



Implementation Guide

# Instance Scheduler on AWS



# Instance Scheduler on AWS: Implementation Guide

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

# Table of Contents

<b>Solution overview .....</b>	<b>1</b>
Features and benefits .....	2
Use cases .....	3
Concepts and definitions .....	3
<b>Architecture overview .....</b>	<b>5</b>
Architecture diagram .....	5
AWS Well-Architected design considerations .....	7
<b>Architecture details .....</b>	<b>10</b>
Scheduler configuration table .....	10
Schedules .....	10
Periods .....	10
Time Zone .....	10
Hibernate Field .....	11
Enforced field .....	11
Retain running field .....	11
SSM maintenance window field .....	11
Override status field .....	12
Instance type .....	13
Schedule definitions .....	13
Periods .....	15
Start and stop times .....	15
Adjacent periods .....	17
Days of the week .....	17
Days of the month .....	17
Months .....	17
Period definitions .....	17
Cross-account instance scheduling .....	19
Enabling cross account scheduling using Account IDs .....	20
Enabling cross account scheduling using AWS Organization ID .....	20
Managing Account IDs with AWS Systems Manager Parameter Store .....	20
Automated tagging .....	20
Scheduler command line interface .....	21
AWS services in this solution .....	22
<b>Plan your deployment .....</b>	<b>24</b>

Cost .....	24
Pricing examples (monthly) .....	25
Security .....	29
AWS Key Management System .....	29
Amazon Identity Access Management .....	29
Supported AWS Regions .....	30
Quotas .....	31
Quotas for AWS services in this solution .....	31
AWS Cloudformation quotas .....	31
AWS Lambda quotas .....	32
<b>Deploy the solution .....</b>	<b>33</b>
Deployment considerations .....	33
Partial automation .....	33
Instance shutdown behavior .....	33
Amazon RDS maintenance window .....	34
Global Configuration Settings .....	34
Performance .....	34
Encrypted Amazon EBS Volumes .....	35
Logging and notifications .....	35
AWS Cloudformation templates .....	35
Deployment process overview .....	36
Step 1: Launch the instance scheduler hub stack .....	37
Step 2 (Optional): Launch the remote stack in secondary accounts .....	44
Configure the solution .....	45
Step 3: Configure schedules and periods .....	46
Step 4: Tag your instances .....	47
<b>Monitoring this solution with AppRegistry .....</b>	<b>48</b>
Activate CloudWatch Application Insights .....	49
Activate AWS Cost Explorer .....	50
Confirm cost tags associated with the solution .....	50
Activate cost allocation tags associated with the solution .....	51
<b>Update the solution .....</b>	<b>52</b>
Updating to v1.5x .....	52
<b>Troubleshooting .....</b>	<b>54</b>
Known issue resolution .....	54
Problem: Instances not being scheduled in a remote account .....	54

Problem: Solution update from any version v1.3.x (or earlier) to v1.5.0 .....	55
Problem: Encrypted EC2 Instances Not Starting .....	56
Contact AWS Support .....	56
Create case .....	57
How can we help? .....	57
Additional information .....	57
Help us resolve your case faster .....	57
Solve now or contact us .....	57
<b>Uninstall the solution .....</b>	<b>58</b>
Using the AWS Management Console .....	58
Using AWS Command Line Interface .....	58
<b>Developer guide .....</b>	<b>59</b>
Source code .....	59
Scheduler CLI .....	59
Prerequisites .....	59
Credentials .....	59
Install the Scheduler CLI .....	60
Command structure .....	61
Common arguments .....	61
Available commands .....	62
Solution resources .....	74
Log files .....	75
Manage schedules using Infrastructure as Code (IaC) .....	76
Sample schedules .....	78
Standard 9-5 working hours .....	78
Stop instances after 5 PM .....	80
Stop instances over the weekend .....	82
Monitor schedule activity in Amazon CloudWatch .....	85
View Instance Scheduler on AWS metrics .....	85
<b>Reference .....</b>	<b>87</b>
Anonymized data collection .....	87
Related resources .....	89
Contributors .....	90
<b>Revisions .....</b>	<b>92</b>
<b>Notices .....</b>	<b>94</b>

# Automate starting and stopping AWS instances

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

The Instance Scheduler on AWS solution automates the starting and stopping of [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 in a production environment 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 Amazon 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.	<a href="#">Cost</a>
The estimated baseline cost for running a small deployment of this solution in the US East (Northern Virginia) Region is USD \$5.00 per month.	
Understand the security considerations for this solution.	<a href="#">the section called "Security"</a>

If you want to . . .	Read . . .
Configure schedules.	<a href="#">Configure the solution</a>
Know which AWS Regions are supported for this solution.	<a href="#">Supported AWS Regions</a>
View or download the AWS CloudFormation template included in this solution to automatically deploy the infrastructure resources (the “stack”) for this solution.	<a href="#">AWS CloudFormation templates</a>
Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) to deploy the solution.	<a href="#">GitHub repository</a>

## Features and benefits

The Instance Scheduler 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 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](#) section.

### Manage schedules using Infrastructure as Code (IaC)

Instance Scheduler on AWS provides an AWS CloudFormation Custom Resource that can be used to manage schedules using Infrastructure as Code. For more information, refer to [Manage Schedules Using Infrastructure as Code \(IaC\)](#) section.

## Integration with SSM Maintenance Windows

For Amazon EC2 instances, Instance Scheduler on AWS can integrate with SSM 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 [SSM maintenance window field](#) section.

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

This solution includes an [AppRegistry](#) resource to register the solution's CloudFormation template and its underlying resources as an application in both 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**

Amazon EC2 and Amazon RDS resources that are able to be scheduled.

**regular business hours**

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

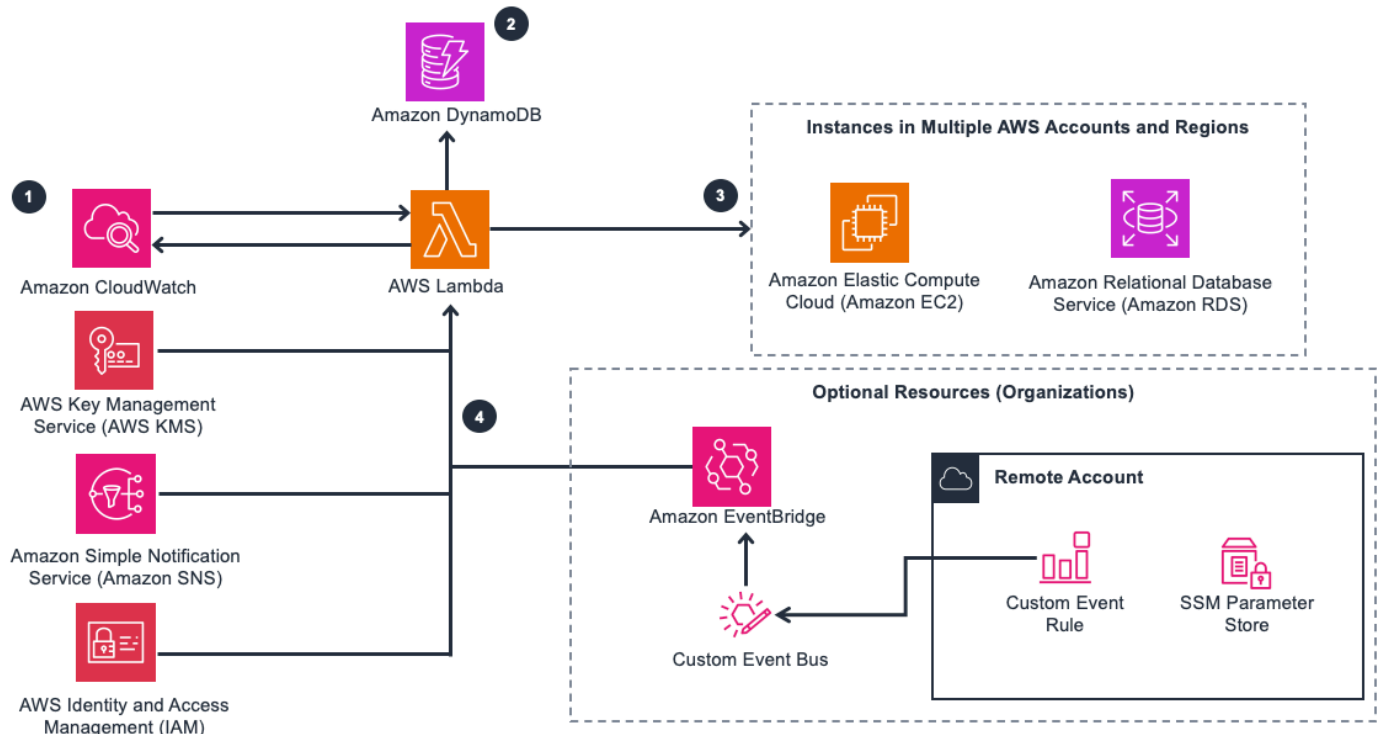
For a general reference of AWS terms, see the [AWS glossary](#) in the AWS General Reference.

# Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

## Architecture diagram

Deploying this solution deploys the following components in your AWS account.



## Instance Scheduler on the AWS Cloud

Instance Scheduler on AWS deploys an [Amazon EventBridge](#) event bus when the customer selects the CloudFormation parameter “Use AWS Organizations”. Optionally, Instance Scheduler on AWS deploys Amazon EventBridge diagram only shows the event bus, which is deployed in addition to the resources shown.

The high-level process flow for the solution components deployed with the AWS CloudFormation template is as follows:

1. The AWS CloudFormation template sets up an [Amazon CloudWatch](#) event at a customer-defined interval. This event invokes the Instance Scheduler [AWS Lambda](#) function. During

configuration, the user defines the AWS Regions and accounts, as well as a *custom tag* that Instance Scheduler on AWS uses to associate schedules with applicable [Amazon EC2](#), [Amazon RDS](#) instances, and clusters.

2. These values are stored in [Amazon DynamoDB](#), and the Lambda function retrieves them each time it runs. You can then apply the custom tag to applicable instances.
3. During initial configuration of the Instance Scheduler, you define a tag *key* you will use to identify applicable Amazon EC2 and Amazon RDS instances. When you create a schedule, the name you specify is used as the tag *value* that identifies the schedule you want to apply to the tagged resource.

For example, a user might use the solution's default tag name (tag key) `Schedule` and create a schedule called `uk-office-hours`. To identify an instance that will use the `uk-office-hours` schedule, the user adds the `Schedule` tag key with a value of `uk-office-hours`.

4. **Optional:** When you select the CloudFormation parameter **Using AWS Organizations** as Yes and provide a valid organization id, an additional resource Event Bus resource is created which will receive events from the CloudWatch events from remote accounts. The events from the remote account will provide account id which will added to the solution configuration in DynamoDB.

#### Note

AWS CloudFormation resources are created from [AWS Cloud Development Kit \(AWS CDK\)](#) constructs.

The Lambda function uses [AWS Identity Access Management](#) (AWS IAM) for permission requirements for your resources, and [AWS Key Management System](#) (AWS KMS) for encryption of the [Amazon Simple Notification Service](#) (Amazon SNS topic) and Dynamo DB tables.

Each time the solution's Lambda function runs, 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, and 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 Amazon EC2 or Amazon RDS 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 rule 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 the name of the schedule, the number of instances associated with that schedule, and the number of running instances as an optional custom metric in Amazon CloudWatch (refer to [Amazon CloudWatch Metrics](#)).

#### 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.
- 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 [the section called “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.

# Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

## Scheduler configuration table

When deployed, the Instance Scheduler on AWS creates an Amazon DynamoDB table that contains global configuration settings. To modify these global configuration settings after the solution is deployed, update the AWS CloudFormation stack. **Do not modify these values in the DynamoDB table.** If you modify the values in the DynamoDB table, you will create a conflict between the stored parameters in the stack and the values in the table.

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](#).

## Schedules

Schedules specify when Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS) instances 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 [Periods](#).

## 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.

## SSM maintenance window field (only applies to EC2 instances)

The `ssm-maintenance-window` field allows you to automatically add an AWS Systems Manager maintenance window as a running period to a schedule. When you specify the name of a maintenance window that exists in the same account and AWS Region as your deployed stack to schedule your Amazon EC2 instances, the solution will start the instance 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, and if the maintenance event is completed.



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 only be picked up by the lambda any time between 1-5 minutes.

This solution uses the AWS Lambda frequency you specified during initial configuration to determine how long before the maintenance window to start your instance. If you set the **Frequency** AWS CloudFormation parameter to 10 minutes or less, the solution will start the instance 10 minutes before the maintenance window. If you set the frequency to greater than 10 minutes, the scheduler will start the instance the same number of minutes as the frequency you specified. For example, if you set the frequency to 30 minutes, the scheduler will start the instance 30 minutes before the maintenance window.

The **Enable SSM Maintenance windows** in the solution stack should be set to Yes and the stack should be updated, so that the solution will start loading the SSM maintenance window into the DynamoDB table, which will be used when the AWS Lambda function runs.

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

## Override status field

Schedules also contain an `override_status` field that allows you to temporarily override the solution's start and stop actions. If you set the field to `running`, the solution will start but not stop the applicable instance. The instance will run until you stop it manually. If you set the field to `stopped`, the solution will stop but not start the applicable instance. The instance will not run until you manually start it.

### Note

If you set the `override_status` field to `running` but use the `enforced` field to prevent an instance from being manually started outside of a running period, the solution will stop the instance. If you set the `override_status` field to `stopped` but use the `enforced` field to prevent an instance from being manually stopped during a running period, the solution will restart the instance.

## Instance type

For Amazon EC2 instances only, a schedule allows you to specify an optional instance type for each period in a schedule. When you specify an instance type in the period, the solution will start the Amazon EC2 instance with the applicable 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, if the specified instance type is compatible with the instance configuration of the running instance. 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
<code>description</code>	An optional description of the schedule.
<code>hibernate</code>	Choose whether to hibernate Amazon EC2 instances running Amazon Linux. When this field is set to <code>true</code> , the scheduler will hibernate instances when it stops them. Note that your instances must <a href="#">turn on hibernation</a> and must meet the <a href="#">hibernation prerequisites</a> .
<code>enforced</code>	Choose whether to enforce the schedule. When this field is set to <code>true</code> , 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.
<code>name</code>	The name used to identify the schedule. This name must be unique.

Field	Description
periods	<p>The name of the periods that are used in this schedule. Enter the name(s) exactly as it appears in the period name field.</p> <p>You can also specify an instance type for the period using the syntax <code>&lt;period-name&gt;@&lt;instance-type&gt;</code>. For example, <code>weekdays@t2.large</code> .</p>
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.
ssm_maintenance_window	<p>Choose whether to add an AWS Systems Manager maintenance window as a running period. Enter the name of a maintenance window.</p> <div> <p> <b>Note</b></p> <p>To use this field, you must also set the <code>use_maintenance_window</code> parameter to <code>true</code>.</p> </div> <div> <p> <b>Note</b></p> <p>This feature only applies to EC2 instances.</p> </div>
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 <code>true</code> .
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 <b>TZ</b> column of the <a href="#">List of tz database time zones</a> .

Field	Description
<code>use_maintenance_window</code>	Choose whether to add a maintenance window as a running period to an instance schedule, or to add an AWS Systems Manager maintenance window as a running period to an Amazon EC2 instance schedule. For more information, refer to <a href="#">SSM Maintenance Window Field</a> .
<code>use_metrics</code>	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.

## Periods

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 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 at the end of each day unless you have scheduled an adjacent period.

Similarly, if you specify a stop time only, the instance must be started manually. If you don't specify either time, the solution uses the days of the week, days of the month, or months rules to start and stop instances.

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 specify both a `begintime` and `endtime` in your period. If no time is specified, this solution will use the time specified in the other periods to determine when to start and stop your instances. For example, if in one period you specify a `begintime` of 9 AM with no `endtime` because you want the instance to run until you manually stop it, Instance Scheduler on AWS will stop the instance at 12:00 AM the following day. The solution will evaluate whether to keep the instance started or stopped until it finds another period for that specific day.

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.



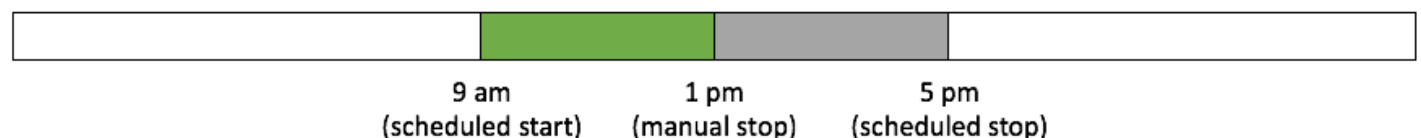
### 9-5 scheduled start and stop

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 manual start

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  $n^{\text{th}}$  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). For more information, refer to [Configure the solution](#).

### 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  $n^{\text{th}}$  day of the month, the last day of the month, or the nearest weekday to a specific date. For more information, refer to [Configure the solution](#).

### Months

The `months` field defines which months an instance will run. You can specify a list of months, a range of months, or every  $n^{\text{th}}$  month. The solution supports abbreviated month names (Jan) and numbers (1). For more information, refer to [Step 3: Configure schedules and periods](#).

### 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
<code>begintime</code>	The time, in HH:MM format, that the instance will start.
<code>description</code>	An optional description of the period.
<code>endtime</code>	The time, in HH:MM format, that the instance will stop.
<code>months</code>	<p>Enter a comma-delimited list of months, or a hyphenated range of months, during which the instance will run. For example, enter <code>jan, feb, mar</code> or <code>1, 2, 3</code> to run an instance during those months. Or, you can enter <code>jan-mar</code> or <code>1-3</code>.</p> <p>You can also schedule an instance to run every <math>n^{\text{th}}</math> month or every <math>n^{\text{th}}</math> month in a range. For example, enter <code>Jan/3</code> or <code>1/3</code> to run an instance every third month starting in January. Enter <code>Jan-Jul/2</code> to run every other month from January to July.</p>
<code>monthdays</code>	<p>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 <code>1, 2, 3</code> or <code>1-3</code> to run an instance during the first three days of the month. You can also enter multiple ranges. For example, enter <code>1-3, 7-9</code> to run an instance from the 1<sup>st</sup> to the 3<sup>rd</sup> and the 7<sup>th</sup> through the 9<sup>th</sup>.</p> <p>You can also schedule an instance to run every <math>n^{\text{th}}</math> day of the month or every <math>n^{\text{th}}</math> day of the month in a range. For example, enter <code>1/7</code> to run an instance every seventh day starting on the 1<sup>st</sup>. Enter <code>1-15/2</code> to run an instance every other day from the 1<sup>st</sup> to the 15<sup>th</sup>.</p>

Field	Description
	Enter <b>L</b> to run an instance on the last day of the month. Enter a date and <b>W</b> to run an instance on the nearest weekday to the specified date. For example, enter <b>15W</b> to run an instance on the nearest weekday to the 15 <sup>th</sup> .
name	The name used to identify the period. This name must be unique.
weekdays	<p>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 <b>0, 1, 2</b> or <b>0-2</b> to run an instance Monday through Wednesday. You can also enter multiple ranges. For example, enter <b>0-2, 4-6</b> to run an instance every day except Thursday.</p> <p>You can also schedule an instance to run every n<sup>th</sup> occurrence of a weekday in the month. For example, enter <b>Mon#1</b> or <b>0#1</b> to run an instance the first Monday of the month.</p> <p>Enter a day and <b>L</b> to run an instance on the last occurrence of that weekday in the month. For example, enter <b>friL</b> or <b>4L</b> to run an instance on the last Friday of the month.</p>

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.

## 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, launch the main solution template (instance-scheduler-on-aws) in the primary account. Then, 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). Update the main 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, launch the main solution template (instance-scheduler-on-aws) in the primary account with the CloudFormation parameter **Using AWS Organizations?** As Yes , and provide the organization ID in the cloudformation parameters **Provide Organization ID OR List of Remote Account IDs**. When the main solution is completely installed, then launch the remote template (instance-scheduler-on-aws-remote) in each applicable secondary account in the same Region as the solution in the main account. When each remote stack is launched successfully, the main solution will update with the Account ID without any further changes in the main 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 the Parameter Store.

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

## Automated tagging

The Instance Scheduler on AWS solution 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
<code>ScheduleMessage=Started by scheduler {scheduler}</code>	<code>ScheduleMessage=Started by scheduler MyScheduler</code>
<code>ScheduleMessage=Started on {year}/{month}/{day}</code>	<code>ScheduleMessage=Started on 2017/07/06</code>
<code>ScheduleMessage=Started on {year}/{month}/{day} at {hour}:{minute}</code>	<code>ScheduleMessage=Started on 2017/07/06 at 09:00</code>
<code>ScheduleMessage=Started on {year}/{month}/{day} at {hour}:{minute} {timezone}</code>	<code>ScheduleMessage=Started on 2017/07/06 at 09:00 UTC</code>

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.

## Scheduler CLI

The solution includes a command line interface (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 in this solution

The following AWS services are included in this solution:

AWS service	Description
<a href="#">Amazon Elastic Compute Cloud</a>	<b>Core.</b> Solution is used to start and stop EC2 instances. The instances are identified by specific tags key/values which are configured in the solution.
<a href="#">Amazon Relational Database Service</a>	<b>Core.</b> Solution is used to change RDS Instances status to Available or Stopped. The instances are identified by specific tags key/value s which are configured in the solution.
<a href="#">Amazon Aurora Clusters</a>	<b>Core.</b> 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.
<a href="#">AWS Lambda</a>	<b>Core.</b> Solution deploys a Lambda Function which contains all the logic to schedule the instances and also to manage any updates to the cloudformation stack using custom resource feature.
<a href="#">Amazon DynamoDB</a>	<b>Core.</b> Solution creates DynamoDb tables to store configuration of the schedules, state information and last actions performed of the instances and a table to store SSM maintenance window for scheduling purpose.
<a href="#">Amazon CloudWatch</a>	<b>Core.</b> Solution stores debugging and information logs.
<a href="#">AWS Identity Access Management</a>	<b>Core.</b> Solution uses IAM to get permissions for scheduling instances.
<a href="#">Amazon Simple Notification Systems</a>	<b>Core.</b> Solution creates a SNS topic to send error messages for the users to subscribe to and troubleshoot in case of any errors.

AWS service	Description
<a href="#">AWS Key Management System</a>	<b>Core.</b> Solution creates a KMS key to encrypt the SNS topic.
<a href="#">AWS Systems Manager</a>	<b>Supporting.</b> Provides application-level resource monitoring and visualization of resource operations and cost data.
<a href="#">AWS EventBridge</a>	<b>Supporting.</b> Solution creates an EventBridge event bus to receive events from remote templates when the user chooses to use the AWS Organizations feature.

# Plan your deployment

This section describes the [cost](#), [network security](#), [quotas](#), and other considerations prior to deploying the solution.

## 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 in small deployment of two accounts and two Regions is approximately **\$5.00 per month**. See sample cost tables below for a more detailed breakdown.

Instance Scheduler on AWS is designed to invoke AWS Lambda functions multiple times per run cycle. For example, if the solution is being used to manage both EC2 and 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 will run five Lambda functions. One for the initial start of the process to handle CloudWatch Events, which is invoked based on the selected frequency (default: five minutes), and each service, account, and Region will be handled by an individual Lambda run (2 accounts x 2 services x 1 Region).

The optional custom [Amazon CloudWatch metrics](#) will cost an additional **\$0.60 - \$1.20 per month for each metrics-enabled schedule (\$0.60 per service the schedule operates in)**. To save costs, this feature can be enabled on a per-schedule basis.

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 services in this solution](#).

The cost of the solution per run will depend on the number of instances being tagged and managed by the solution. As the number of EC2 and RDS 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. For full details, refer to the pricing webpage for each AWS service used in this solution.

## Pricing examples (monthly)

### Small deployment

This pricing example is based on these assumptions:

- Two accounts, two Regions, scheduling both EC2 and RDS
- 100 schedules
- One schedule with metrics enabled
- Scheduling interval: 5 minutes
- AWS Lambda function size: 128 MB
- Average Lambda function runtime: 8 seconds

AWS service	Dimensions	Monthly Cost [USD]
AWS Lambda	<ul style="list-style-type: none"> <li>• 288 scheduling runs per day</li> <li>• 1+8 Lambdas per run</li> <li>• 8 second average Lambda function runtime</li> <li>• (\$0.0000021/second)</li> <li>• (\$0.0000002/Lambda function call)</li> </ul>	~ \$1.40
AWS CloudWatch Metrics (optional)	<ul style="list-style-type: none"> <li>• 1 metrics-enabled schedules * 2 services</li> <li>• (\$0.60/month)</li> <li>• ~ 200,000 PutMetric calls/month</li> <li>• (\$0.01/1000)</li> </ul>	~ \$2.00
AWS DynamoDB	<ul style="list-style-type: none"> <li>• ~ 70,000 WRU/month (\$1.25 per million)</li> <li>• ~ 250,000 RRU/month (\$0.5 per million)</li> </ul>	~ \$0.15

AWS service	Dimensions	Monthly Cost [USD]
	<ul style="list-style-type: none"> <li>negligible storage costs (&lt; \$0.01)</li> </ul>	
AWS Key Managed Service	<ul style="list-style-type: none"> <li>1 KMS key (\$1/month)</li> <li>~ 140,000 api requests/month (\$0.30/10000)</li> </ul>	~ \$1.45
<b>Total:</b>		<b>\$5.00</b>

## Medium deployment

This pricing example is based on these assumptions:

- 50 accounts, four Regions, scheduling both EC2 and RDS
- 100 schedules
- 10 schedules with metrics enabled
- Scheduling interval: 5 minutes
- AWS Lambda function size: 128 MB
- Average Lambda function runtime: 8 seconds

AWS service	Dimensions	Monthly Cost [USD]
AWS Lambda	<ul style="list-style-type: none"> <li>288 scheduling runs per day</li> <li>1+400 Lambdas per run</li> <li>8 second average Lambda function runtime</li> <li>(\$0.0000021/second)</li> <li>(\$0.0000002/Lambda function call)</li> </ul>	~ \$59.00

AWS service	Dimensions	Monthly Cost [USD]
AWS CloudWatch Metrics (optional)	<ul style="list-style-type: none"> <li>• 10 metrics-enabled schedules * 2 services</li> <li>• (\$0.60/month)</li> <li>• ~ 3.5m PutMetric calls/month</li> <li>• (\$0.01/1000)</li> </ul>	~ \$47.00
AWS DynamoDB	<ul style="list-style-type: none"> <li>• ~ 3.5m WRU/month (\$1.25 per million)</li> <li>• ~ 4m RRU/month (\$0.5 per million)</li> <li>• negligible storage costs (&lt; \$0.01)</li> </ul>	~ \$6.00
AWS Key Managed Service	<ul style="list-style-type: none"> <li>• 1 KMS key (\$1/month)</li> <li>• ~ 7m api requests/month (\$0.30/10000)</li> </ul>	~ \$22.00
<b>Total:</b>		<b>\$134.00</b>

## Large deployment

This pricing example is based on these assumptions:

- 120 accounts, six Regions, scheduling both EC2 and RDS
- 100 schedules
- 15 schedules with metrics available
- Scheduling interval: 5 minutes
- AWS Lambda function size: 128 MB
- Average Lambda function runtime: 8 seconds



AWS service	Dimensions	Monthly Cost [USD]
AWS Lambda	<ul style="list-style-type: none"> <li>• 288 scheduling runs per day</li> <li>• 1+1400 Lambdas per run</li> <li>• 8 second average Lambda function runtime</li> <li>• (\$0.0000021/second)</li> <li>• (\$0.0000002/Lambda function call)</li> </ul>	~ \$212.00
AWS CloudWatch Metrics (optional)	<ul style="list-style-type: none"> <li>• 15 metrics-enabled schedules * 2 services</li> <li>• (\$0.60/month)</li> <li>• ~ 12.5m PutMetric calls/month</li> <li>• (\$0.01/1000)</li> </ul>	~ \$143.00
AWS DynamoDB	<ul style="list-style-type: none"> <li>• ~ 12.5m WRU/month (\$1.25 per million)</li> <li>• ~ 13m RRU/month (\$0.5 per million)</li> <li>• negligible storage costs (&lt; \$0.01)</li> </ul>	~ \$19.00
AWS Key Managed Service	1 KMS key (\$1/month)  ~ 25m api requests/month (\$0.30/10000)	~ \$76.00
<b>Total:</b>		<b>\$450.00</b>

**Note**

The cost of the solution will increase if the number of accounts and Regions are increased. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

To ensure the solution is efficiently configured, 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. Limit metrics to only those schedules that will provide useful insight.
5. Select a frequency which 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.

## 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 Key Management System

The solution creates an AWS managed Custom Master Key (CMK), which is used to configure server-side encryption for the SNS topic and the DynamoDB tables.

## Amazon Identity Access Management

The solution's Lambda function requires permissions to start/stop both EC2 and RDS instances, modify instance attributes, update tags for the instances among other permissions. 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 also has access to get/put SSM parameters, access to CloudWatch log groups, KMS key encryption/decryption, and publish messages to SNS topic. For detailed information about each permission provided to the service role, refer to [CloudFormation templates](#).

## Supported AWS Regions

You can deploy Instance Scheduler in any standard and GovCloud AWS Region and in some Opt-in Regions. Once deployed, the solution can be configured to apply the appropriate start or stop actions to tagged Amazon EC2 and Amazon RDS instances in any Region(s) of your account. If you use cross-account instance scheduling, the solution will apply actions to instances in all configured Regions in all accounts.

### Important

Instance Scheduler on AWS actions will 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

Instance Scheduler on AWS has been validated in all Standard AWS Regions and AWS GovCloud. For Opt-in Regions, Instance Scheduler on AWS can target instances within any Opt-in Region for scheduling, but the cloud-formation stacks themselves are currently only available for deployment in the following Opt-in Regions.

## Supported Regions

ap-south-1

ap-east-1

ap-southeast-3

eu-south-1

me-south-1

me-central-1

## 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 *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 [Lambda quotas](#).

# Deploy the solution

## Deployment considerations

Before you launch the automated deployment, review Cost, Architecture, Security, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy Instance Scheduler into your account.

**Time to deploy:** Approximately five minutes

## Partial automation

Users have the option to implement a partially automated solution by default (for example, configure start-only or stop-only actions). An example of this is a team that needs the flexibility to start instances at varying times (manually) but must stop those instances at the end of each day, or vice versa. Refer to [the section called “Schedules”](#) for an example of how this can be done.

## Instance shutdown behavior

### Amazon EC2

This solution is designed to automatically stop Amazon Elastic Compute Cloud (Amazon 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, Amazon 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 AWS Instance Scheduler are configured with a Stop shutdown behavior, otherwise they will be terminated.

### Amazon RDS

This solution is designed to automatically stop, not delete, Amazon RDS instances. You can use the **Create RDS Instance Snapshot** AWS CloudFormation template parameter to create snapshots of RDS 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 that snapshots are not available for Amazon Aurora clusters.

You can use the **Schedule Aurora Clusters** template parameter to start and stop RDS 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 Amazon RDS instance, refer to [Stopping an Amazon RDS DB instance temporarily](#) in the *Amazon RDS User Guide*.

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

## Amazon RDS maintenance window

Every Amazon RDS 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. Note that 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 Amazon RDS 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*.

## Global configuration settings

When you deploy the Instance Scheduler's AWS CloudFormation template, global configuration settings are stored in an Amazon DynamoDB table (ConfigTable). To modify these settings, update the solution stack using the AWS CloudFormation template. Do not change these settings in the DynamoDB table.

## 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 Solution's main AWS Lambda function using the **Memory Size** parameter.

## Encrypted Amazon EBS volumes

If your Amazon EC2 instances contain encrypted Amazon Elastic Block Store (Amazon EBS) volumes, you must grant Instance Scheduler permission to use the customer master key (CMK) to start and stop instances. Add the `kms:CreateGrant` permission to Instance Scheduler role (`<stackname>-SchedulerRole-<id>`) in the account where the instance is located.

## 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 rule 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.

## 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 will also register the spoke account with the hub, requiring no manual configuration.

### Note

If you have previously deployed this solution, see [Update the solution](#) Update the solution.

## 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 through this survey. Data collection is subject to the [AWS Privacy Policy](#).

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 mins (not including configuration)

???

- Launch the AWS CloudFormation template in your AWS account.
- Enter values for the required parameters.
- Review the other template parameters, and adjust if necessary.

???

- Launch the AWS CloudFormation template in your AWS account.
- Enter a value for the required parameter: **Primary account**.

???

- Create schedules and periods for your environment.

???

- Apply the custom tag to applicable resources

## Step 1: Launch the instance scheduler hub stack

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

### Note


If you choose to setup the solution for using remote Account IDs for cross-account scheduling, provide the remote account ids only after installing the `instance-scheduler-on-aws-remote.template`.


**Time to deploy:** Approximately five minutes


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

[Launch solution](#)


2. The template launches in the US East (N. Virginia) Region by default. To launch Instance scheduler 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 STS Limits](#) 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	Description
<b>Instance Scheduler tag name</b>	Schedule	<p>The tag key Instance Scheduler will read to determine the schedule for a resource. The value on a resource specifies the name of the schedule.</p> <p>If you choose to modify the default value, make sure to assign a name that will be easy to apply consistently and correctly across all necessary instances.</p> <div> <b>Note</b> The tag key is case sensitive.</div>

Parameter	Default	Description
		For example, if the value for this parameter is left as the default of schedule, Instance Scheduler will schedule all EC2 instances tagged with the key schedule and the value my-office-hours-schedule according to the schedule with the name my-office-hours-schedule .
<b>Service(s) to schedule</b>	EC2	The services to schedule. Select EC2, RDS, or Both.
<b>Schedule Aurora Clusters</b>	No	Choose whether to schedule Amazon Aurora clusters. To turn on Aurora cluster scheduling, you must select RDS or Both for the <b>Service(s) to schedule</b> parameter.
<b>Create RDS instance snapshot</b>	Yes	Choose whether to create a snapshot before stopping RDS instances. <div data-bbox="1081 1486 1507 1755"> <p> <b>Note</b></p> <p>Snapshots are not available for Amazon Aurora clusters.</p> </div>

Parameter	Default	Description
<b>Scheduling enabled</b>	Yes	Select No to disable scheduling for all services.
<b>Default time zone</b>	UTC	Default IANA time zone identifier for schedules that do not specify a time zone. For a list of valid time zone identifiers, refer to the <b>TZ identifier</b> column of the <a href="#">List of tz database time zones</a> .
<b>This account</b>	Yes	Select Yes to enable scheduling resources in the current (hub) account.
<b>Frequency</b>	5	The interval (in minutes) at which the AWS Lambda function runs. Select 1, 2, 5, 10, 15, 30, or 60.
<b>Enable CloudWatch Metrics</b>	No	<p>Enable per-schedule CloudWatch metrics for all schedules. Can be disabled per-schedule on the schedule definition. For details, refer to <a href="#">Step 3</a>.</p> <div>  <b>Important</b>            Enabling this feature will incur charges of \$0.60/month for each schedule+ service combination in use.         </div>

Parameter	Default	Description
<b>Memory Size</b>	128	Memory size of the AWS Lambda function that schedules Amazon EC2 and Amazon RDS resources. Increase if you are experiencing high memory usage or timeouts.
<b>Namespace</b>	<Optional input>	<p>Provide a unique identifier to differentiate between multiple solution deployments (No Spaces). Example: Dev</p> <p>This parameter is required to be a non-empty value for deployments using AWS Organizations.</p>
<b>Use AWS Organizations</b>	No	Use AWS Organizations to automate spoke account registration.
<b>Organization ID / Remote Account IDs</b>	<Requires input>	(Required) If using AWS Organizations, provide Organization ID, for example, o-xxxxyyy. Otherwise, provide a comma separated list of spoke Account IDs to schedule, such as 1111111111, 2222222222 or {param: ssm-param-name}.

Parameter	Default	Description
<b>Region(s)</b>	<Optional input>	<p>List of Regions where instances will be scheduled. For example, us-east-1 , us-west-1 .</p> <div>  <b>Note</b>            If you leave this parameter blank, the solution will use the current Region.         </div>
<b>Enable CloudWatch Debug Logs</b>	No	Enable debug-level logging in CloudWatch Logs.
<b>Enable SSM maintenance windows</b>	No	Allow schedules to specify a maintenance window name. Instance Scheduler will ensure the instance is running during that maintenance window.
<b>Log retention days</b>	30	The log retention period in days.

Parameter	Default	Description
<b>Started tags</b>	<code>&lt; input InstanceScheduler-LastAction=Started By {scheduler} {year}/{month}/{day} {hour}:{minute}{timezone},&gt;</code>	Comma separated list of tag keys and values of the format <code>key=value, key=value, ...</code> that are set on started instances. Leave blank to disable.  <div>  <b>Note</b>  This value can be removed to disable the feature entirely. </div>
<b>Stopped tags</b>	<code>&lt; input InstanceScheduler-LastAction=Stopped By {scheduler} {year}/{month}/{day} {hour}:{minute}{timezone}&gt;</code>	Comma separated list of tag keys and values of the format <code>key=value, key=value, ...</code> that are set on stopped instances. Leave blank to disable.  <div>  <b>Note</b>  This value can be removed to disable the feature entirely. </div>

- Choose **Next**.
- On the **Configure stack options** page, choose **Next**.
- On the **Review** page, review and confirm the settings. Select the box acknowledging that the template will create IAM resources.
- Choose **Create** 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 completely or successfully installed in the Hub account.

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.

A blue rectangular button with the text "Launch solution" in white, bold, sans-serif font.

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 hub stack is similarly configured there is no further changes required in the main stack to start the scheduling.
7. If the AWS Organization option is set to No then the primary stack should be updated with the new Account ID.

Parameter	Default	Description
Namespace	<Optional input>	Unique identifier used to differentiate between

Parameter	Default	Description
		multiple solution deployments. Must be set to the same value as the Hub stack.
<b>Hub Account ID</b>	<Requires input>	Account ID of the Instance Scheduler Hub stack that should be allowed to schedule resources in this account.
<b>Use AWS Organizations</b>	No	Use AWS Organizations to automate spoke account registration. Must be set to the same value as the Hub stack.

8. Choose **Next**.

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

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

11 Choose **Create** 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

Once the solution has been deployed, you can configure scheduling on the hub control stack.

## Step 3: Configure schedules and periods

### 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 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 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 schedule](#).

## Step 4: Tag your instances

When you deployed the AWS CloudFormation template, you defined the name (tag key) for the solution's *custom tag*. For the Instance Scheduler to recognize an Amazon EC2 or Amazon RDS instance, the tag key on that instance must match the 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 [Tagging Your Amazon EC2 Resources](#) and [Tagging your 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 a resource, use the tag key you defined during initial configuration. Set the tag key to `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 Your Amazon RDS Resources](#).

### EC2 Instances with Encrypted EBS Volumes

If your EC2 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 Instances Not Starting](#).

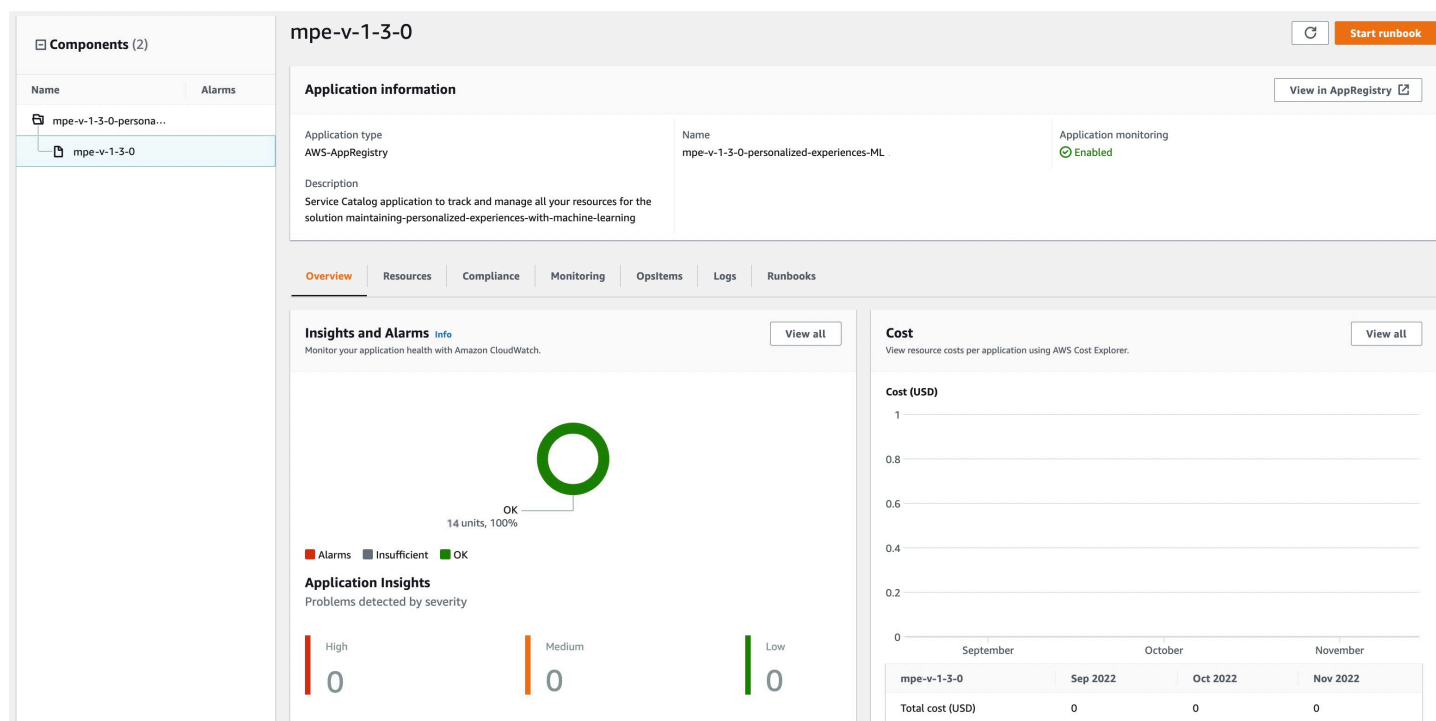
# Monitoring this solution with AppRegistry

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



## Instance Scheduler on AWS in Application Manager

**Note**

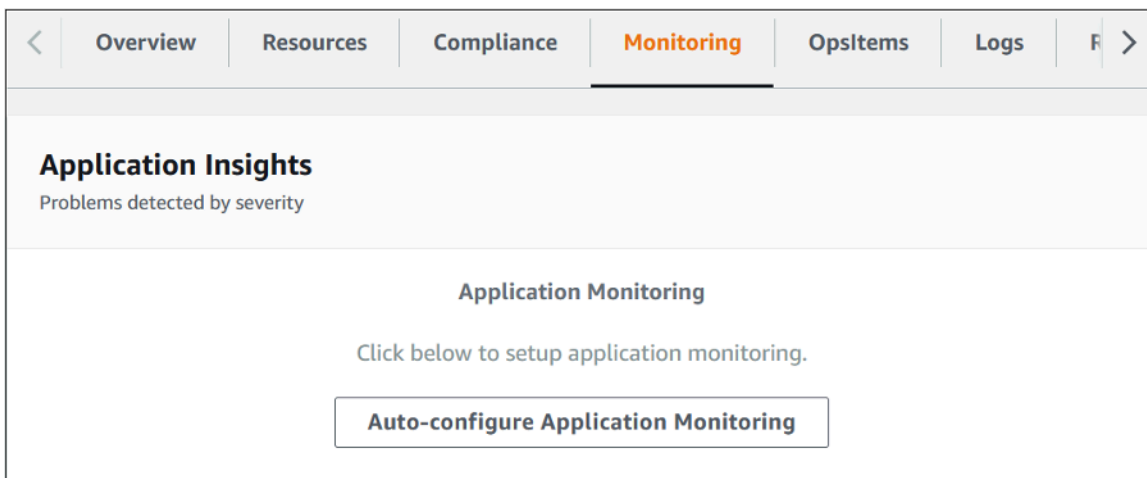
You must activate CloudWatch Application Insights, AWS Cost Explorer, and cost allocation tags associated with this solution. They are not activated by default.

## Activate CloudWatch Application Insights

1. Open the [Systems Manager console](#).
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose **AppRegistry applications**.
4. In **AppRegistry applications**, search for the application name for this solution and select it.

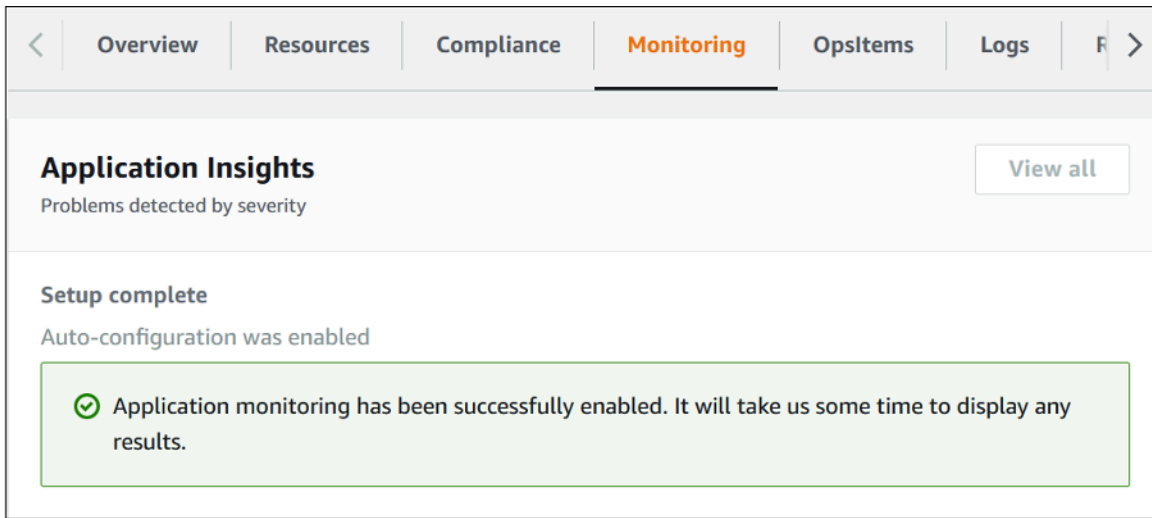
The next time you open Application Manager, you can find the new application for your solution in the **AppRegistry application** category.

5. In the **Components** tree, choose the application stack you want to activate.
6. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Monitoring**.



### Instance Scheduler Application Manager

Monitoring for your applications is now activated and the following status box appears:



## Instance Scheduler Application Manager

# 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**.
3. On the **Welcome to Cost Explorer** page, choose **Launch Cost Explorer**.

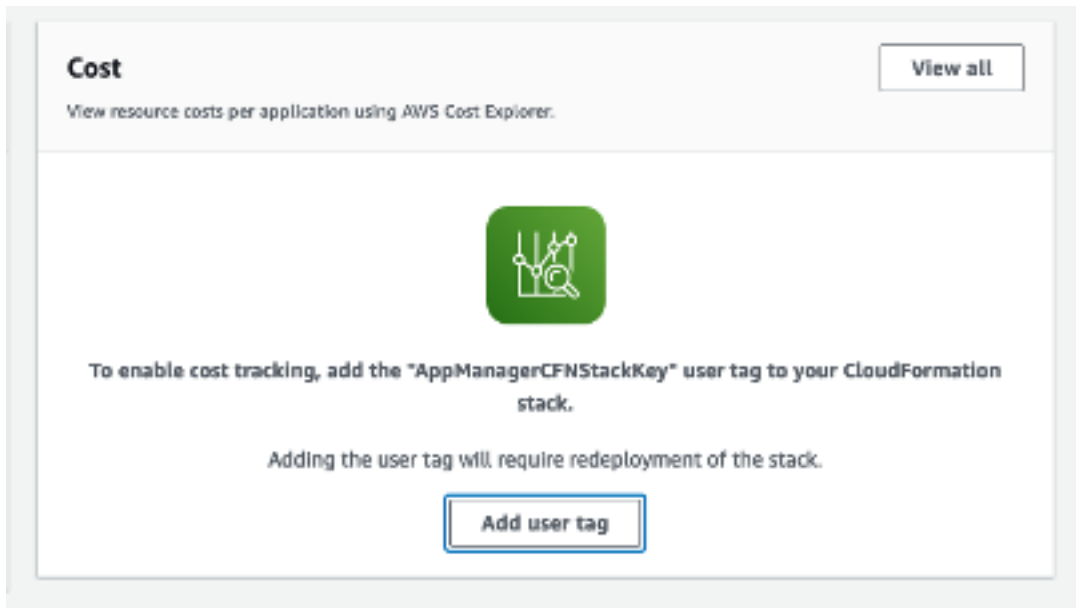
The activation process can take up to 24 hours to complete. Once activated, you can open the Cost Explorer user interface to further analyze cost data for the solution.

## 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:

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.

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 a cost allocation tag 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 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**.

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



# 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 [AWS CloudFormation console](#), select your existing `instance-scheduler-on-aws` hub stack, 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. Refer to [Step 1: Launch the instance scheduler hub stack](#) for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template might create (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 after a few minutes.

## Updating to v1.5x

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:
  - Choose a unique namespace for the solution.
  - Select whether you would like to **Use AWS Organizations** to manage spoke registration going forward.
    - 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:
  - Namespace – same as you chose for the hub account.
  - Use AWS Organizations – same as hub account.
  - 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:
  - If you selected **Yes** for **Use AWS Organizations**, you can skip this step entirely.
  - Update the hub template for a second time (select **Use current template**)
  - Replace **Organization 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.

# 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, see the [Contact AWS Support](#) section for instructions on 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). For example:

Scheduler-ec2-111122223333-<REGION>-20230411

5. 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 (or earlier) 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 filter **Search Resources**.
5. Select the Lambda function in the column **Physical ID**.
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 on AWS is reporting that EC2 instances with encrypted EBS volumes are being started, but they never actually start.

### Resolution:

If your Amazon EC2 instances contain encrypted Amazon Elastic Block Store (Amazon EBS) volumes, you must grant Instance Scheduler permission to use the customer master key (CMK) to start and stop instances. This permission must be granted to the scheduling role ***in the same account as the EC2 instance(s) using the key.***

For the hub account, this role will be called `<stackname>-SchedulerRole-<id>`.

On spoke accounts, this role will be called `<namespace>-SchedulerRole`.

To grant access to a specific KMS key, apply the following policy to the correct scheduler role:

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

## 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 questions 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.

## 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

## 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 Cloud Development Kit \(AWS CDK\)](#). See the [README.md](#) file for additional information.

## 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-yyyyymmdd` 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-1.5.3-py3-none-any.whl*) and place it in a directory on your computer.
2. From the same directory in which you placed the 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 instance_scheduler_cli-1.5.3-py3-none-any.whl
```

3. 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
<code>--stack &lt;stackname&gt;</code>	The name of the scheduler stack. <div> <b>Important</b> This argument is required for all commands.</div>
<code>--region &lt;regionname&gt;</code>	The region where the scheduler stack is deployed. <div> <b>Note</b> You must use this argument when the default configuration and credential files are not installed in the same region as the solution stack.</div>
<code>--profile-name &lt;profilename&gt;</code>	The name of the profile to use to run commands. If no profile name is specified, the default profile is used.
<code>--query</code>	A JMESPath expression that controls the command output. For more information on controlling output, refer to <a href="#">Controlling Command Output from the AWS Command Line Interface</a> in the <i>AWS CLI User Guide</i> .

Argument	Description
<code>--help</code>	Shows valid commands and arguments for the scheduler CLI. When used with a specific command, it shows valid subcommands and arguments for that command.
<code>--version</code>	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 n<sup>th</sup> 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 n<sup>th</sup> 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 n<sup>th</sup> 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

### Description

Deletes an existing period

### Arguments

`--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}

  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
```

## Solution resources

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

Resource name	Type	Description
<b>Main</b>	AWS::Lambda::Function	Instance Scheduler AWS Lambda function.
<b>Scheduler Config Helper</b>	Custom::ServiceSetup	Stores global configuration settings in Amazon DynamoDB.
<b>Scheduler Invoke Permission</b>	AWS::Lambda::Permission	Allows the Amazon CloudWatch event to invoke the Instance Scheduler's AWS Lambda function.
<b>Scheduler Logs</b>	AWS::Logs::LogGroup	CloudWatch Log Group for Instance Scheduler.
<b>Scheduler Policy</b>	AWS::IAM::Policy	Policy that allows scheduler to perform start and stop actions, change Amazon EC2 instance attributes, set

Resource name	Type	Description
		tags, and access scheduler resources.
<b>Scheduler Rule</b>	<code>AWS::Events::Rule</code>	Amazon Eventbridge event rule that invokes the scheduler's Lambda function.
<b>Configuration Metrics Event Rule</b>	<code>AWS::Events::Rule</code>	Amazon EventBridge event rule that periodically invokes the configuration description anonymized metrics function. Disabled when anonymized metrics are disabled.
<b>State Table</b>	<code>AWS::DynamoDB::Table</code>	DynamoDB table that stores last desired state of instances.
<b>Config Table</b>	<code>AWS::DynamoDB::Table</code>	DynamoDB table that stores global configuration, schedule, and period data.
<b>Instance Scheduler SNS Topic</b>	<code>AWS::SNS::Topic</code>	Sends warning and error messages to subscribed email addresses.

## 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-yyyyymmdd`: Logs general scheduler messages
- `SchedulingOrchestratorHandler-yyyyymmdd`: Logs general orchestration information for when scheduling executions are started
- `SchedulerSetupHandler-yyyyymmdd`: Logs the output of configuration actions



- `Scheduler-<service>-<account>-<region>-yyyymmdd`: Logs the 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.

## 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 [???](#). 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 on AWS 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.

## Sample schedules

The 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	Value
begintime	09:00
endtime	16:59

Field	Value
name	weekdays-9-5
weekdays	mon-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	london-working-hours
periods	weekdays-9-5
timezone	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 key and value will be different. For example, if you entered `Sked` as your tag name, your tag key will be `Sked` and the tag value 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	Value
endtime	16:59
name	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	stop-at-5-new-york
periods	stop-at-5
timezone	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 selecting [previously deployed Instance Scheduler Hub Stack](#) and then **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	Value
begintime	09:00
endtime	23:59
name	mon-start-9am
weekdays	mon

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

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

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

Field	Value
begintime	00:00
endtime	16:59
name	fri-stop-5pm
weekdays	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	mon-9am-fri-5pm
periods	mon-start-9am,tue-thu-full-day,fri-stop-5pm
timezone	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> --name fri-stop-5pm --weekdays fri --
begintime 0:00 --endtime 17:00

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

## 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

```

## Monitor schedule activity in Amazon CloudWatch

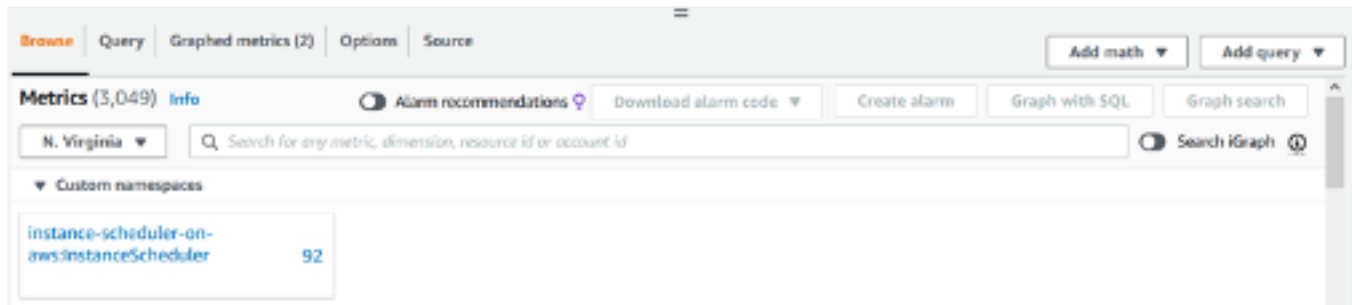
This solution provides an option to create a new custom Amazon CloudWatch metric (metric name). To activate this feature, select Yes for the Enable CloudWatch Metrics parameter (`<stackname>:InstanceScheduler`).

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 statistic of each selected metric to **Sum** and the Period to match the scheduling interval configured for the solution. (Default = 5 minutes. Example image uses an interval of 1 minute).
6. At the top of the graph, select a time range to view.



## Application Insights

# Reference

This section includes information about an optional feature for collecting unique metrics for this solution, 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 [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.

## Related resources

The [Resource Scheduler](#) is 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 regularly scanned to start/stop instances.
- The table identifies which solution is better based on scenarios.

Scenario	Resource scheduler	Instance scheduler
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



# 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
June 2019	Added information on hibernating Amazon EC2 instances, scheduling RDS instances that are part of an Amazon Aurora cluster, new scheduler 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 AWS Instance Scheduler has been validated
March 2020	Bug fixes
June 2020	Clarified time zone information for period rule start and stop times. For information about changes for v1.3.2, refer to the <a href="#">CHANGELOG.md file</a> in the Github repository
September 2020	AWS Cloud Development Kit (AWS CDK) conversion and documentation enhancements. For more information about changes for v1.3.3, refer to the <a href="#">CHANGELOG.md file</a> in the Github repository
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 <a href="#">CHANGELOG.md file</a> in the GitHub repository.

Date	Change
May 2022	Minor update and bug fixes. For more information about v1.4.1, refer to the <a href="#">CHANGELOG.md file</a> in the GitHub repository.
January 2023	Minor update and bug fixes. For more information about v1.4.2, refer to the <a href="#">CHANGELOG.md file</a> in the GitHub repository.
May 2023	Deprecated <code>override_status</code> scheduling flag, add 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 <a href="#">CHANGELOG.md</a> file in the GitHub repository.
July 2023	<p>Added guidance on how to manage Infrastructure as Code, and updating to version 1.5.x.</p> <p>Additional Troubleshooting, Features and Benefits added. For more information about v1.5.1, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.</p>
September 2023	Documentation update: Added AWS Support section for contacting support.
October 2023	<p>Minor update, security patching and added additional sample cost tables in the documentation.</p> <p>For more information about v1.5.2, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.</p>
October 2023	Release v1.5.3: Security patch. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
December 2023	Documentation update to resolve discrepancies, added actional section the AppRegistry section. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
February 2024	Release v1.5.4: Security patch. For more information, refer to the <a href="#">CHANGELOG.md</a> file 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 current AWS 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. The responsibilities and liabilities of AWS 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.

The Instance Scheduler on AWS solution is licensed under the terms of the Apache License Version 2.0 available at <https://www.apache.org/licenses/LICENSE-2.0>