aws

Implementation Guide

Instance Scheduler on AWS

Instance Scheduler on AWS: Implementation Guide

Table of Contents

Automate starting and stopping AWS instances

Publication date: *October 2020 ([last update](): June 2024)*

The Instance Scheduler on AWS solution automates the starting and stopping of various AWS services including [Amazon Elastic Compute Cloud]() (Amazon EC2) and [Amazon Relational Database Service]() (Amazon RDS) instances.

This solution helps reduce operational costs by stopping resources that are not in use and starting resources when their capacity is needed. For example, a company can use Instance Scheduler on AWS to automatically stop instances outside of business hours every day. If you leave all of your instances running at full utilization, this solution can result in up to 70% cost savings for those instances that are only necessary during regular business hours (weekly utilization reduced from 168 hours to 50 hours).

Instance Scheduler on AWS leverages Amazon Web Services (AWS) resource tags and [AWS Lambda]() to automatically stop and restart instances across multiple AWS Regions and accounts on a customer-defined schedule. This solution also allows you to use hibernation for stopped EC2 instances.

This implementation guide provides an overview of the Instance Scheduler on AWS solution, its reference architecture and components, considerations for planning the deployment, and configuration steps for deploying the solution to the AWS Cloud.

This guide is intended for IT infrastructure architects, administrators, and DevOps professionals who want to implement Instance Scheduler on AWS in their environment.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution. The estimated cost for running this solution in the US East (N. Virginia) Region is USD $13.15 per month.	[Cost]()
Understand the security considerations for this solution.	[AWS Well-Architected Security]()

If you want to . . .	Read . . .
	[Security](#)
Configure schedules.	[Scheduler configuration table](#)
Know which AWS Regions are supported for this solution.	[Supported AWS Regions](#)
View or download the AWS CloudForm ation template included in this solution to automatically deploy the infrastructure resources (the "stack") for this solution.	[AWS CloudFormation templates](#)
Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) (AWS CDK) to deploy the solution.	[GitHub repository](#)

Features and benefits

The Instance Schedule on AWS solution provides the following features:

Cross-account instance scheduling

This solution includes a template that creates the [AWS Identity and Access Management](#) (IAM) roles necessary to start and stop instances in secondary accounts. For more information, refer to the [Cross-account instance scheduling](#) section.

Automated tagging

Instance Scheduler on AWS can automatically add tags to all instances that it starts or stops. The solution also includes macros that allow you to add variable information to the tags. For more information, refer to the [Automated tagging](#) section.

Configure schedules or periods using Scheduler CLI

This solution includes a command line interface (CLI) that provides commands for configuring schedules and periods. The CLI allows customers to estimate cost savings for a given schedule. For more information, refer to the [Scheduler CLI](#).

Manage schedules using Infrastructure as Code (IaC)

This solution provides an AWS CloudFormation Custom Resource that you can use to manage schedules using Infrastructure as Code (IaC). For more information, refer to Manage Schedules Using Infrastructure as Code.

Integration with Systems Manager Maintenance Windows

For Amazon EC2 instances, Instance Scheduler on AWS can integrate with AWS Systems Manager maintenance windows, defined in the same Region as those instances, to start and stop them in accordance with the maintenance window. For more information, refer to the Systems Manager maintenance window field section.

Integration with Service Catalog AppRegistry and Application Manager, a capability of AWS Systems Manager

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both Service Catalog AppRegistry and Application Manager. With this integration, you can centrally manage the solution's resources.

Use cases

Running instances only during working hours

If you leave all of your instances running at full utilization, this solution can result in up to 76% cost savings for those instances that are only necessary during regular business hours (weekly utilization reduced from 168 hours to 40 hours). For more information, see the Sample schedule.

Stopping instances after working hours

If you want to make sure that development instances are off after hours and until needed again, you can use this solution to set an end period without a start period. For more information, see the Sample schedule.

Concepts and definitions

This section describes key concepts and defines terminology specific to this solution:

schedule

Group of one or more periods that an instance is bound by.

period

Running period(s) defined by a start and stop time.

instance

A supported resource that is able to be scheduled. For example, an Amazon EC2 instance or an Amazon RDS cluster Amazon EC2 and Amazon RDS.

regular business hours

9:00 to 17:00 (9 AM – 5 PM) ET on weekdays

For a general reference of AWS terms, see the [AWS Glossary](#).

Cost

You are responsible for the cost of the AWS services used while running Instance Scheduler on AWS. As of the latest revision, the cost for running this solution a small deployment in two accounts and two Regions is approximately **$13.15 per month.** See the following sample cost tables for a more detailed breakdown.

Instance Scheduler on AWS is designed to invoke AWS Lambda functions multiple times per run cycle. For example, if you use the solution to manage both Amazon EC2 and Amazon RDS instances in one Region for two accounts (one account where the solution is deployed and the other account is a cross account), the solution runs five Lambda function invocations:

- One to handle the initial Orchestration Request from Event Bridge, which is invoked based on the selected frequency (default: five minutes).
- An additional Lambda invocation for each service, account, and Region.
- If [Auto Scaling Group scheduling](#) is enabled, another orchestration invocation runs for all accounts/regions every hour.

Custom Operational Metrics add additional costs based on the number of schedules and instance types (such as m2.medium, t3.large) being scheduled by the solution. If you don't want to track these metrics, you can turn off this feature to save costs. Refer to [Operational insights dashboard](#) for more details on these metrics and their associated costs.

This solution uses on-demand scaling for its Amazon DynamoDB tables to provide sufficient read and write capacity.

See the pricing webpage for each AWS service in this solution.

The cost of the solution per run depends on the number of instances being tagged and managed by the solution. As the number of EC2 and RDS DB instances increases, the Lambda run time also increases proportionately.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change.

> ⓘ **Note**
>
> For cost optimization purposes, Instance Scheduler groups all Amazon RDS-related services into a single invocation. So even if you enable Amazon RDS, Amazon Aurora, Amazon Neptune, and Amazon DocDB scheduling; this will still be considered just "RDS" for the purpose of cost calculations.

Pricing examples (monthly)

Small deployment

This pricing example is based on these assumptions:

- Two accounts, two Regions, scheduling all possible services
- 3 schedules in active use
- 20 instances of 3 different sizes
- Scheduling interval: 5 minutes
- Lambda function size: 128 MB
- Average Lambda function runtime: 8 seconds

AWS service	Dimensions	Monthly Cost [USD]
AWS Lambda	288 + 24 scheduling runs per day	~$1.50

AWS service	Dimensions	Monthly Cost [USD]
	1+8 Lambda functions per run 8 second average Lambda runtime ($0.0000021/second) ($0.0000002/Lambda function call)	
Operational Metrics (optional)	CloudWatch dashboard ($3/month) 3 per-instance-type metrics ($0.90/month) 3 per-schedule metrics * 2 services ($0.60/month) ~80,000 PutMetric calls/month ($0.01/1000)	~$10.00
Amazon DynamoDB	~75,000 WRU/month ($1.25 per million) ~100,000 RRU/month ($0.5 per million) negligible storage costs (< $0.01)	~$0.15
AWS KMS	1 AWS KMS key ($1/month) ~140,000 API requests/month ($0.30/10000)	~$1.50

AWS service	Dimensions	Monthly Cost [USD]
Total:		~$13.15

Medium deployment

This pricing example is based on these assumptions:

- 50 accounts, 4 Regions, scheduling all supported services
- 10 schedules in active use
- 200 instances of 10 different sizes
- Scheduling interval: 5 minutes
- Lambda function size: 128 MB
- Average Lambda runtime: 8 seconds
- 5 EC2 Maintenance Windows

AWS service	Dimensions	Monthly Cost [USD]
AWS Lambda	288 + 24 scheduling runs per day 1+400 Lambda functions per run 8 second average Lambda runtime ($0.0000021/second) ($0.0000002/Lambda function call)	~$64.00
Operational Metrics (optional)	CloudWatch dashboard ($3/ month) 10 per-instance-type metrics	~$60.00

AWS service	Dimensions	Monthly Cost [USD]
	($0.90/month) 10 per-schedule metrics * 2 services ($0.60/month) ~3.5m PutMetric calls/month ($0.01/1000)	
Amazon DynamoDB	~7m WRU/month ($1.25 per million) ~8m RRU/month ($0.5 per million) Storage costs (<$0.01)	~$12.00
AWS KMS	1 AWS KMS key ($1/month) ~7m API requests/month ($0.30/10000)	~$22.00
Total:		~$158.00

Large deployment

This pricing example is based on these assumptions:

- 120 accounts, 6 Regions, scheduling both Amazon EC2 and Amazon RDS

- 100 schedules in active use

- 2000 instances of 50 different sizes

- 100 EC2 Maintenance Windows

- Scheduling interval: 5 minutes

- Lambda function size: 128 MB

- Average Lambda function runtime: 8 seconds

AWS service	Dimensions	Monthly Cost [USD]
AWS Lambda	288 + 24 scheduling runs per day 1+1440 Lambda functions per run 8 second average Lambda function runtime ($0.0000021/second) ($0.0000002/Lambda call)	~$230.00
Operational Metrics (optional)	CloudWatch dashboard ($3/month) 50 per-instance-type metrics ($0.90/month) 100 per-schedule metrics * 2 services ($0.60/month) ~3.5m PutMetric calls/month ($0.01/1000)	~$300.00
Amazon DynamoDB	~26m WRU/month ($1.25 per million) ~26m RRU/month ($0.5 per million) Storage costs (<$0.01)	~$40.00
AWS KMS	1 KMS key ($1/month) ~25m API requests/month ($0.30/10000)	~$80.00

AWS service	Dimensions	Monthly Cost [USD]
Total:		~$650.00

To efficiently configured the solution, consider the following:

1. Deploy the solution in a Region where the cost of the Lambda function is lowest.

2. Do not change the memory of the Lambda function (CloudFormation parameter **Memory** unless absolutely required). This will increase the cost of the solution significantly.

3. Remove unused schedules from the solution configurations.

4. Select a frequency that reduces the number of Lambda function runs per day. For example, if the schedules are hours apart, set the frequency (**Frequency** CloudFormation parameter) to one-hour increments. By default, the solution is set to five minutes which means the Lambda function will be invoked 288 times a day, whereas a one-hour frequency will run 24 times a day.

Supported AWS Regions

You can deploy Instance Scheduler in any AWS Region, including AWS GovCloud (US) Regions and some opt-in Regions (Regions that are disabled by default). After you deploy the solution, you can configure it to apply the appropriate start or stop actions to tagged EC2 and RDS DB instances in any Region(s) of your account. If you use cross-account instance scheduling, the solution applies actions to instances in all configured Regions in all accounts.

> ⚠️ **Important**
>
> Instance Scheduler on AWS actions affect appropriately tagged instances in all AWS Regions of your account, even though the Lambda function is running in a single Region.

You can use multiple deployments of the solution to schedule a large number of instances, or instances in many accounts and Regions. When you deploy multiple schedulers, use a different tag name for each stack, and configure a set of non-overlapping Regions for each deployment.

Each deployment checks every instance in every configured Region in an account for the tag key that identifies resources it should schedule. If the Regions for multiple deployments overlap, each instance will be checked by multiple deployments.

> **ⓘ Note**
>
> For opt-in Regions, Instance Scheduler on AWS can target instances within any opt-in Region for scheduling, but the CloudFormation stacks themselves are currently only available for deployment in the following opt-in Regions.

Cross-account instance scheduling using account IDs or AWS Organization ID

This solution includes a template (instance-scheduler-on-aws-remote.template) that creates the AWS Identity and Access Management (IAM) roles and other necessary resources to enable the solution to start scheduling in the secondary accounts. You can review and modify permissions in the remote template before you launch the stack.

Enabling cross account scheduling using Account IDs

To apply automated start-stop schedules to resources in secondary accounts:

1. Sign in to the AWS Management Console and select the button to launch the instance-scheduler-on-aws AWS CloudFormation template in the primary account.

2. Launch the remote template (instance-scheduler-on-aws-remote) in each applicable secondary account. When each remote stack is launched, it creates a cross-account role Amazon Resource Name (ARN).

3. Update the primary solution stack with the Account ID in the **Provide Organization ID** or **List of Remote Account IDs** parameters to allow the solution to perform start and stop actions on instances in the secondary accounts.

Enabling cross account scheduling using AWS Organization ID

To apply automated start-stop schedules to resources in secondary accounts:

1. Sign in to the AWS Management Console and select the button to launch the instance-scheduler-on-aws AWS CloudFormation template in the primary account.

2. Set the CloudFormation parameter **Using AWS Organizations?** as Yes,and provide the organization ID in the **Provide Organization ID** OR **List of Remote Account IDs** CloudFormation parameters.

3. After deploying the stack in the primary account, launch the remote template (`instance-scheduler-on-aws-remote`) in each applicable secondary account in the same Region as the solution in the primary account. When each remote stack is launched successfully, the primary solution account will be updated with the account id without any further changes in the primary account.

Managing Account IDs with AWS Systems Manager Parameter Store

Use AWS Systems Manager Parameter Store to store remote account IDs. You can store remote Account IDs as a list parameter where every item is an account ID, or as a string parameter that contains a comma-delimited list of remote account IDs. The parameter has the format {param:*name*} where the name is the name of the parameter in Parameter Store.

To leverage this feature, you must launch the Instance Scheduler on AWS hub stack in the same account as your parameter store.

Services supported for scheduling

Instance Scheduler on AWS currently supports scheduling of the following services:

- Amazon EC2
- Amazon EC2 Auto Scaling groups
- Amazon RDS
- Amazon Aurora clusters
- Amazon DocumentDB
- Amazon Neptune

Instance shutdown behavior

Amazon EC2

This solution is designed to automatically stop EC2 instances and assumes that instance *shutdown behavior* is set to Stop, not Terminate. Note that you cannot restart an Amazon EC2 instance after it is terminated.

By default, EC2 instances are configured to stop, not terminate, when shut down, but you can modify this behavior. Therefore, make sure that the instances you control using the Instance

Scheduler on AWS are configured with a Stop shutdown behavior; otherwise, they will be terminated.

Amazon RDS, Amazon Neptune, and Amazon DocumentDB

This solution is designed to automatically stop, not delete, RDS, Neptune, and DocDB instances. You can use the **Create RDS Instance Snapshot** AWS CloudFormation template parameter to create snapshots of RDS DB instances before the solution stops the instances. Snapshots are kept until the next time the instance is stopped and a new snapshot is created.

> ⓘ **Note**
>
> Snapshots are not available for Amazon Aurora clusters. You can use the **Schedule Aurora Clusters** template parameter to start and stop RDS DB instances that are part of an Aurora cluster or that manage Aurora databases. You must tag the cluster (not the individual instances) with the tag key you defined during initial configuration and the schedule name as the tag value to schedule that cluster.

For more information about limitations to starting and stopping an RDS DB instance, refer to [Stopping an Amazon RDS DB instance temporarily](#) in the *Amazon RDS User Guide*.

When an RDS DB instance is stopped, the cache is cleared, which might lead to slower performance when the instance is restarted.

Amazon RDS maintenance window

Every RDS DB instance has a weekly [maintenance window](#) during which any system changes are applied. During the maintenance window, Amazon RDS will automatically start instances that have been stopped for more than seven days to apply maintenance. Amazon RDS will not stop the instance once the maintenance event is complete.

The solution allows you to specify whether to add the preferred maintenance window of an RDS DB instance as a running period to its schedule. The solution will start the instance at the beginning of the maintenance window and stop the instance at the end of the maintenance window if no other running period specifies that the instance should run, and if the maintenance event is completed.

If the maintenance event is not completed by the end of the maintenance window, the instance will run until the scheduling interval after the maintenance event is completed. For more information

about the Amazon RDS maintenance window, refer to [Maintaining a DB instance](#) in the *Amazon RDS User Guide*.

Amazon EC2 Auto Scaling groups

We designed this solution to automatically stop Amazon EC2 Auto Scaling groups by using scheduled scaling actions. You can use the solution to configure scheduled scaling actions on the Auto Scaling group (ASG). When an ASG is stopped by a scheduled scaling action, its minimum, desired, and maximum capacities will be set to `0` until the ASG is automatically started again. This will return the minimum, desired, and maximum capacities to their original values.

Architecture

This section provides a reference implementation architecture diagram, AWS Well-Architected design considerations, security components, scheduler configurations, and AWS services used in this solution.

Architecture diagram

Deploying this solution deploys the following components in your AWS account.
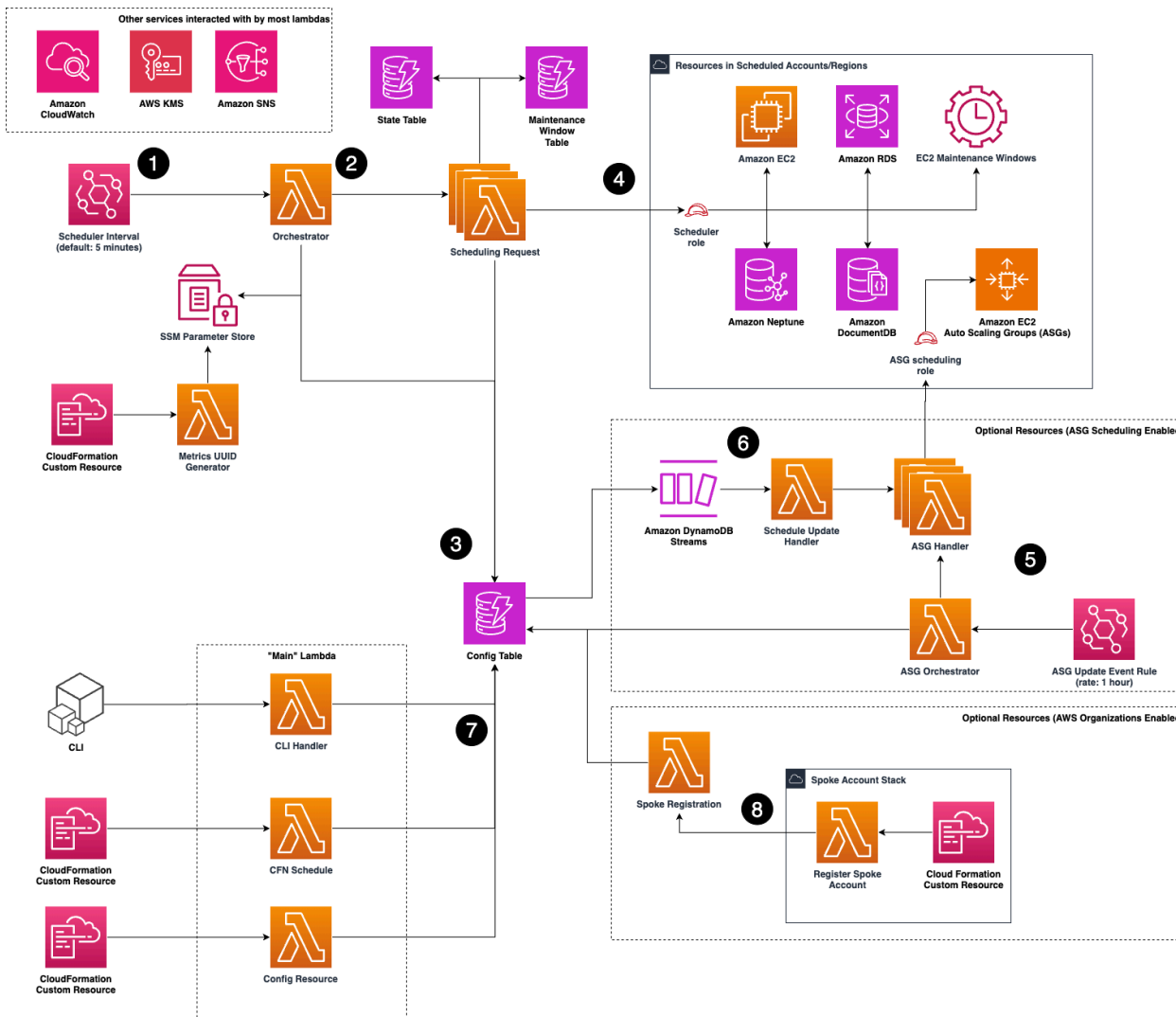
Instance Scheduler on the AWS Cloud

1. The solution deploys an [Amazon EventBridge Rule](#) with a configurable scheduling interval. This scheduling interval defines how frequently the solution runs and takes action to schedule your instances.

2. Each scheduling interval invokes an [AWS Lambda](#) Orchestration function. This will determine the list of AWS accounts, Regions, and services that need to be scheduled. The scheduling interval will then invoke multiple scheduling request Lambda functions in parallel to perform scheduling activities.

3. A collection of schedules and periods are stored in an [Amazon DynamoDB](#) configuration table to control the scheduling behavior of the solution. You can configure any number of schedules/ periods in this table and the solution will schedule instances accordingly.

4. Each scheduling request inspects resources in its particular target (account/Region/service) to find resources that have been tagged for scheduling by using schedules defined in the solution's configuration table. The scheduling request handler then inspect the configured schedules and performs required scheduling actions.

5. If ASG scheduling is enabled, Instance Scheduler on AWS deploys an hourly [Amazon EventBridge rule](#) and associated orchestration and handler resources to manage [Scheduled Scaling Actions](#) for [Amazon EC2 Auto Scaling groups](#) that have been tagged for scheduling by the solution.

6. In addition to the hourly scan, the solution also tracks updates to schedules in the configuration table. When a schedule is updated, a secondary orchestration Lambda function is invoked to ensure that ASG scheduled scaling actions are kept up to date with the most recent schedule configuration.

7. The solution provides multiple ways to create/update schedules in the solution's configuration table, along with several example schedules to use as a starting point. Configuration methods include: The DynamoDB console, a scheduler CLI, and an [AWS CloudFormation Custom resource](#).

8. If AWS Orgs Mode is enabled and a valid Org ID is provided when the solution is deployed, Instance Scheduler on AWS automatically registers newly deployed spoke stacks with the solution hub stack. The hub and spoke stacks must be deployed in the same Region and in accounts that are members of the same [AWS Organization](#).

> (i) **Note**
>
> AWS CloudFormation resources are created from [(AWS CDK)](#) constructs.

All Lambda functions used by this solution leverage AWS IAM for permission requirements for your resources, and AWS KMS for encryption of the Amazon Simple Notification Service (Amazon SNS topic) and DynamoDB tables.

Each time the solution performs a scheduling interval, it checks the current state of each appropriately tagged instance against the targeted state (defined by one or more periods in a schedule in the instance tag) in the associated schedule. The schedule interval then applies the appropriate start or stop action, as necessary.

For example, if the Lambda function is invoked on a Friday at 9 AM (ET) and it identifies a stopped EC2 or RDS DB instance with a Schedule=office-hours tag, it will check Amazon DynamoDB for the office-hours schedule configuration details. If the office-hours schedule contains a period that indicates that the instance should run Monday through Friday from 9 AM ET to 5 PM ET, the Lambda function will start that instance.

The Lambda function also records information about your resources and displays them in an optional Amazon CloudWatch Custom dashboard. Information recorded includes the number of instances tagged for each schedule, the sizes of those instances, and whether or not those instances are currently in a running or stopped state. For more information on this custom dashboard, refer to Operational insights dashboard.

> **ⓘ Note**
>
> *Stopping* an Amazon EC2 instance is different from *terminating* an Amazon EC2 instance. By default, Amazon EC2 instances are configured to stop, not terminate, when shut down, but you can modify this behavior. Before using this solution, verify that instances are set to stop or terminate as appropriate.

AWS Well-Architected design considerations

We designed this solution with best practices from the AWS Well-Architected Framework which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework were applied when building this solution.

Operational excellence

This section describes how we architected this solution using the principles and best practices of the operational excellence pillar.

- The solution pushes metrics to Amazon CloudWatch to provide observability into its components (such as its infrastructure and Lambda functions).

- AWS X-Ray traces Lambda functions.

- Uses Amazon SNS for error reporting.

Security

This section describes how we architected this solution using the principles and best practices of the security pillar.

- All inter-service communications use IAM roles.

- All multi-account communications use IAM roles.

- All roles used by the solution follow least-privilege access. In other words, they only contain minimum permissions required so that the service can function properly.

- All data storage including DynamoDB tables have encryption at rest.

Reliability

This section describes how we architected this solution using the principles and best practices of the reliability pillar.

- The solution uses serverless AWS services wherever possible (such as Lambda and DynamoDB) to ensure high availability and recovery from service failure.

- Data processing uses Lambda functions. The solution stores data in DynamoDB, so it persists in multiple Availability Zones by default.

Performance efficiency

This section describes how we architected this solution using the principles and best practices of the [performance efficiency pillar](#).

- The solution uses serverless architecture.

- You can launch the solution in any AWS Region that supports the AWS services used in this solution (such as Lambda and DynamoDB). For details, refer to [Supported AWS Regions](#).

- The solution is automatically tested and deployed every day. Our solution architects and subject matter experts review the solution for areas to experiment and improve.

Cost optimization

This section describes how we architected this solution using the principles and best practices of the [cost optimization pillar](#).

- The solution uses serverless architecture, and customers pay only for what they use.

- The compute layer defaults to Lambda, which uses a pay-per-use model.

Sustainability

This section describes how we architected this solution using the principles and best practices of the [sustainability pillar](#).

- The solution uses managed and serverless services to minimize the environmental impact of the backend services.

- The solution's serverless design is aimed at reducing carbon footprint compared to the footprint of continually operating on-premises servers.

Scheduler configuration table

When deployed, Instance Scheduler on AWS creates an Amazon DynamoDB table that contains global configuration settings.

Global configuration items contain a type attribute with a value of **config** in the configuration table. Schedules and periods contain type attributes with values of **schedule** and **period**, respectively. You can add, update, or remove schedules and periods from the configuration table using the DynamoDB console or the solution's command line interface. However, you don't edit any items with a type of **config** because these items are managed by the solution.

Scheduler CLI

The solution includes a CLI that provides commands for configuring schedules and periods. The CLI allows you to estimate cost savings for a given schedule. The cost estimates provided by the schedule CLI are for approximation purposes only. For more information about configuring and using the scheduler CLI, refer to Scheduler CLI.

AWS services used in this solution

AWS service	Description
AWS Lambda	**Core.** Solution deploys a Lambda function that contains all the logic to schedule the instances and to manage updates to the CloudForm ation stack using a custom resource feature**.**
Amazon DynamoDB	**Core.** Solution creates DynamoDB tables to store schedule configuration, state informati on, last actions performed of the instances , and a table to store Systems Manager maintenance window for scheduling purpose.
Amazon CloudWatch	**Core.** Solution stores debugging and informati on logs.
AWS IAM	**Core.** Solution uses IAM to get permissions for scheduling instances.
Amazon SNS	**Core.** Solution creates an SNS topic to send error messages for the users to subscribe to and troubleshoot in case of any errors.

AWS service	Description
AWS KMS	**Core.** Solution creates an AWS KMS key to encrypt the SNS topic.
Amazon EventBridge	**Core.** Solution creates an EventBridge Solution creates EventBridge scheduled rules that invoke AWS lambda on a consistent interval"
AWS Systems Manager	**Supporting.** Provides application-level resource monitoring and visualization of resource operations and cost data.
Amazon EC2	**Scheduled.** Solution is used to start and stop EC2 instances. The instances are identified by specific tags key/values which are configured in the solution.
Amazon RDS	**Scheduled.** Solution is used to change RDS DB instances status to Available or Stopped. The instances are identified by specific tags key/ values which are configured in the solution.
Amazon Aurora	**Scheduled.** Solution is used to change Aurora clusters status to Available or Stopped. The clusters are identified by specific tags key/ values which are configured in the solution.
Amazon Neptune	**Scheduled.** Solution is used to change Neptune instances status to Available or Stopped. The instances are identified by specific tags key/values which are configured in the solution.

AWS service	Description
Amazon DocumentDB	**Scheduled.** Solution is used to change DocumentDB instances status to Available or Stopped. The instances are identified by specific tags key/values which are configured in the solution.
Amazon EC2 Auto Scaling groups	**Scheduled.** Solution is used to manage scheduled scaling rules for EC2 Auto Scaling groups. These rules will start/stop Auto Scaling groups in accordance with an associated schedule. Groups are identified by specific tags key/values which are configured in the solution.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared responsibility model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

AWS KMS

The solution creates an AWS managed Customer managed key, which is used to configure server-side encryption for the SNS topic and the DynamoDB tables.

Amazon IAM

The solution's Lambda function requires permissions to start/stop both to EC2 and RDS DB instances, modify instance attributes, and update tags for the instances. All the necessary permissions are provided by the solution to Lambda service role created as part of the solution template.

Additionally, the Lambda service role has access to get/put Systems Manager parameters, access to CloudWatch log groups, AWS KMS key encryption/decryption, and publish messages to SNS

topics. For detailed information about each permission provided to the service role, refer to the [CloudFormation templates](#).

Encrypted EC2 EBS Volumes

When scheduling EC2 instances attached to EBS volumes encrypted by AWS KMS, you must grant Instance Scheduler on AWS permission to use the associated AWS KMS key(s). This allows Amazon EC2 to decrypt the attached EBS volumes during the started function. This permission must be granted to the scheduling role in the same account as the EC2 instance(s) using the key.

To grant permission to use an AWS KMS key with Instance Scheduler on AWS, add the AWS KMS key's ARN to the Instance Scheduler on AWS stack (hub or spoke) in the same account as the EC2 instance(s) using the key(s):

Kms Key Arns for EC2
comma-separated list of kms arns to grant Instance Scheduler kms:CreateGrant permissions to provide the EC2 service with Decrypt permissions for encrypted EBS volumes. This allows the scheduler to start EC2 instances with attached encrypted EBS volumes. provide just (*) to give limited access to all kms keys, leave blank to disable. For details on the exact policy created, refer to security section of the implementation guide (https://aws.amazon.com/solutions/implementations/instance-scheduler-on-aws/)

```
Enter CommaDelimitedList
```

KMS Ket Arns for EC2

This will automatically generate the following policy and add it to the scheduling role for that account:

```
{

  "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"
        }
    ]
}
```

Getting started

This guide contains a brief overview and instructions to deploy the solution quickly. This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation templates specify the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the templates.

Deployment process overview

> ⚠️ **Important**
>
> This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. AWS owns the data gathered though this survey. Data collection is subject to the Privacy notice.

To opt out of this feature, download the template, modify the AWS CloudFormation mapping section, and then use the AWS CloudFormation console to upload your updated template and deploy the solution.

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 5-10 minutes (not including configuration).

Step 1: Launch the instance scheduler stack

1. Launch the AWS CloudFormation template in your AWS account.

2. Enter values for the required parameters.

3. Review the other template parameters, and adjust if necessary.

Step 2 (Optional): Launch the remote stack in secondary accounts

1. Launch the AWS CloudFormation template in your AWS account.

2. Enter values for the required parameters.

AWS CloudFormation templates

This solution uses [AWS CloudFormation templates and stacks](#) to automate its deployment. The CloudFormation templates specify the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the templates.

You can download the CloudFormation templates for this solution before deploying it.

View template

instance-scheduler-on-aws.template - Use this template to launch the solution and all associated components. The default configuration deploys an AWS Lambda function, an Amazon DynamoDB table, an Amazon CloudWatch event, and CloudWatch custom metrics, but you can also customize the template based on your specific needs.

View template

instance-scheduler-on-aws-remote.template - Use this template to launch the cross-account role used by the solution to schedule instances in spoke accounts. For deployments using AWS Organizations, deploying the template also registers the spoke account with the hub, requiring no manual configuration.

> ⓘ **Note**
>
> If you previously deployed this solution, see [Update the solution](#) for update instructions.

Step 1: Launch the instance scheduler hub stack

Follow the step-by-step instructions in this section to deploy the solution into your account.

Time to deploy: Approximately five minutes

Launch solution

1. Sign in to the [AWS Management Console](#) and select the button to launch the **instance-scheduler-on-aws.template** AWS CloudFormation template.

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.

4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, see [IAM and AWS STS quotas](#) in the *AWS Identity and Access Management User Guide*.

5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default
Schedule tag key	Schedule
Scheduling interval (minutes)	5
Default time zone	UTC
Scheduling enabled	Yes
Enable xxx Scheduling	Enabled

Parameter	Default
Start tags	InstanceScheduler-LastA n=Started By {scheduler {year}/{month}/{day} {h inute}{timezone},>
Stop tags	InstanceScheduler-LastA n=Stopped By {scheduler {year}/{month}/{day} {h inute}{timezone},>
Enable EC2 SSM maintenance windows	No
KMS Key ARNs for EC2	*<Optional Input>*
Create RDS instance snapshots on stop	No
ASG scheduled tag key	scheduled
ASG action name prefix	is-
Use AWS Organizations	No
Namespace	default

Parameter	Default
Organization ID/Remote Account IDs	*<Optional Input>*
Region(s)	*<Optional Input>*
Enabled hub account scheduling	Yes
Log retention period (days)	30
Enable CloudWatch Debug Logs	No
Operational Monitoring	Enabled
Memory Size	128
Protect DynamoDB Tables	Enabled

6. Choose **Next**.

7. On the **Configure stack options** page, choose **Next**.

8. On the **Review and create** page, review and confirm the settings. Check the box acknowledging that the template will create IAM resources.

9. Choose **Submit** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately five minutes.

Step 2 (Optional): Launch the remote stack in secondary accounts

> ⚠️ **Important**
>
> The remote stack must be deployed in the same Region as the hub stack.

This automated AWS CloudFormation template configures secondary account permissions that will allow the hub stack to schedule instances in other accounts. Install the remote template only after the primary/hub stack has been successfully installed in the Hub account.

Launch solution

1. Sign in to the AWS Management Console of the applicable secondary account and select the button to launch the instance-scheduler-on-aws-remote AWS CloudFormation template.

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar. If the hub stack is configured to use AWS Organizations, then deploy the remote template in the same region as the hub stack.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.

4. On the **Specify Details** page, assign a name to your remote stack.

5. Under **Parameters**, review the parameter for the template, and modify it.

6. If the AWS Organizations option is enabled and the hub stack is similarly configured, there are no further changes required in the main stack to start the scheduling.

7. If the AWS Organization option is set to No, then the hub stack should be updated with the new Account ID.

Parameter	Default	Description
Hub Account ID	*<Requires Input>*	Account ID of the Instance Scheduler on AWS hub stack that will schedule resources in this account.
Use AWS Organizations	No	Use AWS Organizations to automate spoke account registration. Must be set to the same value as the hub stack.
Namespace	`default`	Unique identifier used to differentiate between multiple solution deployments. Must be set to the same value as the hub stack.
Kms Key ARNs for EC2	<Optional Input>	Comma-separated list of KMS ARNs to grant the solution kms:CreateGrant permissions to provide the EC2 service with decrypt permissions for encrypted EBS volumes. This allows the scheduler to start EC2 instances with attached encrypted EBS volumes. Provide (*) to give limited access to all KMS keys; leave blank to disable. For details on the exact policy created, refer to Encrypted EC2 EBS Volumes

5. Choose **Next.**

6. On the **Options** page, choose **Next**.

7. On the **Review and create** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create IAM resources.

8. Choose **Submit** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately five minutes.

Configure the solution

Now that the solution has been deployed, you can begin configuring schedules and tagging instances for scheduler. To learn more about how to do these things, refer to Configure schedules and Tag instances for scheduling.

Operator guide

This guide is intended for users and operators of this solution and contains details on how to configure schedules, monitor the solution, update the solution and other advanced features.

Configure schedules

Once the solution has been successfully deployed, you can begin to configure schedules. Instance Scheduler on AWS supports two methods of managing schedules as described below.

> ⓘ **Note**
>
> The solution can support any number of schedules, each of which can contain one or more periods that define when instances controlled by that schedule should be running. For more information, refer to Schedules and Periods.

Using Infrastructure as Code (recommended)

Instance Scheduler on AWS provides an AWS CloudFormation CustomResource that you can use to manage your schedules and periods using Infrastructure as Code (IaC).

For information on how to manage schedules using IaC, please refer to Manage Schedules Using Infrastructure as Code (IaC).

Using the Amazon DynamoDB Console and Instance Scheduler on AWS CLI

> ⚠ **Important**
>
> If you used the custom resource to manage any schedules using IaC, you must not use the DynamoDB console or scheduler CLI to delete or modify those schedules or their periods. If you do, you will create a conflict between the stored parameters in CloudFormation and the values in the table. Also, do not use periods managed by CloudFormation in schedules created using the DynamoDB console or the scheduler CLI.

When deploying the Instance Scheduler on AWS hub stack, the solution created an Amazon DynamoDB table containing several sample periods and schedules that you can use as a reference to create your own custom periods and schedules. To create a schedule in DynamoDB, modify one of the schedules in the configuration table (ConfigTable) or create a new one. To create a schedule using the CLI, first Install the Scheduler CLI and then use the Available commands.

> **ⓘ Note**
>
> For examples of how to create several sample schedules using IaC, DynamoDB, and the InstanceScheduler CLI, please refer to Sample schedules.

This section provides instruction and reference on how to use, monitor and update the solution as well as troubleshooting and support information.

Tag instances for scheduling

When you deployed the AWS CloudFormation template, you defined the name (tag key) for the solution's *custom tag*. For Instance Scheduler on AWS to recognize an Amazon EC2 or Amazon RDS instance, the tag key on that instance must match this custom tag key. Therefore, it is important that you apply tags consistently and correctly to all applicable instances. You can continue to use existing tagging best practices for your instances while using this solution. For more information, refer to Tag your Amazon EC2 resources and Tagging Amazon RDS resources.

On the AWS Management Console, use the Tag Editor to apply or modify tags for multiple resources at a time. You can also apply and modify tags manually in the console.

Setting the tag value

When you apply a tag to an instance, use the tag key you defined during initial configuration (by default the tag key is Schedule) and set the tag value to the name of the schedule that should apply to the instance. If you would like to change the tag key, you can do so by updating the solution parameters.

> **ⓘ Note**
>
> For Amazon RDS instances, the tag value can be from 1 to 256 Unicode characters in length and cannot be prefixed with aws:. The string can contain only the set of Unicode letters,

digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). For more information, refer to Tagging Amazon RDS resources.

EC2 instances with encrypted EBS volumes

If your EC2 DB instances have EBS volumes encrypted with customer-managed KMS keys, you must give the Instance Scheduler role the KMS:CreateGrant permission to be able to start those instances. For more information, refer to Encrypted EC2 EBS Volumes.

Schedule reference

Schedules specify when instances tagged with that schedule should run. Each schedule must have a unique name, which is used as the tag *value* that identifies the schedule you want to apply to the tagged resource.

Periods

Each schedule must contain at least one period that defines the time(s) the instance should run. A schedule can contain more than one period. When more than one period is used in a schedule, Instance Scheduler on AWS will apply the appropriate start action when at least one of the periods is true. For more information, refer to Period reference.

Time zone

You can also specify a time zone for the schedule. If you do not specify a time zone, the schedule will use the default time zone you specify when you launch the solution. For a list of acceptable time zone values, refer to the **TZ** column of the List of TZ database time zones.

Hibernate field

The hibernate field allows you to use hibernation for stopped Amazon EC2 instances. If this field is set to true, your EC2 instances must use an Amazon Machine Image (AMI) that supports hibernation. For more information, refer to Supported Linux AMIs and Supported Windows AMIs in the *Amazon EC2 User Guide*. Hibernation saves the contents from the instance memory (RAM) to your Amazon Elastic Block Store (Amazon EBS) root volume. If this field is set to true, instances are hibernated instead of stopped when the solution stops them.

If you set the solution to use hibernation, but your instances are not [configured for hibernation](#) or they do not meet the [hibernation prerequisites](#), the solution logs a warning and the instances are stopped without hibernation. For more information, refer to [Hibernate your On-Demand Instance or Spot Instance](#) in the *Amazon EC2 User Guide*.

Enforced field

Schedules contain an enforced field that allows you to prevent an instance from being manually started outside of a running period, or manually stopped during a running period. If this field is set to true and a user manually starts an instance outside of a running period, the solution will stop the instance. If this field is set to true, it also restarts an instance if it was manually stopped during a running period.

Retain running field

The retain_running field prevents the solution from stopping an instance at the end of a running period if the instance was manually started before the beginning of the period. For example, if an instance with a period that runs from 9 AM to 5 PM is manually started before 9 AM, the solution will not stop the instance at 5 PM.

Systems Manager maintenance window field (only applies to EC2 instances)

The ssm-maintenance-window field allows you to automatically add AWS Systems Manager maintenance windows as a running periods to a schedule. When you specify the name of a maintenance window that exists in the same account and AWS Region as your Amazon EC2 instances, the solution will start the instance at least 10 minutes before the start of the maintenance window and stop the instance at the end of the maintenance window if no other running period specifies that the instance should run.

Once the SSM Maintenance window is created and the schedule is configured with the name of the SSM maintenance window, the changes are picked up at the next scheduled run of the Lambda. For example, if you selected a frequency of 5 minutes for the scheduler Lambda to run, the maintenance window changes will be picked up by the Lambda on the next 5-minute interval.

Instance Scheduler on AWS will ensure that your instances are started at least 10 minutes before the maintenance window begins. Depending on the value you set for the **Scheduling Interval** AWS CloudFormation parameter, this may result in your instance being started 10+interval minutes

before the beginning of the maintenance window in order to guarantee that the instance starts at least 10 minutes early. For example, if you set the Scheduling Interval to 30 minutes, the scheduler will start the instance between 10-40 minutes before the beginning of the maintenance window.

> ⓘ **Note**
>
> In order to use this feature, the Enable EC2 SSM Maintenance Windows CloudFormation parameter in the solution hub stack must be set to `yes`.

For more information, refer to [AWS Systems Manager Maintenance Windows](#) in the *AWS Systems Manager user guide*.

Instance type

For Amazon EC2 instances only, a schedule allows you to specify an optional desired instance type for each period in a schedule. When you specify an instance type in the period, the solution will automatically resize EC2 instances to match the requested instance type.

To specify an instance type, use the syntax *<period-name>@<instance-type>*. For example, weekends@t2.nano. Note that if you specify an instance type for a period that schedules Amazon EC2 instances and Amazon RDS instances, the instance type will be ignored for Amazon RDS instances.

If the instance type of a running instance is different than the instance type specified for the period, the solution will stop the running instance and restart the instance with the specified instance type. For more information, refer to [Change the instance type](#) in the *Amazon EC2 User Guide for Linux Instances*.

Schedule definitions

The Instance Scheduler on AWS configuration table in Amazon DynamoDB contains schedule definitions. A schedule definition can contain the following fields:

Field	Description
description	An optional description of the schedule.

Field	Description
hibernate	Choose whether to hibernate Amazon EC2 instances running Amazon Linux. When this field is set to true, the scheduler will hibernate instances when it stops them. Note that your instances must turn on hibernation and must meet the hibernation prerequisites.
enforced	Choose whether to enforce the schedule. When this field is set to true, the scheduler will stop a running instance if it is manually started outside of the running period or it will start an instance if it is stopped manually during the running period.
name	The name used to identify the schedule. This name must be unique.
periods	The name of the periods that are used in this schedule. Enter the name(s) exactly as it appears in the period name field. You can also specify an instance type for the period using the syntax *<period-name>@<instance-type>*. For example, weekdays@t2.large.
retain_running	Choose whether to prevent the solution from stopping an instance at the end of a running period if the instance was manually started before the beginning of the period.

Field	Description
ssm_maintenance_window	Choose whether to add AWS Systems Manager maintenance window(s) as additional running period for this schedule. Accepts a StringSet of maintenance window names that will be matched against the names of windows in the same account/region as scheduled EC2 instances. **Note:** This feature only applies to EC2 instances.
stop_new_instances	Choose whether to stop an instance the first time it is tagged if it is running outside of the running period. By default, this field is set to true.
timezone	The time zone the schedule will use. If no time zone is specified, the default time zone (UTC) is used. For a list of acceptable time zone values, refer to the **TZ** column of the [List of tz database time zones](#).
use_metrics	Choose whether to turn on CloudWatch metrics at the schedule level. This field overwrites the CloudWatch metrics setting you specified at deployment. **Note:** Enabling this feature will incur charges of $0.90/month per schedule or scheduled service.

Period reference

Periods contain conditions that allow you to set the specific hours, days, and months an instance will run. A period can contain multiple conditions, but all conditions must be true for the Instance Scheduler on AWS to apply the appropriate start or stop action.
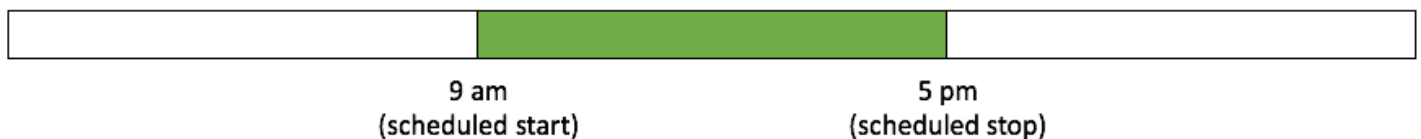
Start and stop times

The `begintime` and `endtime` fields define when the Instance Scheduler on AWS will start and stop instances. If you specify a start time only, the instance must be stopped manually. Note that if you specify a value in the [weekdays](#) field, the solution uses that value to determine when to stop the instance. For example, if you specify a `begintime` of 9 AM with no `endtime` and a weekdays value of Monday through Friday, the instance will be stopped at 11:59 PM on Friday unless you have scheduled an adjacent period.

Similarly, if you only specify a stop time, the instance must be started manually. If you don't specify either time, this solution uses the days of the week, days of the month, or months rules to start and stop instances at the beginning/end of each day as appropriate.

The `begintime` and `endtime` values for your period must be in the time zone specified in the schedule. If you do not specify a time zone in the schedule, the solution will use the time zone specified when you launch the solution.

If your schedule contains multiple periods, we recommend that you always specify both a `begintime` and `endtime` in your periods.

If you start an instance before the specified start time, the instance will run until the end of the running period. For example, a user might define a period that starts an instance daily at 9 AM and stops that instance at 5 PM.


9-5 scheduled start and stop

If you manually start that instance at 5 AM, the solution will stop the instance at 5 PM. If you use the [retain running field](#), the solution will not stop the instance at 5 PM.

5 AM scheduled stop

If you stop an instance before the specified stop time, the instance will not run until the beginning of the next running period. Continuing from the previous example, if the user stops the instance at 1 PM on Wednesday, the solution will not start the instance until 9 AM on Thursday.
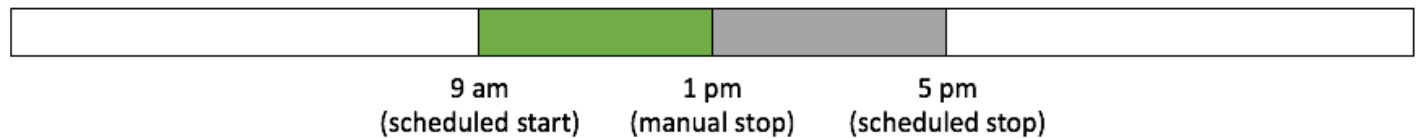
5 PM scheduled stop

Adjacent periods

The solution will not stop running instances if the schedule contains two adjacent running periods. For example, if you have a schedule with one period with an `endtime` of 11:59 PM and another period with a `begintime` of midnight the following day, the solution will not stop running instances, if there are no `weekdays, monthdays,or months` rules that stop the instances.

To implement a schedule that runs instances from 9 AM Monday to 5 PM Friday, the solution requires three periods. The first period runs applicable instances from 9 AM to 11:59 PM Monday. The second period runs the instances from midnight Tuesday to 11:59 PM Thursday. The third period runs the instances from midnight Friday to 5 PM Friday. For more information, refer to Sample schedule.

Days of the week

The `weekdays` field defines which days during the week an instance will run. You can specify a list of days, a range of days, the nth occurrence of that day in a month, or the last occurrence of that day in a month. The solution supports abbreviated day names (Mon) and numbers (0).

Days of the month

The `monthdays`, field defines which days during the month an instance will run. You can specify a list of days, a range of days, every n[th] day of the month, the last day of the month, or the nearest weekday to a specific date.

Months

The `months` field defines which months an instance will run. You can specify a list of months, a range of months, or every n[th] month. The solution supports abbreviated month names (Jan) and numbers (1).

Period definitions

The Instance Scheduler on AWS configuration table in Amazon DynamoDB contains period definitions. A period definition can contain the following fields. Note that some fields support Cron non-standard characters.

> ⚠️ **Important**
>
> You must specify at least one of the following items: begintime, endtime, weekdays, months, or monthdays.

Field	Description
`begintime`	The time, in **HH:MM** format, that the instance will start.
`description`	An optional description of the period.
`endtime`	The time, in **HH:MM** format, that the instance will stop.
`months`	Enter a comma-delimited list of months, or a hyphenated range of months, during which the instance will run. For example, enter `jan, feb, mar` or `1, 2, 3` to run an instance

Field	Description
	during those months. Or, you can enter `jan-mar` or `1-3`. You can also schedule an instance to run every n^{th} month or every n^{th} month in a range. For example, enter `Jan/3` or `1/3` to run an instance every third month starting in January. Enter `Jan-Jul/2` to run every other month from January to July.
`monthdays`	Enter a comma-delimited list of days of the month, or a hyphenated range of days, during which the instance will run. For example, enter `1, 2, 3` or `1-3` to run an instance during the first three days of the month. You can also enter multiple ranges. For example, enter `1-3, 7-9`to run an instance from the 1st to the 3rd and the 7th through the 9th. You can also schedule an instance to run every n^{th} day of the month or every n^{th} day of the month in a range. For example, enter `1/7` to run an instance every seventh day starting on the 1st. Enter `1-15/2` to run an instance every other day from the 1st to the 15th. Enter L to run an instance on the last day of the month. Enter a date and W to run an instance on the nearest weekday to the specified date. For example, enter `15W` to run an instance on the nearest weekday to the 15th.
`name`	The name used to identify the period. This name must be unique.

Field	Description
weekdays	Enter a comma-delimited list of days of the week, or a range of days of the week, during which the instance will run. For example, enter 0, 1, 2 or 0-2 to run an instance Monday through Wednesday. You can also enter multiple ranges. For example, enter 0-2, 4-6 to run an instance every day except Thursday. You can also schedule an instance to run every n[th] occurrence of a weekday in the month. For example, enter Mon#1 or 0#1 to run an instance the first Monday of the month. Enter a day and L to run an instance on the last occurrence of that weekday in the month. For example, enter friL or 4L to run an instance on the last Friday of the month.

When a period contains multiple conditions, note that all conditions must be true for Instance Scheduler on AWS to apply the appropriate action. For example, a period that contains a weekdays field with a value of Mon#1 and a months field with a value of Jan/3 will apply the action on the first Monday of the quarter.

Automated tagging

The Instance Scheduler on AWS can automatically add tags to all instances it starts or stops. You can specify a list of tag names or tagname=tagvalue pairs in the **Started tags** and **Stopped tags** parameters. The solution also includes macros that allow you to add variable information to the tags:

- {scheduler}: The name of the scheduler stack
- {year}: The year (four digits)
- {month}: The month (two digits)

- `{day}`: The day (two digits)
- `{hour}`: The hour (two digits, 24-hour format)
- `{minute}`: The minute (two digits)
- `{timezone}`: The time zone

The following table gives examples of different inputs and the resulting tags.

Example parameter input	Instance Scheduler tag
`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`

When you use the **Started tags** parameter, the tags are automatically deleted when the scheduler stops the instance. When you use the **Stopped tags** parameter, the tags are automatically deleted when the instance is started.

Sample schedules

Instance Scheduler on AWS allows you to automatically start and stop Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS) instances. The following section provides some example schedules that can be adapted to many common use cases.

Standard 9-5 working hours

This schedule shows how to run instances on weekdays from 9 AM to 5 PM in London.

Periods

This period will start instances at 9 AM and stop instances at 5 PM on weekdays (Mon-Fri).

Field	Type	Value
begintime	String	09:00
endtime	String	16:59
name	String	weekdays-9-5
weekdays	StringSet	mon-fri

Schedule

The schedule name provides the tag value that must be applied to instances and the timezone that will be used.

Field	Type	Value
name	String	london-working-hours
periods	StringSet	weekdays-9-5
timezone	String	Europe/London

Instance tag

To apply this schedule to instances, you must add the `Schedule=london-working-hours` tag to the instances. If you change the default tag name in the AWS CloudFormation **Instance Scheduler tag name** parameter, your tag will be different. For example, if you entered `Sked` as your tag name, your tag will be `Sked=london-working-hours`. For more information, refer to Tag your resources in the *Amazon Elastic Compute Cloud User Guide*.

Scheduler CLI

To configure the above schedule using the Instance Scheduler CLI, use the following commands:

```
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
```

Custom resource

The following CloudFormation template will create the above schedule using the schedule custom resource.

To deploy this template, you will need to provide the **ServiceInstanceScheduleServiceToken** ARN that can be found in the AWS CloudFormation console by selecting the previously deployed Instance Scheduler Hub Stack and then select **Outputs**.

```
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
```

Stop instances after 5 PM

Instances can be started freely at any time during the day and this schedule will ensure that a stop command is automatically sent to them at 5 PM ET every day.

Periods

This period will stop instances at 5 PM every day.

Field	Type	Value
endtime	String	16:59
name	String	stop-at-5

Schedule

The schedule name provides the tag value that must be applied to instances and the timezone that will be used.

Field		Value
name	String	stop-at-5-new-york
periods	StringSet	stop-at-5
timezone	String	America/New York

Instance tag

To apply this schedule to instances, you must add the `Schedule=stop-at-5-new-york` tag to the instances. If you changed the default tag name in the AWS CloudFormation **Instance Scheduler tag name** parameter, your tag will be different. For example, if you entered Sked as your tag name, your tag will be `Sked=stop-at-5-new-york` . For more information, refer to [Tag your resources](#) in the *Amazon Elastic Compute Cloud User Guide*.

Scheduler CLI

To configure the above schedule using the [Instance Scheduler CLI](#), use the following commands:

```
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
```

Custom resource

The following CloudFormation template will create the above schedule using the schedule custom resource.

To deploy this template, you will need to provide the **ServiceInstanceScheduleServiceToken** ARN that can be found in the AWS CloudFormation console by clicking on the previously deployed Instance Scheduler Hub Stack and selecting **Outputs**.

```
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'
```

Stop instances over the weekend

This schedule shows how to run instances from Monday 9 AM ET to Friday 5 PM ET. Since Monday and Friday are not full days, this schedule includes three periods to accommodate: Monday, Tuesday-Thursday, and Friday.

Periods

The first period starts tagged instances at 9 AM Monday and stops at midnight. This period includes the following fields and values.

Field	Type	Value
begintime	String	09:00
endtime	String	23:59
name	String	mon-start-9am
weekdays	StringSet	mon

The second period runs tagged instances all day Tuesday through Thursday. This period includes the following fields and values.

Field		Value
name	String	tue-thu-full-day
weekdays	StringSet	tue-thu

The third period stops tagged instances at 5 PM on Friday. This period includes the following fields and values.

Field		Value
begintime	String	00:00
endtime	String	16:59
name	String	fri-stop-5pm
weekdays	StringSet	fri

Schedule

The schedule combines the three periods into the schedule for tagged instances. The schedule includes the following fields and values.

Field		Value
name	String	mon-9am-fri-5pm
periods	StringSet	mon-start-9am,tue-thu-full-day,fri-stop-5pm
timezone	String	America/New_York

Instance tag

To apply this schedule to instances, you must add the `Schedule=mon-9am-fri-5pm` tag to the instances. Note that if you changed the default tag name in the AWS CloudFormation **Instance Scheduler tag name** parameter, your tag will be different. For example, if you entered Sked as your tag name, your tag will be `Sked=mon-9am-fri-5pm`. For more information, refer to Tag your resources in the *Amazon Elastic Compute Cloud User Guide*.

Scheduler CLI

To configure the above schedule using the Instance Scheduler CLI, use the following commands:

```
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
```

Custom resource

The following CloudFormation template will create the above schedule using the schedule custom resource.

To deploy this template, you will need to provide the **ServiceInstanceScheduleServiceToken** ARN that can be found in the AWS CloudFormation console by selecting the previously deployed Instance Scheduler Hub Stack and then select **Outputs**.

```
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
```

Solution resources

The following resources are created as part of the Instance Scheduler on AWS stack.

Resource name	Type	Description
Main	`AWS::Lambda::Funct ion`	Instance Scheduler on AWS Lambda function.
Scheduler Config Helper	`Custom::ServiceSetup`	Stores global configura tion settings in Amazon DynamoDB.
Scheduler Invoke Permission	`AWS::Lambda::Permi ssion`	Allows the Amazon CloudWatch event to invoke the Instance Scheduler's AWS Lambda function.
Scheduler Logs	`AWS::Logs::LogGroup`	CloudWatch Log Group for Instance Scheduler.
Scheduler Policy	`AWS::IAM::Policy`	Policy that allows scheduler to perform start and stop actions, change Amazon EC2 instance attributes, set tags, and access scheduler resources.
Scheduler Rule	`AWS::Events::Rule`	Amazon EventBridge event rule that invokes the scheduler's Lambda function.
Configuration Metrics Event Rule	`AWS::Events::Rule`	Amazon EventBridge event rule that periodically invokes the configuration description anonymized metrics function. Disabled when anonymized metrics are disabled.
State Table	`AWS::DynamamoDB::T able`	DynamoDB table that stores last desired state of instances.

Resource name	Type	Description
Config Table	`AWS::DynamamoDB::Table`	DynamoDB table that stores global configuration, schedule, and period data.
Instance Scheduler SNS Topic	`AWS::SNS::Topic`	Sends warning and error messages to subscribed email addresses.

Scheduler CLI

The Instance Scheduler on AWS command line interface (CLI) allows you to configure schedules and periods, and estimate cost savings for a given schedule.

Prerequisites

The CLI in this solution requires Python 3.8+

Credentials

To use the scheduler CLI, you must have credentials for the AWS CLI. For more information, refer to [Configuration and credential file settings](#) in the *AWS CLI User Guide*.

Your credentials must have the following permissions:

- `lambda:InvokeFunction` – To invoke the InstanceSchedulerMain function in the scheduler stack, and to update the schedule and period information in the scheduler configuration database from the command line

- `cloudformation:DescribeStackResource` – To retrieve the physical resource ID of the AWS Lambda function from the stack to handle the CLI request

Requests made by the scheduler CLI and responses are logged in the `AdminCliRequestHandler-yyyymmdd` log stream.

> **ⓘ Note**
>
> If you specify a profile using the profile-name argument, the profile you specify must
> have these permissions. For more information about the profile-name argument, refer to
> [Common Arguments](#).

Install the Scheduler CLI

1. [Download](#) the scheduler CLI package *(**instance_scheduler_cli.zip***) and place it in a directory on
   your computer.

2. Unzip the zip archive into its own directory (instance_scheduler_cli).

3. From the same directory in which you placed the unzipped CLI package, install the scheduler-cli
   to your environment:

> **ⓘ Note**
>
> Scheduler-CLI requires Python 3.8 or above and the latest version of pip, if you do
> not have both of these installed on your local machine, please refer to [pip's official](#)
> [documentation](#) for installation instructions before attempting to install the Scheduler-
> CLI.

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

5. Verify the installation succeeded with:

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

> **ⓘ Note**
>
> If preferred, an [sdist of the CLI](#) and can be installed using the same process as above.

Command structure

The scheduler CLI uses a multipart structure on the command line. The next part specifies the scheduler CLI python script. The scheduler CLI has commands that specify the operations to perform on periods and schedules. The specific arguments for an operation can be specified on the command line in any order.

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

Common arguments

The scheduler CLI supports the following arguments that all commands can use:

Argument	Description
`--stack <stackname>`	The name of the scheduler stack. **Important:** This argument is required for all commands.
`--region <regionname>`	The name of the region where the scheduler stack is deployed. **Note:** You must use this argument when the default configuration and credential files are not installed in the same region as the solution stack.
`--profile-name <profilename>`	The name of the profile to use to run commands. If no profile name is specified, the default profile is used.
`--query`	A JMESPath expression that controls the command output. For more information on controlling output, refer to [Controlling Command Output from the AWS Command Line Interface](#) in the *AWS CLI User Guide*.

Argument	Description
`--help`	Shows valid commands and arguments for the scheduler CLI. When used with a specific command, it shows valid subcommands and arguments for that command.
`--version`	Shows the version number of the scheduler CLI.

Available commands

- [create-period](#)
- [create-schedule](#)
- [delete-period](#)
- [delete-schedule](#)
- [describe-periods](#)
- [describe-schedules](#)
- [describe-schedule-usage](#)
- [update-period](#)
- [update-schedule](#)
- [help](#)

create-period

Description

Creates a period. A period must contain at least one of the following items: `begintime`, `endtime`, `weekdays`, `months`, or `monthdays`.

Arguments

`--name`

    The name of the period

Type: String

Required: Yes

`--description`

A description of the period

Type: String

Required: No

`--begintime`

The time when the running period starts. If `begintime` and `endtime` are not specified, the running period is 00:00 – 23:59.

Type: String

Constraints: `H:MM` or `HH:MM` format

Required: No

`--endtime`

The time when the running period stops. If `begintime` and `endtime` are not specified, the running period is 00:00 – 23:59.

Type: String

Constraints: `H:MM` or `HH:MM` format

Required: No

`--weekdays`

The days of the week for the period

Type: String

Constraints: Comma-delimited list of abbreviated day names (mon) or numbers (0). Use – to specify a range. Use / to specify every nth day of the week.

Required: No

```
--months
```

The months of the period

Type: String

Constraints: Comma-delimited list of abbreviated months names (jan) or numbers (1). Use – to specify a range. Use / to specify every nth month.

Required: No

```
--monthdays
```

The days of the month for the period

Type: String

Constraints: Comma-delimited list of abbreviated months names (jan) or numbers (1). Use – to specify a range. Use / to specify every nth day of the month.

Required: No

Example

```
$ 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

Description

Creates a schedule.

Arguments

`--name`

    The name of the schedule

    Type: String

    Required: Yes

`--description`

    A description of the schedule

    Type: String

    Required: No

`--enforced`

    Enforces the scheduled state for the instance

    Required: No

`--use-metrics`

    Collect Amazon CloudWatch metrics

    Required: No

`--periods`

    A list of running periods for the schedule. If multiple periods are specified, the solution will start
    an instance if one of the periods evaluates to `true`.

    Type: String

    Constraints: Comma-delimited list of periods. Use `<period-name>@`*`<instance type>`* to
    specify an instance type for a period. For example, `weekdays@t2.large`.

    Required: Yes

`--retain-running`

    Prevents an instance from being stopped by the solution at the end of a running period, if the
    instance was manually started before the beginning of the period.

Required: No

`--ssm-maintenance-window`

Adds an AWS Systems Manager maintenance window as a running period to an Amazon EC2 instance schedule. To use this command, you must use the `use-maintenance-window` command.

Type: String

Required: No

`--do-not-stop-new-instances`

Do not stop an instance the first time it is tagged if it is running outside of a running period

Required: No

`--timezone`

The time zone the schedule will use

Type: Array of strings

Required: No (If this argument is not used, the default time zone from main solution stack is used.)

`--use-maintenance-window`

Adds an Amazon RDS maintenance window as a running period to an Amazon RDS instance schedule, or an AWS Systems Manager maintenance window as a running period to an Amazon EC2 instance schedule

Required: No

Example

```
$ 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`

    The name of the applicable period

    Type: String

    Required: Yes

> ⚠️ **Important**
>
>   If the period is used in existing schedules, you must remove it from those schedules *before* you delete it.

Example

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

delete-schedule

Description

Deletes an existing schedule

Arguments

`--name`

> The name of the applicable schedule
>
> Type: String
>
> Required: Yes

Example

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

describe-periods

Description

Lists the configured periods for the Instance Scheduler stack

Arguments

`--name`

> The name of a specific period you want described
>
> Type: String
>
> Required: No

Example

```
$ 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

Description

Lists the configured schedules for the Instance Scheduler stack.

Arguments

`--name`

The name of a specific schedule you want described

Type: String

Required: No

Example

```
$ 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

Description

Lists all the periods running within a schedule and calculates the billing hours for instances. Use this command to simulate a schedule to calculate potential savings, and running periods after creating or updating a schedule.

Arguments

`--name`

The name of the applicable schedule

Type: String

Required: Yes

`--startdate`

The start date of the period used for calculation. The default date is the current date.

Type: String

Required: No

`--enddate`

The end date of the period used for calculation. The default date is the current date.

Type: String

Required: No

Example

```
$ 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

Description

Updates an existing period

Arguments

The `update-period` command supports the same arguments as the `create-period` command. For more information on the arguments, refer to the [create period command](#).

> ⚠️ **Important**
>
>   If you do not specify an argument, that argument will be removed from the period.

update-schedule

Description

Updates an existing schedule

Arguments

The `update-schedule` command supports the same arguments as the `create-schedule` command. For more information on the arguments, refer to the [create schedule command](#).

> ⚠️ **Important**
>
> If you do not specify an argument, that argument will be removed from the schedule.

help

Description

Displays a list of valid commands and arguments for the scheduler CLI.

Example

```
$ 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
```

When used with a specific command, the `--help` argument shows valid subcommands and arguments for that command.

Specific command example

```
$ 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
```

Update global configuration settings

When you first deployed Instance Scheduler's Hub template in AWS CloudFormation, a number of global configuration settings were selected as parameter inputs. These global configuration parameters can be updated at any time within the CloudFormation console.

To update the global configuration of Instance Scheduler, log into the account/region containing your hub deployment and go to the AWS CloudFormation console. Find the Instance Scheduler Hub Stack and select **Update -> Use Existing Template**. Update any global configuration parameters you would like to change, and then select **next -> next -> submit** to perform a CloudFormation update of the relevant solution resources.

Manage schedules using Infrastructure as Code (IaC)

Instance Scheduler on AWS provides a custom resource (`ServiceInstanceSchedule`) that you can use to configure and manage schedules through AWS CloudFormation. The custom resource uses PascalCase keys for the same data as the Instance Scheduler config table in Amazon DynamoDB (see template below for examples). For more information on the fields for schedules, refer to Schedule Definitions. For more information on the fields for periods, refer to Period Definitions.

When you use the custom resource to create a schedule, the name of that schedule is the logical resource name of the custom resource by default. To specify a different name, use the Name property of the custom resource. The solution also adds the stack name to the schedule name as a prefix by default. If you do not want to add the stack name as a prefix, use the `NoStackPrefix` property.

When you use the Name and `NoStackPrefix` properties, make sure you choose unique schedule names. If a schedule with the same name already exists, the resource will not be created or updated.

To get started managing schedules using IaC, copy and paste the following sample template and customize as many or as few schedules as you like. Save the file as a .template file (for example: `my-schedules.template`), and then deploy your new template using AWS CloudFormation. For examples of completed schedule templates, refer to [Sample Schedules](#).

```
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'
```

When deploying the template, you must provide ServiceTokenARN for your deployment
of Instance Scheduler on AWS. This ARN can be found within CloudFormation by
navigating to your deployed Instance Scheduler stack, selecting **Outputs**, and looking for
`ServiceInstanceScheduleServiceToken.`

> ⚠️ **Important**
>
> Do not use the DynamoDB console or scheduler CLI to delete or modify schedules and
> periods that were configured using the custom resource. If you do, you will create a conflict
> between the stored parameters in the stack and the values in the table. Also, do not use
> periods configured using the custom resource in schedules created using the DynamoDB
> console or the scheduler CLI.

Before you delete the main Instance Scheduler stack, you must delete all additional stacks that
contain schedules and periods created using the custom resource because the custom resource
stacks contain dependencies on the main stack's DynamoDB table.

In the configuration DynamoDB table, schedules and periods that were configured with the custom resource can be identified by the **configured_in_stack** attribute. The attribute contains the Amazon Resource Name of the stack that was used to create the item.

Advanced features

EC2 Auto Scaling Group scheduling

Instance Scheduler on AWS supports scheduling of EC2 Auto Scaling groups (ASGs) using Scheduled Scaling Actions. This differs from the implementation of EC2/RDS scheduling and will be further explained in this section

Refer to [Scheduled scaling for Amazon EC2 Auto Scaling](#) for more information on Scheduled Scaling Actions.

ASG scheduling overview

ASGs can be scheduled by applying a schedule tag as described in [Tagging instances for scheduling](#)

Scheduled scaling rules will then be managed for your ASG by two systems:

First, an ASG orchestrator Lambda function runs hourly and initiates an ASG handler function for each of your scheduled accounts/regions. This function will look for newly tagged ASGs or ASGs whose configured scheduled scaling actions have become stale. It will then reconfigure all scheduled scaling actions that start with the ASG action name prefix (specified at solution deployment) to match the associated schedule.

Second, when a schedule is updated in the scheduler configuration table, a DynamoDB stream will initiate (via the Schedule Update Handler Lambda function) addition ASG handler requests that will update the scheduled scaling actions on all ASGs tagged with the newly updated schedule.

Definition of Running/Stopped for ASGs

When an Auto Scaling Group is configured, a user specifies a minimum, desired, and maximum capacity for that ASG. Instance Scheduler refers to these values as the min-desired-max of an ASG.

When Instance Scheduler first configures Scheduled Scaling Actions for an ASG, the currently configured min-desired-max values will be used to define the ***running state*** of the ASG. If the ASG is currently configured with a min-desired-max of 0-0-0, Instance Scheduler will report an error and will not configure any Scheduled Scaling Actions until a new min-desired-max has been configured that can be used to define a running state for the ASG.

When updating the Scheduled Scaling Actions for an ASG, Instance Scheduler will look at the current min-desired-max at the time of the update and use those values to define the new **running state** for the schedule. If the min-desired-max is currently 0-0-0 at the time of the update, the previous **running state** will be used.

For all ASGs the **stopped state** is defined as a min-desired-max of 0-0-0.

ASG Scheduled Tag

When an auto scaling group is scheduled by the solution, an auto scaling group scheduled tag is added to the auto scaling group. The tag contains the below information in JSON format:

Key	Value Type	Value
schedule	String	Schedule name aligned with the scheduler configuration table.
ttl	String	Until when the tag is valid.
min_size	Integer	Auto scaling group min size when scheduled.
max_size	Integer	Auto scaling group max size when scheduled.
schedule	Integer	Auto scaling group desired capacity when scheduled.

The presence of a valid Scheduled Tag that has not yet expired its TTL indicates to Instance Scheduler that an ASG has been correctly configured for scheduling. This tag can be manually deleted to forcefully cause Instance Scheduler to reconfigure Scheduled Scaling Actions on an ASG during the next ASG scheduling run.

Limitations

ASG scheduling is performed by converting Instance Scheduler on AWS schedules into scheduled scaling rules compatible with the ASG service. This translation works best for simple single-period schedules that do not use complex cron expressions.

The following schedule features are not supported for ASG scheduling:

- Advanced schedule flags such as *enforced* and *retain running.*

- Nth weekday, nearest weekday, and last weekday expressions in periods.

- Multi-period schedules with immediately adjacent or overlapping periods.*

*when configuring scheduled scaling actions for multi-period schedules, Instance Scheduler on AWS directly translate the beginning/end of periods to start/stop actions for the ASG even when another overlapping or adjacent period would normally cause that action to be skipped.

Monitor the solution

Logging and notifications

Instance Scheduler on AWS leverages Amazon CloudWatch Logs for logging. This solution logs processing information for each tagged instance, the results of the period evaluation for the instance, the desired state of the instance during that period, the applied action, and debugging messages. For more information, refer to [Solution resources](#).

Warning and error messages are also published to a solution-created Amazon SNS topic, which sends messages to a subscribed email address. For details, refer to What is Amazon SNS? in the *Amazon SNS Developer Guide*. You can find the name of the Amazon SNS topic in the **Outputs** tab of the solution stack.

Log files

The Instance Scheduler on AWS creates a log group that contains the default AWS Lambda log files and a log group that contains the following log files:

- `InstanceScheduler-yyyymmdd:` Logs general scheduler messages

- `SchedulingOrchestratorHandler-yyyymmdd:` Logs general orchestration information for when scheduling executions are started

- `SchedulerSetupHandler-yyyymmdd:` Logs the output of configuration actions

- `Scheduler-<service>-<account>-<region>-yyyymmdd:` Logs scheduling activity in each service, account, and region

- `CliHandler-yyyymmdd:` Logs requests from the admin CLI

- `Eventbus_request_handler-yyyymmdd`: Logs the calls to the EventBus resources, if the solution is deployed to AWS organizations.

- `CollectConfigurationDescription-yyyymmdd`: Logs configuration description metric data that is sent periodically

Operational insights dashboard

Instance Scheduler on AWS comes with an Operational Insights dashboard that allows you to monitor the operation of the solution and get insight into the running hours that have been saved by using this solution.

To use this dashboard, ensure that **Operational Monitoring** is set to "enabled" in the solution's hub stack parameters in AWS CloudFormation. Then go to AWS CloudWatch and select "Dashboards" from the navigation menu. The dashboards name will be *{stack-name}-Operational-Insights-Dashboard*

The dashboard will display various operational metrics about the operations of your solution including counts of how many instances are currently being managed by the solution, information on when and how many instances are running throughout the day, and an estimation of how many running hours have been saved by shutting instances down. Sample data below:
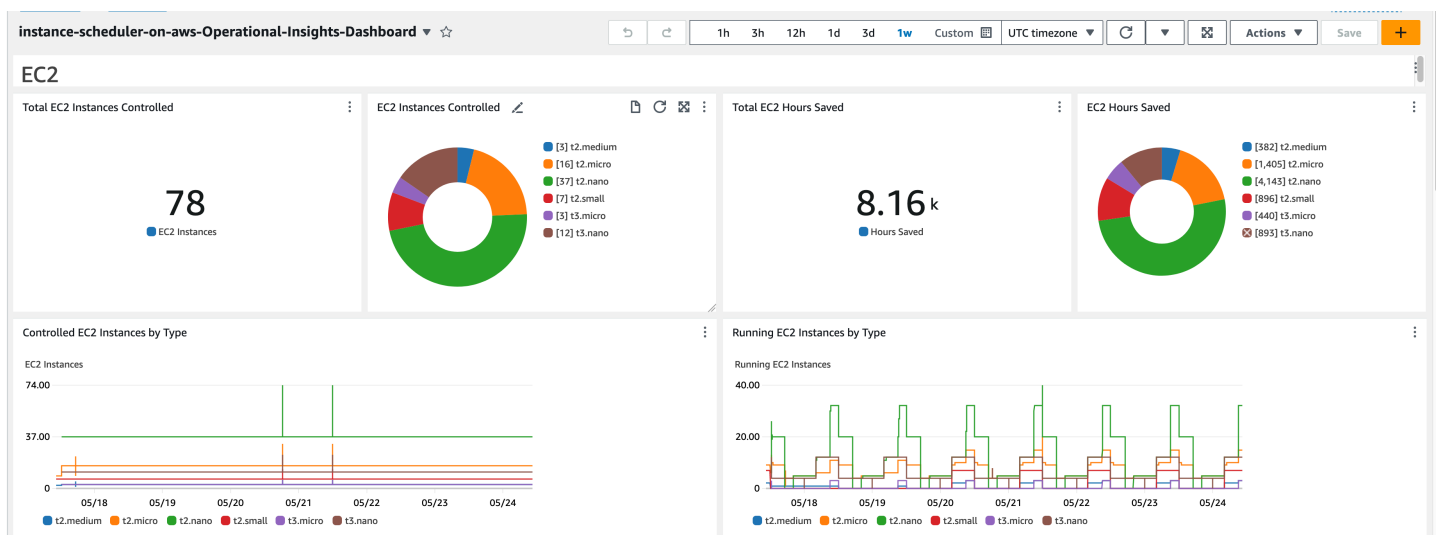
*Instance Scheduler AWS stack in CloudWatch*
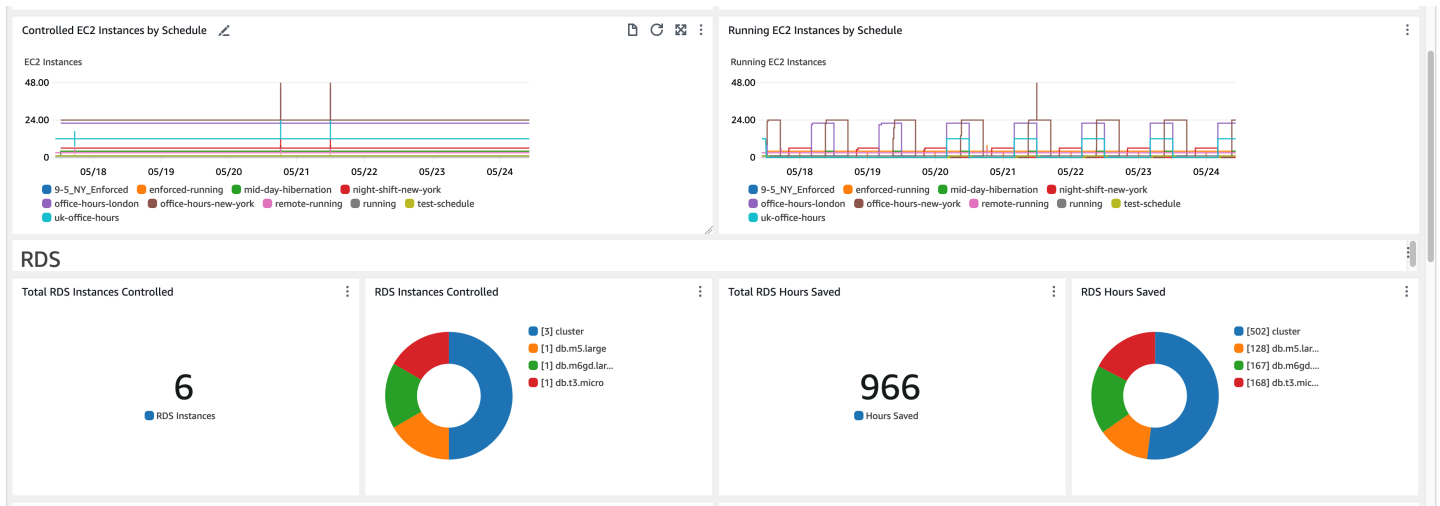
Controlled EC2 instances by Schedule

> **ⓘ Note**
>
> The information in these graphs is dependent upon the scheduling interval configured on the solution hub stack. When updating the solution's scheduling interval, the dashboard will only display scheduling metrics from after the most recent update to the scheduling interval.

The dashboard also provides insight into the health of Lambda functions that are critical to the operation of the solution. If the average Lambda duration for any of the depicted Lambda functions begins to approach the yellow zone, it may be time to increase the Lambda size property on the solution hub stack.
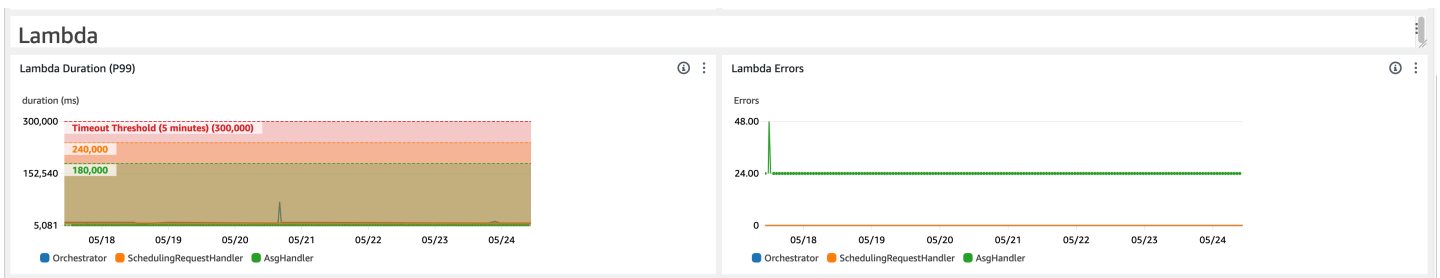
Lambda duration

Depicted: AsgHandler is raising consistent errors over multiple days. This indicates a potential problem with ASG scheduling and should prompt further investigation into the logs for that Lambda

Additional costs associated with this feature

This operational dashboard is powered by custom CloudWatch metrics collected by the solution that will incur additional cost. This feature can be turned off by disabling "Operational Monitoring" on the solution hub stack. This feature costs an additional $3.00/month plus additional scaling costs based on the size of your deployment. The costs are as follows:

Custom CloudWatch Dashboard	$3
Per-schedule metrics	$0.60 per schedule*
Per-instance-type metrics	$0.90 per instance type*
API usage	~$0.10 per account/region

*These costs are tracked per service category (EC2/RDS) and only for schedules/instance types actually used for scheduling. For example, if you have 15 schedules configured, with three for RDS and five for EC2, the total cost will be 8*$0.60 or $4.80/month. Inactive schedules will not be billed.*

Monitor the solution with Service Catalog AppRegistry

The solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both Service Catalog AppRegistry and AWS Systems Manager Application Manager.

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.

- View operations data for the resources of this solution in the context of an application, such as deployment status, CloudWatch alarms, resource configurations, and operational issues.

  The following figure depicts an example of the application view for Instance Scheduler on AWS stack in Application Manager.

Solution stack in Application Manager

Note: You must activate CloudWatch Application Insights, AWS Cost Explorer, and cost allocation tags associated with this solution. They are not activated by default.

Activate CloudWatch Application Insights

1. Sign in to the Systems Manager console.

2. In the navigation pane, choose **Application Manager**.

3. In **Applications**, search for the application name for this solution and select it.

   The application name will have App Registry in the **Application Source** column, and will have a combination of the solution name, Region, account ID, or stack name.

4. In the **Components** tree, choose the application stack you want to activate.

5. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Insights**.
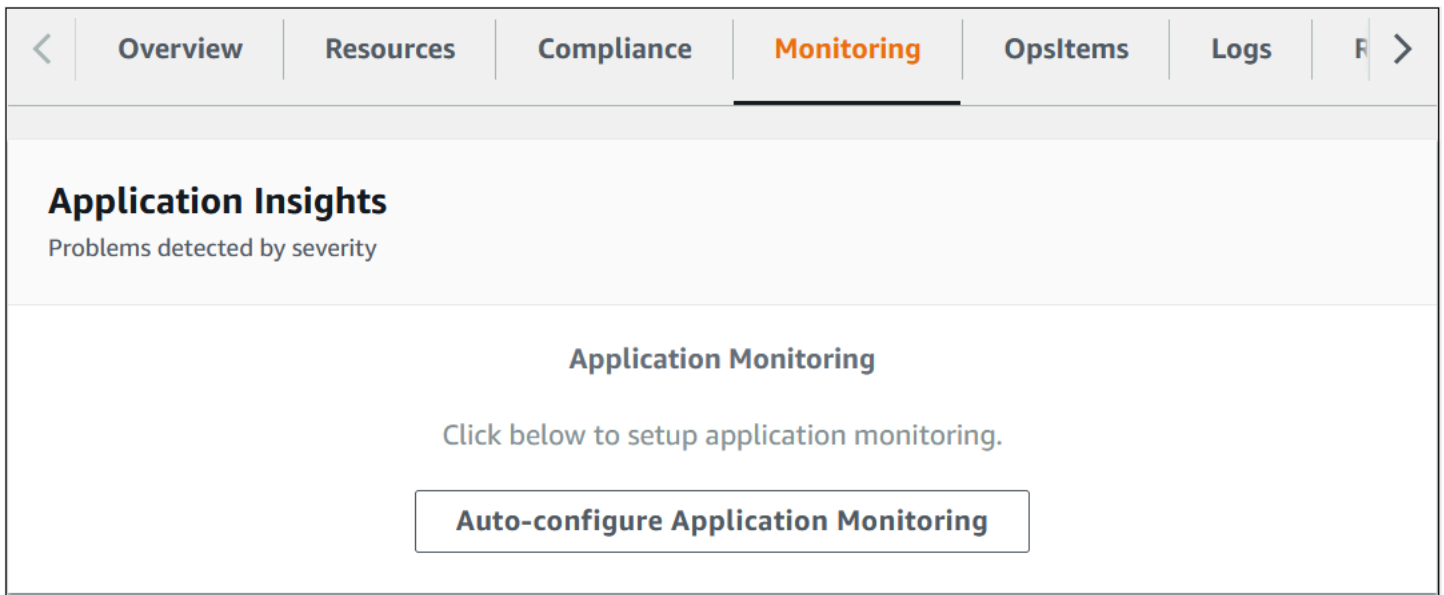
Monitoring for your applications is now activated and the following status box appears:
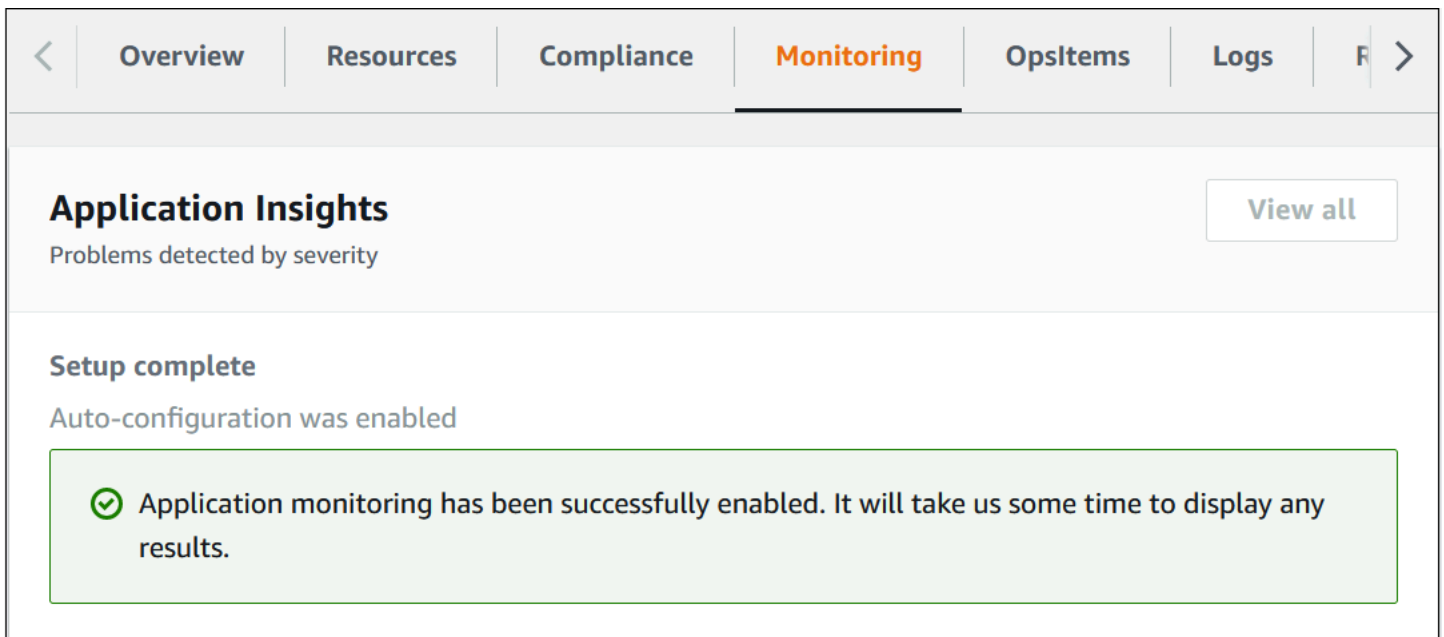

Amazon CloudWatch metrics

This solution provides an option to create a new custom Amazon CloudWatch metric (metric name). To activate this feature, select Yes for the **Enable CloudWatch Metrics** CloudFormation parameter (*<stackname>*:InstanceScheduler).

Each time the AWS Lambda function runs, it updates the metric data for each applicable instance and then applies the appropriate start or stop action. This data includes the name of the schedule, the number of instances associated with that schedule, and the number of running instances.

View Instance Scheduler on AWS metrics

1. In the AWS Management Console, open the Amazon CloudWatch console.

2. In the **Custom Namespaces** drop-down field, choose *<stackname>*:InstanceScheduler.

3. Select the schedule and service dimensions.

4. Select the schedules and services for which you want to view the status.

5. Under **Graphed Metrics** set the statstic of each selected metric to **Sum** and the Period to match the scheduling interval configured for the solution (default = 5, example image uses an interval of 1).

6. At the top of the graph, select a time range to view.
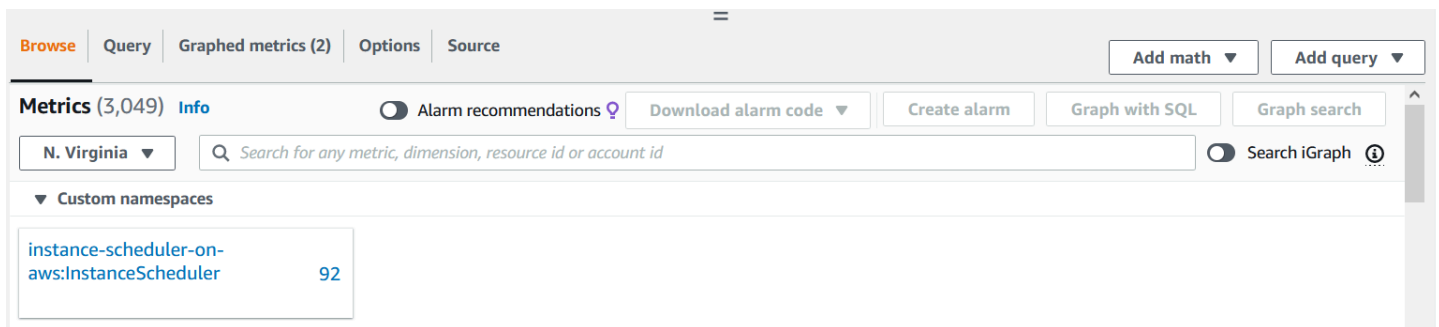
Screenshot of metrics
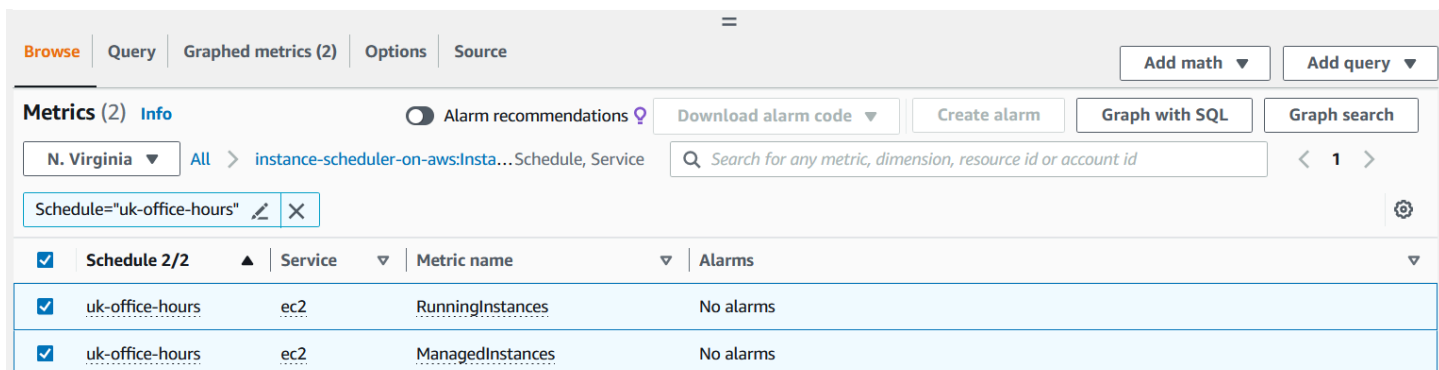
Screenshot of metrics
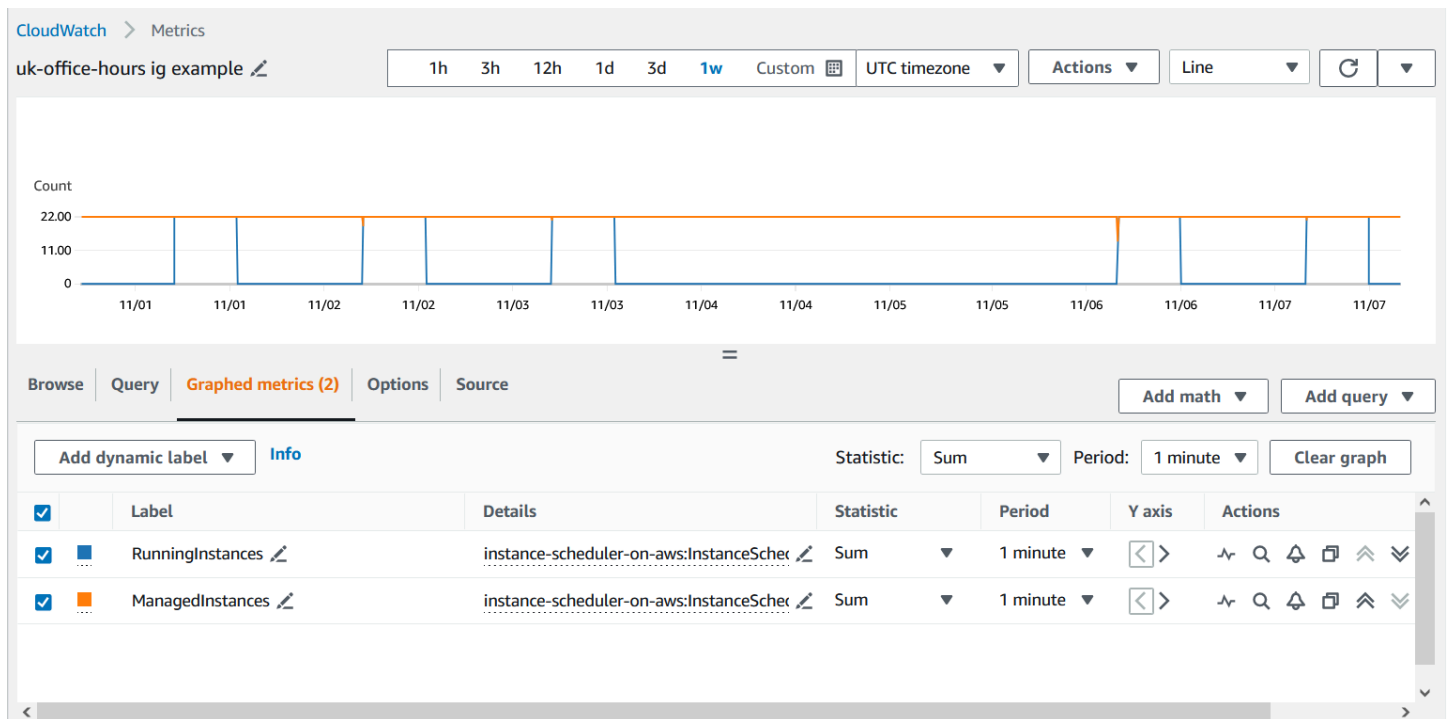
Screenshot of metrics example

Confirm cost tags associated with the solution

After you activate cost allocation tags associated with the solution, you must confirm the cost allocation tags to see the costs for this solution. To confirm cost allocation tags:

1. Sign in to the Systems Manager console.

2. In the navigation pane, choose **Application Manager**.

3. In **Applications**, choose the application name for this solution and select it.

4. In the **Overview** tab, in **Cost**, select **Add user tag**.
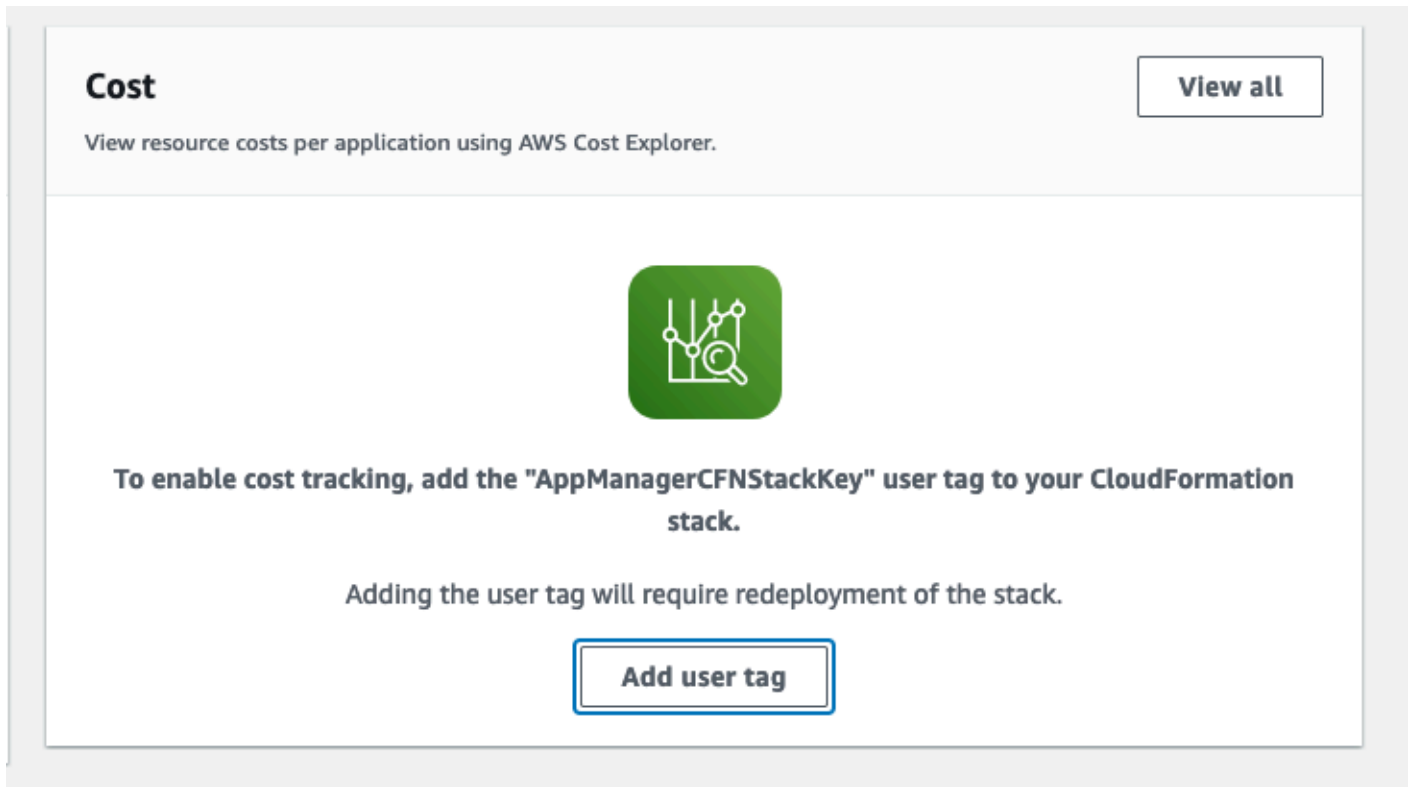
5. On the **Add user tag** page, enter `confirm`, then select **Add user tag**.

The activation process can take up to 24 hours to complete and the tag data to appear.

Activate cost allocation tags associated with the solution

After you activate Cost Explorer, you must activate the cost allocation tags associated with this solution to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization. To activate cost allocation tags:

1. Sign in to the [AWS Billing and Cost Management and Cost Management console](#).

2. In the navigation pane, select **Cost Allocation Tags**.

3. On the **Cost allocation tags** page, filter for the `AppManagerCFNStackKey` tag, then select the tag from the results shown.

4. Choose **Activate**.

AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer which must be first activated. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time. To activate Cost Explorer for the solution:

1. Sign in to the [AWS Cost Management console](#).
2. In the navigation pane, select **Cost Explorer** to view the solution's costs and usage over time.

Performance

If the solution's AWS Lambda function does not process all scheduled instances before its next invocation, the solution logs the error in Amazon CloudWatch Logs, and sends an Amazon Simple Notification Service (Amazon SNS) notification to the error SNS topic. To help ensure that all instances are processed before the next invocation, you can change the default interval at which the Lambda function runs or launch multiple deployments of the solution with different tag names.

If you increase the default interval, this might reduce the granularity of your schedules. For example, a Lambda function set to run on a 15-minute interval will only perform start and stop actions every 15 minutes.

To schedule a large number of instances, we recommend using an interval of at least five minutes and increasing the memory size of the Instance Scheduler's main AWS Lambda function using the **Memory Size** parameter.

Update the solution

If you have previously deployed the solution, follow this procedure to update the Instance Scheduler on AWS CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the [AWS CloudFormation console](#), select your existing `instance-scheduler-on-aws`, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:

- Select **Amazon S3 URL**.

- Copy the link of the [latest template](#).

- Paste the link in the **Amazon S3 URL** box.

- Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.

4. Under **Parameters**, review the parameters for the template and modify them as necessary. For details about the parameters, see [Step 1. Launch the instance scheduler stack](#).

5. Choose **Next**.

6. On the **Configure stack options** page, choose **Next**.

7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.

8. Choose **View change set** and verify the changes.

9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive an **UPDATE_COMPLETE** status in a few minutes.

Repeat the above steps for the aws-instance-scheduler-remote stacks in each of your spoke accounts.

Updating to v1.5.x

Version 1.5.0 changes the way Instance Scheduler on AWS manages cross-account scheduling roles and introduces the ability to optionally manage them through your AWS Organization.

If you don't want to use AWS Organizations, Instance Scheduler on AWS only requires the Account IDs of the spoke accounts you wish to schedule instead of full role ARNs, however the update process will differ slightly if you do not plan to use AWS Organizations.

To update to v1.5.x, do the following:

1. Update the hub template using the normal update instructions while updating the following parameters:

   a. Choose a unique namespace for the solution.

   b. Select whether you would like to **Use AWS Organizations** to manage spoke registration going forward.

     i. If you selected **Yes** replace **Organization ID/Remote Account IDs** with the ID of your AWS Organization.

2. Update all remote stacks in spoke accounts with the following parameters:

   a. Namespace – same as you chose for the hub account.

   b. Use AWS Organizations – same as hub account.

   c. Hub Account ID – Account ID of the hub account (should be unchanged from before).

3. Update the hub template for a second time to configure the remote Account IDs in hub account:

   a. If you selected **Yes** for **Use AWS Organizations** you can skip this step entirely.

   b. Update the hub template for a second time (select **Use current template**).

   c. Replace **Organizations ID/Remote Account IDs** with a comma separated list of {param: ssm-param-name} and/or remote Account IDs for each of your spoke accounts.

Updating from v1.5.0

Instance Scheduler v1.5.0 has a known compatibility issue with AppRegistry that prevents it from being able to directly upgrade to newer versions of the solution.

In order to upgrade v1.5.0 to a newer version of the solution, you must first update to an intermediary stack using the following templates:

Hub Stack: https://solutions-reference.s3.amazonaws.com/aws-instance-scheduler/v1.5.0/aws-instance-scheduler-1.5.0-u.template

Remote Stack: https://solutions-reference.s3.amazonaws.com/aws-instance-scheduler/v1.5.0/aws-instance-scheduler-remote-1.5.0-u.template

Installation of these stacks will disable AppRegistry integration on your deployment, allowing newer versions of the solution to recreate the association.

Updating to v3.x

Updating from 1.5.x to 3.x can be done without interruption to scheduling with the exception of the following:

- "CloudWatch Metrics" feature in 1.5.x has been replaced with the Operational Insights Dashboard.

- Per-schedule metrics in CloudWatch have been moved from Schedule/Service/MetricName →
  Schedule/Service/SchedulingInterval/MetricName
- KMS key ARNs for use with encrypted EBS volumes on EC2 DB instances must now be provided
  to the hub/spoke CloudFormation stack in their respective accounts. (For more information, refer
  to Encrypted EC2 EBS Volumes.)
- The CloudFormation parameter for scheduled services has been broken up into individual
  parameters for each supported service.
- Instance Scheduler 3.0 is not backwards compatible with older versions of the Instance Scheduler
  CLI.

In addition to the above, the schema of the Maintenance Window table has been updated and will
be replaced as part of the update. This will reset tracking for EC2 maintenance windows for the
first few minutes after updating to v3.x and in rare cases may cause instances currently within a
maintenance window to be stopped prematurely immediately following the update, after which
scheduling operations will continue as normal.

Troubleshooting

This section provides troubleshooting instructions for deploying and using the solution.

Known issue resolution provides instructions to mitigate known errors. If these instructions don't address your issue, Contact AWS Support provides instructions for opening an AWS Support case for this solution.

Known issue resolution

Problem: Instances not being scheduled in a remote account

If you notice instances are not being scheduled in a remote account.

Resolution

Update the hub stack with the secondary account ID or complete the following task:

1. In the primary account, navigate to the CloudWatch console

2. In the navigation pane, select **Logs** > **Log Groups**.

3. Select the log group named *<STACK_NAME>*-logs

4. Search for the log stream for the Account ID (remote account).

5. For example, If there is no log stream named with the account ID, go to the DynamoDB console and select the table named *<STACK_NAME>*-<ConfigTable>-<RANDOM>.

6. Select **Explore Items** and select **Run.**

7. Select the item type Config.

8. Check if the attribute remote_account_ids has the account ID.

9. Check if the Account ID is not visible in this attribute.

10 If the solution is configured to aws organizations, then uninstall and reinstall the remote template in the remote account.

11 If the solution is configured to use remote Account IDs, update the cloudformation parameter **Provide Organization Id OR List of Remote Account IDs** with the list of account IDs where the instances are to be scheduled and where the remote template is deployed.

Problem: Solution update from any version v1.3.x to v1.5.0

Lambda function is not working, for example, scheduling is not being performed.

Resolution

1. Ensure that the update to the CloudFormation stack has been completed.

2. Go to the CloudFormation console and select the solution stack .

3. Select the Resources tab.

4. Search for **Main** in the **Search Resources** filter.

5. Select the Lambda function in the **Physical ID** column.

6. In the Lambda console select **Configuration**.

7. Select the environment **Variables**.

8. Ensure the following environment variables are available.

   - ACCOUNT
   - CONFIG_TABLE
   - DDB_TABLE_NAME
   - ENABLE_SSM_MAINTENANCE_WINDOWS
   - ISSUES_TOPIC_ARN
   - LOG_GROUP
   - MAINTENANCE_WINDOW_TABLE
   - METRICS_URL
   - SCHEDULER_FREQUENCY
   - SEND_METRICS
   - SOLUTION_ID
   - STACK_ID
   - STACK_NAME
   - START_EC2_BATCH_SIZE
   - STATE_TABLE
   - TAG_NAME
   - TRACE
   - USER_AGENT

- USER_AGENT_EXTRA

- UUID_KEY

Problem: Encrypted EC2 instances not starting

Instance Scheduler is reporting that EC2 instances with encrypted EBS volumes are being started, but they never actually start.

Resolution

Refer to Encrypted EC2 EBS Volumes for how to grant Instance Scheduler access to be able to schedule EC2 instances with encrypted EBS volumes

Contact AWS Support

If you have AWS Developer Support, AWS Business Support, or AWS Enterprise Support, you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

Create case

1. Sign in to Support Center.

2. Choose **Create case**.

How can we help?

1. Choose **Technical**.

2. For **Service**, select **Solutions**.

3. For **Category**, select **Instance Scheduler on AWS (Linux or Windows)**.

4. For **Severity**, select the option that best matches your use case.

5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

Additional information

1. For **Subject**, enter text summarizing your question or issue.

2. For **Description**, describe the issue in detail.

3. Choose **Attach files**.

4. Attach the information that AWS Support needs to process the request.

Help us resolve your case faster

1. Enter the requested information.

2. Choose **Next step: Solve now or contact us**.

Solve now or contact us

1. Review the **Solve now** solutions.

2. If you can't resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

Uninstall the solution

> ⚠️ **Important**
>
> When uninstalling the solution, make sure to uninstall all custom schedule stacks before uninstalling the solution itself.

You can uninstall the Instance Scheduler on AWS solution from the AWS Management Console or by using the AWS Command Line Interface. To uninstall the solution, delete the hub stack in AWS Cloud Formation along with all installed remote stacks. You can then remove any scheduling tags that had been applied to instances for scheduling purposes.

> ℹ️ **Note**
>
> If **Protect DynamoDB Tables** is enabled on the solution's hub stack, CloudFormation will retain the solution's DynamoDB tables and KMS key rather than deleting them. If you wish to delete these resources, ensure that this property is set to **Disabled** before deleting the hub stack. Alternatively, you can delete them manually after the hub stack has already been deleted.

Using the AWS Management Console

1. Sign in to the [AWS CloudFormation console](#).
2. On the **Stacks** page, select this solution's installation stack.
3. Choose **Delete**.

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to [What Is the AWS Command Line Interface](#) in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name
```

```
        <installation-stack-name>
```

Developer guide

This section provides the source code for the solution and list the sections added here and include links to each subtopic.

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others.

The Instance Scheduler on AWS templates are generated using the AWS CDK. See the README.md file for additional information.

Reference

This section includes information about an optional feature for collecting unique metrics for this solution, quotas, pointers to related resources, and a list of builders who contributed to this solution.

Anonymized data collection

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected periodically and sent to AWS:

- **Solution ID** - The AWS Solution identifier.
- **Unique ID (UUID)** - Randomly generated, unique identifier for each Instance Scheduler on AWS deployment.
- **Timestamp** - Data-collection timestamp.
- **Scheduling Actions** - Count of how often instance scheduler takes certain actions against instances and how long it took to perform those actions.

  Example data:

  ```
  num_unique_schedules: 4
  num_instances_scanned: 23
  duration_seconds: 6.7
  actions: [
    {
      action: Started
      instanceType: a1.medium
      instances: 8
      service: ec2
    },
    …
  ]
  ```

- **Instance Count** - Count of the number of instances and schedules being processed in each Region.

  Example data:

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

- **Deployment Description** An overview description of the deployment:

  Example data:

```
services: [ec2, rds]
regions: [us-east-1, us-east-2]
num_accounts: 6
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** How often each feature of the Scheduler CLI is used.

  Example data:

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

AWS owns the data gathered through this survey. Data collection is subject to the [AWS Privacy Policy](#). To opt out of this feature, complete the following steps before launching the CloudFormation template.

1. Download the `instance-scheduler-on-aws.template` [AWS CloudFormation template](#) to your local hard drive.

2. Open the CloudFormation template with a text editor.

3. Modify the CloudFormation template mapping section from:

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

to

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

4. Sign in to the [AWS CloudFormation console](#).

5. Select **Create stack**.

6. On the **Create stack** page, **Specify template** section, select **Upload a template file**.

7. Under **Upload a template file**, choose **Choose file** and select the edited template from your local drive.

8. Choose **Next** and follow the steps in the [Launch the stack](#) section of this guide.

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the [services implemented in this solution](#). For more information, see [AWS service quotas](#).

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the Service endpoints and quotas page in the PDF instead.

AWS CloudFormation quotas

Your AWS account has AWS CloudFormation quotas that you should be aware of when launching the stack in this solution. By understanding these quotas, you can avoid limitation errors that would prevent you from deploying this solution successfully. For more information, see AWS CloudFormation quotas in the in the *AWS CloudFormation User's Guide*.

AWS Lambda quotas

Your account has AWS Lambda, Concurrent Execution quota of 1000, if the solution is used in an account where there are other workloads running and using Lambda then this Quota should be set to an appropriate value. This value is adjustable, for more information refer to AWS Lambda Getting started guide.

Related resources

The Resource Scheduler is a similar to Instance Scheduler on AWS, but its implementation differs in the following ways:

Instance Scheduler on AWS uses a Lambda function to frequently evaluate the schedules stored in its configuration and checks if the instances are in the desired state. The Resource scheduler quicksetup uses start and stop times to perform start and stop actions using SSM runbooks. This occurs one time when the current time is equal to the start time or the current time is past the start time.

Instance Scheduler on AWS currently enables scheduling for EC2, RDS, and Aurora Clusters. Resource scheduler only schedules or starts and stops EC2 instances.

Use Resource scheduler to identify EC2 instances and start/stop them at specific times.

Use Instance Scheduler on AWS when the accounts have to regulary scanned to start/stop instances.

The table identifies which solution is better based on scenarios.

Scenario	Resource Scheduler	Instance Scheduler on AWS
Schedule Amazon Neptune instances	No	Yes
Schedule Amazon DocumentDB instances	No	Yes
Schedule Auto Scaling group instances	No	Yes
Schedule EC2 instances	Yes	Yes
Schedule RDS instances	No	Yes
Schedule Aurora Clusters	No	Yes
Manage Schedules in a Single Account (hub account)	No	Yes
Manage Schedules in individual accounts	Yes	No
Change Calendar Integration	Yes	No
Start and Stop actions only	Yes	No
Monitor instances periodically and start and stop based on instance current state	No	Yes

Contributors

- Arie Leeuwesteijn
- Mahmoud ElZayet
- Ruald Andreae
- Nikhil Reddy

- Caleb Pearson

- Jason DiDomenico

- Max Granat

- Pratyush Das

- Amanda Jones

- Kevin Hargita

- Beomseok Lee

Revisions

Date	Change
February 2018	Initial release
July 2018	Added clarification on begintime and endtime; added an example schedule configuration
October 2018	Added information on encrypted Amazon EBS volumes; added clarification on stack name length and Amazon RDS tag restrictions
May 2019	Added information on hibernating Amazon Ec2 instances, scheduling RDS DB instances that are part of an Amazon Aurora cluster, new schedule CLI arguments, SSM maintenance windows, Parameter Store integration, and clarification on schedules with adjacent running periods
October 2019	Added information regarding the AWS Regions where Instance Scheduler on AWS has been validated
March 2020	Bug fixes
June 2020	Clarified time zone information for period rule start and stop times; for information about updates and changes for v1.3.2, refer to the CHANGELOG.md file in the GitHub repository.
September 2020	AWS Cloud Development Kit (AWS CDK) (AWS CDK) conversion and documentation enhancements; for more information about changes to v1.3.3, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
October 2020	Updated the template in Appendix D and updated the documentation for the Hibernate Field.
April 2021	Updated the SSM Maintenance Window functionality for EC2 instance scheduling, Configured solution to work in AWS GovCloud, and bug fixes. For more information about v1.4.0, refer to the CHANGELOG.md file in the GitHub repository.
May 2022	Minor update and bug fixes. For more information about v1.4.1, refer to the CHANGELOG.md file in the GitHub repository.
January 2023	Minor update and bug fixes. For more information about v1.4.2, refer to the CHANGELOG.md file in the GitHub repository.
May 2023	Deprecated override_status scheduling flag, Resource Scheduler vs Instance Scheduler, add feature for AWS Organizations, simplify cross account scheduling to eliminate the need for cross account roles. For more information about v1.5.0, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
July 2023	Added guidance on how to manage Infrastructure as Code, and updating to version 1.5.x. Additional Troubleshooting, Features and Benefits added. For more information about v1.5.1, refer to the CHANGELOG.md file in the GitHub repository.
October 2023	Minor update, security patching, and added additional sample cost tables in the documentation. For more information about v1.5.2, refer to the CHANGELOG.md file in the GitHub repository.
October 2023	Release v1.5.3 Security patch. For more information, refer to the CHANGELOG.md file in the GitHub repository.
December 2023	Documentation update to resolve discrepencies, added actional section to the AppRegistry section. For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2024	Release v1.5.4 Security patch. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2024	Release v1.5.5 Security patch. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
June 2024	Release v3.0.0: Major release. Added support for EC2 Auto Scaling Groups, Amazon Neptune, and Amazon DocumentDB. Added operational insights dashboard. Documentation update to include Operator and Developer guides. For more information, refer to the CHANGELOG.md file in the GitHub repository.
June 2024	v3.0.1: Minor update. Modified CLI install, Uninstall solution instructions and updated links. For more information, refer to the CHANGELOG.mdfile in the GitHub repository.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Instance Scheduler on AWS is licensed under the terms of the Apache License Version 2.0.