。

ソリューションに関連するコスト配分タグのアクティブ化

Cost Explorer をアクティブにしたら、このソリューションに関連するコスト配分タグをアクティブにして、このソリューションのコストを確認する必要があります。コスト配分タグは、組織の管理アカウントからのみアクティブ化できます。次の手順で、コスト配分タグをアクティブ化します。

1. AWS Billing and Cost Management コンソールにサインインします。
2. ナビゲーションペインで、[**コスト配分タグ**] を選択します。
3. **コスト配分タグ**ページで、`AppManagerCfnStackKey` タグでフィルタリングし、表示された結果からタグを選択します。
4. [**アクティブ化**] を選択します。

アクティベーションプロセスが完了してタグデータが表示されるまでに最大 24 時間かかる場合があります。

ソリューションのアップデート

このソリューションをすでにデプロイしている場合は、この手順に従って AWS での Instance Scheduler ソリューションの CloudFormation スタックを更新し、このソリューションのフレームワークの最新バージョンを取得します。

1. AWS Cloudformation コンソールにサインインし、既存の **Instance Scheduler** スタック (ハブスタック) を選択して、[**更新**] を選択します。
2. [**既存テンプレートを置き換える**] を選択します。
3. 次の手順でテンプレートを指定します。
    a. [**Amazon S3 URL**] を選択します。

   b. 最新テンプレートのリンクをコピーします。

   c. **Amazon S3 URL** テキストボックスにリンクを貼り付けます。

   d. テンプレートの正しい 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。もう一度、[**次へ**] を選択します。

4. **パラメータ**で、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。パラメータの詳細については、「ステップ 1: Instance Scheduler スタックの起動」を参照してください。
5. [**次へ**] を選択します。
6. **スタックオプションの設定**ページで、[**次へ**] を選択します。
7. **レビュー**ページで、設定を確認します。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チェックボックスを必ずオンにします。
8. [**変更セットのプレビュー**] を選択して、変更を確認します。
9. [**スタックの更新**] を選択して、スタックをデプロイします。

スタックのステータスは、AWS CloudFormation コンソールの**ステータス**列で確認できます。数分後に UPDATE_COMPLETE ステータスが表示されます。

v1.5.x へのアップデート

バージョン 1.5.0 では、AWS での Instance Scheduler ソリューションがクロスアカウントスケジューリングロールを管理する方法が変更され、オプションで AWS Organizations を使用してロールを管理できるようになりました。

AWS Organizations を使用したくない場合は、Instance Scheduler ではフルロールの ARN ではなく、スケジュールを行いたいメンバーアカウントのアカウント ID のみが必要になります。ただし、AWS Organizations を使用する予定がない場合は、更新プロセスが若干異なります。

v1.5.x にアップデートするには、次の操作を行います。

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** パラメータで、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 Organizations の ID に置き換えます。

2. 次のパラメータでスポークアカウントの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. Use AWS Organizations — 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)。

3. ハブスタックのテンプレートをもう一度更新して、ハブアカウントのリモートアカウント ID を設定します。

a. **Use AWS Organizations** で **Yes** を選択した場合は、このステップをすべて省略できます。

b. ハブスタックのテンプレートをもう一度更新します ([**現在のテンプレートを使用**] を選択します)。

c. **組織 ID / リモートアカウント ID** を、各スポークアカウントの `{param: ssm-param-name}` および / またはリモートアカウント ID のカンマ区切りリストに置き換えます。

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する際の既知の問題の解決方法について説明します。

このセクションで説明されていない問題に対して AWS サポートを利用する方法については、「AWS サポート」セクションを参照してください。

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ていない

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。

解決方法

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。(例: Scheduler-ec2-111122223333-<REGION>-20230411)
5. 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、Amazon 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「Config (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場合。
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 パラメーターの **Provide Organization Id OR List of Remote Account IDs** を更新します。

問題：任意のバージョン v1.3.x から v1.5.0 へのソリューションのアップデート

Lambda 関数が機能していません (例: スケジューリングが実行されていない)。

解決方法

1. CloudFormation スタックの更新が完了していることを確認します。
2. CloudFormation コンソールに移動し、このソリューションのスタックを選択します。
3. [リソース] タブを選択します。
4. **リソースの検索**フィルターで [**Main**] を検索します。
5. 物理 ID 列で Lambda 関数を選択します。
6. Lambda コンソールで [設定] を選択します。
7. 環境変数を選択します。
8. 次の環境変数が利用可能であることを確認します。
    a. ACCOUNT
    b. CONFIG_TABLE
    c. DDB_TABLE_NAME
    d. ENABLE_SSM_MAINTENANCE_WINDOWS
    e. ISSUES_TOPIC_ARN
    f. LOG_GROUP
    g. MAINTENANCE_WINDOW_TABLE
    h. METRICS_URL
    i. SCHEDULER_FREQUENCY

j. SEND_METRICS

k. SOLUTION_ID

l. STACK_ID

m. STACK_NAME

n. START_EC2_BATCH_SIZE

o. STATE_TABLE

p. TAG_NAME

q. TRACE

r. USER_AGENT

s. USER_AGENT_EXTRA

t. UUID_KEY

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。

解決方法

Amazon EC2 インスタンスに暗号化された Amazon Elastic Block Store (Amazon EBS) ボリュームが含まれている場合は、AWS での Instance Scheduler ソリューションに、カスタマーマスターキー (CMK) を使用してインスタンスを起動および停止するアクセス許可を付与する必要があります。このアクセス許可は、**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スケジューリングロール**に付与する必要があります。

ハブアカウントの場合、このロールは、*<stackname>*-SchedulerRole-*<id>* になります。

スポークアカウントの場合、このロールは、*<namespace>*-SchedulerRole になります。

特定の KMS キーへのアクセスを許可するには、次のポリシーを適切なスケジューラーロールに適用します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InstanceSchedulerKMSAccess",
            "Effect": "Allow",
            "Action": ["kms:CreateGrant"],
            "Resource": ["keyARN"]
        }
    ]
}
```

AWS サポート

AWS ビジネスサポートまたは AWS エンタープライズサポートをご利用の場合は、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

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 or 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、インターフェー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**] に、問題の詳細を入力します。
3. [**ファイルを添付**] を選択します。
4. AWS サポートがリクエストを処理するため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**のソリューションを確認します。
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。

ソリューションのアンインストール

AWS マネジメントコンソールまたは AWS Command Line Interface を使用して、AWS での Instance Scheduler ソリューションを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。

> **重要:** ソリューションをアンインストールする場合は、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 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をインストールしたスタックを選択します。
3. [**削除**] を選択します。

AWS Command Line Interface の使用

AWS Command Line Interface (AWS CLI) がご自分の環境で使用できるかどうかを確認します。インストール手順については、*AWS CLI* ユーザーガイドの「AWS Command Line Interface とはどのようなものですか」を参照してください。AWS CLI が使用可能になったことを確認したら、次のコマンドを実行します。

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

開発者ガイド

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 ソリューションのテンプレートは、AWS Cloud Development Kit (AWS CDK) を使用して作成されています。詳細については、README.md ファイルを参照してください。

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「設定ファイルと認証情報ファイル」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 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。

> **注意:** 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、「共通の引数」を参照してください。

Scheduler CLI のインストール

1. Scheduler CLI パッケージ ***(instance_scheduler_cli-1.5.1-py3-none-any.whl)*** をダウンロードして、ご自身のコンピューターのディレクトリに配置します。
2. Scheduler 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。

> **注記**: Scheduler CLI には Python 3.8 以降と最新バージョンの pip が必要です。ご自分のローカルマシンにこれら両方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 instance_scheduler_cli-1.5.1-py3-none-any.whl
```

3. インストールが成功したことを確認します。

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

> **注記**: 必要に応じて Scheduler 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。

コマンドの構造

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>
```

共通の引数

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 <stackname>`	Instance Scheduler 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 <regionname>`	Instance Scheduler 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 AWS 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 <profilename>`	コマンドの実行に使用するプロファイルの名前。プロファイル名を指定しない場合、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 CLI から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`	Instance Scheduler の 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` 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`

期間の説明

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` 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` 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 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: 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 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: 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 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>@<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: `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 インススケジュールに追加します。このコマンドを使用するには、`use-maintenance-window` コマンドを使用する必要があります。

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

タイプ: 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 インススケジュールに追加します。

必須: いいえ

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

説明

既存の期間を削除します。

引数

`--name`

該当する期間の名前

タイプ: 文字列

必須: はい

> **重要:** 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

AWS での Instance Scheduler ソリューションの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

AWS での Instance Scheduler ソリューションの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": [
      {
         "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"
      },
      {
         "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。

> **重要:** 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。

> **重要:** 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。

特定のコマンドのサンプル

```
$ 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
```

ソリューションリソース

次のリソースは、AWS での Instance Scheduler ソリューションのスタックの一部として作成されます。

リソース名	タイプ	説明
Main	`AWS::Lambda::Function`	AWS Lambda の Instance Scheduler 関数
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 AWS Lambda の Instance Scheduler 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。

リソース名	タイプ	説明
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::DynamoDB::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::DynamoDB::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。

ログファイル

AWS での Instance Scheduler ソリューションは、デフォルトの AWS Lambda ログファイルを含むロググループと、次のログファイルを含むロググループを作成します。

- `InstanceScheduler-yyyymmdd`: 一般的なスケジューラメッセージをログに記録します。
- `SchedulingOrchestratorHandler-yyyymmdd`: スケジューリングの実行が開始された場合の一般的なオーケストレーション情報を記録します。
- `SchedulerSetupHandler-yyyymmdd`: 設定アクションの出力をログに記録します。
- `Scheduler-<service>-<account>-<region>-yyyymmdd`: 各サービス、アカウント、リージョンのスケジューリングアクティビティを記録します。
- `CliHandler-yyyymmdd`: Admin CLI からのリクエストをログに記録します。
- `Eventbus_request_handler-yyyymmdd`: ソリューションが AWS Organizations にデプロイされている場合に、EventBus リソースへの呼び出しを記録します。

- `CollectConfigurationDescription-yyyymmdd`: 定期的に送信される設定説明のメトリクスデータをログに記録します。

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

AWS での Instance Scheduler ソリューションの AWS CloudFormation のカスタムリソース (`ServiceInstanceSchedule`) を使用して、AWS CloudFormation を通してスケジュールを設定およびできます。カスタムリソースでは、PascalCase 名と Amazon DynamoDB で Instance Scheduler の設定テーブルと同じ構文を使用します (次のテンプレートを参照)。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開始するには、次のサンプルテンプレートをコピーして貼り付け、必要に応じてスケジュールをカスタマイズしてから、AWS CloudFormation を使用して更新したテンプレートをデプロイします。

```
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'
```

このテンプレートをデプロイする場合は、AWS での Instance Scheduler ソリューションのデプロイ用に 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 AWS CloudFormation 内で確認できます。

> **重要**: Amazon 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、Amazon 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。
>
> AWS での Instance Scheduler ソリューションの主要な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、主要なスタックの DynamoDB テーブルへの依存関係が含まれているためです。

Amazon 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。

サンプルスケジュール

AWS での Instance Scheduler ソリューションを使用すると、Amazon Elastic Compute Cloud (Amazon EC2) および Amazon Relational Database Service (Amazon RDS) 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。

午前 9 時 ～ 午後 5 時までの標準労働時間

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	値
begintime	09:00
endtime	16:59
name	weekdays-9-5
weekdays	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	値
name	london-working-hours
periods	weekdays-9-5
timezone	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 (`Schedule`) 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「リソースのタグ付け」を参照してください。

Scheduler CLI

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
```

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上記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、`ServiceInstanceScheduleServiceTokenARN` を提供する必要があります。この ARN は AWS CloudFormation コンソールで、前述でデプロイした Instance Scheduler スタックの「出力」をクリックすると表示されます。

```
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の停止

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	値
endtime	16:59
name	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	stop-at-5-new-york
periods	stop-at-5
timezone	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は、タグが異なります。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「リソースのタグ付け」を参照してください。

Scheduler CLI

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上記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、`ServiceInstanceScheduleServiceTokenARN` を提供する必要があります。この ARN は AWS CloudFormation コンソールで、前述でデプロイした Instance Scheduler スタックの「出力」をクリックすると表示されます。

```
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	値
begintime	09:00
endtime	23:59
name	mon-start-9am
weekdays	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	tue-thu-full-day
weekdays	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	00:00
endtime	16:59
name	fri-stop-5pm
weekdays	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	mon-9am-fri-5pm
periods	mon-start-9am,tue-thu-full-day,fri-stop-5pm
timezone	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は、タグが異なることに注意してください。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「リソースのタグ付け」を参照してください。

Scheduler CLI

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> --name fri-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上記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、`ServiceInstanceScheduleServiceTokenARN` を提供する必要があります。この ARN は AWS CloudFormation コンソールで、前述でデプロイした Instance Scheduler スタックの「出力」をクリックすると表示されます。

```
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
```

参照資料

このセクションには、このソリューション固有のメトリクスを収集するためのオプション機能、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。

匿名化されたデータの収集

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。有効にすると、定期的に次の情報が収集され、AWS に送信されます。

- **Solution ID** - AWS ソリューションの識別子
- **一意の ID (UUID)** - AWS での Instance Scheduler ソリューションのデプロイごとにランダムに生成された一意の識別子

- **Timestamp** - データ収集タイムスタンプ
- **Scheduling Actions** - Instance Scheduler がインスタンスに対して特定のアクションを実行する頻度と、それらのアクションを実行するのにかかった時間のカウント

  サンプルデータ:

```
duration_seconds: 6.7
actions: [
  {
    action: Started
    instanceType: a1.medium
    instances: 8
    service: ec2
  },
  …
]
```

- **Instance Count** – 各リージョンで処理されたインスタンスとスケジュールの数のカウント

  サンプルデータ:

```
service: [ec2]
regions: [us-east-1]
num_instances: 35
num_schedules: 6
```

- **Deployment Description** – デプロイの規模と範囲の見積もり

  サンプルデータ:

```
services: [ec2, rds]
regions: [us-east-1, us-east-2]
num_accounts: 6
```

- **Configuration Description** – Instance Scheduler の各機能の使用頻度

  サンプルデータ:

```
num_schedules: 12
num_cfn_schedules: 3
default_timezone: UTC
schedule_aurora_clusters: True,
```

```
        create_rds_snapshots: False,
        schedule_interval_minutes: 10,
        memory_size_mb: 128,
        using_organizations: False,
        enable_ec2_ssm_maintenance_windows: True,
        num_started_tags: 1,
        num_stopped_tags: 0,
        schedule_flag_counts: {
          stop_new_instances: 10,
          enforced: 3,
          retain_running: 0,
          hibernate: 0,
          override: 0,
          use_ssm_maintenance_window: 2,
          use_metrics: 0,
          non_default_timezone: 4,
        },
```

- **CLI Usage** – Scheduler CLI の各機能の使用頻度

  サンプルデータ:

  ```
  command_used: describe-schedule-usage
  ```

AWS は、このアンケートで収集されたデータを所有します。データ収集には、AWS プライバシーポリシーが適用されます。この機能を無効にするには、CloudFormation テンプレートを起動する前に、次の手順を実施してください。

1. CloudFormation テンプレートをローカルのハードドライブにダウンロードします。
2. テキストエディタで CloudFormation テンプレートを開きます。
3. CloudFormation テンプレートのマッピングセクションを次のように変更します。

   ```
   "Send": {
    "AnonymousUsage": {
     "Data": "Yes"
   }
   ```

   次のように変更します。

```
"Send": {
 "AnonymousUsage": {
  "Data": "No"
}
```

4. AWS CloudFormation コンソールにサインインします。
5. [スタックの作成] を選択します。
6. スタックの作成ページのテンプレートの指定セクションで、[テンプレートファイルのアップロード] を選択します。
7. **テンプレートファイルのアップロード**で、[**ファイルの選択**] を選択し、ローカルドライブから編集したテンプレートを選択します。
8. [**次へ**] を選択し、このガイドの「ソリューションのデプロイ」セクションの「Instance Scheduler スタックの起動」の手順に従います。

関連リソース

Resource Scheduler は AWS での Instance Scheduler ソリューションと似ていますが、その実装は次の点で異なります。

- AWS での Instance Scheduler ソリューションでは 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 ソリューションでは、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 ソリューションを使用してください。

- この表は、シナリオに基づいてどちら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	Instance Scheduler
EC2 インスタンスのスケジュール	はい	はい
RDS インスタンスのスケジュール	いいえ	はい
Aurora クラスターのスケジュール	いいえ	はい
単一のアカウント (ハブアカウント) でのスケジュール管理	いいえ	はい
個々のアカウントのスケジュール管理	はい	いいえ
カレンダー統合の変更	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監視し、インスタンスの現在の状態に基づいた起動と停止	いいえ	はい

寄稿者

- Arie Leeuwesteijn
- Mahmoud ElZayet
- Ruald Andreae
- Nikhil Reddy
- Caleb Pearson
- Jason DiDomenico
- Max Granat
- Pratyush Das

改訂履歴

日付	変更
2018 年 2 月	初回リリース
2018 年 7 月	begintime と endtime の明確化を追加。スケジュール設定のサンプルを追加。
2018 年 10 月	暗号化された Amazon EBS ボリュームに関する情報を追加。スタック名の長さと Amazon RDS タグの制限に関する説明を追加。
2019 年 5 月	Amazon EC2 インスタンスの休止、Amazon Aurora クラスターの一部である RDS インスタンスのスケジュール、新しいスケジュール CLI の引数、SSM のメンテナンスウィンドウ、パラメータストアの統合、および隣接する実行期間のスケジュールの明確化に関する情報を追加。
2019 年 10 月	AWS での Instance Scheduler ソリューションが検証された AWS リージョンに関する情報を追加。
2020 年 3 月	バグ修正
2020 年 6 月	期間ルールの開始時刻と停止時刻のタイムゾーン情報を明確化。v1.3.2 の更新と変更については、GitHub リポジトリの CHANGELOG.md ファイルを参照してください。
2020 年 9 月	AWS Cloud Development Kit(AWS CDK) 変換とドキュメントの機能強化。v1.3.3 の変更の詳細については、GitHub リポジトリの CHANGELOG.md ファイル を参照してください。
2020 年 10 月	付録 D のテンプレートを更新し、hibernate フィールドのセクションを更新しました。
2021 年 4 月	Amazon EC2 インスタンスのスケジューリング用の SSM メンテナンスウィンドウの機能をアップデート、このソリューションを AWS GovCloud で動作するように設定、およびバグ修正。v1.4.0 の詳細については、GitHub リポジトリの CHANGELOG.md ファイルを参照してください。
2022 年 5 月	マイナーアップデートとバグ修正。v1.4.1 の詳細については、GitHub リポジトリの CHANGELOG.md ファイルを参照してください。
2023 年 1 月	マイナーアップデートとバグ修正。v1.4.2 の詳細については、GitHub リポジトリの CHANGELOG.md ファイルを参照を参照してください。

日付	変更
2023 年 5 月	スケジューリングフラグの override_status の廃止 (Resource Scheduler と Instance Scheduler) により、AWS Organizations 向けの機能が追加され、クロスアカウントスケジューリングが簡略化され、クロスアカウントロールが不要になりました。 v1.5.0 の詳細については、GitHub リポジトリの CHANGELOG.md ファイルを参照を参照してください。
2023 年 7 月	Infrastructure as Code を管理する方法とバージョン 1.5.x への更新方法に関するガイダンスの追加。その他にも、トラブルシューティング、機能、および利点を追加。 v1.5.1 の詳細については、GitHub リポジトリの CHANGELOG.md ファイルを参照してください。

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。このドキュメントは、(a) 情報提供のみを目的としており、(b) AWS の現行製品とプラクティスを表したものであり、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義務や確約を意味するものではありません。AWS の製品やサービスは、明示または暗示を問わず、いかなる保証、表明、条件を伴うことなく「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で行われるいかなる契約の一部でもなく、そのような契約の内容を変更するものでもありません。

AWS での Instance Scheduler ソリューションは、Apache Software Foundation で公開されている Apache ライセンスバージョン 2.0 の条項に基づいてライセンス提供されます。