Ops Automator

AWS Implementation Guide

Arie Leeuwesteijn

Mahmoud ElZayet

Ruald Andreae

Mike O'Brien

July 2017

Last updated: August 2020 (refer to revisions)

Contents

About this guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Ops Automator solution in the AWS Cloud. It includes links to AWS CloudFormation templates that launch, configure, and run the AWS services required to deploy this solution on AWS, 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.

Overview

Amazon Web Services (AWS) offers its customers several methods to help automate manual tasks or processes such as archiving, backup and data recovery, resource selection and scheduling, and configuration management. Automating these tasks can help increase reliability, reduce costs, and reduce operational complexity so you can focus on delivering applications and services at a high velocity. However, it can be a challenge to automate operational tasks on various AWS resources that exist across multiple Regions and accounts.

To help customers more easily manage cross-account and cross-region automation, AWS offers the Ops Automator solution. This solution is built on a core framework that provides the infrastructure for task audit trails, logging, resource selection, scaling, AWS API retries, completion handing for long tasks, and concurrency handling.

The Ops Automator solution is easy to deploy and features an extensive suite of actions that use time-based, interval-based, or event-based triggers to automatically manage AWS resources.

Ops Automator can use resource tags to identify which resources will receive automated actions, allowing you to customize automation at the individual-resource level.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost for running this solution depends on the number of actions you run and what the actions do. As of the date of publication, the cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$10 per month**, considering the following factors:

- 1 GB of Amazon DynamoDB data storage at the default throughput capacity
- 1 GB of Amazon CloudWatch Logs ingested
- 1 million AWS Lambda function run
- 1 million CloudWatch custom events generated

By default, this solution enables auto scaling for its Amazon DynamoDB tables to provide sufficient read and write capacity for resource tracking and configuration data. If you use this solution to manage a large number of resources, auto scaling can increase Amazon DynamoDB charges.

This pricing does not reflect variable charges incurred from the implemented actions, data transfer fees, or snapshot storage costs. 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 with the default parameters builds the following environment in the AWS Cloud.

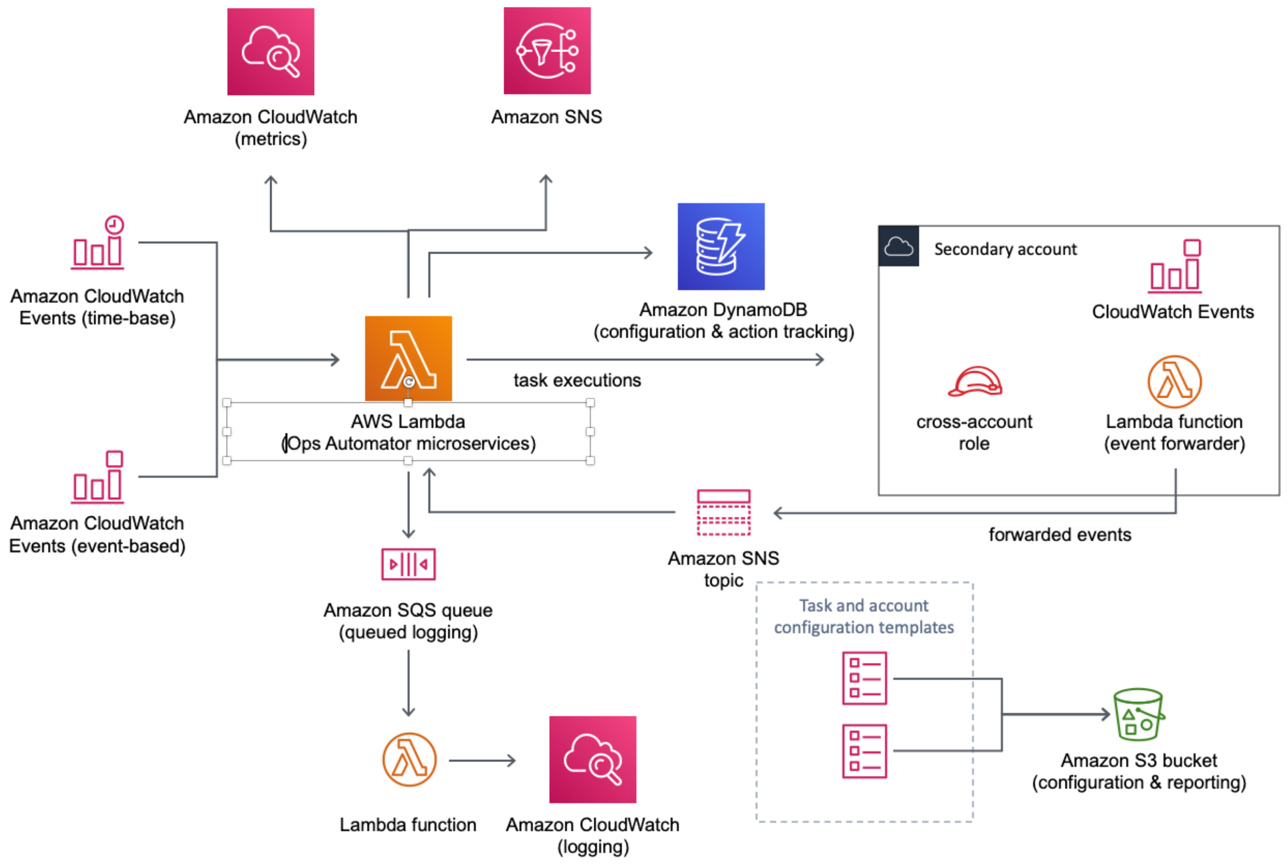
Amazon CloudWatch (metrics)
Amazon SNS
Amazon CloudWatch Events (time-base)
Amazon DynamoDB (configuration & action tracking)
Secondary account
CloudWatch Events
task executions
AWS Lambda (Ops Automator microservices)
cross-account role
Lambda function (event forwarder)
Amazon CloudWatch Events (event-based)
Amazon SNS topic
forwarded events
Amazon SQS queue (queued logging)
Task and account configuration templates
Amazon S3 bucket (configuration & reporting)
Lambda function
Amazon CloudWatch (logging)

Figure 1: Ops Automator architecture

This solution includes an AWS CloudFormation template that must be deployed in the primary account. For guidance on choosing a primary account, refer to the Security section.

This template launches all solution components, including a set of microservices (AWS Lambda functions) that manage triggering events, resource selection, task execution, concurrency control, and completion; Amazon DynamoDB tables that store task-related data; Amazon CloudWatch for logging and metrics collection; Amazon Simple Notification Service (Amazon SNS) for event forwarding and notifications; Amazon Simple Queue Service (Amazon SQS) for logging; and Amazon Simple Storage Service (Amazon S3) for task output.

The primary template automatically generates additional AWS CloudFormation templates in an Amazon S3 bucket. These templates allow you to create cross-account AWS Identity and Access Management (IAM) roles to perform actions in secondary accounts, and to forward events. You can modify and build upon these templates to create custom actions that extend the solution's functionality. The solution also includes additional templates that you can use as references to configure and combine multiple task stacks to create end-to-end scenarios that handle complex tasks, such as vertical scaling for [Amazon Elastic Compute Cloud](#) (Amazon EC2) instances.

During initial configuration of the primary AWS CloudFormation template, you define a tag *key* which you use to identify resources that will receive automated actions. When you deploy a *task* template, the stack name you specify is used as one or more tag *values* that identify the tasks you want the solution to perform on the tagged resource. For example, a user might assign the custom tag name (tag key) `OpsAutomatorTaskList` and create a task stack called `Delete7` that deletes a resource after seven days. To identify the resource for deletion, the user adds the `OpsAutomatorTaskList` tag key with a value of `Delete7`.

Or, you can use a tag filter expression to specify the tasks you want to perform on resources. The expression overwrites the selection of resources based on the values in the `OpsAutomatorTaskList` tag key. A tag filter expression enables a more granular selection of resources. Tag filter expressions allow the use of wildcards, regular expressions, and Boolean operators for tag names and values. For more information, refer to [Appendix D](#).

You can also configure tasks to send all started and ended actions to an Amazon SNS topic, and to send detailed metrics to Amazon CloudWatch Metrics.

Task execution process

The automated process begins when either a time-based, interval-based, or event-based trigger invokes the solution's *scheduler* Lambda function. The scheduler checks the task configuration to determine which tasks must be run. Based on the metadata from the action that is configured for the task, the scheduler runs one or more Lambda functions that use tags to identify and select resources across accounts and regions. Then, the scheduler creates task execution items, and starts one or more Lambda functions that implements the logic of the action on the resource or resources. Most actions contain completion logic. If the action contains completions logic, the framework periodically calls this logic to check whether the execution step has completed. The framework continues to do this until the logic of the action decides that the action is complete, or the action has timed out.

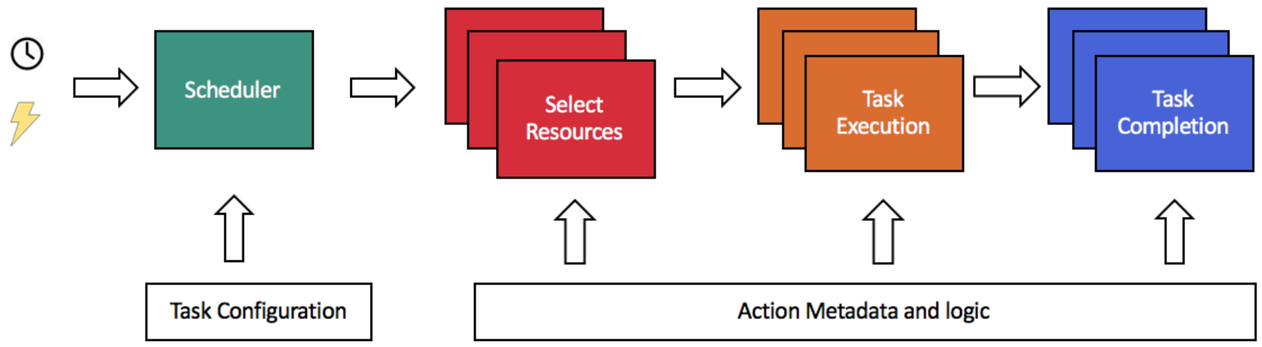

Figure 2: Task execution process

Solution components

Included actions

Ops Automator features an extensive suite of actions that you can use to configure automated tasks. An *action* consists of metadata that specifies which resources and properties to select from AWS services, the aggregation level of the selected resources, the permissions required to access and modify that resource, and the code that implements the action logic. An action can be used in multiple tasks with different parameters or resource selection criteria.

To initiate an action, you must launch the specific task template. This creates a new *task* that defines the time interval or resource event that triggers the action, the action parameters that define how the task should be performed, the tags that determine which resources will receive the actions, and the accounts and Regions where the resources are located. For more information about the task-configuration templates included with this solution, refer to Appendix A.

For a complete list of Ops Automator actions, refer to Appendix B.

Combining actions

With Ops Automator, you can combine multiple task stacks to create end-to-end scenarios that handle complex tasks. Ops Automator actions can dynamically tag created and affected resources for follow up actions. For example, you can configure a task to create snapshots for Amazon Elastic Block Store (Amazon EBS) volumes and tag those snapshots. Another task can select those snapshots and delete them automatically using a retention count or a number of retention days. For a sample configuration, refer to Appendix G.

You can set actions to be triggered by resource-specific or tag-modification events, enabling you to construct event-driven workflows.

The solution includes an example template you can use to combine and deploy multiple tasks as custom resources from a single AWS CloudFormation template. You can find the template in the solution's Amazon Simple Storage Service (Amazon S3) bucket in the **TaskConfigurations/ScenarioTemplates** folder. For more information, refer to Appendix H.

Role and task templates

When you deploy Ops Automator, the solution automatically creates an Amazon S3 bucket in the primary account that contains folders with configuration templates and scripts for creating tasks.

One folder (`AccountsConfiguration`) contains the templates required to create the AWS Identity and Access Management (IAM) role necessary to perform tasks in secondary accounts, and to forward events from these secondary accounts to the primary account.

Another folder (`TaskConfiguration`) contains templates to create tasks based on parameters you define.

The bucket also contains an HTML file (`ActionsConfiguration.html`) that lists all available actions. When you download this file and open it in a web browser, you can use the links in it to open the AWS Management Console and create a task for the selected action.

If you delete the Ops Automator stack in the primary account, all task stacks and configurations are deleted.

Task execution across accounts

Role configuration

To perform tasks on resources in secondary accounts, you must launch the role configuration template (`AccountRoleConfiguration.template`) in each secondary account. When the role configuration stack is launched in a secondary account, it creates a trust relationship with the IAM role in the Ops Automator primary stack that allows the solution's Lambda function to make the required API calls across accounts.

The role configuration template includes AWS CloudFormation parameters for every available Ops Automator action grouped by AWS service. To give the Lambda function permission to perform a specific action in the secondary account, set the applicable parameter to `Yes`. For example, to allow the solution to create backups in Amazon DynamoDB in the secondary account, set the **DynamoDB Create backup** parameter to `Yes`.

The template includes a parameter (**Custom Rolename**) that allows you to specify a custom role name that you can use for task execution in secondary accounts. A custom role name allows you to split Ops Automator permissions into smaller subsets for groups of tasks, or to specify a name that complies with company naming standards.

Important: You must deploy the `AccountRoleConfiguration` template in the primary account if you want to perform tasks on resources in the primary account. You must also use the same value for the **Custom Rolename** parameter across all stacks. For example, if you deploy the Ops Automator stack in account A and want to run the Amazon EC2 Create Snapshot task in accounts A and B, you must deploy the template in account A and account B. You must also use the same **Custom Rolename** in accounts A and B.

The solution is configured to use the default role name (`<ops-automator-stackname>-OpsAutomatorActionsRole`). If you want to use a custom role name, you must specify the custom name in the **Cross-Account Role name** parameter in the applicable task template. The solution uses this role to run the task in all applicable secondary accounts.

Important: We recommend using AWS CloudFormation StackSets to deploy these stacks in an easy and consistent way.

Event forwarding

Ops Automator tasks can be triggered by events. For example, a task that starts an Amazon Elastic Compute Cloud (Amazon EC2) instance can be triggered when tags are set on an instance. To enable the solution to trigger these event-based tasks across accounts and Regions, the events from those accounts and Regions must be forwarded to the primary account. Use the event forwarder template (`AccountForwardEvents`) to forward events from secondary accounts and Regions to the primary account. You must deploy this template in each applicable secondary account and Region to forward events.

When the event forwarder stack is launched, it creates Amazon CloudWatch rules for selected events and uses an AWS Lambda function to forward events to the primary Ops Automator Lambda function via an Amazon Simple Notification Service (Amazon SNS) topic. Applicable accounts are automatically allowed access to put events in the SNS topic.

The event forwarder template includes AWS CloudFormation parameters for every available event. To give the Lambda function permission to forward a specific event to the primary account, set the applicable parameter to `Yes`. For example, to allow the solution to forward tag-change events for Amazon EC2, set the **EC2 Tag Change events** parameter to `Yes`.

Important: We recommend using AWS CloudFormation StackSets to deploy these stacks in an easy and consistent way.

Placeholders

Ops Automator features several placeholders you can use in parameters such as names, prefixes, descriptions, and tag names and values set by Ops Automator actions. Placeholders have the format `{name}`. When tasks are run, placeholders are replaced with dynamic values. This allows the solution to give created resources dynamic names and descriptions, and to set dynamic tag names and values on created or affected resources. For more information, refer to Appendix E.

Performance

Ops Automator is designed to accommodate task processing for a large number of resources. By default, this solution enables on-demand provisioning for its Amazon DynamoDB tables to provide sufficient read and write capacity to store tracking and configuration data for each resource it manages. Some actions include parameters that enable you to select larger Lambda functions to run specific steps of a task.

Daily backups

Ops Automator creates a daily backup of the Amazon DynamoDB configuration table. Configuration backup files are stored in the `Backups` folder in the solution's Amazon S3 bucket. The solution's primary template includes a parameter where you define the retention period for these backup files.

Logging and notifications

Ops Automator leverages Amazon CloudWatch Logs for logging. The solution logs which AWS Lambda function handled each event; when each task was run; which tasks were run; the state of implemented tasks; whether each task completed; which resources were selected for each task; the execution time of the next task; and debugging, warning, and error messages. For more information, refer to Appendix F.

Warning and error messages are also published to a solution-created Amazon Simple Notification Service (Amazon SNS) topic. You can find the name of the Amazon SNS topic in the **Outputs** tab of the primary solution stack. The topic name is the value of the **IssueSNSTopic** key.

You can also configure tasks to send notifications for every started or completed task using the **TaskNotifications** parameter in the task template. If you set this parameter, a notification is sent to the Amazon SNS topic when the task execution starts and completes. You can find the name of the Amazon SNS topic in the **Outputs** tab of the primary solution stack. The topic name is the value of the **NotificationSNSTopic** key.

Amazon S3 buckets

This solution creates two Amazon S3 buckets. One bucket (`<ops-automator-stackname>-task-resources-suffix`) is used to store resource information for resources that are larger than 16 MB. The other bucket (`<ops-automator-stackname>-reporting-suffix`) is used to store reports generated by tasks. The folder structure of this bucket is `/actionname/taskname/reportname`.

Considerations

Solution updates

Ops Automator v2.x has been redesigned to improve throughput, simplify the configuration, and add event-triggered tasks.

If you are using a v1.x stack, you cannot update to v2.x. To use the latest version of the solution, delete the v1.x stack and follow the Automated Deployment steps to launch v2.x.

If you are using a v2.x stack, you can update to the current version, however we recommend deploying in a test environment first to confirm a successful deployment.

Encrypted Amazon EBS volumes

If your Amazon EC2 instances contain encrypted Amazon Elastic Block Store (Amazon EBS) volumes, you must grant Ops Automator permission to use the customer's primary key (CMK) to perform actions. Add the `kms:CreateGrant` permission to the Ops Automator role (`<stackname>-OpsAutomatorLambdaRole -<id>`).

Regional deployments

Some Ops Automator features are not available in the AWS GovCloud (US) Regions. Additionally, the solution uses Amazon Resource Names (ARNs) to perform tasks in secondary accounts. In the AWS GovCloud (US) Regions, ARNs have an identifier that is different from the one in other AWS Regions. Because not all solution features are supported in the Region and the difference in ARNs, Ops Automator is not available in the AWS GovCloud (US) Regions at this time.

AWS CloudFormation template

This solution uses AWS CloudFormation to automate the deployment of the Ops Automator solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment.

View template

ops-automator.template: Use this template to launch the Ops Automator and all associated components. The default configuration deploys AWS Lambda functions, Amazon DynamoDB tables, Amazon CloudWatch events, AWS CloudFormation templates, AWS Identity and Access Management roles, Amazon Simple Storage Service (Amazon S3) buckets, and Amazon Simple Notification Service (Amazon SNS) topics.

Automated deployment

Before you launch this solution, review the considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy Ops Automator into your account.

Time to deploy: 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 Ops Automator stack in the primary account

- Launch the AWS CloudFormation template into your primary AWS account.
- Enter values for required parameters: **Stack Name**.
- Review the other template parameters, and adjust if necessary.
- (Optional) Create an Amazon Elastic Container Service (Amazon ECS) image.

Step 2. Launch a task template in the primary account

- Launch the applicable task-configuration AWS CloudFormation template into the primary account.
- Review the template parameters, and adjust if necessary.

Step 3. (Optional) Launch a role template in the secondary account(s)

- Launch the applicable role AWS CloudFormation template into the secondary account with applicable resources.
- Enter values for required parameters: **Stack Name**.
- Review the template parameters, and adjust if necessary.

Step 4. (Optional) Launch the event forwarder template in secondary account(s)

- Launch the applicable event forwarder AWS CloudFormation template into the secondary account with applicable resources.
- Enter values for required parameters: **Stack Name**.
- Review the template parameters, and adjust if necessary.

Step 5. Tag your resources

- Apply the custom tag to applicable resources.

Step 1. Launch the Ops Automator stack in the primary account

You must deploy this AWS CloudFormation template in your primary account. Launch this template using an AWS Identity and Access Management (IAM) role specifically created for this purpose. For more information, refer to the Security section.

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

1. Sign in to the AWS Management Console and use the button to the right to launch the `ops-automator` AWS CloudFormation template.

   Launch Solution

   Optionally, you can download the template as a starting point for your own implementation.
2. The template launches in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the Region selector in the console navigation bar.

> **Note**: Ops Automator is not available in the AWS GovCloud (US) Regions at this time.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Task Scheduler Tag Name	`OpsAutomatorTaskList`	The tag key (name) that identifies applicable resources. The tag value will contain the list of tasks to be performed on tagged resources. Refer to Step 5 for detailed information.
Enable CloudWatch Metrics	`Yes`	Choose whether to collect CloudWatch Metrics data for Ops Automator. You can configure detailed metrics for individual tasks can be configured at the task-level.
Schedule active?	`Yes`	Choose whether to activate the scheduling task feature.
Clean up task tracking table?	`Yes`	Choose whether to clean the task tracking table.
Export Task Tracking Table to Amazon S3	`No`	Choose whether to export the task tracking table to Amazon S3.
Hours to keep tasks?	`168`	The number of hours to keep a task before it is automatically deleted from the tracking table.
Keep failed tasks?	`Yes`	Choose whether to store failed tasks in the Amazon DynamoDB table.
Log Retention Days	`30`	The number of days to keep logs before they are automatically deleted.
Days to keep configuration backups	`7`	The number of days to keep a configuration backup file before it is automatically deleted.
ECS/Fargate	`No`	Choose whether Ops Automator should use Fargate to run tasks.
Cluster VPC	`<blank>`	Optional - Existing VPC Id to use for Fargate.
Cluster Subnets	`<blank>`	Optional - Comma separated list of two existing VPC Subnet Ids where ECS instances will run. Required if setting VPC.
Cluster Availability Zones	`<blank>`	Optional - Comma-delimited list of VPC availability zones in which to create subnets. Required if setting VPC.

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 receive a status of CREATE_COMPLETE in approximately 15 minutes.

(Optional) Create an Amazon ECS image

If you chose the Amazon ECS/AWS Fargate option in Step 1.5 (modifying the parameters), you must create a Docker image and upload it to the ECS repository, `ops-automator`. This process assumes that you are familiar with Docker and have the Docker software installed locally. Refer to Appendix J for detailed instructions.

Step 2. Launch a task template in the primary account

Before you configure a task, review the information in Appendix A for the applicable action.

> **Note:** If you used the `ActionsConfiguration.html` file to launch the task, continue to Step 2.7. For more information on the file, refer to Role and task templates.

1. In the primary account's Amazon Simple Storage Service (Amazon S3) console, navigate to the bucket for the Ops Automator solution stack.

> **Note:** You can find the name of the S3 bucket in the AWS CloudFormation stack **Outputs** tab. The bucket name is the value of the **ConfigurationBucketName** key.

2. In the **TaskConfiguration** folder, select the applicable template.
3. Copy the **Link** value.
4. In the AWS CloudFormation console, select **Create Stack**.
5. Select **Specify an Amazon S3 template URL**.
6. Paste the template link into the text box and select **Next**.
7. Enter a **Stack name**.
8. Under **Parameters**, review the parameters for the template and modify them as necessary. For more information, refer to Appendix A.
9. Select **Next**.
10. Select **Next**. Then, 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.
11. Choose **Create** to deploy the stack.

> **Note:** If you delete the Ops Automator stack, all task stacks and configurations will be deleted.

Step 3. (Optional) Launch a role template in secondary account(s)

Use this procedure to create a role to perform tasks on resources in secondary accounts.

> **Note:** This role template is generated by the primary Ops Automator stack. The template will only set up a trust relationship between a secondary account and the primary account for which this template was generated. If you run multiple Ops Automator stacks in a single account, verify that you select the template from the Ops Automator stack you want to give the secondary account access to.

1. In the primary account's Amazon S3 console, navigate to the bucket for the Ops Automator solution stack.

   > **Note:** You can find the name of the S3 bucket in the AWS CloudFormation stack **Outputs** tab. The bucket name is the value of the **ConfigurationBucketName** key.

2. In the **AccountsConfiguration** folder, select the `AccountRoleConfiguration` template.
3. Select **Download** and note the location of the downloaded template.
4. In the secondary account's AWS CloudFormation console, select **Create Stack**.

   > **Important:** You must deploy the `AccountRoleConfiguration` template in the primary account if you want to perform tasks on resources in the primary account. You must also use the same value for the **Custom Rolename** parameter across all stacks. For more information, refer to Role configuration.

5. Select **Upload a template to Amazon S3**.
6. Select **Choose File**.
7. Navigate to the downloaded template and select **Choose**. Then, select **Next**.
8. Enter a **Stack name** and select **Next**.
9. To give the Ops Automator Lambda function in the primary account access to actions in the secondary account, set the applicable parameters to `Yes`. For example, to allow the solution to create backups in Amazon DynamoDB in the secondary account, set the **DynamoDB Create backup** parameter to `Yes`.
10. Optional: Enter a **Custom Rolename**. For more information, refer to Role Configuration.

11. Select **Next**. Then, 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.
12. Choose **Create** to deploy the stack.
13. After the stack deploys, navigate to the stack **Outputs** tab and copy the **Value** of the **CrossAccountRoleArn** key.

Step 4. (Optional) Launch the event forwarder template in secondary account(s)

Use this procedure to forward events from secondary accounts to the primary account. Launch this template in each applicable account and each applicable Region.

> **Important:** To use actions triggered by events across accounts and Regions, you must deploy the event forwarder AWS CloudFormation template (`AccountForwardEvents`) in each applicable account and Region, and you must deploy the account role configuration template (`AccountRoleConfiguration`) in each account.

1. In the primary account's Amazon S3 console, navigate to the bucket for the Ops Automator solution stack.

> **Note:** You can find the name of the S3 bucket in the AWS CloudFormation stack **Outputs** tab. The bucket name is the value of the **ConfigurationBucketName** key.

2. In the **Accounts Configuration** folder, select the `AccountForwardEvents` AWS CloudFormation template.
3. Copy the **Link** value.
4. In the secondary account, navigate to the AWS CloudFormation console and under **StackSets**, select **View stacksets**.
5. On the **StackSets** page, select **Create StackSet**.
6. Select **Specify an Amazon S3 template URL**.
7. Paste the template link into the text box and select **Next**.
8. Enter a **StackSet name**.

9. To forward events from this account to the primary account, set the applicable parameters to `Yes`. For example, to allow the solution to forward tag-change events for Amazon EC2, set the **EC2 Tag Change events** parameter to `Yes`.

10. Select **Next**.

11. Select **Next**. Then, 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.

12. Choose **Create** to deploy the stack.

Step 5. Tag your resources

When you deployed the AWS CloudFormation template, you defined the tag key for the solution's custom tag. For Ops Automator to recognize a resource, the tag key on that resource must match the custom tag name stored in the solution's Amazon DynamoDB table. Therefore, it is important that you apply tags consistently and correctly to all applicable resources. You can continue to use existing tagging strategies for your resources while using this solution.

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

Setting the tag value

As you apply a tag to a resource, use the tag key you defined during initial configuration and set the tag value to the name of an Ops Automator task stack to perform that task on the resource. For example, a user might define `OpsAutomatorTaskList` as the tag key. Then, the user creates a stack called `CopyResource`. To identify the resources to be copied, the user assigns the `OpsAutomatorTaskList` tag key with a value of `CopyResource` to each resource.

To perform multiple tasks on a single resource, use a comma-separated list of those tasks as the tag value. Continuing from the previous example, a user can assign the tag `OpsAutomatorTaskList` tag key with the value `CopyResource, DeleteResource` to identify resources to be copied, then deleted.

Security

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

IAM roles

AWS Identity and Access Management (IAM) roles enable customers to assign granular access policies and permissions to services and users on the AWS Cloud. Ops Automator creates an IAM role in the primary account that includes all the required permissions to perform actions on specific resources.

When you launch a role AWS CloudFormation template in a secondary account, the solution creates the applicable IAM role with least-privilege access in the secondary account. This IAM role in the secondary account has a trust policy with the IAM role in the primary account, allowing the primary account to access resources in the secondary account.

Solution-generated roles

Ops Automator automatically creates stacks and IAM roles that can grant additional permissions to the solution's main AWS Lambda function. To mitigate the risk of unauthorized access to the solution's main Lambda function and roles, AWS recommends that you deploy the solution in an isolated and tightly controlled management account, and limit access to that account.

To further isolate the solution, AWS recommends that you create a separate IAM role with the following permissions in the primary account:

```
        {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1499433973000",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:DeleteTopic",
                "sns:ListTopics",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "iam:CreateRole",
                "iam:UpdateRole",
                "iam:DeleteRole",
                "iam:AttachRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:GetRole",
                "iam:PassRole",
                "iam:ListRoles",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "dynamodb:CreateTable",
                "dynamodb:UpdateTable",
                "dynamodb:DeleteTable",
                "dynamodb:DescribeTable",
                "events:*",
                "logs:*",
                "lambda:*",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:DeleteBucket",
                "s3:PutObjectAcl",
                "s3:ListBucket",
                "s3:GetObject",
                "s3:GetLifecycleConfiguration",
                "s3:PutLifecycleConfiguration",
                "cloudformation:*",
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Use this role to launch the `ops-automator` AWS CloudFormation template. This restricts unauthorized access to the solution's AWS Lambda functions and roles.

For secondary accounts, use IAM roles to prevent non-administrators from creating, deleting, or updating tags. AWS recommends you check the access levels created by the solution templates and modify them, if necessary.

Additional resources

AWS services

- AWS Lambda
- Amazon DynamoDB
- Amazon EC2
- Amazon CloudWatch
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS CloudFormation

Appendix A: Task-configuration templates

Ops Automator automatically generates a separate AWS CloudFormation template for each action. Review the template parameters and modify them as necessary.

Common template parameters

Each action-specific template has a set of common parameters, and a set of parameters that are specific to the applicable action. Review the common parameters and modify them as necessary.

Parameter	Default	Description
Task description	<Optional input>	Description of the task. For example, `Create a snapshot every 30 minutes`.
Task interval	<Optional input>	For tasks that support intervals, enter the scheduled expression (cron syntax) that specifies when to run the task. For example, enter `0/30****` to run the task every 30 minutes. **Note:** Some tasks have a constraint on the minimum time between executions. When you create the task, the cron expression will be validated against that restraint. If the expression is not valid, the stack will not create and an error message will be logged.
Tag filter	<Optional input>	Filter used to select resources. You can use this instead of adding the task name to the list of values in the **OpsAutomatorTaskList** tag. For example, `Owner=DBAdmin, Stack=Test`. For more information on tag filters, refer to Appendix D. **Note:** For actions that can delete or terminate resources, you cannot use "*" as the name of the tag in the filter.
Timeout	`60`	The number of minutes the solution waits for a task to complete before reporting timing out. Specify a value of at least 1 minute. Ops Automator will continue to check for task completion until the timeout is reached, then emit a Timeout error if it has not completed. **Note:** This parameter will only show for actions that the solution checks for completion.
Regions	<Default region>	List of Regions where the task will run. For example, `us-east-1, eu-west-1`.

Parameter	Default	Description
		Note: Use this parameter for actions that use regional resources. This parameter will not show for actions that use global resources. For example, IAM and Amazon S3.
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. For more information, refer to Role configuration.
Accounts	<Optional input>	Comma-delimited list of account roles used by the task. **Note:** To allow cross-account operations, a cross-account role must be created. For more information, refer to Role configuration.
Cross account role name	<Optional input>	The name of a cross-account role that allows accounts to run this task. Leave this parameter blank to use the default role (*`<ops-automator-stackname>`*`-OpsAutomatorActionsRole`). For more information, refer to Role configuration.
Timezone	`UTC`	The time zone used for scheduling the task.
Task enabled	`Yes`	Select `No` to temporarily disable execution of the task.
Enable debugging	`No`	Choose whether to log detailed debugging information.
Resource selection scope for *<eventname>* event	`resource`	Choose the scope for selecting resources for tasks triggered by an event. Select `resource` or `region`.
Collect Metrics	`No`	Choose whether to collect CloudWatch metrics for the task.
Task Notifications	`No`	Choose whether to send notifications for started/ended tasks to an Amazon SNS topic.

Task memory allocation settings

Some actions include parameters that enable you to select larger AWS Lambda functions to run specific steps of a task. Review the memory allocation parameters and modify them as necessary.

Parameter	Default	Description
Resource selection memory	`Standard` or action-specific default	The memory size of the Lambda function that selects resources. Set this parameter to a higher value if you expect to select a large number of resources or you expect the selection of resources to take more than the 15-minute Lambda timeout or task interval. Choose `Standard`, `Medium`, `Large`, `XLarge`, `XXLarge`, or `XXXLarge`. If you selected Fargate when you installed Ops Automator then Amazon ECS will allow you to choose Fargate for this action.

Parameter	Default	Description
Selection reserved memory (if Resource selection memory = ECS)	`128`	Reserved memory (MB) to select resources for the task using an ECS task in a Fargate container. This value is only used if ECS is selected as the value for the **Resource selection memory** parameter. Specify a value between `4` and `8192` MB. Increase this value if resource selection is running out of memory.
Execution memory	`Standard` or action-specific default	The memory size of the Lambda function that runs the completion check. Set this parameter to a higher value if you expect the completion check to take more than the 15-minute Lambda timeout or task interval. Choose `Standard`, `Medium`, `Large`, `XLarge`, `XXLarge`, or `XXXLarge`.
Execution reserved memory (if Execution Memory = ECS)	`128`	Reserved memory (MB) for task execution using an ECS task in a Fargate container. This value is only used if ECS is selected as the value for the **Execution memory** parameter. Specify a value between `4` and `8192` MB. Increase this value if your task is running out of memory.

Amazon EBS snapshot events

The following parameters are for actions that can be triggered by Amazon Elastic Block Store (Amazon EBS) snapshot events.

Parameter	Default	Description
Snapshot copied	`No`	Choose whether to run the EBS snapshot targeted task when the snapshot is copied.
Snapshot for volume copied	`No`	Choose whether to run the EBS volume targeted task when the volume is copied.
Snapshot created	`No`	Choose whether to run the EBS snapshot targeted task when the snapshot is created.
Snapshot for volume created	`No`	Choose whether to run the EBS volume targeted task when the snapshot for a volume is created.
Snapshot shared account	`No`	Choose whether to run the EBS snapshot targeted task when a snapshot is shared with the task account.

Amazon EC2 state events

The following parameters are for actions that can be triggered by specific Amazon Elastic Compute Cloud (Amazon EC2) state events.

Parameter	Default	Description
Instance started	`No`	Choose whether to run a task when an Amazon EC2 instance is started.
Instance stopped	`No`	Choose whether to run a task when an Amazon EC2 instance is stopped.

Parameter	Default	Description
Instance terminated	`No`	Choose whether to run a task when an Amazon EC2 instance is terminated.

Amazon EC2 tag events

The following parameters are for actions that can be triggered by specific Amazon EC2 tag events.

Parameter	Default	Description
EC2 Tag Change Events	`No`	Choose whether to run a task when Amazon EC2 instance tags are modified.
EBS Snapshot Events	`No`	Choose whether to run a task when Amazon EBS snapshot tags are modified.

Appendix B: Available actions

Amazon EC2 create snapshot

The create snapshot action enables the solution to automatically create snapshots of Amazon Elastic Block Store (Amazon EBS) volumes attached to Amazon Elastic Compute Cloud (Amazon EC2) instances. The minimum interval between task executions is 60 minutes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
Snapshot Volume		
Volume tag filter	<Optional input>	Specify a tag filter expression that will be the basis for selecting which volumes to snapshot. Leave this parameter blank to select all root and/or data volumes. For more information about tagging your resources, refer to Step 5 and Appendix D.
Copy root volume	`Yes`	Create a snapshot of a root Amazon EC2 instance volume.
Copy data volumes	`Yes`	Create snapshots of Amazon EC2 instance data volumes.
Snapshot Naming and Description		
Snapshot name	`volumeid-yyyymmddhhmm`	The name of the snapshot. Leave blank to use the default snapshot name.
Snapshot name prefix	<Optional input>	Prefix of the snapshot name.

Parameter	Default	Description
Set snapshot name	`Yes`	Set the name of the snapshot.
Snapshot description	<Optional input>	The description of the snapshot. Leave this parameter blank to use the default description: `Snapshot created by` *`[task]`* `for` *`[root]`* (if it is a root volume) `volume` *`[volume-id]`* `(device` *`[device]`*`) of instance` *`[instance-id]`*.
Tagging		
Instance tags	<Optional input>	Enter tags to set on an instance after snapshots have been created. **Note:** If the task is triggered by setting tags, do not set tags that will cause the task to trigger again.
Snapshot tags	<Optional input>	Tags that will be added to snapshots. Use a list of `tagname=tagvalue` pairs. For example, if you create a task called `DeleteEC2Snapshots`, you can enter a value of `OpsAutomatorTaskList=DeleteEC2Snapshots` in this parameter to allow Ops Automator to delete the snapshot based on the parameters specified in the `DeleteEC2Snapshots` task.
Copied instance tags	<Optional input>	Enter a tag filter to copy tags from the instance to the snapshot. For example, enter `*` to copy all tags from the instance to the snapshot. For more information on tag filters, refer to Appendix D.
Copied volume tags	<Optional input>	Enter a tag filter to copy tags from the volume to the snapshot. For example, enter `*` to copy all tags from the volume to the snapshot. For more information on tag filters, refer to Appendix D.
Volume tags	<Optional input>	Tags that will be added to source Amazon EBS volumes after the snapshot has been created.
Snapshot Sharing		
Cross account role name for tagging of shared snapshots	<Optional input>	The name of a cross-account role in accounts that are sharing the snapshot. This role is used to create tags for shared snapshots in these accounts. Leave this parameter blank to use the default role (*`<ops-automator-stackname>`*`-OpsAutomatorActionsRole`). This role gives permission to use the `Ec2SetTags` action.
Create tags for shared snapshots	`No`	Choose whether to create tags for shared snapshots in applicable accounts.
Accounts with create-volume permissions	<Optional input>	List of accounts with permissions to create volumes from the snapshot.

The following table contains the event scope for triggering events.

Event	Scope
Instance started	`resource`
Instance stopped	`resource`
Tags changed for EC2 instance	`resource`

The following table contains task tag and naming placeholders.

Placeholder	Tags	Parameters	Description
device	`SnapshotTags`	**Snapshot description**	The device name of the volume on the instance.
instance-id	`SnapshotTags`	**Snapshot name** **Snapshot name prefix** **Snapshot description**	The ID of the instance.
snapshot-ids	`InstanceTags`		List of IDs of the snapshots created for the instance.
volume-id	`SnapshotTags`	**Snapshot name** **Snapshot name prefix** **Snapshot description**	The ID of the volume for which the snapshot was created.
snapshot-id	`VolumeTags`	`Yes`	The ID of the created snapshot.

Amazon EC2 delete snapshot

The delete snapshot action enables the solution to automatically delete snapshots of Amazon Elastic Block Store (Amazon EBS) volumes attached to Amazon Elastic Compute Cloud (Amazon EC2) instances older than a customer-defined number of days. Or, customers can configure this action to keep only the latest snapshots. The minimum interval between task executions is 15 minutes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
Retention days	*<Requires input>*	The retention period in days. Set this parameter to `0` to use **Retention count**.
Retention count	*<Requires input>*	The number of snapshots to retain for a volume. The maximum allowed value is `1000`. Set this parameter to `0` to use **Retention days**.

Parameter	Default	Description
		Note: If both **Retention days** and **Retention count** are set to `0`, the solution will return an error. You must only specify one.

The following table contains the event scope for triggering events.

Event	Scope
Snapshot for volume copied	`resource`
Snapshot for volume created	`resource`

Amazon EC2 copy snapshot

The copy snapshot template enables the solution to automatically copy snapshots of Amazon Elastic Block Store (Amazon EBS) volumes attached to Amazon Elastic Compute Cloud (Amazon EC2) instances between accounts and Regions. The minimum interval between task executions is 60 minutes. The maximum number of snapshots you can copy per account to a destination Region concurrently is five.

The diagram below describes the tagging parameters you can use when copying a shared snapshot.

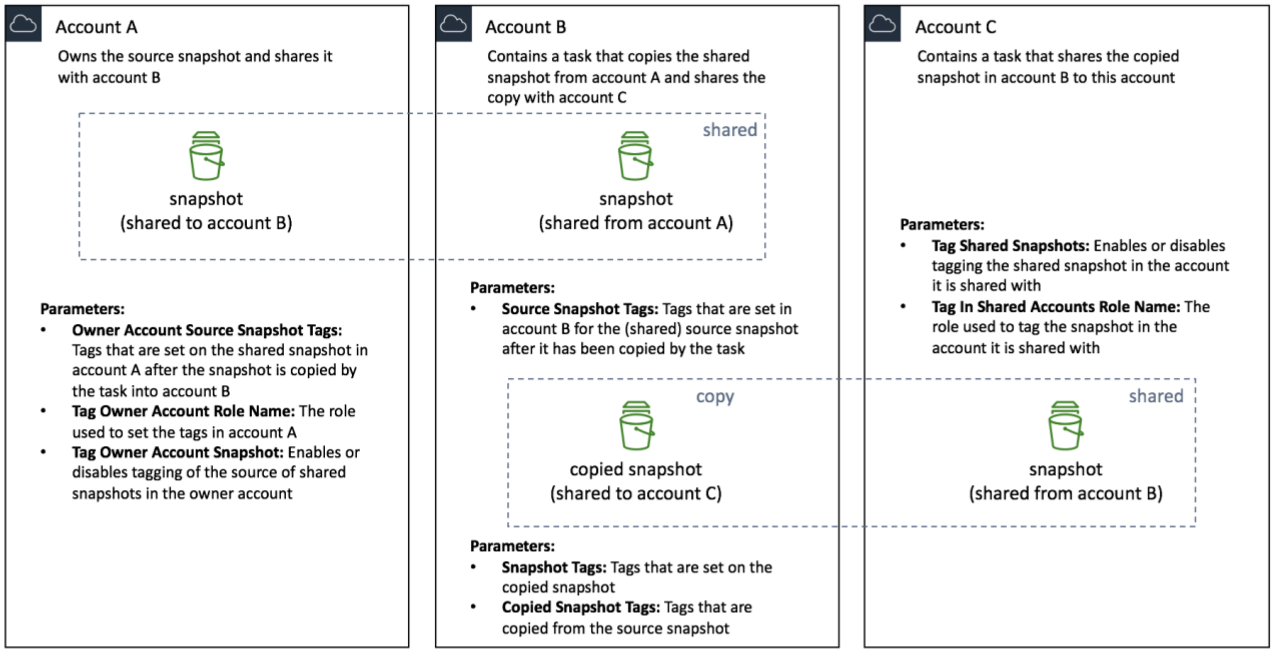
Account A
Owns the source snapshot and shares it with account B
snapshot
(shared to account B)
Parameters:
Owner Account Source Snapshot Tags: Tags that are set on the shared snapshot in account A after the snapshot is copied by the task into account B
Tag Owner Account Role Name: The role used to set the tags in account A
Tag Owner Account Snapshot: Enables or disables tagging of the source of shared snapshots in the owner account
Account B
Contains a task that copies the shared snapshot from account A and shares the copy with account C
shared
snapshot
(shared from account A)
Parameters:
Source Snapshot Tags: Tags that are set in account B for the (shared) source snapshot after it has been copied by the task
copy
copied snapshot
(shared to account C)
Parameters:
Snapshot Tags: Tags that are set on the copied snapshot
Copied Snapshot Tags: Tags that are copied from the source snapshot
Account C
Contains a task that shares the copied snapshot in account B to this account
Parameters:
Tag Shared Snapshots: Enables or disables tagging the shared snapshot in the account it is shared with
Tag In Shared Accounts Role Name: The role used to tag the snapshot in the account it is shared with
shared
snapshot
(shared from account B)

Figure 3: Tagging parameters when copying a shared snapshot

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
Snapshot Copy		
Copied shared snapshots owned by accounts	<Optional input>	Comma-delimited list of accounts to copy shared snapshots from. Leave this parameter blank to copy shared snapshots from all accounts.
Delete source snapshot	`No`	Choose whether to delete a source snapshot after it has been copied. Only owned source snapshots can be deleted. To delete a shared snapshot after it has been copied, use the **OwnerAccountSourceSnapshotTags** and **TagOwnerAccountRoleName** parameters to create tags for the shared snapshot in the source account. Then, configure an `Ec2RemoveSnapshot` task for the account that selects the snapshots to be deleted.
Snapshot copy description	<Optional input>	The description for the copied snapshot.
Destination region	*<Requires input>*	AWS Region to copy the snapshot to.
Snapshot types copied	<Optional input>	Choose which type of snapshots are copied: snapshots owned by the account, snapshots shared with the account, or both.
Snapshot Copy Tagging		
Copied tags from source snapshot	<Optional input>	Choose whether to copy tags from the source snapshot.
Snapshot tags	<Optional input>	Tags to add to the copied snapshot. Use a list of `tagname=tagvalue` pairs. **Note:** If snapshots are shared with other accounts, use the **Create tags for shared snapshots** parameter to create the tags in the accounts that share the snapshot.
Source Snapshot Tagging		
Source snapshot tags in accounts sharing the copied snapshot	<Optional input>	Tags to add to the copied snapshot in accounts that are sharing the snapshot.
Cross account role name for tagging source snapshots in accounts sharing the snapshot	<Optional input>	The name of a cross-account role in accounts that own a shared snapshot. This role is used to create tags for shared snapshots in these accounts. Leave this parameter blank to use the default role (*`<ops-automator-stackname>`*`-OpsAutomatorActionsRole`). This role gives permission to use the `Ec2SetTags` action.

Parameter	Default	Description
Source snapshot tags	<Optional input>	Tags to add to the copied snapshot in the account that runs the task for the shared snapshot.
Create tags for the shared source snapshot in the account that owns the shared snapshot	`No`	Choose whether to create tags for the copied snapshot in the account that owns the shared snapshot.
Snapshot Sharing		
Name of the account role for tagging in accounts with restore permissions	<Optional input>	The name of a cross-account role in accounts that share a snapshot. This role is used to create tags for shared snapshots in these accounts. Leave this parameter blank to use the default role (*`<ops-automator-stackname>`*`-OpsAutomatorActionsRole`) or the *`<ops-automator-stackname>`*`-NoneActionsRole`. This role must give permission to use the `Ec2SetTags` action.
Create tags for shared snapshots	`No`	Choose whether to create tags for copied, shared snapshots in the accounts that share the copied snapshots.
Accounts with create-volume permissions	<Optional input>	List of accounts that can create volumes from the copied snapshot.
Permissions and Encryption		
Encrypted	`No`	To enable encryption of the copied snapshot, select `Yes`.
KMS Key ID	<Optional input>	The full ARN of an AWS Key Management Service (AWS KMS) customer master key (CMK) to use when creating the snapshot copy. If you do not specify a CMK, the solution will use the default CMK for Amazon EBS. For instructions on how to find a key ARN, refer to Creating keys in the *AWS KMS Developer Guide*. **Note:** If you specify an alternative CMK, it must exist in the same AWS Region that the snapshots are copied to. Also, the account or role that the Ops Automator uses, or an applicable cross-account role, must have permission to use the key. For more information, refer to Appendix C.

The following table contains the event scope for triggering events.

Event	Scope
Snapshot created	`resource`

Event	Scope
Snapshot shared with account	`resource`
Tags changed for EC2 instance	`resource`

The following table contains task tag and naming placeholders.

Placeholder	Tags	Parameters	Description
destination-region	`SourceSnapshotTags`		The Region the snapshot is copied to.
copy-snapshot-id	`SourceSnapshotTags`		The ID of the copied snapshot.
owner-account	`SnapshotTags`	**Snapshot Description**	The account number of the account that owns the snapshot.
source-description		**Snapshot Description**	The description of the source snapshot.
source-region	`SnapshotTags`	**Snapshot Description**	The Region of the source snapshot.
source-snapshot-id	`SnapshotTags`	**Snapshot Description**	The ID of the source snapshot.
source-volume-id	`SnapshotTags`	**Snapshot Description**	The volume from which the source snapshot was taken.

Amazon EC2 replace instance

This action replaces your existing Amazon EC2 instance with a new instance of a different size. The solution automatically launches the new instance with the same Amazon Machine Image (AMI) and replaces the instance to the next defined size up or the next defined size down. The solution is integrated with Elastic Load Balancing to automatically register the new instance with the same load balancer or target group.

Note that all Amazon Elastic Block Store (Amazon EBS) volumes are created and defined in the AMI, but no data is copied from the replaced instance. Amazon EBS volumes on the original instance will be deleted, so any data on those volumes will be lost.

To replace instances with encrypted Amazon EBS volumes, you must add the `kms:CreateGrant` permission to the Ops Automator role. For more information, refer to Encrypted Amazon EBS Volumes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
Replacement Options		
Instance tags	<Optional input>	Tags to add to the new instance. Use a list of `tagname=tagvalue` pairs. **Important:** If tag events are used to start this task, do not specify tags that will trigger a new execution of this task.
Copied instance tags	*	The tag filter expression to copy tags from the original instance to the replacement instance. By default, all tags are copied except for the tags that are part of the scaling tag filter expressions. **Important:** If tag events are used to start this task, do not specify tags that will trigger a new execution of this task.
Replacement mode	`ReplaceByType`	Choose how you want to replace your instances. Choose, `ReplaceByType` or `ReplaceByStep`. For more information, refer to Replacing with a New Instance of a Different Size.
Replace by Type		
Replace if same type	`False`	Replace the instance even if the new instance type is the same are the current type.
New instance type(s)	*<Requires input>*	Specify an instance type or types for the new, scaled instance. We recommend specifying a list of types to provide alternatives in case an instance type is not available. **Note:** The list must be sorted from smallest to largest.
Replace by Step		
Assumed Type	*<Requires input>*	The default instance type that will be used if the instance type of an existing instance is not within the range defined in the **Instance Types** parameter. **Note:** You must specify an instance type that is specified in the **Instance Types** parameter.
Scale up tag filter	<Optional input>	The tag filter expression that determines when an instance is scaled up.
Scaling range	*<Requires input>*	A comma-delimited list of valid, compatible instance types. **Note:** The list must be sorted from smallest to largest and must contain at least two instance types.
Scale down tag filter	<Optional input>	The tag filter expression that determines when an instance is scaled down.

Parameter	Default	Description
Try next in range	`True`	Choose whether to try the next instance type in the range if an instance type is not available. If this parameter is set to `False`, the instance will not be scaled if the next type is not available.

The following table contains the event scope for triggering events.

Event	Scope
EC2 tag change events	`resource`

The following table contains task tag and naming placeholders.

Placeholder	Tags	Parameters	Description
new-instance-type	`NewInstanceTags`		The type of the new instance.
org-instance-id			The ID of the original instance.
org-instance-type	`NewInstanceTags`		The type of the replaced instance.

Amazon EC2 resize instance

This action stops your existing instance, resizes the instance to the next defined size up or the next defined size down, then starts the instance again.

During the resizing, Amazon Elastic Block Store (Amazon EBS) volumes on the instance will remain attached and the data will persist. However, any data on the ephemeral storage of your instance will be lost. To keep your data, it must be stored on an attached Amazon EBS volume.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
Instance Options		
Resizing mode	`ReplacebyType`	Choose how you want to replace your instances. Choose, `ReplaceByType` or `ReplaceByStep`. For more information, refer to Resizing an Existing Instance.
Instance tags	*	Tags to add to the new instance. Use a list of `tagname=tagvalue` pairs.

Parameter	Default	Description
		Important: If tag events are used to start this task, do not specify tags that will trigger a new execution of this task.
Replace by Type		
New instance type(s)	*<Requires input>*	Specify an instance type or types for the new, scaled instance. We recommend specifying a list of types to provide alternatives in case an instance type is not available. **Note:** The list must be sorted from smallest to largest.
Replace by Step		
Assumed Type	*<Requires input>*	The default instance type that will be used if the instance type of an existing instance is not within the range defined in the **Instance Types** parameter. **Note:** You must specify an instance type that is specified in the **Instance Types** parameter.
Scale up tag filter	<Optional input>	The tag filter expression that determines when an instance is scaled up.
Scaling range	*<Requires input>*	A comma-delimited list of valid, compatible instance types. **Note:** The list must be sorted from smallest to largest and must contain at least two instance types.
Scale down tag filter	<Optional input>	The tag filter expression that determines when an instance is scaled down.
Try next in range	`True`	Choose whether to try the next instance type in the range if an instance type is not available. If this parameter is set to `false`, the instance will not be scaled if the next type is not available.

The following table contains the event scope for triggering events.

Event	Scope
EC2 tag change events	`resource`

The following table contains task tag and naming placeholders.

Placeholder	Tags	Parameters	Description
new-instance-type	`NewInstanceTags`		The new type of the resized instance.

Placeholder	Tags	Parameters	Description
org-instance-type	`ResizedInstanceTags`		The original type of the resized instance.

Amazon EC2 tag instance by CPU utilization

This action checks the CPU utilization metrics of your Amazon EC2 instances against thresholds you define. When CPU utilization is above or below the specified thresholds, the instance is tagged. The tagging can be used to trigger the actions that vertically scale your instances.

The **Task Interval** AWS CloudFormation parameter defines the interval at which the solution checks CPU utilization. The minimum interval is five minutes and the maximum interval is 24 hours (1440 minutes).

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
CPU Utilization Thresholds and Tagging Parameters		
CPU threshold high	`80`	The high threshold for average CPU utilization that will determine whether an instance is tagged.
CPU high tags	*<Requires input>*	Tags to add to the instances with average CPU utilization above the high threshold. Use a list of `tagname=tagvalue` pairs.
CPU threshold low	`10`	The low threshold for average CPU utilization that will determine whether an instance is tagged.
CPU low tags	*<Requires input>*	Tags to add to the instances with average CPU utilization below the low threshold. Use a list of `tagname=tagvalue` pairs.
Replace by Type		
New instance type(s)	*<Requires input>*	Specify an instance type or types for the new, scaled instance. We recommend specifying a list of types to provide alternatives in case an instance type is not available. **Note:** The list must be sorted from smallest to largest.
Replace by Step		
Assumed Type	*<Requires input>*	The default instance type that will be used if the instance type of an existing instance is not within the range defined in the **Instance Types** parameter.

Parameter	Default	Description
		Note: You must specify an instance type that is specified in the **Instance Types** parameter.
Scale up tag filter	<Optional input>	The tag filter expression that determines when an instance is scaled up.
Scaling range	*<Requires input>*	A comma-delimited list of valid, compatible instance types. **Note:** The list must be sorted from smallest to largest and must contain at least two instance types.
Scale down tag filter	<Optional input>	The tag filter expression that determines when an instance is scaled down.
Try next in range	`True`	Choose whether to try the next instance type in the range if an instance type is not available. If this parameter is set to `false`, the instance will not be scaled if the next type is not available.

Amazon DynamoDB set capacity template

The Amazon DynamoDB set capacity template enables the solution to automatically provision throughput capacity for reads and writes to DynamoDB. The minimum interval between task executions is 15 minutes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Default	Description
Table name	*<Requires input>*	The name of the Amazon DynamoDB table.
Table read units	*<Requires input>*	The provisioned number of read units for the table.
Table write units	*<Requires input>*	The provisioned number of write units for the table.
Global Secondary Index 1		
Table read units	`1`	The provisioned number of read units for the table.
Table write units	`1`	The provisioned number of write units for the table.
Index name	`None`	The name of the secondary global index. Leave this parameter blank or set the parameter to `None` if it is not used.
Global Secondary Index 2		
Table read units	`1`	The provisioned number of read units for the table.
Table write units	`1`	The provisioned number of write units for the table.
Index name	`None`	The name of the secondary global index. Leave this parameter blank or set the parameter to `None` if it is not used.

Parameter	Default	Description
Global Secondary Index 3		
Table read units	`1`	The provisioned number of read units for the table.
Table write units	`1`	The provisioned number of write units for the table.
Index name	`None`	The name of the secondary global index. Leave this parameter blank or set the parameter to `None` if it is not used.
Global Secondary Index 4		
Table read units	`1`	The provisioned number of read units for the table.
Table write units	`1`	The provisioned number of write units for the table.
Index name	`None`	The name of the secondary global index. Leave this parameter blank or set the parameter to `None` if it is not used.
Global Secondary Index 5		
Table read units	`1`	The provisioned number of read units for the table.
Table write units	`1`	The provisioned number of write units for the table.
Index name	`None`	The name of the secondary global index. Leave this parameter blank or set the parameter to `None` if it is not used.

For more information on throughput capacity, refer to Provisioned Throughput in the *Amazon DynamoDB Developer Guide*.

Appendix C: Configuring keys for encrypting snapshots

Ops Automator allows you to use AWS Key Management Service (AWS KMS) to encrypt the snapshot copies this solution creates. The solution is designed to use the default customer master key (CMK) for Amazon Elastic Block Store for copying snapshots, but you can specify an alternate CMK. If you specify an alternate CMK, it must exist in the same AWS Region as the copied snapshot. The account or role that Ops Automator uses, or applicable cross-account role, must have permission to use the role to encrypt copied snapshots.

Encrypting snapshots in the primary account

Use this procedure to configure your CMK to encrypt snapshots in the primary account where the Ops Automator stack is deployed.

1. Sign into the AWS Management Console and open the AWS Identity and Access Management (IAM) console.
2. In the left navigation pane, select **Encryption keys**.

3. Choose the destination AWS Region for your copied snapshots.
4. Verify that the desired key exists in the Region. If the key does not exist in that Region, create a key.
5. Add a key policy to the key to allow the Ops Automator IAM role to use the key to encrypt snapshots. The name of the role is `<stackname>-OpsAutomatorLambdaRole-<id>`.

Encrypting snapshots in a secondary account(s)

Use this procedure to configure your CMK to encrypt snapshots in secondary account(s).

1. Sign into the AWS Management Console and open the AWS Identity and Access Management (IAM) console.
2. In the left navigation pane, select **Encryption keys**.
3. Choose the destination AWS Region for your copied snapshots.
4. Verify that the desired key exists in the Region. If the key does not exist in that Region, create a key.
5. Add a key policy to the key to allow the copy snapshot cross-account role in the secondary account to use the key to encrypt snapshots.
6. In each secondary account, add the permission to use the key to the cross-account role that allows the primary account to run the copy snapshot task in the secondary account.
   a. Navigate to the IAM console.
   b. Choose the cross-account role that allows the primary account to run the copy snapshot task in the secondary account. For more information, refer to Step 3.
   c. Edit the cross-account role's policy. Specify the key's Amazon Resource Name and the permitted actions. For more information, refer to Sharing Custom Encryption Keys More Securely Between Accounts by Using AWS Key Management Service.

Appendix D: Resource selection

Ops Automator uses a custom tag key (name) to identify resources that will receive automated actions. The default tag key is `OpsAutomatorTaskList`, but you can modify it during initial configuration of the solution's AWS CloudFormation template. The value of the `OpsAutomatorTaskList` tag contains a comma-separated list of tasks you want the solution to perform on that resource. For more information, refer to Step 5.

Or, you can use a tag filter expression to specify the tasks you want to perform on resources. The expression overwrites the selection of resources based on the values in the `OpsAutomatorTaskList` tag key. A tag filter expression enables a more granular selection of resources. They have the format `<tag-name>[=<tag-value>]`. The tag names and values can contain wildcards, regular expressions, and Boolean operators. For example, a tag filter with a value of `*` will select all resources. To use a regular expression, the tag name or value must start with `\`.

Tag filter format	Description
`*`	Any tagged resource. **Note:** For actions that can delete or terminate resources, you cannot use "*" as the name of the tag in the filter.
`A*`	Tag keys that start with "A".
`*A*`	Tag keys that contain "A".
`*A`	Tag keys that end with "A".
`\.*\d$`	Tag keys that end with a digit.
`A=B`	Tag keys "A" with value "B".
`A=B*`	Tag keys "A" with a value that starts with "B".
`*=B`	Any tag with a value "B".
`*=B*`	Any tag with a value that starts with "B".
`*=\.*\d$`	Any tag with a value that ends with a digit.
`A=B,C=D`	Tag keys "A" with value "B" or tag keys "C" with a value "D".

A tag filter expression can also contain multiple filters combined using logical operators. Use `&` for `AND`, `|` for `OR`, or `!` for `NOT`. You can also group filters using parentheses.

Expression	Description
`A=*&B=*`	Tags "A" and "B" with any value.
`A=*|B=*`	Tag "A" or "B" with any value.
`((A=*&B=*)|C=*)`	Tags "A" and "B", or tag "C" with any value.
`!A=*`	Not tag "A".
`A=!B`	Tag "A" not having value "B".
`(!A=*)&(C=!D)`	Not tag "A" and tag "C" not having value "D".

The following table gives examples of different tag filters and the resulting Ops Automator action.

Tag filter	Ops Automator Action
`Owner=DBAdmin`	Perform the task on resources with the `Owner` tag key with a value of `DBAdmin`.
`Owner`	Perform the task on resources with the `Owner` tag key with any value.
`*=DBAdmin`	Perform the task on resources with any tag key with a value of `DBAdmin`.
`*test`	Perform the task on resources with a tag key that ends with `test`.
`test*`	Perform the task on resources with a tag key that starts with `test`.
`*=*test`	Perform the task on resources with any tag key with a value that ends with `test`.
`*=test*`	Perform the task on resources with any tag key with a value that starts with `test`.
`(Owner=DBAdmin)&(Stack=Test)`	Perform the task on resources with the `Owner` tag key with a value of `DBAdmin` and resources with the `Stack` tag key with a value of `Test`.
`(Owner=DBAdmin)|(Stack=Test)`	Perform the task on resources with the `Owner` tag key with a value of `DBAdmin` or resources with the `Stack` tag key with a value of `Test`.
`(Owner=DBAdmin)&(Stack=!Test)`	Perform the task on resources with the `Owner` tag key with a value of `DBAdmin` and resources with the `Stack` tag key with a value that is not `Test`.

Appendix E: Tag and parameter placeholders

Placeholders can be used in parameters such as names, prefixes, and descriptions, and tag names and values set by Ops Automator actions. Placeholders have the format `{`*`name`*`}`. When tasks are implemented, placeholders are replaced with dynamic values. This allows the solution to give created resources dynamic names and descriptions, and to set dynamic tag names and values on created or affected resources.

The following table contains a list of common placeholders.

Placeholder	Description
`{account}`	The account in which the task is run.
`{stack}`	The name of the Ops Automator stack.
`{date}`	The date in YYYYMMDD format.
`{datetime}`	The date in YYYYMMDDHHMMSS format.
`{day}`	The day in DD format.
`{hour}`	The hour in HH format.
`{iso-date}`	The date in ISO format.

Placeholder	Description
`{iso-datetime}`	The date and time in ISO format.
`{iso-time}`	The time in ISO format.
`{iso-weekday}`	The weekday in ISO format.
`{minute}`	The minute in MM format.
`{month}`	The month in MM format.
`{monthname}`	The abbreviated name of the month. For example, Jan.
`{monthname-long}`	The full name of the month. For example, January.
`{region}`	The Region in which the task is run.
`{second}`	The seconds in SS format.
`{task}`	The name of the implemented task.
`{task-id}`	The unique ID of the implemented task.
`{task-group}`	The unique ID of the task group. A task group is a collection of tasks that are the result of a task started by the scheduler. One task can have multiple task executions depending on the aggregation level of the task. Each task will have its own task ID, but the tasks will share a group ID.
`{time}`	The time in HHMMSS format.
`{timezone}`	The time zone configured for the task.
`{weekday}`	The abbreviated name of the weekday. For example, Mon.
`{weekday-long}`	The full name of the weekday. For example, Monday.
`{year}`	The year in YYYY format.

Ops Automator actions can have action-specific placeholders for tags.

The following table contains placeholders that are specific to the `Ec2ReplaceInstance` and `Ec2ResizeInstance` actions.

Placeholder	Description
`{instance-id}`	The instance ID of the volume from which the snapshot was taken.
`{volume-id}`	The volume ID of the volume from which the snapshot was taken.
`{device}`	The device of the volume from which the snapshot was taken.

Placeholder	Description
`{snapshot-ids}`	The snapshot IDs for snapshots taken for all volumes of an instance.
`{snapshot-id}`	The snapshot ID for a snapshot taken from a volume.

The following table contains a list of special placeholders.

Placeholder	Description
`{task-tag}`	Expands to the value of the tag used to specify the list of tasks for the Ops Automator stack. You can use this placeholder to set follow-up actions for resources affected or created by a task. For example, `{task-tag}=`*`NameoftheFollowUpAction`*
`{delete}`	Use this placeholder for tags in tag parameters. Setting a tag to this value will delete the tag. For example, *`TagName`*`={delete}` will delete the *`TagName`* tag.
`{ssm:name}`	Expands to the value of a parameter in the AWS Systems Manager Parameter Store for the account with the Ops Automator stack. Parameters of the types string and string list are supported. This placeholder allows you to share parameters between tasks and manage then centrally.

Appendix F: Logging

Ops Automator creates a log group `(`*`<ops-automator-stackname>`*`-logs`). The solution logs information for every step of a task that is run, which may result in many log streams. The log streams this solution creates have structured names to help make it easier to find logs for a specific task or component of the framework.

The solution creates the following general log streams:

- `SetupHelperHandler-YYYYMMDD` – Logs the output of the setup process when you deploy or update the Ops Automator stack. Check this stream if deploying or updating the stack fails.
- `OpsAutomatorMain-YYYMMMDD` – Logs errors that are handled in specific modules and logged in their log streams. This stream is only created if errors occur.

- `ScheduleHandler-YYYYMMDD` – Logs the output from the Lambda function that schedules configured tasks. This stream contains information that shows which tasks are run and the next time the task will run. Check this stream if a task does not run at the expected time.
- `CompletionHandler-YYYYMMDD` – Logs the output from the Lambda function that runs the completion logic. The output of the completion logic is logged in the stream for the specific task.
- `TaskConfigAdminAPI-YYYYMMDD` – Logs the output from configuration steps performed through the Configuration Admin API.
- `ConfigurationResourcesHandler-YYYYMMDD` – Logs the output from the custom resource handler that creates tasks. Check this log when creating a task stack fails because it includes all parameter validation errors.
- `TaskTrackingHandler-YYYYMMDD` – Logs detailed information from core task processing. This steam is created if you enable detailed debugging.
- `Ec2StateEventHandler-<account>-<region>-YYYYMMDD` – Logs information about processing Amazon EC2 state events for a specific account and/or Region.
- `Ec2TagEventHandler-<account>-<region>-YYYYMMDD` – Logs information about processing Amazon EC2 tag modification events for a specific account and/or Region.
- `EbsSnapshotEventHandler-<account>-<region>-YYYYMMDD` – Logs information about processing Amazon EBS snapshot events for a specific account and/or Region.

The solution also creates the following task-specific log streams:

- `SelectResourcesHandler-<taskname>-<account>-<region>-YYYYMMDD`
- `SelectResourcesHandler-<taskname>-<account>-YYYYMMDD`
- `SelectResourcesHandler-<taskname>-YYYYMMDD`

These log streams log the output from the AWS Lambda function that selects the resources for a task. Check these logs if the expected resources are not selected for a task. When detailed logging is enabled, these streams contain detailed information about found resources and the filtering used to include or exclude the resources from the action execution. These streams are named based on the aggregation level of the resources.

- `<taskname>-<account>-<region>-<resource>-<task-id>-YYYYMMDD`
- `<taskname>-<account>-<region>-<task-id>-YYYYMMDD`
- `<taskname>-<account>-<task-id>-YYYYMMDD`

- `<taskname>-<task-id>-YYYYMMDD`

These log streams log the output from the execution and, if enabled, completion logic of the action. Check these logs if an action fails to run or complete. These streams are named based on the aggregation level of the resources. For example, if the task is for a specific resource, the log stream name will include the name of the specific resource. If the task is for a specific Region, the stream name will include the Region name.

The solution also creates the following Lambda service log groups:

- `/aws/lambda/<stackname>-Standard`
- `/aws/lambda/<stackname>-Medium`
- `/aws/lambda/<stackname>-Large`
- `/aws/lambda/<stackname>-XLarge`
- `/aws/lambda/<stackname>-XXLarge`
- `/aws/lambda/<stackname>-XXXLarge`

These log groups contain the default log streams for the Lambda functions. These streams do not contain any Ops Automator-specific information.

Messages

This solution also logs error, warning, and debugging messages. Each message has the format: `yyyy-mm-dd - hh:mm:ss.mmm - <category> - <text>`. The category can have the value INFO for informational messages, DEBUG for detailed debugging messages, WARNING for warning messages, or ERROR for error messages.

Issues SNS topic

Ops Automator creates an Amazon Simple Notification Service (Amazon SNS) topic where all warnings and errors are posted. Messages include the error, and the name of the log group and the log stream where the error is logged. The name of the Amazon SNS topic is located in the Ops Automator stack output named **IssueSNSTopic**.

The messages posted to the topic contain the following attributes:

Attribute	Description
`log-group`	The name of the log group of the stream where the message was logged.
`log-stream`	The name of the log stream where the message was logged.
`category`	`ERROR` or `WARNING`.
`message`	The error or warning message.

Notifications SNS topic

If enabled, Ops Automator creates an Amazon SNS topic where notifications for every started or completed task are posted. Use the **Task Notifications** parameter to enable this feature. The name of the Amazon SNS topic is located in the Ops Automator stack output named **NotificationsSNSTopic**.

The messages posted to the topic contain the following attributes:

Attribute	Description
`Id`	The task ID.
`TaskName`	The name of the task.
`Account`	The account in which the action is run.
`Resources`	The task resources.
`Parameters`	The parameters of the task.
`Time`	The date and time in ISO format.
`Type`	Can be either `task-started` or `task-ended`.
`Status`	The status of the implemented task: `completed`, `timed-out`, or `failed`. Included when the `Type` is `task-ended`.
`ActionResult`	The result output of a completed task. Included when the `Status` is `completed`.
`Error`	The error reported by a failed task. Included when the `Status` is `failed`.

Out-of-band logging

Ops Automator uses Amazon CloudWatch logs to log all output. To help deal with high volume and concurrency, the solution includes a layer on top of CloudWatch logging that enables the solution to log out-of-band information. Logged messages are buffered and written to an Amazon Simple Queue Service (Amazon SQS) queue. A Lambda function (`<stackname>-OpsAutomatorCloudWatchQueueHandler`) processes the queued messages and writes them to the log streams. Because of the buffering and queuing, there might be a delay between when messages are generated and when they are written to the log stream.

Appendix G: Sample deployment configuration

Ops Automator enables customers to perform a sequence of tasks on resources in their accounts. The following section shows how to configure a sequence of tasks that will take snapshots of all Amazon Elastic Block Store (Amazon EBS) volumes in the primary account at 1:00 AM daily and retain the last seven snapshots. This example shows how to combine an interval-based task (creating the snapshots) with an event-based task (deleting the snapshots).

First, deploy the `ops-automator` template in your primary account. For more information, refer to Step 1.

Next, deploy the `Ec2DeleteSnapshot` *task* template in the primary account. For more information, refer to Step 2. Use the following values:

Parameter	Value
Stack name	`Delete7`
Task description	`Retain the last 7 snapshots`
Task interval	(Leave blank)
Tag filter	(Leave blank)
Regions	(Enter the applicable AWS Region(s). For example, `us-east-1, eu-west-1`)
This account	`Yes`
Accounts	(Leave blank)
Cross-account role name	(Leave blank)
Timezone	`UTC`
Task enabled	`Yes`
Collect metrics	`Yes` (optional)
Enable debugging	`No`
Snapshot for volume copied	`No`
Snapshot for volume created	`Yes`
Retention days	`0` **Note:** Set this parameter to `0` to retain snapshots using **Retention count**.
Retention count	`7`

Then, deploy the `Ec2CreateSnapshot` *task* template in the primary account using the following values:

Parameter	Value
Stack name	`BackupDaily`

Parameter	Value
Task description	`Create a snapshot at 1 am daily`
Task interval	`0 1 * * ?`
Tag filter	(Leave blank)
Regions	(Enter the applicable AWS Region(s). For example, `us-east-1, us-west-2`)
This account	`Yes`
Accounts	(Leave blank)
Cross-account role name	(Leave blank)
Timezone	`UTC`
Task enabled	`Yes`
Enable debugging	`No`
Instance stopped	`No`
Instance started	`No`
Resource selection memory	`Medium`
Copy root volume	`Yes`
Copy data volumes	`Yes`
Copied instance tags	`*`
Copied volume tags	`*`
Volume tag filter	(Leave blank)
Set snapshot name	`Yes`
Snapshot name prefix	`auto-`
Snapshot name	(Leave blank)
Volume tags	`LastSnapshot={snapshot-id}`
Instance tags	`LastSnapshots={snapshot-ids}`
Snapshot tags	`OpsAutomatorTaskList=Delete7`

When deployed using the configuration above, Ops Automator does the following:

1. Create a snapshot of any EBS volumes attached to Amazon Elastic Cloud Compute (Amazon EC2) instances in the primary account with the `BackupDaily` tag at 1:00 AM. If a snapshot already exists for the volume, the solution takes an incremental snapshot.
2. Copy the Amazon EC2 instance and volume tags to the snapshot.

3. Attach a new tag (`OpsAutomatorTaskList=Delete7`) to the snapshot. This tag identifies applicable snapshots of a specific volume for deletion after the retention count (seven snapshots).
4. After the creation of a snapshot, the solution checks the number of snapshots for the volume and retains the last seven snapshots.

Appendix H: Creating tasks with custom resources

Ops Automator features a custom resource you can use to create templates that include multiple tasks to help implement full end-to-end scenarios. The solution includes example templates in the **TaskConfiguration/ScenarioTemplates** folder.

First, create the custom resource.

1. In the resource section, create a custom resource with a type of `Custom::TaskConfig`.
2. Set the service token to the Amazon Resource Name (ARN) of the Ops Automator standard AWS Lambda function:
   `arn:aws:lambda:${AWS::Region}:${AWS::AccountId}:function<stackname>-OpsAutomatorStandard`.

Then, generate an AWS CloudFormation resource snippet.

1. In the **TaskConfiguration/ScenarioTemplates** folder, use an action template to create a task.
2. After the task has been created, download the `BuildTaskCustomResource.py` script from the **TaskConfiguration/Scripts** folder.

   Note: You must have Python and boto3 installed, and you must have a profile that is allowed to read from the Ops Automator configuration table.

3. At a command prompt, run the following command:

```
python BuildTaskCustomResource.py <task name> <optional profile
name>
```

An AWS CloudFormation snippet for the custom resource is created and written to the standard output. You can modify the snippet and include it in your custom templates. Note that the snippet gives the same name as the name of the task that was used to generate it. You must change the name of the snippet or delete the task stack you used to create the snippet to allow the custom resource to successfully create tasks.

We strongly recommend that you use the AWS CloudFormation template and user interface to set the required properties for a task. Then, use the included helper script to create a custom snippet to use in your custom templates.

The following table contains a list of properties you can use in custom resources.

Property	Type and example value	Description
Accounts	List of strings `“777788889999”, “000000000000”`	A list of account roles used by the task.
Action	String	The name of the action.
Debug	Boolean `True | False`	Choose whether to enable detailed logging.
Description	String	The task description.
Enabled	Boolean `True | False`	Choose whether to enable the task.

Event scope	Dictionary with event structure `source/detail/event`.
	Scope values are `region | resource`.
	The dictionary below shows the structure with all supported events use scopes. Currently, `Ec2SetTags` supports this parameter.
	
	`source: "aws.ec2"`
	`detail: "EC2 Instance State-change Notification"`
	`events: "started", "stopped", "terminated"`
	
	For example:
	
	`{`
	`  "aws.ec2": {`
	`          "EC2 Instance State-change Notification": {`
	`              "started": "region",`
	`               "stopped": "region",`
	`               "terminated": "region`
	`          }`
	`     }`
	`}`
		Scope for selecting resources for tasks triggered by an event. The default value is `resource`. This property is only stored in the configuration database, if it is set to `region`.

Source event tag filter	String	If **Event scope** is set to `region`, this property is used to filter the

Property	Type and example value	Description
		events that trigger the action using the tags on the resource that is the source of the event. Review Appendix A to determine which actions support this property.

Events	Dictionary containing the events structure `source/detail/list of events`
	
	`source: "aws.ec2"`
	` detail: "EBS Snapshot Notification"`
	`  events: "copySnapshot","createSnapshot",`
	`                         "shareSnapshot"`
	`   detail:  "EC2 Instance State-change`
	`Notification`
	`          events: "terminated",`
	`"running","stopped"`
	
	`source: "ec2.aws.tag"`
	` detail: "TagChangeOnResource"`
	`  events: "ChangedInstanceTags",`
	`                        "ChangedSnapshotTags"`
	
	For example:
	
	`{`
	`    "aws.ec2": {`
	`         "EC2 Instance State-`
	`change Notification":`
	`              [ "running",`
	`"stopped"]`
	`     }`
	`}`
		The events that trigger the task.

Interval	String	The scheduled expression (cron syntax) that specifies when to run the task.
Parameters	Dictionary	Action-specific parameters. The dictionary contains the parameters as `parameter-name:parameter-value` pairs.

Property	Type and example value	Description
Regions	List of strings `“eu-west-1”, “eu-central-1”`	List of Regions where the task will run.
Tag filter	String	Expression used to select resources for the action.
Completion size	String `Standard | Medium | Large | XLarge | XXLarge | XXXLarge`	The size of the Lambda function used for completion checking for the action.
Cross-account role name	String	Custom role name used for cross-account action execution.
Execute size	String `Standard | Medium | Large | XLarge | XXLarge | XXXLarge`	The size of the Lambda function used for action execution.
Task metrics	Boolean `True | False`	Choose whether to collect CloudWatch metrics for the task.
Name	String	Unique name of the task.
Task notifications	Boolean `True | False`	Choose whether to send notifications for started/ended task executions to an Amazon SNS topic.
Select size	String `Standard | Medium | Large | XLarge | XXLarge | XXXLarge`	The size of the Lambda function used for resource selection.
Task timeout	String `60`	The time, in minutes, the solution waits for a task to complete before reporting timing out.
This account	Boolean `True | False`	Choose whether to run the task on resources in this account.
Timezone	String	The time zone used for scheduling tasks.

You can use the following helper script to create a custom snippet for your custom templates.

```
{
   "Properties": {
      "Accounts": [
         "000000000000","111111111111"
      ],
      "Action": "Ec2CreateSnapshot",
      "CompletionSize": "Medium",
      "Debug": "True",
      "Enabled": "True",
      "Events": {},
      "ExecuteSize": "Medium",
      "Interval": "0 ) * ** ?",
      "Name": " create-snapshot",
      "Parameters": {
         "BackupDataVolumes": "True",
         "BackupRootVolumes": "True",
         "CreateVolumePermission": [
            "222222222222"
         ],
         "InstanceTags": "last-snapshots={snapshot-ids}",
         "SetSnapshotName": "True",
         "SnapshotDescription": "Snapshot for volume {volume-id}",
         "SnapshotName": "{instance-id}-{volume-id}-{datetime}",
         "TagSharedSnapshots": "True",
         "VolumeTagFilter": null,
         "VolumeTags": "last-snapshot={snapshot-id}"
      },
      "Regions": [
         "eu-west-1"
      ],
      "SelectSize": "Medium",
      "ServiceToken": "arn:aws:lambda:eu-west-1:ACCOUNT-
NR:function:STACKNAME-OpsAutomator-Standard",
      "TaskMetrics": "True",
      "TaskNotifications": "False",
      "TaskTimeout": "30",
      "ThisAccount": "False",
      "Timezone": "UTC"
   },
   "Type": "Custom::TaskConfig"
}
```

The following custom resource snippet creates a task is triggered by a create-snapshot event. The task deletes snapshots and retains the last three snapshots.

```
  {
     "Properties": {
        "Accounts": [
           "000000000000"
        ],
        "Action": "Ec2DeleteSnapshot",
        "CompletionSize": "Medium",
        "Debug": "True",
        "Description": "Deletes snapshots, keeps latest 3",
        "Enabled": "True",
        "Events": {
           "aws.ec2": {
              "EBS Snapshot Notification": [
                 "createSnapShotForVolume"
              ]
           }
        },
        "ExecuteSize": "Medium",
        "Name": "delete-snapshot-3 ",
        "Parameters": {
           "RetentionCount": "3",
           "RetentionDays": "0"
        },
        "Regions": [
           "eu-west-1"
        ],
        "SelectSize": "Medium",
        "ServiceToken": "arn:aws:lambda:eu-west-
1:ACCOUNT_NUM:function:STACK-OpsAutomator-Standard",
        "TaskMetrics": "True",
        "TaskNotifications": "False",
        "ThisAccount": "False"
     },
     "Type": "Custom::TaskConfig"
  }
```

Appendix I: Ops Automator scenarios

Ops Automator includes templates in the `TaskConfiguration/ScenarioTemplates` folder that implement end-to-end scenarios.

Vertical scaling

The vertical scaling template sets up two tasks for vertical scaling. The Ops Automator's handler AWS Lambda function retrieves the configuration for the *CPU metrics* task from Amazon DynamoDB and runs that task at a defined interval. The *CPU metrics* task checks the CPU utilization metrics of your Amazon Elastic Compute Cloud (Amazon EC2) instances against thresholds you define during deployment. When CPU utilization is below the low threshold, the instance is tagged to be scaled down. When utilization is above the high threshold, the instance is tagged to be scaled up. If utilization is within both thresholds, the instance is not changed.

Once an instance is tagged, the solution triggers the *vertical scaling* task to resize an existing instance, or replace the instance with a new instance with a different size, depending on which configuration you choose. You can specify a range of instance types for resizing.

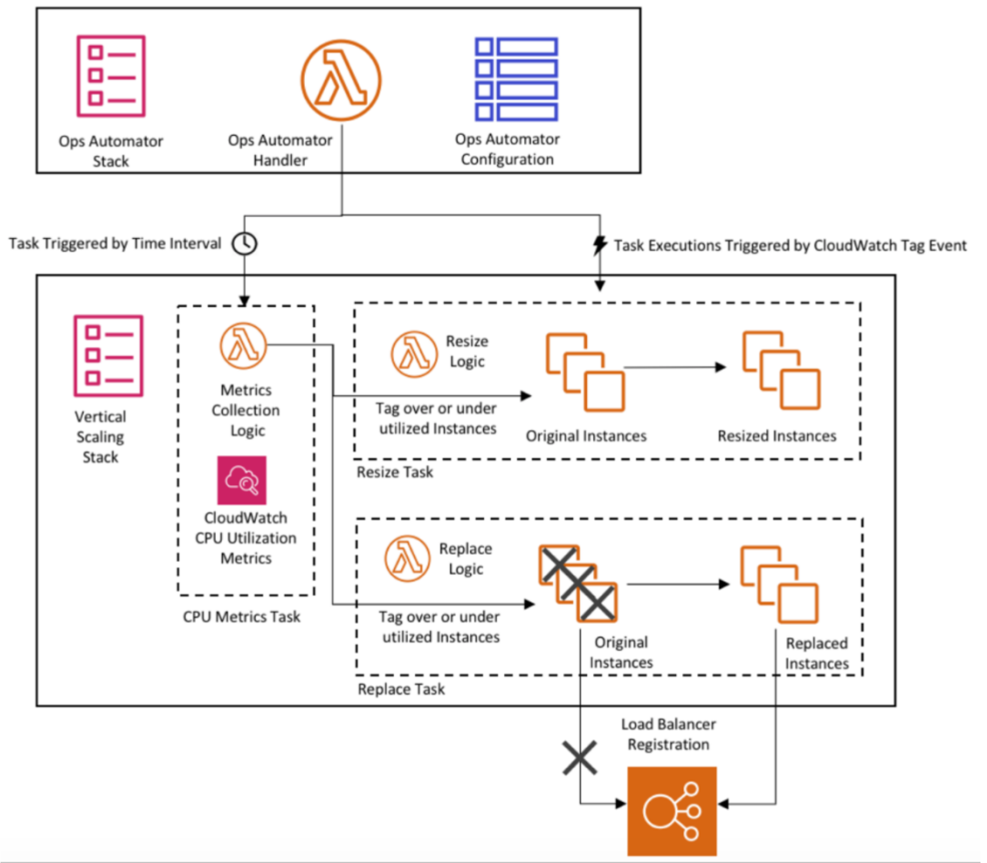
Ops Automator Stack
Ops Automator Handler
Ops Automator Configuration
Task Triggered by Time Interval
Task Executions Triggered by CloudWatch Tag Event
Vertical Scaling Stack
Metrics Collection Logic
CloudWatch CPU Utilization Metrics
CPU Metrics Task
Resize Logic
Tag over or under utilized Instances
Original Instances
Resized Instances
Resize Task
Replace Logic
Tag over or under utilized Instances
Original Instances
Replaced Instances
Replace Task
Load Balancer Registration

Figure 4: Ops Automator vertical scaling process

Scaling method

This solution includes two methods for scaling: resizing your existing instance or replacing your instance with a new instance of a different size. When you specify a vertical scaling method, the solution creates the applicable task (*resize* or *replace*).

Resizing an existing instance

When you choose to resize your existing instances, the solution automatically stops your existing instance, resizes the instance to the next defined size up or the next defined size down, then starts the instance again.

During the resizing, Amazon Elastic Block Store (Amazon EBS) volumes on the instance will remain attached and the data will persist. However, any data on the ephemeral storage of your instance will be lost. To keep your data, it must be stored on an attached Amazon EBS volume.

This approach may be best for applications that can experience downtime, and use cases where the data on volumes must be preserved.

You can choose to resize your instances by type or by step using the **Resizing mode** AWS CloudFormation template parameter. When you choose to resize by type, the solution resizes your instance with a type you specify in the **New instance type(s)** parameter. We recommend you specify more than one type in this parameter to provide an alternative if the specified type is not available. When you choose to resize by step, the solution resizes your instance with the next defined size up or down in a range you specify in the **Scaling Range** parameter.

Replacing with a new instance of a different size

When you choose to replace your existing instance with a new instance of a different size, the solution automatically launches a new instance with the same Amazon Machine Image and settings that is the next defined size up or the next defined size down. The solution is integrated with Elastic Load Balancing to automatically register (classic load balancers and application load balancers) the new instance with the same load balancer or target group.

Amazon EBS volumes on the original instance will be deleted, so any data on those volumes will be lost.

This approach may be best for applications that cannot experience downtime, and use cases where the data on volumes does not need to be preserved.

You can choose to replace your instances by type or by step using the **Replacement mode** AWS CloudFormation template parameter. When you choose to replace by type, the solution replaces your instance with a new instance with a type you specify in the **New instance type(s)** parameter. We recommend you specify more than one type in this parameter to provide an alternative if the specified type is not available. When you choose to replace by step, the solution replaces your instance with a new instance that is the next defined size up or down in a range you specify in the **Scaling Range** parameter.

Scaling range

You must specify at least two valid Amazon EC2 instance types for the **Scaling Range** parameter, and the instances types must be compatible. For more information, refer to Compatibility for Resizing Instances in the *Amazon EC2 User Guide for Linux Instances*.

The solution will not scale to instances that are not within the defined range. If an instance that exceeds the high threshold is at the top of the instance type range, the solution will not scale the instance and a message will be logged in Amazon CloudWatch. If an instance that falls below the low threshold is at the bottom of the range, the solution will not scale the instance and a message will be logged in CloudWatch.

Assumed instance type

You can use the **Assumed Type** AWS CloudFormation parameter to specify an instance type to use for instances that are not within the range you define. This instance type is used as the basis for scaling.

For example, a customer might specify a range of t2.micro, t2.small, and t2.medium, and an assumed type of t2.small. If that customer has a t3.small instance that exceeds the high threshold, the solution will use t2.small as the original instance type and scale the instance to t2.medium.

If you do not specify an assumed type, the solution logs an error in Amazon CloudWatch when it has to scale an instance that is not within the range you defined.

Launch the vertical scaling template

> **Note:** If you used the `ActionsConfiguration.html` file to launch the task, continue to step 7. For more information on the file, refer to Role and Task Templates.

1. In the primary account's Amazon S3 console, navigate to the bucket for the Ops Automator solution stack.

> **Note:** You can find the name of the S3 bucket in the AWS CloudFormation stack **Outputs** tab. The bucket name is the value of the **ConfigurationBucketName** key.

2. In the **TaskConfiguration/ScenarioTemplates** folder, select the `Ec2VerticalScaling` AWS CloudFormation template.
3. Copy the **Link** value.
4. In the AWS CloudFormation console, select **Create Stack**.
5. Select **Specify an Amazon S3 template URL**.
6. Paste the template link into the text box and select **Next**.
7. Enter a **Stack name**.

> **Note:** This name will also be the name of the scaling plan that is used to tag Amazon EC2 instances for scaling.

8. Under **Parameters**, review the parameters for the template and modify them as necessary.

Parameter	Default	Description
Ops Automator Stack	*<stackname>*	The name of the Ops Automator AWS CloudFormation stack. This parameter is auto-populated with the name you chose for your Ops Automator stack.
Scaling Interval	`5`	Specify how often, in minutes, to check CPU utilization metrics.
Scaling Method	`replace`	Choose whether to restart your instances with a new size, or replace your instance with a new, resized instance. Select `resize` or `replace`.
Instance Types	`t2.micro, t2.small, t2.medium, t2.large`	A comma-delimited list of valid, compatible instance types. **Note:** The list must be sorted from smallest to largest and must contain at least two instance types.
Assumed Type	`t2.small`	The default instance type that will be used if the instance type of an existing instance is not within the range defined in the **Instance Types** parameter. **Note:** You must specify an instance type that is specified in the **Instance Types** parameter.
CPU High Utilization Threshold	`90`	The high threshold for average CPU utilization that will determine whether an instance is scaled up.
CPU Low Utilization Threshold	`10`	The low threshold for average CPU utilization that will determine whether an instance is scaled down.
Instance Tags	`LastScalingAction=Instance scaled from type {org-instance-type} to type {new-instance-type} at {iso-datetime} by task {task} in Ops Automator stack {stack}`	Tags to set on the resized Amazon EC2 instance.
Account List	<Optional input>	A comma-delimited list of account IDs for instance resizing. **Note:** You must enable the `Ec2TagCpuInstance`, `Ec2ReplaceInstance`, and `Ec2ResizeInstance` actions in the cross-account role in each account. For more information, refer to Task Execution Across Accounts.
Region List	<Optional input>	A comma-delimited list of AWS Regions for instance resizing

Parameter	Default	Description
		Note: You must launch the `event forwarder` stack in each applicable Region. For more information, refer to Task Execution Across Accounts.
Detailed Logging	`No`	Choose whether to generate detailed Amazon CloudWatch logs.

9. Select **Next**.
10. Select **Next**. Then, on the **Review** page, review and confirm the settings. Select the checkbox acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
11. Choose **Create** to deploy the stack.
12. After the stack is created, add the `Scaling=<vertical-scaling-stackname>` tag to applicable instances.

Appendix J: AWS Fargate (Amazon ECS) implementation

Ops Automator is a developer framework that runs AWS Lambda tasks by default. In cases where parts of tasks take longer than the 15 minute time limit for Lambda functions, the framework can be configured to run these steps as Amazon Elastic Container Service (Amazon ECS) tasks using an AWS Fargate cluster. Fargate is a serverless compute engine that removes the need to provision and manage servers.

This appendix describes how to set up a Fargate cluster that is used by the Ops Automator framework for long-running tasks. Beginning with Ops Automator v2.2.0 you can either deploy with Fargate, or add it later. The following procedure assumes you deployed the Ops Automator using the AWS Management Console and an AWS CloudFormation template.

Note: If you previously deployed Ops Automator from the GitHub repository, refer to the README file to update Ops Automator in your environment.

Setup

Overview

Use the following procedure to deploy or update Ops Automator using the AWS Fargate on Amazon ECS option.

1. Deploy Ops Automator with AWS Fargate.
2. Set up the build environment.
3. Build and deploy the Docker container.
4. Deploy Ops Automator actions using Amazon ECS.

Deploy Ops Automator with AWS Fargate

You will deploy this solution's template following the procedure in Automated deployment. When you review the parameters, choose the ECS/Fargate option by changing the default value to **True**. Refer to Step 1 for details.

You can choose to run Fargate in an existing Amazon Virtual Private Cloud (Amazon VPC), or create a new one.

- To use an existing Amazon VPC, you must provide the VPC ID and subnet IDs for at least two subnets. The Fargate cluster deploys in the AWS VPC and subnets that you specify. Note that public subnets are required.
- If you do not provide a VPC ID, the solution creates a new VPC and public subnets for the Fargate cluster.

Set up the build environment

The following resources are required to set up the build environment:

- A workstation with Docker, Python with Boto3, and the AWS Command Line Interface (AWS CLI) installed.
- An AWS Identity and Access Management (IAM) role that allows AWS CloudFormation `describe-stacks`, `sts get-caller-identity`, and `ecr get-login` parameters.

Download and install the following Amazon ECS image tools and files:

- Dockerfile.orig – Dockerfile for the image
- build-and-deploy-image.sh – Script to build and upload the image to ECR
- ops-automator-ecs-runner.py – Python script that runs on the image
- requirements.txt – Required python modules loaded on the image

Build and deploy the Docker container

In your terminal, navigate to the folder where you downloaded the Amazon ECS files and run the following command.

```
./build-and-deploy-image.sh -s <stack-name> -r <region>
```

Replace `<stack-name>` with the name you entered for your Ops Automator CloudFormation stack. Replace `<region>` with the Region where your stack is currently running.

This script pulls down the required files to build a Docker image based on the AWS Linux Docker optimized Amazon Machine Images (AMI). It installs the `ops-automator-ecs-runner.py` script on the image, then uploads the image to the **ops-automator** repository created by the Ops Automator stack.

Deploy Ops Automator actions using Amazon ECS

The Amazon ECS option is now available for Actions. Continue with Step 2 in Automated deployment.

Appendix K: Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products.

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier
- **Timestamp:** Data-collection timestamp
- **Task Data:** Task-specific data (refer to the following table)

Action	Task data
DynamoDB Set Capacity	The number of old and updated read and write units for DynamoDB tables and indexes..
EC2 Copy Snapshots	The number of copied snapshots.
EC2 Create Snapshot	The number of created snapshots and total volumes.
EC2 Delete Snapshot	The number of deleted snapshots.
EC2 Replace Instance	The number of replaced instances, and the old and new instance types.
EC2 Resize Instance	The number of resized instances, and the old and new instance types.
EC2 Tag CPU Instance	The number of checked, overutilized, and underutilized instances.

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 as follows:

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

to

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

Source code

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
July 2017	Initial release
December 2017	Added considerations for large numbers of resources; new options for encryption and an AWS KMS customer master key for EC2 snapshot copies
December 2017	Added instructions for configuring an alternative customer master key for snapshot encryption
April 2018	Added clarification on AWS GovCloud (US) Region availability
May 2019	Added information about the vertical scaling functionality
June 2019	Added information about the new functionality for version 2
October 2019	Upgraded the solution to Python 3.7
August 2020	Added information about using Amazon ECS as the automation platform; for additional information about changes to v2.2, refer to the CHANGELOG.md file in the GitHub repository