EC2 Scheduler

AWS Implementation Guide

Lalit Grover

September 2016

Last updated: September 2017 (see revisions)

Notice: EC2 Scheduler has been superseded by AWS Instance Scheduler.

In 2016, the EC2 Scheduler was launched to help AWS customers easily configure custom start and stop schedules for their Amazon Elastic Compute Cloud (Amazon EC2) instances. In 2018, AWS launched the AWS Instance Scheduler, a new and improved scheduling solution that enables customers to schedule Amazon EC2 instances, Amazon Relational Database Service (Amazon RDS) instances, and more. We encourage customers to migrate to AWS Instance Scheduler for future updates and new features.

Legacy templates, scripts, and documentation for EC2 Scheduler are available in our GitHub repository.

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the EC2 Scheduler on the Amazon Web Services (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 on the AWS Cloud.

Overview

EC2 Scheduler is a solution that automates the starting and stopping of Amazon Elastic Compute Cloud (Amazon EC2) instances.

The EC2 Scheduler leverages Amazon EC2 resource tags and AWS Lambda to automatically stop[1] and restart Amazon EC2 instances on a customer-defined schedule. The solution is easy to deploy and can help reduce operational costs. For example, an organization can use the EC2 Scheduler in a production environment to automatically stop Amazon EC2 instances every day, outside of business hours. For customers who leave all of their Amazon EC2 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).

Cost

You are responsible for the cost of the AWS services used while running the EC2 Scheduler. There is no additional cost for deploying the automated solution. As of the date of publication, the cost for running this solution with default settings in the US East (N. Virginia) Region is approximately $1.27 per month in AWS Lambda charges, or less if you have Lambda free tier[2] monthly usage credit. This is independent of the number of Amazon EC2 instances you are running. The optional custom Amazon CloudWatch metric, if enabled, will cost an additional $0.50/month per Amazon EC2 instance. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

[1] Note that *stopping* an Amazon EC2 instance is different than *terminating* an Amazon EC2 instance. By default, Amazon EC2 instances are configured to stop, not terminate, when shut down, but customers can modify this behavior. Before using this solution, verify that instances are set to stop or terminate as appropriate.

[2] https://aws.amazon.com/lambda/pricing/

Architecture Overview

Automatic deployment of this solution configures the following components and functionality.

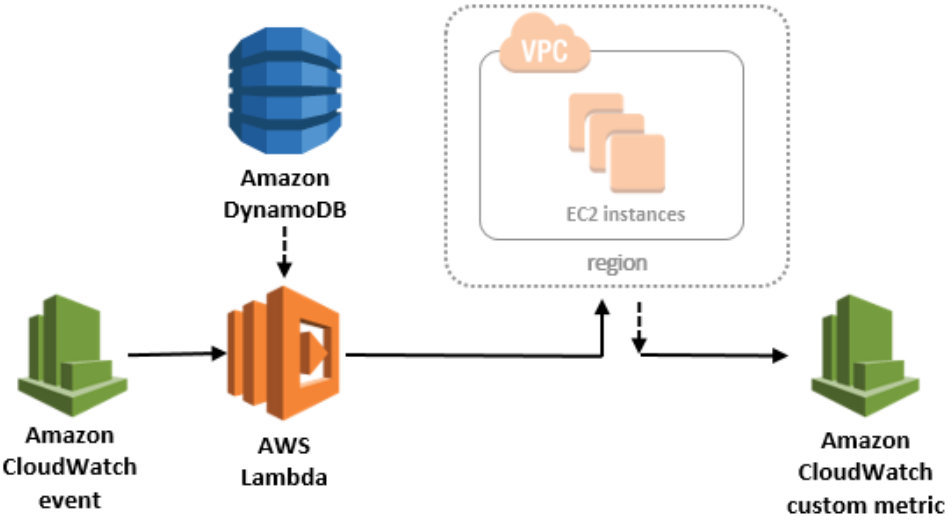


Figure 1: EC2 Scheduler on the AWS Cloud

The AWS CloudFormation template sets up an Amazon CloudWatch event at a customer-defined interval. This event invokes the EC2 Scheduler AWS Lambda function. During configuration, the user defines default start and stop parameters (in UTC) as well as a *custom tag* that the EC2 Scheduler will use to identify applicable Amazon EC2 instances. These values are stored in Amazon DynamoDB, and the Lambda function retrieves them each time it runs. The customer then applies the custom tag to applicable Amazon EC2 instances.

For example, a user might assign the custom tag the name (tag key) `scheduler:ec2-startstop`, and define default start and stop times of 15:00 and 1:00 UTC (3:00 pm and 1:00 am respectively) to be applied Monday through Thursday.

Note that this solution allows the user to define custom start and stop parameters that override the default values for an individual instance. Users can also append additional characters to the custom tag name (e.g. `scheduler:ec2-startstop:group1`) to easily identify different start-stop scenarios from the same EC2 Scheduler deployment. See Step 2. Tag Your Amazon EC2 Instances for detailed information about tag names and using the tag value to set custom parameters.

Each time the solution's Lambda function runs, it calls the Amazon EC2 API to check the current state of each appropriately tagged instance against the targeted state (defined by the solution's default values or custom parameters in the instance tag), and then applies the appropriate start or stop action, as necessary.

Continuing from the previous example, if the Lambda function is invoked on a Tuesday at 1:30 am (UTC) and it identifies a running Amazon EC2 instance with a `scheduler:ec2-startstop` tag, it will check if the tag value contains custom override parameters that confirm this behavior. If there are no custom parameters, the Lambda function will stop that instance in accordance with the user-defined default settings.

The Lambda function also records the instance ID and targeted state as an optional custom metric in Amazon CloudWatch (see Amazon CloudWatch Metrics).

Design Considerations

Weekly Functionality

The current version of the EC2 Scheduler allows customers to create weekly schedules. For example, you can set it to run on specific days of the week or all days, however the current version does not support biweekly or monthly intervals. If you want to temporarily stop or modify EC2 Scheduler actions for a particular week or month, you must manually modify or remove resource tags on applicable Amazon EC2 instances.

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.

Start and stop actions can be further modified and customized at the instance level. See Step 2. Tag Your Amazon EC2 Instances for more information.

Regional Deployments

Customers can deploy the EC2 Scheduler in any AWS Region. Once deployed, the EC2 Scheduler applies the appropriate start or stop actions to tagged Amazon EC2 instances in all AWS Regions[3] of a customer's account.

> **Important:** EC2 Scheduler actions will affect appropriately tagged instances in all AWS Regions[3] of your account, even though the Lambda function is running in a single region.

Customers can run active deployments of the EC2 Scheduler in different AWS Regions. Customers can also run multiple EC2 Schedulers in the same AWS Region, however they must be sure to assign different names to the respective custom tag names and Amazon DynamoDB tables during initial deployment.

[3] At the time of publication, this does not include AWS GovCloud (US) or the China (Beijing) Region.

Multiple deployments of the solution enable customers to define separate custom tag keys with different default start-stop parameters for each deployment, reducing the manual management of custom tag values. For example, two teams working on a different office schedule might require different default start and stop times for large groups of instances. Separate EC2 Scheduler deployments give each team full control over their default parameters and custom tag keys so that they can run two fully independent EC2 Schedulers.

Instance Shutdown Behavior

This solution is designed to automatically stop 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 customers can modify this behavior.[4] Therefore, make sure that the instances you control using the EC2 Scheduler are configured with a `Stop` shutdown behavior, otherwise they will be terminated.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the EC2 Scheduler on the AWS Cloud. It includes the following CloudFormation template, which you can download before deployment:

View template

ec2-scheduler.template: This is the primary solution template you use to launch the EC2 Scheduler and all associated components. The default configuration deploys AWS Lambda functions, an Amazon DynamoDB table, an Amazon CloudWatch event, and CloudWatch custom metrics, but you can also customize the template based on your specific network needs.

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 the EC2 Scheduler into your account.

Time to deploy: Approximately five (5) minutes.

[4] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/terminating-instances.html#Using_ChangingInstanceInitiatedShutdownBehavior

What We'll Cover

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 Stack

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

Step 2. Tag Your Amazon EC2 Instances

- Apply the custom tag to applicable instances.

Step 1. Launch the Stack

This automated AWS CloudFormation template deploys the EC2 Scheduler in AWS Lambda, and configures related components. Please make sure that you've verified the settings for your Amazon EC2 instances before launching the stack.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Log in to the AWS Management Console and click the button to the right to launch the `ec2-scheduler` AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

   Launch Solution

2. The template is launched in the US East (N. Virginia) Region by default. To launch the EC2 Scheduler 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 EC2 Scheduler stack.
5. Under **Parameters**, review the parameters for the template, and modify them as necessary.

> **Note:** The Lambda function checks your current preferences in Amazon DynamoDB each time it runs, so you can manually modify these values at any time.

This solution uses the following default values.

Parameter	Default	Description
CustomTagName	`scheduler:ec2-startstop` <Or user defined>	This tag identifies instances to receive automated actions; each instance that you want the solution to start and stop must have a tag key that matches this name. If you choose to modify the default tag key, make sure to use a name that will be easy to apply consistently and correctly across all applicable instances.
Schedule	`rate(5 minutes)`	Enter the scheduled expression for the CloudWatch Event rule that invokes the EC2 Scheduler Lambda function. For example: `rate([5 minutes|1 hour|1 day])` or `cron(0/5 * * * ? *)`
DefaultStartTime	`0800`	Default start time. **Important:** All times are in UTC in 24-hour format, no colon.
DefaultStopTime	`1800`	Default stop time **Important:** All times are in UTC in 24-hour format, no colon.
DefaultDaysActive	`all`	Default active day
DynamoDBTableName	`EC2-Scheduler`	Name of the Amazon DynamoDB table where default settings are stored. **Important:** If you deploy this solution twice in the same AWS Region, you **must** use different names for this parameter.
ReadCapacityUnits	`1`	Provisioned read throughput
WriteCapacityUnits	`1`	Provisioned write throughput
CloudWatchMetrics	`Enabled`	Collect custom metric data (see Amazon CloudWatch Metrics). **Important:** Leaving this feature enabled will incur charges of $0.50/month per instance. To opt out of this feature, choose `Disabled`.
SendAnonymousData	`Yes`	Send anonymous data to AWS to help us understand Amazon EC2 usage and related cost savings across our customer base as a whole. To opt out of this feature, choose `No`. For more information, see the appendix.

6. Choose **Next.**
7. On the **Options** page, choose **Next**.
8. 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.
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 roughly five (5) minutes.

> **Note:** In addition to the primary AWS Lambda function `ec2-scheduler`, this solution includes the `solution-helper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.
>
> When running this solution, you will see both Lambda functions in the AWS Lambda console, but only the primary `ec2-scheduler` function is regularly active. However, do not delete the `solution-helper` function as it is necessary to manage associated resources.

Step 2. Tag Your Amazon EC2 Instances

When you deployed the CloudFormation template, you defined the name (tag key) for the solution's *custom tag*. For the EC2 Scheduler to recognize an Amazon EC2 instance, the tag key on that Amazon EC2 instance must match (or start with) the custom tag name stored in the Amazon DynamoDB table. Therefore, it is important that you apply tags consistently and correctly to all applicable Amazon EC2 instances. You can continue to use existing tagging strategies for your instances while using this solution (see Tagging Your Amazon EC2 Resources for more information on instance tagging).

> **Note:** After initial deployment, you can manually change the custom tag key in DynamoDB if necessary.

On the AWS Management Console, use the Tag Editor to apply or modify tags for multiple Amazon EC2 instances at a time. You can also apply and modify tags manually in the Amazon EC2 console.

Setting the Tag Value

As you apply a tag, set the tag value accordingly to activate different EC2 Scheduler actions:

- To apply the default actions to an instance, set the tag value to `true` or `default`. The EC2 Scheduler will retrieve default parameters from the Amazon DynamoDB table.
- To temporarily exclude an instance from EC2 Scheduler actions, set the tag value to `none` or to an empty string (clear the field).
- To permanently exclude an instance from EC2 Scheduler actions, remove the solution tag.
- To apply custom start and stop parameters on an instance-by-instance basis, see the next section.

Applying Custom Parameters

You can apply custom start and stop parameters to an instance which will override the default values you set during initial deployment. To do this, modify the tag value to specify the alternative settings.

The EC2 Scheduler will read tag values, looking for four possible custom parameters in following order: `<start time>; <stop time>; <time zone>; <active day(s)>`

You must separate each value with a semicolon. The following table gives acceptable input values for each field

Tag Value Field	Acceptable input values
<start time>	`none` or time in 24-hour format (UTC, with no colon)
<stop time>	`none` or time in 24-hour format (UTC, with no colon)
<time zone>	`utc` **Note:** The time zone parameter is set to UTC and cannot be modified at this time.
<active day(s)>	`all`, `weekdays`, or `mon`, `tue`, `wed`, `thu`, `fri`, `sat`, and/or `sun`.

The following table gives examples of different tag values and the resulting EC2 Scheduler actions.

Example Tag Value	EC2 Scheduler Action
`0800;1800;utc;all`	The instance will start at 0800 hours and stop at 1800 hours on all days of the week.
`1000;1700;utc;weekdays`	The instance will start at 1000 hours and stop at 1700 hours Monday through Friday.
`1030;1700;utc;mon,tue,fri`	The instance will start at 1030 hours and stop at 1700 hours on Monday, Tuesday, and Friday only.
`0815;1745;utc;wed,thu`	The instance will start at 0815 hours and stop at 1745 hours on Wednesday and Thursday only.
`none;1800;utc;weekdays`	The instance stop at 1800 hours Monday through Friday. The instance will need to be started manually.
`0800;none;utc;weekdays`	The instance start at 0800 hours Monday through Friday. The instance will need to be stopped manually.
`none;none;utc;weekdays`	The instance must be started and stopped manually (no automated actions).
`0800`	The instance will start at 0800 hour and stop at the default stop time on the default active days stored in the Amazon DynamoDB table.

Example Tag Value	EC2 Scheduler Action
`0800;1800`	The instance will start at 0800 hours and stop at 1800 hours on the default active days stored in the Amazon DynamoDB table.
`0800;1800;utc`	The instance will start at 0800 hours and stop at 1800 hours on the default active days stored in the Amazon DynamoDB table.
`default`	The instance will start and stop on the default schedule.
`True`	The instance will start and stop on the default schedule.
`<EMPTY>`	There will be no action taken on the instance.
`<Random String>`	There will be no action taken on the instance.

Modifying Tag Keys

During initial deployment, the name you assign to the custom tag is stored in Amazon DynamoDB. When the AWS Lambda function runs, it will look for all instance tags that contain or start with that exact name. After it reads the full name, it ignores any other characters on the tag key string.

As mentioned in the Architecture Overview section, you have the option to append additional characters to the tag key. This feature allows you to apply multiple sets of custom parameters to the same instance using a single EC2 Scheduler deployment, because the tag keys are unique. It also allows you to create logical groupings of resources for easier filtering.

For example, a team deploys the solution using the custom tag key `scheduler:ec2-startstop` and default parameters of a 1200 start time and 2100 stop time (UTC) on weekdays. The following example scenarios explain how they can use multiple tags.

Example 1: There is a subset of instances that require three different start and stop schedules within a single week: longer days on Monday and Wednesday, late days on Tuesday, Thursday, and Friday, and short days on weekends. Instead of deploying the full solution three times with three different default settings, the team uses a single deployment and applies three tags to those instances. They choose to append the tag keys with `:long`, `:late`, and `:short` to help identify and manage the three schedules for a single instance.

Tag Key	Tag Value
`scheduler:ec2-startstop:long`	`0800;2200;utc;mon,wed`
`scheduler:ec2-startstop:late`	`1500;2400;utc;tue,thu,fri`
`scheduler:ec2-startstop:short`	`1400;1800;utc;sat,sun`

Example 2: There is a different subset of instances that need to start earlier (0800 UTC) on Tuesdays but use the default times for the rest of the week. The team would apply two tags to those instances: one that uses the default tag key and another that they append with `:alt`.

Tag Key	Tag Value
`scheduler:ec2-startstop`	`default` (or `true`)
`scheduler:ec2-startstop:alt`	`0800;none;utc;tue`

In this scenario, the EC2 Scheduler would start the applicable instances earlier on Tuesdays, but revert to the default stop time (2100 UTC). Although the second (`:alt`) tag specifies no stop time (`none`), the instances also have the first tag, which runs the EC2 Scheduler and applies the default start and stop times on weekdays (Tuesdays included).

> **Note:** When applying multiple, appended solution tags, be aware of overlapping day or time ranges. The EC2 Scheduler function will apply actions as specified in **all** tag values.
>
> In Example 2, if the second tag value specified a custom start time of 1300 UTC, it would have no effect because the EC2 Scheduler would have already started the instance at the default start time of 1200 UTC.

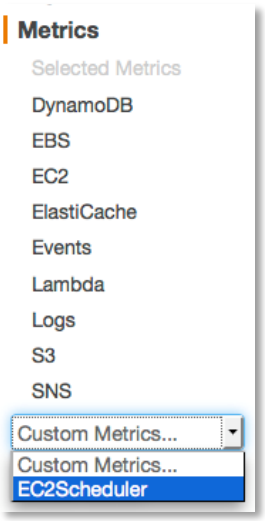
Amazon CloudWatch Metrics

This solution creates a new custom Amazon CloudWatch metric named `EC2Scheduler`. 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 instance ID, the desired state for that instance, and the AWS Region where the instance exists. The desired instance state is recorded as either `1` (running) or `0` (stopped). If the Lambda function runs and there are no tagged instances, it will record that there are no instances to start or stop.

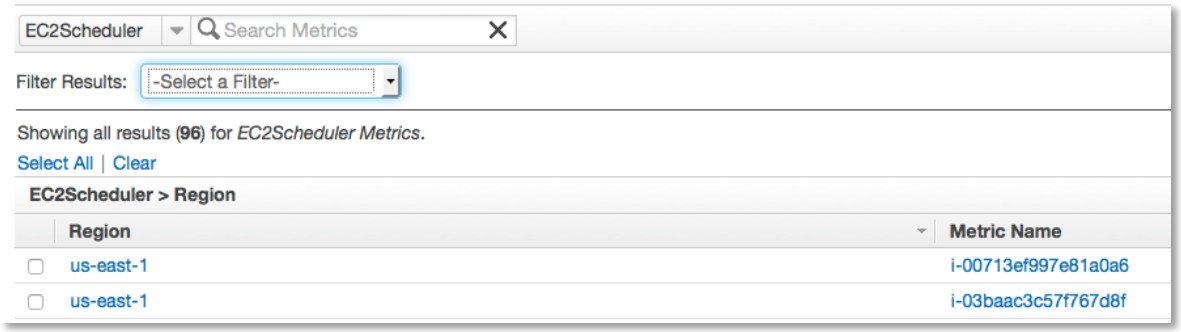
This metric allows users to review the expected change in state of their instances over time. It can also aid in troubleshooting instances that are in an incorrect state. For example, if an instance's actual state is (or was) not equal to its desired state as recorded in the metric, this can help determine whether the problem is related to the EC2 Scheduler (such as an incorrect tag) or if the instance failed to start or stop for some other reason.

View EC2 Scheduler Metrics

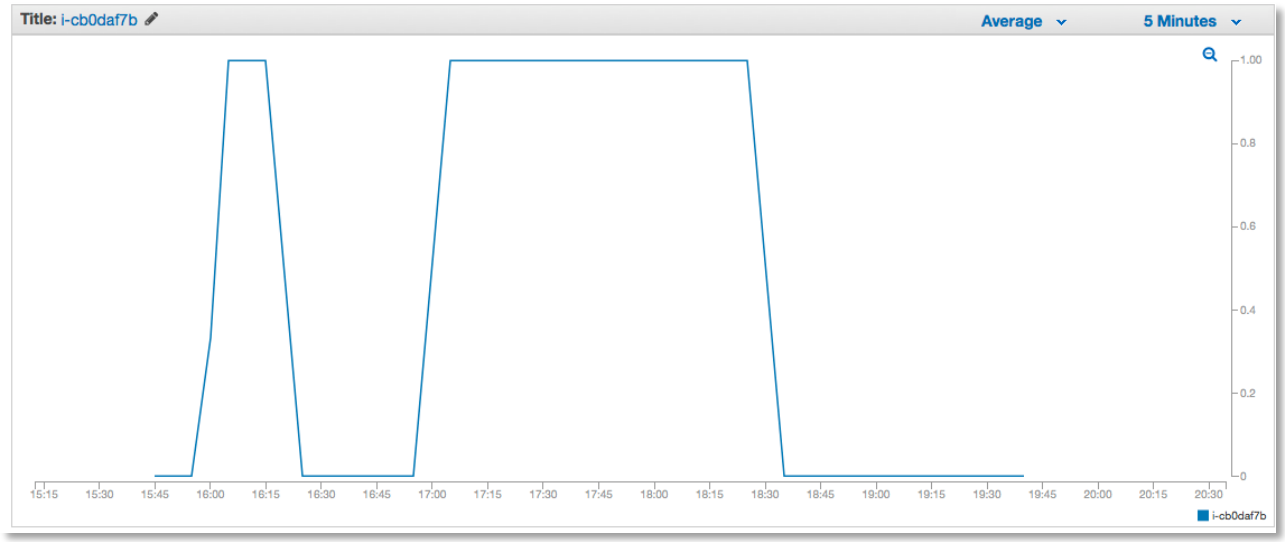
1. In the AWS Management Console, open the Amazon CloudWatch console.
2. In the **Custom Metrics** drop-down field, choose `EC2Scheduler`.



3. Select the Amazon EC2 instance(s) that you want to view the status of.



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.



Additional Resources

AWS services documentation

- AWS CloudFormation
- Amazon EC2 User Guide for Linux instances
- Information on Instance Initiated Shutdown Behavior
- AWS Lambda
- Amazon DynamoDB
- Amazon CloudWatch

Appendix: Collection of Anonymous Data

This solution includes an option to send anonymous usage data to AWS. We use this data to better understand how customers use this solution to improve the services and products that we offer. When enabled, the following information is collected and sent to AWS each time the EC2 Scheduler Lambda function runs:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each EC2 Scheduler deployment
- **Timestamp:** Data-collection timestamp
- **Instance Data:** Count of the state and type of instances that are managed by the EC2 Scheduler in each AWS Region
  Example data:

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

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, set the **SendAnonymousData** parameter to `No`.

Send Us Feedback

We welcome your questions and comments. Please post your feedback on the AWS Solutions Discussion Forum.

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change	In sections
September 2016	Initial release	--
September 2016	Removed reference to outdated instance tag limit.	Step 2
September 2017	Clarified template parameters and tag key options.	Step 1, Modifying Tag Keys

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. 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 EC2 Scheduler solution is licensed under the terms of the Amazon Software License available at https://aws.amazon.com/asl/.