Instance Scheduler on AWS

Implementation Guide

Instance Scheduler on AWS: Implementation Guide

Table of Contents

Automate starting and stopping AWS instances

Publication date: *October 2020 (last update (p. 51): January 2023)*

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 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 discusses architectural considerations and configuration steps for deploying the Amazon Web Services (AWS) Instance Scheduler in the AWS Cloud. It includes links to AWS CloudFormation templates that launch, configure, and run the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting in the AWS Cloud.

Cost

You are responsible for the cost of the AWS services used while running Instance Scheduler on AWS. As of January 2023, the cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$9.90 per month** in AWS Lambda charges, or less if you have Lambda free tier monthly usage credit. This is independent of the number of Amazon EC2 instances you are running. The optional custom Amazon CloudWatch metric (p. 26) will cost an additional **$0.90 per month per schedule or scheduled service**. By default, this solution uses Auto Scaling for its Amazon DynamoDB tables to provide sufficient read and write capacity.

Instance Scheduler on AWS is designed to run different numbers of AWS Lambda functions 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 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.

The costs in the following table are based on the following assumptions:

1. The solution is deployed in US East (N. Virginia) Region.
2. The solution is managing both EC2 and RDS instances.
3. The solution is managing instances in an additional account.
4. The total number of executions per day is 1,440 runs (Lambda is scheduled to run every five minutes).
5. The average run time for each Lambda is assumed as eight seconds (this depends upon the number of instances being scheduled).
6. The memory selected for the AWS Lambda 128 MB.

AWS service	Dimensions	Cost (per month) [USD]
AWS Lambda	1,440 runs/24 hours 40 seconds per run (8 seconds for each Lambda) ($0.0000021/second/ run)	$7.25
AWS CloudWatch Metrics (enabled)	1 month per schedule or scheduled service	$0.90
AWS DynamoDB	1,080,000 (Write/ month) ($1.25 per million requests)	$1.25
AWS DynamoDB	1,080,000 (Read/ month) ($0.5 per million requests)	$0.50

AWS service	Dimensions	Cost (per month) [USD]
AWS DynamoDB	<1GB (First 25 GB is free)	$0.0
	Total	$9.90

The cost is independent of the number of Amazon EC2 and RDS instances you are running. By default, this solution uses on-demand scaling for its Amazon DynamoDB tables to provide sufficient read and write capacity. The AWS services for this solution are listed under Additional Resources (p. 28).

Prices are subject to change. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

Architecture overview

Deploying this solution builds the following environment in the AWS Cloud.

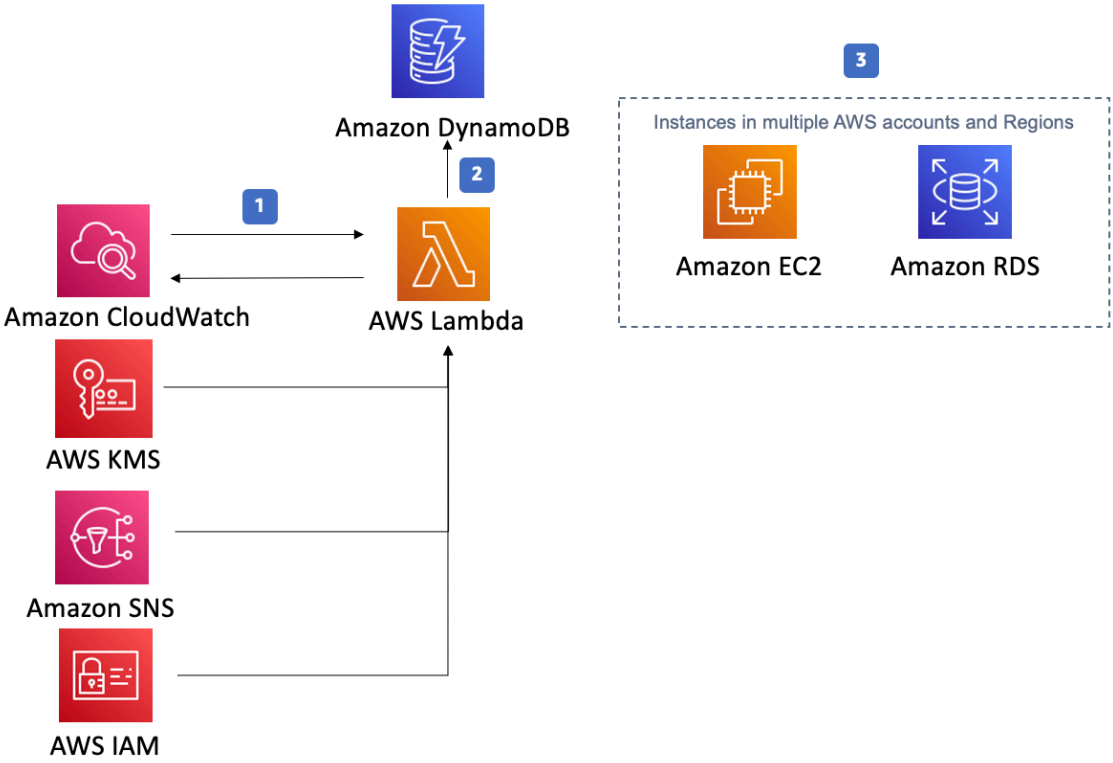

Figure 1: Instance Scheduler on the AWS Cloud

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`.

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 (p. 6) 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 (p. 26)).

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.

Solution components

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 (p. 29).

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, the Instance Scheduler will apply the appropriate start action when at least one of the period rules is true. For more information, refer to Period Rules (p. 9).

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. Your EC2 instances will have an Amazon Machine Image (AMI) when the hibernate field is set to `true`. For more information, refer to Hibernate your On-Demand or Reserved Linux instance in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*. 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 when the solution stops them.

If you set the solution to use hibernation, but your instances are not enabled 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 or Reserved Linux instance in the *Amazon Elastic Compute Cloud User Guide for Linux instances*.

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

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.

The 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 CloudFormation parameter **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 that 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 Compatibility for Resizing Instances in the *Amazon EC2 User Guide for Linux Instances*.

Schedule definitions

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

Field	Description
`description`	An optional description of the schedule.
`hibernate`	Choose whether to hibernate Amazon EC2 instances running Amazon Linux. When this field is set to `true`, the scheduler will hibernate instances when it stops them. Note that your instances must turn on hibernation and must meet the hibernation prerequisites.
`enforced`	Choose whether to enforce the schedule. When this field is set to `true`, the scheduler will stop a running instance if it is manually started outside of the running period or it will start an instance if it is stopped manually during the running period.
`name`	The name used to identify the schedule. This name must be unique.
`override_status`	When this field is set to `running`, the instance will be started but not stopped until you stop it manually. When this field is set to `stopped`, the instance will be stopped but not started until you start it manually.
`periods`	The name of the periods that are used in this schedule. Enter the name(s) exactly as it appears in the period `name` field. You can also specify an instance type for the period using the syntax <*period-name*>@<*instance-type*>. For example, `weekdays@t2.large`.
`retain_running`	Choose whether to prevent the solution from stopping an instance at the end of a running period if the instance was manually started before the beginning of the period.
`ssm_maintenance_window`	Choose whether to add an AWS Systems Manager maintenance window as a running period. Enter the name of a maintenance window. **Note** To use this field, you must also set the `use_maintenance_window` parameter to `true`.
`stop_new_instances`	Choose whether to stop an instance the first time it is tagged if it is running outside of the running period. By default, this field is set to `true`.

Field	Description
`timezone`	The time zone the schedule will use. If no time zone is specified, the default time zone (UTC) is used. For a list of acceptable time zone values, refer to the **TZ** column of the List of TZ Database Time Zones.
`use_maintenance_window`	Choose whether to add an Amazon RDS maintenance window as a running period to an Amazon RDS 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 Amazon RDS Maintenance Window (p. 15) and SSM Maintenance Window Field (p. 7).
`use_metrics`	Choose whether to turn on CloudWatch metrics at the schedule level. This field overwrites the CloudWatch metrics setting you specified at deployment. **Note** Enabling this feature will incur charges of $0.90/month per schedule or scheduled service.

Period rules

Period rules contain conditions that allow you to set the specific hours, days, and months an instance will run. A period rule 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` (p. 10) 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 00:00 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.

Figure 2: 9-5 scheduled start and stop

If the user manually starts that instance at 5 am, the solution will stop the instance at 5 pm. Note that if you use the retain running field (p. 7), the solution will not stop the instance at 5 pm.

Figure 3: 5AM 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.

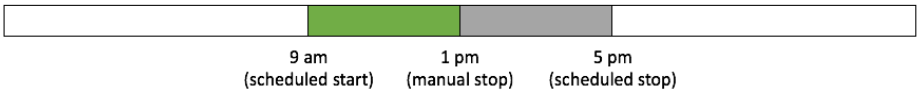

Figure 4: 5PM scheduled stop

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 (p. 45).

Days of the week

The `weekdays` field defines which days during the week an instance will run. You can specify a list of days, a range of days, the nth occurrence of that day in a month, or the last occurrence of that day in a month. The solution supports abbreviated day names (Mon) and numbers (0). For more information, refer to Step 2 (p. 23).

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 nth day of the month, the last day of the month, or the nearest weekday to a specific date. For more information, refer to Step 2 (p. 23).

Months

The `months` field defines which months an instance will run. You can specify a list of months, a range of months, or every nth month. The solution supports abbreviated month names (Jan) and numbers (1). For more information, refer to Step 2 (p. 23).

Period definitions

The Instance Scheduler 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.

Field	Description
Important You must specify at least one of the following items: `begintime`, `endtime`, `weekdays`, `months`, or `monthdays`.	
`begintime`	The time, in HH:MM format, that the instance will start.
`description`	An optional description of the period rule
`endtime`	The time, in HH:MM format, that the instance will stop.
`months`	Enter a comma-delimited list of months, or a hyphenated range of months, during which the instance will run. For example, enter `jan, feb, mar` or `1, 2, 3` to run an instance during those months. Or, you can enter `jan-mar` or `1-3`. You can also schedule an instance to run every nth month or every nth month in a range. For example, enter `Jan/3` or `1/3` to run an instance every third month starting in January. Enter `Jan-Jul/2` to run every other month from January to July.
`monthdays`	Enter a comma-delimited list of days of the month, or a hyphenated range of days, during which the instance will run. For example, enter `1, 2, 3` or `1-3` to run an instance during the first three days of the month. You can also enter multiple ranges. For example, enter `1-3, 7-9` to run an instance from the 1st to the 3rd and the 7th through the 9th. You can also schedule an instance to run every nth day of the month or every nth day of the month in a range. For example, enter `1/7` to run an instance every seventh day starting on the 1st. Enter `1-15/2` to run an instance every other day from the 1st to the 15th. Enter **L** to run an instance on the last day of the month. Enter a date and `W` to run an instance on the nearest weekday to the specified date. For example, enter `15W` to run an instance on the nearest weekday to the 15th.
`name`	The name used to identify the period rule. This name must be unique.
`weekdays`	Enter a comma-delimited list of days of the week, or a range of days of the week, during which the instance will run. For example, enter `0, 1, 2` or `0-2` to run an instance Monday through Wednesday. You can also enter multiple ranges. For example, enter `0-2, 4-6` to run an instance every day except Thursday. You can also schedule an instance to run every nth occurrence of a weekday in the month. For example, enter `Mon#1` or `0#1` to run an instance the first Monday of the month.

Field	Description
	Enter a day and `L` to run an instance on the last occurrence of that weekday in the month. For example, enter `friL` or `4L` to run an instance on the last Friday of the month.

When a period rule contains multiple conditions, note that all conditions must be true for the Instance Scheduler on AWS to apply the appropriate action. For example, a period rule 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

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

To apply automated start-stop schedules to resources in secondary accounts, launch the main solution template (`instance-scheduler`) in the primary account. Then, launch the remote template (`instance-scheduler-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 each cross-account role ARN by entering the appropriate ARN(s) in the **Cross-account roles** parameter to allow the solution to perform start and stop actions on instances in the secondary accounts.

AWS Systems Manager Parameter Store

You can use AWS Systems Manager Parameter Store to store cross-account role ARNs. You can store cross-account ARNs as a list parameter where every item is an ARN, or as a string parameter that contains a comma-delimited list of ARNs. 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 stack in the same account as your parameter store.

Automated tagging

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

- `{scheduler}`: The name of the scheduler stack
- `{year}`: The year (four digits)
- `{month}`: The month (two digits)
- `{day}`: The day (two digits)
- `{hour}`: The hour (two digits, 24-hour format)
- `{minute}`: The minute (two digits)
- `{timezone}`: The time zone

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

Example Parameter Input	Instance Scheduler Tag
`ScheduleMessage=Started by scheduler {scheduler}`	`ScheduleMessage=Started by scheduler MyScheduler`
`ScheduleMessage=Started on {year}/{month}/{day}`	`ScheduleMessage=Started on 2017/07/06`
`ScheduleMessage=Started on {year}/{month}/{day} at {hour}:{minute}`	`ScheduleMessage=Started on 2017/07/06 at 09:00`
`ScheduleMessage=Started on {year}/{month}/{day} at {hour}:{minute} {timezone}`	`ScheduleMessage=Started on 2017/07/06 at 09:00 UTC`

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

Scheduler command line interface

The 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. 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 (p. 29).

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared 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 the 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 Lambda function created by the solution 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 the CloudFormation templates (p. 18).

Design considerations

Partial automation

Users have the option to implement a partially automated solution by default (i.e., configure start-only or stop-only actions). An example of this is a team that needs the flexibility to stop instances at varying times (manually) but must start those instances simultaneously each morning, or vice versa.

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

Note that this solution is designed to automatically stop, not delete, Amazon Relational Database Service (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 Instance Scheduler 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 database 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 Instance Scheduler 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 fifteen-minute interval will only perform start and stop actions every 15 minutes.

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

Encrypted Amazon EBS volumes

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

Regional deployments

You can deploy Instance Scheduler on AWS in any AWS Region. Once deployed, the solution applies the appropriate start or stop actions to tagged Amazon EC2 and Amazon RDS instances in all Regions of your account. If you use cross-account instance scheduling, the solution will apply actions to instances in all 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 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.

Instance Scheduler on AWS has been validated in the following AWS Regions:

- us-east-1 (N.Virginia)
- us-east-2 (Ohio)
- us-west-1 (N.California)
- us-west-2 (Oregon)
- eu-west-1 (Ireland)
- eu-west-2 (London)
- eu-west-3 (Paris)
- eu-central-1 (Frankfurt)
- sa-east-1 (Sao Paulo)
- ap-northeast-1 (Tokyo)
- ap-northeast-2 (Seoul)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapore)
- ap-southeast-2 (Sydney)
- ca-central-1 (Canada)
- AWS GovCloud (US-East, US-West)

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 (p. 40).

Warning and error messages are also published to a solution-created Amazon SNS topic which sends messages to a subscribed email address (refer to Subscribing to an Amazon SNS Topic 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 to automate the deployment of AWS Instance Scheduler on the AWS Cloud. It includes the following AWS CloudFormation templates, which you can download before deployment:

aws-instance-scheduler.template: Use this template to launch the Instance Scheduler 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.

aws-instance-scheduler-remote.template: Use this template to configure permissions for instances in secondary accounts. The default configuration creates the AWS Identity and Access Management (IAM) roles necessary to start and stop instances in a secondary account.

Automated deployment

Before you launch the automated deployment, please review the architecture, configuration, 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 (5) minutes

Update the stack

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

1. Sign in to the AWS CloudFormation console, select your existing `aws-instance-scheduler` CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
4. Select **Amazon S3 URL**.
5. Copy the link of the latest template.
6. Paste the link in the **Amazon S3 URL** box.
7. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
8. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to Step 1. Launch the Stack for details about the parameters.
9. Choose **Next**.
10. On the **Configure stack options** page, choose **Next**.
11. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
12. Choose **View change set** and verify the changes.
13. 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 see a status of UPDATE_COMPLETE in approximately 15 minutes.

Deployment overview

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the instance scheduler stack (p. 20)

- Launch the AWS CloudFormation template into your AWS account
- Enter values for the required parameter: **Stack Name**
- Review the other template parameters, and adjust if necessary

Step 2. Configure periods (p. 23)

- Create a period and set the applicable fields for the period

Step 3. Configure schedules (p. 24)

- Create a schedule and set the applicable fields for the schedule

Step 4. Tag your instances (p. 24)

- Apply the custom tag to applicable resources

Step 5. Launch the remote stack in secondary accounts (Optional) (p. 24)

- Launch the AWS CloudFormation template into your AWS account
- Enter values for the required parameter: **Primary account**

Step 1. Launch the instance scheduler stack

This automated AWS CloudFormation template deploys the Instance Scheduler in AWS Lambda, and configures related components. Please make sure that you've verified the settings (p. 24) for your instances before launching the stack.

Note
You are responsible for the cost of the AWS services used while running this solution. For more details, visit to the Cost (p. 2) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and click the button below to launch the `instance-scheduler-on-aws` AWS CloudFormation template.

   You can also download the template as a starting point for your own implementation.
2. The template is launched in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.
3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your solution stack.

   Note
   The stack name cannot contain more than 128 characters.
5. Under **Parameters**, review the parameters for the template, and modify them as necessary.

   This solution uses the following default values.

Parameter	Default	Description
Instance Scheduler tag name	`Schedule`	This tag identifies instances to receive automated actions, and

Parameter	Default	Description
		also allows for custom start-stop parameters. 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.
Service(s) to schedule	`EC2`	The services to schedule. Select `EC2`, `RDS`, or `Both`.
Schedule Aurora Clusters	`No`	Choose whether to schedule Amazon Aurora clusters. To turn on Aurora cluster scheduling, you must select `RDS` or `Both` for the **Service(s) to schedule** parameter.
Create RDS instance snapshot	`Yes`	Choose whether to create a snapshot before stopping RDS instances **Note** Snapshots are not available for Amazon Aurora clusters.
Scheduling enabled	`Yes`	Select `No` to temporarily turn off scheduling.
Region(s)	<Optional input>	List of Regions where instances will be scheduled. For example, `us-east-1`, `us-west-1`. **Note** If you leave this parameter blank, the solution will use the current Region.
Default time zone	`UTC`	Default time zone for schedules. For a list of acceptable time zone values, refer to the **TZ** column of the List of TZ Database Time Zones.

Parameter	Default	Description
Cross-account roles	<Optional input>	Comma-delimited list of cross-account roles. For example, `arn:aws:iam::111122223333:role/<stackname>SchedulerCrossAccountRo` If you store your cross-account ARNs in the AWS Systems Manager Parameter Store, use the format `{param:name}`. For more information, refer to AWS Systems Manager Parameter Store (p. 12). **Note** Enter the secondary account **CrossAccountRoleArn** value(s) in this parameter.
This account	`Yes`	Select `Yes` to allow the task to select resources in this account. **Note** If you set this parameter to `No`, you must configure cross-account roles.
Frequency	5	The frequency in minutes at which the AWS Lambda function runs. Select `1`, `2`, `5`, `10`, `15`, `30`, or `60`.
Enable CloudWatch Metrics	No	Choose whether to collect data using CloudWatch Metrics for all schedules. You can override this default setting for an individual schedule when you configure it (refer to Step 3 (p. 24)). **Important** Enabling this feature will incur charges of $0.90/month per schedule or scheduled service.
Memory Size	128	The memory size of the solution's main AWS Lambda function. Increase the default size to schedule a large number of Amazon EC2 and Amazon RDS instances.

Parameter	Default	Description
Enable CloudWatch Logs	No	Choose whether to log detailed information in CloudWatch Logs.
Enable SSM maintenance windows	No	Allow the solution to load SSM maintenance windows, so that they can be used for Amazon EC2 instance scheduling.
Log retention days	30	The log retention period in days
Started tags	<Optional input>	Tags to add to started instances. Use a list of `tagname=tagvalue` pairs.
Stopped tags	<Optional input>	Tags to add to stopped instances. Use a list of `tagname=tagvalue` pairs.
Send anonymous usage data	Yes	Send anonymous data to AWS to help us understand solution usage and related cost savings across our customer base as a whole. To opt out of this feature, select No. For more information, refer to the Collection of operational metrics (p. 48).

6. Choose **Next**.
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. 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.

Step 2. Configure periods

When you deploy the AWS CloudFormation template, the solution creates an Amazon DynamoDB table that contains sample period rules and schedules that you can use as a reference to create your own custom period rules and schedules.

To create a period rule, you can use the Amazon DynamoDB console, the scheduler CLI, or the AWS CloudFormation custom resource that you can create. For details about this stack resource type, refer to Custom resource (p. 43).

Note
If you use the custom resource to create a period, you must not use the DynamoDB console or scheduler CLI to delete or modify that period. 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.

To create a period rule in DynamoDB, modify one of the periods in the configuration table (`ConfigTable`). To create a period in the scheduler CLI, use the applicable commands (p. 30). To create a period using the custom resource, add the applicable fields to the solution's custom resource.

For an example period configuration, refer to Sample schedule (p. 45).

Step 3. Configure schedules

To create a schedule, you can use the Amazon DynamoDB console, the scheduler CLI, or the AWS CloudFormation custom resource that you can create. For details about this stack resource type, refer to Custom resource (p. 43).

Note
If you use the custom resource to create a schedule, you must not use the DynamoDB console or scheduler CLI to delete or modify that schedule. If you do, you will create a conflict between the stored parameters in the stack and the values in the table.

To create a schedule in DynamoDB, modify one of the schedules in the configuration table (`ConfigTable`). To create a schedule in the scheduler CLI, use the applicable commands (p. 30). To create a schedule using the custom resource, add the applicable fields to the solution's custom resource.

For an example schedule configuration, refer to Sample schedule (p. 45).

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 name stored in the Amazon DynamoDB table. Therefore, it is important that you apply tags consistently and correctly to all applicable instances. You can continue to use existing tagging strategies 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 same scheduler name in the Amazon DynamoDB table to schedule your resource. You can also update the scheduler name in the Amazon DynamoDB table.

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.

Step 5. Launch the remote stack in secondary accounts (Optional)

This automated AWS CloudFormation template configures secondary account permissions.

1. Navigate to the AWS Instance Scheduler stack **Outputs** tab and copy the **Value** of **SchedulerRole**.
2. Sign in to the AWS Management Console of the applicable secondary account and click the button to below to launch the `instance-scheduler-on-aws-remote` AWS CloudFormation template.

   

   You can also download the template as a starting point for your own implementation.
3. The template is launched in the US East (N. Virginia) Region by default. To launch the Instance Scheduler in a different AWS Region, use the region selector in the console navigation bar.
4. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
5. On the **Specify Details** page, assign a name to your remote stack.
6. Under **Parameters**, review the parameter for the template, and modify it.

Parameter	Default	Description
Primary account	*<Requires Input>*	Enter the account number of the account with the primary Instance Scheduler stack. This parameter gives the solution permission to schedule Amazon EC2 and Amazon RDS instances in this account.

7. Choose **Next**.
8. On the **Options** page, choose **Next**.
9. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
10. 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.

Amazon CloudWatch metrics

This solution creates a new custom Amazon CloudWatch metric (*<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 metrics

1. In the AWS Management Console, open the Amazon CloudWatch console.
2. In the **Custom Namespaces** drop-down field, choose *<stackname>*:InstanceScheduler.

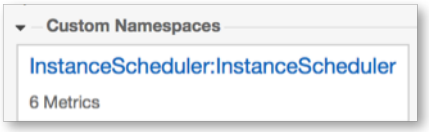

Figure 2: Custom Namespaces field

3. Select the schedule and service dimensions.

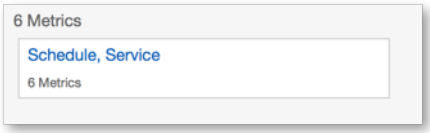

Figure 3: Select instance

4. Select the schedule and service that you want to view the status of.

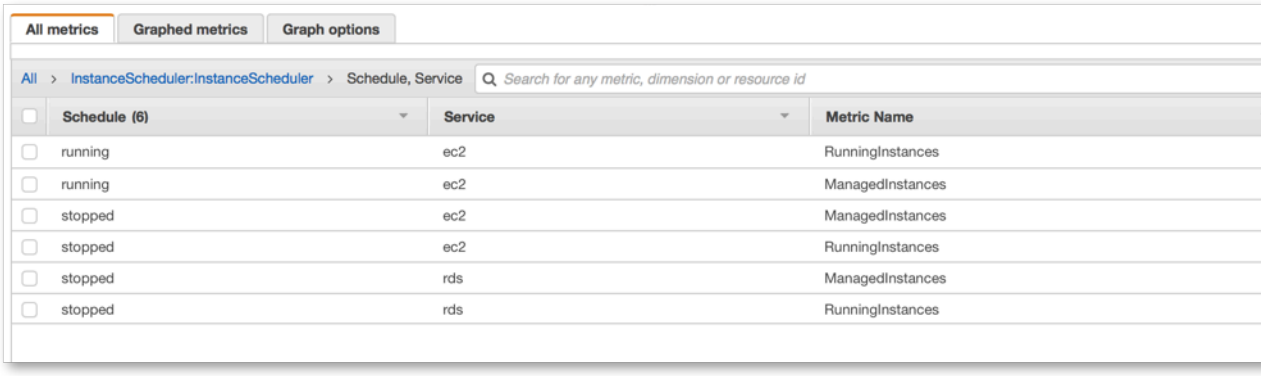

Figure 4: Select schedule

At the bottom of the page, an individual graph will appear for each instance you selected, as shown in the following example. Note that a value of `0` is a stopped instance and a value of `1.00` is a running instance.

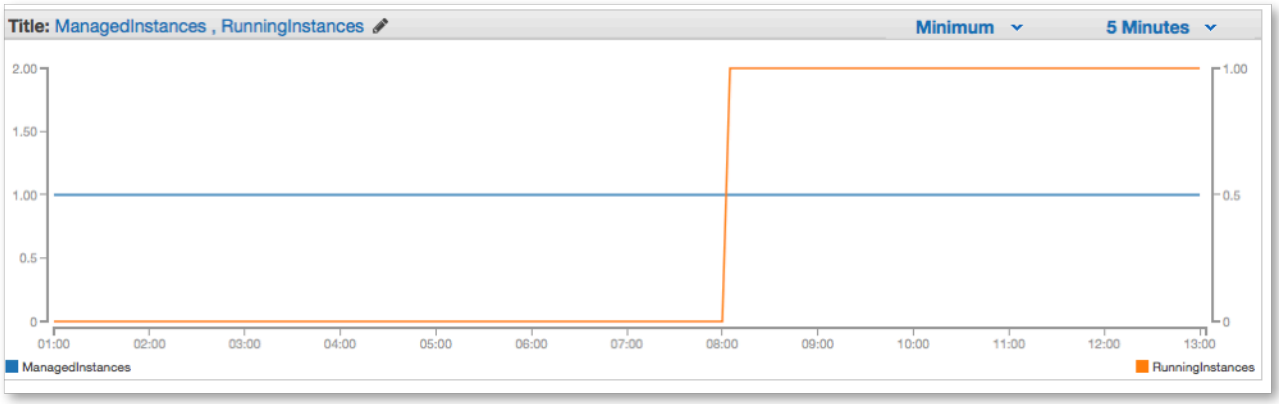

Figure 5: Metrics graph

Additional resources

- Amazon Elastic Compute Cloud
- Amazon Relational Database Service
- AWS CloudFormation
- AWS Lambda
- Amazon DynamoDB
- Amazon CloudWatch
- AWS Key Management Service
- AWS Identity and Access Management
- Amazon Simple Notification Service

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.

Credentials

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

Your credentials must have the following permissions:

- `lambda:InvokeFunction` – To invoke the `InstanceSchedulerMain` function in the scheduler stack, and to update the schedule and period information in the scheduler configuration database from the command line
- `cloudformation:DescribeStackResource` – To retrieve the physical resource ID of the AWS Lambda function from the stack to handle the CLI request

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

> **Note**
> If you specify a profile using the `profile-name` argument, the profile you specify must have these permissions. For more information about the `profile-name` argument, refer to Common Arguments (p. 30).

Install the Scheduler CLI

1. Download and unzip the scheduler CLI package.
2. In the `scheduler-cli` directory, run the `setup` Python script:

```
python setup.py install
```

Command structure

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

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

Common arguments

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

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

Available commands

create-period

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 stop. 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^{th} 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^{th} month.

Required: No

`--monthdays`

The days of the month for the period

Type: String

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

Required: No

Example

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

create-schedule

Description

Creates a schedule

Arguments

`--name`

The name of the schedule

Type: String

Required: Yes

`--description`

A description of the schedule

Type: String

Required: No

`--enforced`

Enforces the scheduled state for the instance

Type: Boolean

Constraints: Value must be `true` or `false`. By default, this parameter is set to `false`.

Required: No

`--use-metrics`

Collect Amazon CloudWatch metrics

Type: Boolean

Constraints: Value must be `true` or `false`. By default, this parameter is set to `false`.

Required: No

`--override-status`

Overrides the status of an instance to keep the instance in the specified state

Type: String

Constraints: Acceptable values are `running` or `stopped`

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 period rules is 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 (if `override-status` is not used)

`--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

Type: Boolean

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`

Add 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 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 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 (p. 30).

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 (p. 32).

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

help

Description

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

Example

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

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

subcommands:
  Valid subcommands

  {create-period,create-schedule,delete-period,delete-schedule,describe-periods,describe-
schedule-usage,describe-schedules,update-period,update-schedule}
                        Commands help
    create-period       Creates a period
    create-schedule     Creates a schedule
    delete-period       Deletes a period
    delete-schedule     Deletes a schedule
    describe-periods    Describes configured periods
    describe-schedule-usage
                        Calculates periods and billing hours in which
                        instances are running
    describe-schedules  Described configured schedules
    update-period       Updates a period
    update-schedule     Updates a schedule
```

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

Specific Command Example

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

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

Solution resources

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

Resource name	Type	Description
Main	`AWS::Lambda::Function`	Instance Scheduler AWS Lambda function
Scheduler Config Helper	`Custom::ServiceSetup`	Stores global configuration settings in Amazon DynamoDB
Scheduler Invoke Permission	`AWS::Lambda::Permission`	Allows the Amazon CloudWatch event to invoke the Instance Scheduler's AWS Lambda function
Scheduler Logs	AWS::Logs::LogGroup	CloudWatch Log Group for Instance Scheduler
Scheduler Policy	`AWS::IAM::Policy`	Policy that allows scheduler to perform start and stop actions, change Amazon EC2 instance attributes, set tags, and access scheduler resources
Scheduler Rule	`AWS::Events::Rule`	Amazon CloudWatch event rule that invokes the scheduler's Lambda function
State Table	`AWS::DynamoDB::Table`	DynamoDB table that stores last desired state of instances
Config Table	`AWS::DynamoDB::Table`	DynamoDB table that stores global configuration, schedule, and period data
Config Table Auto Scaling Read Target	`AWS::ApplicationAutoScaling::ScalableTarget`	Auto Scaling target for configuration table read capacity
Config Table Auto Scaling Write Target	`AWS::ApplicationAutoScaling::ScalableTarget`	Auto Scaling target for configuration table write capacity
Configuration Table Auto Scaling Read Policy	`AWS::ApplicationAutoScaling::ScalingPolicy`	Auto Scaling policy for configuration table read capacity
Configuration Table Auto Scaling Write Policy	`AWS::ApplicationAutoScaling::ScalingPolicy`	Auto Scaling policy for configuration table write capacity
State Table Auto Scaling Read Target	`AWS::ApplicationAutoScaling::ScalableTarget`	Auto Scaling target for state table read capacity
State Table Auto Scaling Write Target	`AWS::ApplicationAutoScaling::ScalableTarget`	Auto Scaling target for state table write capacity

Resource name	Type	Description
State Table Auto Scaling Read Policy	`AWS::ApplicationAutoScaling::ScalingPolicy`	Auto Scaling policy for state table read capacity
State Table Auto Scaling Write Policy	`AWS::ApplicationAutoScaling::ScalingPolicy`	Auto Scaling policy for state table write capacity
Instance Scheduler SNS Topic	`AWS::SNS::Topic`	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-yyyymmdd`: Logs general scheduler messages
- `CloudWatchEventHandler-yyyymmdd`: Logs general Amazon CloudWatch event rule information
- `SchedulerSetupHandler`: Logs the output of configuration actions
- `Scheduler-<service>-<account>-<region>-yyyymmdd`: Logs the service, account, and region of each scheduled instance
- `AdminCliRequestHandler-yyyymmdd`: Logs requests from the admin CLI

Custom resource

You can use the AWS CloudFormation custom resource (`ServiceInstanceSchedule`) to configure schedules and periods. Download the template and update the `ServiceToken` value to the one generated in your instance scheduler stack and use that template to create a new stack to add schedules and periods. The custom resource uses the same names and syntax as the fields in the configuration Amazon DynamoDB table. For more information on the fields for schedules, refer to Schedule definitions (p. 8). For more information on the fields for periods, refer to Period definitions (p. 11).

When using the custom resource to configure a schedule or period, you must include the account, AWS Region, and name of the deployed schedule stack in the `ServiceToken` field.

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.

The following sample custom resource code shows a schedule (`NL-OfficeHours`) that contains two periods.

```
AWSTemplateFormatVersion: 2010-09-09
Metadata:
  'AWS::CloudFormation::Designer': {}
Resources:
  DutchOfficehours:
    Type: 'Custom::ServiceInstanceSchedule'
    Properties:
      Name: NL-OfficeHours
      NoStackPrefix: 'True'
      Description: Office hours in NL
      ServiceToken: >-
        arn:aws:lambda:[Region]:[AccountId]:function:[stackName]-InstanceSchedulerMain
      Enforced: 'True'
      Hibernate: 'True'
      Timezone: Europe/Amsterdam
      Periods:
        - Description: Opening hours on weekdays
          BeginTime: '08:00'
          EndTime: '18:00'
          WeekDays: Mon-Fri
        - Description: Opening hours in weekend
          BeginTime: '09:00'
          EndTime: '17:00'
          WeekDays: Sat-Sun
```

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

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

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

Sample schedule

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 shows how to configure the periods and schedule required to run instances 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 `Schedule` as the tag key and `mon-9am-fri-5pm` as the tag value for the instances. You can run the following AWS CLI command:

```
aws ec2 create-tags --tags "Key=Schedule,Value=mon-9am-fri-5pm" --resources "[INSTANCE_ID]"
```

If you changed 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 `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 (p. 29), 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
```

Uninstall the solution

To uninstall the Instance Scheduler on AWS solution, use the following procedures.

Using the AWS Management Console

1. Sign in to the AWS CloudFormation console.
2. 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>
```

Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how you use this solution to improve the services and products that we offer. When turned on, the following information is collected and sent to AWS each time the Instance Scheduler AWS Lambda function runs:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Instance Data:** Count of the state and type of instances that are managed by the Instance Scheduler in each AWS Region

  Example data:

  - `Running: {t2.micro: 2}, {m3.large:2}`
  - `Stopped: {t2.large: 1}, {m3.xlarge:3}`
  - `Resized: {t2.large-t2.small:3}`

The following information is collected and sent to AWS at stack creation or deletion:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Stack Version:** The version of the stack that is created or deleted
- **Status:** The status of the stack (`stack_created` or `stack_deleted`)
- **Region:** The Region where the stack is deployed

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section from:

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

to

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

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). Refer to the README.md file for additional information.

Contributors

The following individuals contributed to this document:

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

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 CHANGELOG.md file 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 CHANGELOG.md file 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 CHANGELOG.md file in the GitHub repository.
May 2022	Minor update and bug fixes. For more information about v1.4.1, refer to the CHANGELOG.md file in the GitHub repository.
January 2023	Minor update and bug fixes. For more information about v1.4.2, refer to the CHANGELOG.md file 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

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.