Firewall Automation for Network Traffic on AWS

Implementation Guide

Firewall Automation for Network Traffic on AWS: Implementation Guide

Table of Contents

Welcome 1
Cost 2
Architecture overview 3
Deployment considerations 5
    Amazon VPC CIDR Block 5
    AWS Network Firewall configuration 5
    Deployment environment 5
    AWS Transit Gateway 5
Solution components 6
    AWS CodeCommit 6
    AWS CodePipeline 6
    Amazon CloudWatch 6
    Amazon Simple Storage Service 6
    Amazon VPC 6
Security 7
    AWS Key Management Service 7
    AWS Identity and Access Management roles 7
Design considerations 8
    Regional deployments 8
AWS CloudFormation template 9
Automated deployment 10
    Prerequisites 10
        Update the AWS Network Firewall log destination 10
    Deployment overview 10
    Step 1. Launch the stack 10
    Step 2. Modify the Network Firewall, firewall policies, and rule groups 14
Additional resources 15
Configuring resources for AWS Network Firewall 16
    AWS CodeBuild validation stage 16
Troubleshooting 19
Uninstall the solution 20
    Using the AWS Management Console 20
    Using AWS Command Line Interface 20
Collection of operational metrics 21
Source code 22
Revisions 23
Contributors 24
Notices 25
AWS glossary 26

AWS Solution for automating AWS Network Firewall deployments

Publication date:*February 2021 (last update (p. 23): April 2021)*

Firewall Automation for Network Traffic on AWS configures the AWS resources needed to filter network traffic. With this solution, you can inspect hundreds or thousands of Amazon VPCs and accounts in one place. This solution saves you time by automating the process of provisioning a centralized AWS Network Firewall to inspect traffic between VPCs. You can also centrally configure and manage your AWS Network Firewall, firewall policies, and rule groups.

This solution utilizes AWS Network Firewall to provide granular visibility and control of your network traffic. This allows you to accomplish network segmentation, egress domain filtering, and intrusion prevention through event-driven logging. You can enable AWS Network Firewall in your Amazon VPC environments with just a few clicks in the AWS Management Console. AWS Network Firewall automatically scales with network traffic to provide high availability protections without the need to set up or maintain the underlying infrastructure. This solution also helps you collaborate and manage the changes to the AWS Network Firewall configuration by using GitOps workflow.

This implementation guide describes architectural considerations and configuration steps for deploying Firewall Automation for Network Traffic on AWS in the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

This guide is intended for IT architects, DevOps professionals, technology professionals, network engineers, and security engineers who have practical experience architecting in the AWS Cloud.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of February 2021, the estimated cost for running this solution for two network firewall endpoints in two availability zones, 5 GB of traffic per day, with default settings in the US East (N. Virginia) Region is **approximately $620.55 per month**. This includes estimated charges for AWS CodePipeline, AWS CodeBuild, and Amazon Simple Storage Service (Amazon S3).

AWS Service	Dimensions	Total Cost (per month)
AWS Network Firewall (Endpoint)	2 endpoints/24 hours ($0.395/endpoint/hour)	$568.80
AWS Network Firewall (data processed)	5 GB ($0.065/GB)	$9.75
AWS Transit Gateway (VPC attachment)	24 hours ($0.05/hour)	$36.00
AWS Transit Gateway (data processed)	10 GB ($0.02/GB)	$6.00
Amazon Code Services (CodePipeline, CodeBuild, CodeCommit)		Depends on number of AWS CodePipeline executions
Amazon S3		Depends on number of AWS CodePipeline executions and Network Firewall Log Activity
	Total	**$620.55**

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

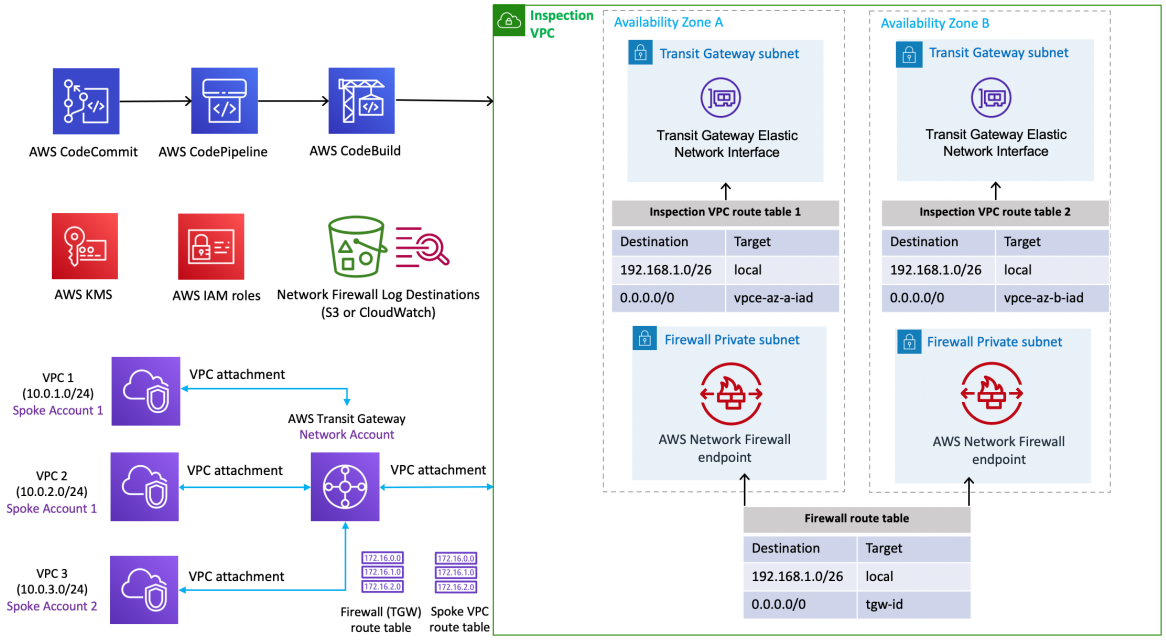

Figure 1: Firewall Automation for Network Traffic on AWS architecture

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

The AWS CloudFormation template deploys an inspection VPC with a total of four subnets in randomly selected availability zones in the Region where the solution is deployed. Two of the subnets are used to create VPC Transit Gateway attachments if you provide an existing AWS Transit Gateway ID. The other two subnets are used to create AWS Network Firewall endpoints in two randomly selected availability zones. The template creates a new AWS CodeCommit repository and a default network firewall configuration that allows all traffic. The template also includes a set of examples to help you create new rule groups. You can modify the configuration package in the CodeCommit repository. This invokes the AWS CodePipeline to run the following stages:

Validation stage–AWS Network Firewall configuration is validated using AWS Network Firewall APIs with dry run mode enabled. This allows the user to find any unexpected issues before attempting an actual change. This stage also checks the JSON file structure and checks if all the referenced files in the configuration exist in the package.

Deployment stage–A new Network Firewall, Network Firewall policy, and rule groups are created in this stage. If any of the resources already exist the resources are updated. This stage also helps with detecting any changes and remediates by applying the latest configuration from the AWS CodeCommit repository. The rule groups changes will roll back to the original state if one of the rule group changes fails. The appliance mode activates for the TGW-VPC attachment to avoid asymmetric traffic. For more information, refer to Appliance in a shared services VPC.

This solution also creates Amazon VPC route tables for each availability zone with a default route destination with the target as Amazon VPC endpoint for AWS Network Firewall. A shared route table with firewall subnets is also created with default route destination with the target as the transit gateway

ID. This route is only created if the transit gateway ID is provided in the AWS CloudFormation input parameters.

Deployment considerations

Amazon VPC CIDR Block

The Amazon VPC and related resource configuration cannot be updated using the CloudFormation update stack workflow. In order to update the VPC CIDR block, the Amazon VPC would have to be deleted and recreated. We recommend consulting your network engineering team to obtain a dedicated CIDR block for the inspection VPC.

AWS Network Firewall configuration

This solution is deployed with a default network firewall policy, which does not disrupt your existing network. This allows you to design and deploy custom network firewall policies, as well as stateful and stateless rule groups. This also includes existing Suricata stateful rules. For more information about Suricata, refer to Stateful Suricata compatible IPS rule groups in the *Network Firewall Guide*.

Note
You can also use AWS Firewall Manager to centrally configure and manage firewall rules for this solution.

Deployment environment

This solution can be deployed multiple times in the same Region to allow users to set up a new AWS Network Firewall and related resources for an existing AWS Transit Gateway. You can deploy this solution without an AWS Transit Gateway to test it before making any network changes. If you don't provide a transit gateway ID, this solution will not create the TGW-VPC attachment. This ensures that your network engineers can customize the AWS Network Firewall configuration and update the firewall policies before making any network changes.

AWS Transit Gateway

This solution works with an existing AWS Transit Gateway to create a VPC transit gateway attachment if the AWS Transit Gateway ID is provided. The association and propagation to the existing Transit Gateway route tables are also created if you provide the route table ID and transit gateway ID. For details, refer to the CloudFormation parameters (p. 10) section.

To create AWS Transit Gateways and manage Amazon VPCs and peering attachments, we recommend using the AWS Serverless Transit Network Orchestrator solution.

Solution components

AWS CodeCommit

This solution creates an AWS CodeCommit repository with the default configuration and examples.

AWS CodePipeline

AWS CodePipeline validates, tests, and implements changes based on updates to the configuration package in the AWS CodeCommit repository.

Amazon CloudWatch

If you select `CloudWatchLogs` for the `Log Destination for the Network Firewall` parameter, this solution will create a log group for your logs. Your alert and flow logs collect log records and consolidate them into log files. For more information, refer to the Network Firewall developer guide.

Amazon Simple Storage Service

The solution creates the following Amazon Simple Storage Service (Amazon S3) buckets:

Source Code Bucket–This bucket hosts versions of the source code used by the AWS CodeBuild stage to validate and deploy Network Firewall resources and update related resources.

CodePipeline Artifacts Bucket–This bucket stores input and output artifacts created by the CodePipeline stages. AWS CodePipeline zips and transfers the files for input or output artifacts as appropriate for the action type in the stage.

Network Firewall Log Destination Bucket–This Amazon S3 bucket is only created if you select `Amazon S3` for the `Log Destination for the Network Firewall` parameter.

Amazon VPC

This solution creates an inspection Amazon VPC with four subnets to support Transit Gateway attachment and AWS Network Firewall endpoints.

Security

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

AWS Key Management Service

This solution creates two AWS Key Management Service (AWS KMS) encryption keys. One of the keys is used to encrypt objects in the Amazon S3 artifact, source code buckets, and AWS CodeBuild projects. The second key is used to encrypt the AWS Network Firewall log destinations, which depends on whether you select `Amazon CloudWatch` or `Amazon S3 bucket`. By default, only IAM roles provisioned by this solution have permission to perform encrypt or decrypt operations with this key. Automatic key rotation is enabled by default.

AWS Identity and Access Management roles

AWS Identity and Access Management (IAM) roles allow you to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant permissions to AWS CodePipeline and AWS CodeBuild stages in order to access Amazon S3 buckets and manage AWS Network Firewall resources.

Design considerations

Regional deployments

This solution uses AWS Network Firewall, which is currently available in select AWS Regions only. You must launch this solution in an AWS Region where AWS Network Firewall is available. For the most current availability by Region, refer to the AWS Regional Services List.

The following table lists the supported AWS Regions for this solution.

Region ID	Region Name
us-east-1	US East (N. Virginia)
us-east-2	US East (Ohio)
us-west-1	US West (N. California)
us-west-2	US West (Oregon)
ap-south-1	Asia Pacific (Mumbai)*
ap-northeast-2	Asia Pacific (Seoul)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)
ap-northeast-1	Asia Pacific (Tokyo)
ca-central-1	Canada (Central)
eu-west-2	Europe (London)
eu-central-1	Europe (Frankfurt)
eu-west-1	Europe (Ireland)
eu-west-3	Europe (Paris)
eu-north-1	Europe (Stockholm)
sa-east-1	South America (São Paulo)

AWS CloudFormation template

This solution uses AWS CloudFormation to automate the deployment of the Firewall Automation for Network Traffic on AWS in the AWS Cloud. It includes the following CloudFormation template, which you can download before deployment:

aws-network-firewall-deployment-automations-for-aws-transit-gateway.template: Use this template to launch the solution and all associated components. The default configuration deploys AWS Network Firewall, Amazon VPC, AWS Transit Gateway, AWS CodeCommit Repository, AWS CodeBuild, AWS CodePipeline, and Amazon CloudWatch. You can also customize the template to meet your specific needs.

Note
AWS CloudFormation resources are created from AWS Cloud Development Kit (CDK) (AWS CDK) constructs.

Automated deployment

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

Time to deploy: Approximately 7 minutes. The AWS CodePipeline can take up to 10 minutes to deploy AWS Network Firewall resources.

Prerequisites

The solution requires AWS Network Firewall to be available in the Region. For more information, refer to Deployment considerations (p. 5).

Update the AWS Network Firewall log destination

If you have previously deployed this solution, any updates made to the stack will require you to manually initiate the AWS CodePipeline to update to the AWS Network Firewall log destination. AWS Network Firewall configuration should not be updated to manually release changes. To start the AWS CodePipeline manually, refer to Start a pipeline manually in the *AWS CodePipeline User Guide*.

To modify the AWS Network Firewall, firewall policy, and rule groups, refer to Configuring resources for network firewall (p. 3).

Deployment overview

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

Step 1. Launch the stack (p. 10)

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: Stack Name, Transit Gateway ID, Transit Gateway route table(s), Firewall Logging Configuration,
- Review the other template parameters, and adjust if necessary.

Step 2. Modify AWS Network Firewall, firewall policies, rule groups (p. 14)

- After the stack is successfully created, the AWS CodePipeline is initiated by CloudFormation.
- Modify the AWS Network Firewall, firewall policies, and rule group. For details, refer to Configuring resources for network firewall (p. 16).

Step 1. Launch the stack

This automated AWS CloudFormation template deploys Firewall Automation for Network Traffic on AWS in the AWS Cloud.

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

1. Sign in to the AWS Management Console and select the button to launch the `aws-network-firewall-deployment-automations-for-aws-transit-gateway` AWS CloudFormation template.

Alternatively, 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 the solution in a different AWS Region, use the Region selector in the console navigation bar.

   Note
   This solution uses AWS Network Firewall, which is currently available in select AWS Regions only. You must launch this solution in an AWS Region that supports this service. For the most current availability by Region, refer to the AWS Regional Services List.

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 this solution template and modify them as necessary. This solution uses the following default values:

VPC configuration

Parameter	Default	Description
Provide the CIDR bock for the inspection VPC	`192.168.1.0/26`	CIDR block for VPC. Must be `/26` or larger CIDR block.

Transit Gateway configuration

Parameter	Default	Description
Provide the existing AWS Transit Gateway ID you wish to attach to the Inspection VPC	`<Optional input>`	The existing AWS Transit Gateway ID in the current Region. Example: `tgw-a1b2c3d4e5` **Note** If AWS Transit Gateway ID is removed/updated and the stack is updated, the AWS Transit Gateway Attachment

Parameter	Default	Description
		is will not be deleted in the account. The AWS Transit Gateway attachment has to be deleted manually.
Provide the AWS Transit Gateway Route Table to be associated with the Inspection VPC TGW Attachment	`<Optional input>`	Existing AWS Transit Gateway route table id. Example: Firewall Route Table. Example: `tgw-rtb-0a1b2c3d` **Note** If the AWS Transit Gateway route table ID is removed and stack is updated, the AWS Transit Gateway Attachment is not deleted in the account. The AWS Transit Gateway attachment has to be deleted manually.
Provide the AWS Transit Gateway Route Table to receive 0.0.0.0/0 route to the Inspection VPC TGW Attachment	`<Optional input>`	Existing AWS Transit Gateway route table ID for propagation. Example: Spoke VPC Route Table. Example: `tgw-rtb-183ae12f` **Note** If the AWS Transit Gateway ID/ AWS Transit Gateway route table ID and Transit Gateway route table ID for default route is removed and stack is updated, the default route in the AWS Transit Gateway route table, route entry for 0.0.0.0/0 is not deleted. The route has to be deleted manually.

Firewall Logging configuration

Parameter	Default	Description
Select the type of log destination for the Network Firewall	`CloudWatchLogs`	The type of storage destination for logs. You can send logs to an Amazon S3 bucket or an Amazon CloudWatch log group. **Note** The default value is `CloudWatchLogs`. This solution will create a log group for the firewall logs. You can also store logs in an Amazon S3 bucket. If no logging needs to be configured, select `ConfigureManually`. If this parameter is being updated after first deployment, the AWS CodePipeline must be initiated manually to update the log destination. For details, refer to Solution components (p. 6).
Select the type of log to send to the defined log destination	`FLOW`	The type of log to send. Alert logs report traffic that matches a stateful rule with an action setting that sends an alert log message. Flow logs are standard network traffic flow logs. **Note** You can set this to ALERT logs or enable both types of logs. For details, refer to Logging network traffic from AWS Network Firewall in the *Network Firewall Developer Guide*.
Select the log retention period for Network Firewall Logs	`90`	Log retention period in days. This setting is also applicable to Inspection VPC Flow Logs retention period.

6. Choose **Next**.
7. On the **Configure stack 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 stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a **CREATE_COMPLETE** status in approximately 7 minutes.

After AWS CloudFormation completes the stack creation, the AWS CodePipeline created by the solution will continue to run until all the AWS Network Firewall resources are created.

Step 2. Modify the Network Firewall, firewall policies, and rule groups

After successfully deploying the stack, AWS CodePipeline initiates the AWS CodeBuild stages. Each stage validates and deploys the AWS Network Firewall components. After the deployment stage completes, you can view the AWS Network Firewall and firewall policy.

To modify the default AWS Network Firewall, firewall policy, and created rule groups, refer to Configuring resources for network firewall (p. 16).

Additional resources

AWS services

- AWS CloudFormation
- AWS CodeCommit
- AWS CodeBuild
- AWS CodePipeline
- Amazon CloudWatch
- AWS Network Firewall
- Amazon S3
- AWS Transit Gateway
- Amazon VPC

Configuring resources for AWS Network Firewall

After deploying the solution, you can customize the resources for your network. This solution creates an AWS CodeCommit repository to store all the AWS Network Firewall configuration files. These files can be updated and new resources can be created in the respective folders. After the changes are committed and pushed to the AWS CodeCommit repository, this solution uses the configuration package to create or update AWS Network Firewall resources. The changes to the firewall, firewall policy, and rule groups can be reviewed after the AWS CodePipeline has finished running successfully. We recommend monitoring the pipeline status to confirm that the changes were deployed successfully. You can also review AWS CodeBuild stage logs in the AWS CodePipeline.

> **Note**
> All references to the `FirewallPolicyArn` and `ResourceARN` attributes should contain the reference path to the actual JSON files. These values are used by this solution to retrieve the configurations. Refer to the example configurations that are provided in the AWS CodeCommit repository.

An unique string is added to the AWS Network Firewall and firewall policy to allow you to deploy the solution more than once in a Region. The deployed resources have a unique name for each Region.

If there are existing resources in the AWS Network Firewall that have the same name as those being referenced in the solution, they will be updated with the configuration provided in the AWS CodeCommit repository. Before committing changes, we recommend reviewing the resource names for any resources previously created in the AWS Network Firewall console in the account and Region.

AWS CodeBuild validation stage

The solution creates two AWS CodeBuild stages. The first stage validates the configuration files (firewall, firewall policy, and rule group) and checks if the JSON format is valid. This solution uses these files to validate the AWS Network Firewall APIs to ensure that the attributes defined in the files have valid data. If any files have formatting issues or invalid data, the AWS CodeBuild stage will be in a Failed state and the deployment of the files to AWS Network Firewall will not continue. The AWS CodeBuild validation stage will provide error details for the files, similar to the ones in the following log example.

```
[TIMESTAMP] : "-----------INVALID FILES START-----------"
[TIMESTAMP]: {
   “path”: “./firewallPolicies/firewall-policy-1.json”,
   “error”: “Unexpected key ‘key’ found in params.FirewallPolicy”
}
[TIMESTAMP]: "-----------INVALID FILES END-----------"
[TIMESTAMP]: “Validation failed.”
[TIMESTAMP]: “Error in firewall config validation” : “Validation failed.”
```

Once the solution is deployed, the AWS CodeCommit repository will have the following default directory structure.

- **Examples**– This directory has example configuration files.

- **Firewalls**– This directory contains the firewall configuration in JSON format. It includes the attributes as a document in the CreateFirewallAPI action.

  Note
  `FirewallPolicyArn` has a value which exactly matches the file path of the firewall policy file in the code commit repository.

As shown in the following example JSON file, this solution uses `firewall-policy-1.json` for the firewall policy in the `./firewallPolicies/firewall-policy-1.json` commit repository path.

```
 {
“FirewallName”: “Firewall-1”,
“FirewallPolicyARN”: “./firewallPolicies/firewall-policy-1.json”,
“Description”: “Network Firewall 1”.
“DeleteProtection”: true,
“SubnetChangeProtection”: true
 }
```

- **FirewallPolicies**– This directory contains the firewall policy configuration in JSON format which will have attributes as documented in CreateFirewallPolicy, the attribute ResourceArn will have a value which exactly matches the file path of the rule group file in the code commit repository. Below is an example of the network firewall policy.

```
{
"FirewallPolicyName": "Firewall-Policy-1",
"Description": "Firewall Policy 1",
"FirewallPolicy": {
  "StatelessDefaultActions": [
    “aws:drop”
   ],
  "StatelessRuleGroupReferences": [
    {
      "Priority": 30,
      “ResourceArn”:“./ruleGroups/stateless-fwd-to-stateful.example.json”
    },
    {
      "Priority": 20,
      “ResourceArn”:“./ruleGroups/stateless-pass-action.example.json”

    }
  ],
  "StatefulRuleGroupReferences":[
    {
      “ResourceArn”:“./ruleGroups/stateful-domainblock.example.json”
    },
    {
      “ResourceArn”:“./ruleGroups/suricata-rule-reference.json”
    }
   ]
  }
}
```

  Note
  `ResourceArn` attribute in the Firewall policy file should have the file path to the rule group file in the AWS CodeCommit repository.

- **RuleGroup**– This directory contains the rule groups configuration in JSON format which will have attributes as documented in CreateRuleGroup. The rule group can be defined by providing details in the `RuleGroup` attribute or the rules (Suricata flat format) attribute, as shown in the following stateful rule group file example.

```
{
  "RuleGroupName": "StatefulRulesExample1",
  "RuleGroup": {
  "RulesSource": {
    "RulesSourceList": {
      "TargetTypes": ["HTTP_HOST"],
      "Targets": [
        "test.example.com",
        "test2.example.com"
        ],
    "GeneratedRulesType": "DENYLIST"
    }
    }
  },
  "Type": "STATEFUL",
  "Description": "Stateful Rule",
  "Capacity": 100
}
```

In this following example Suricata file, the rules attribute references the drop.rules file where the rules are defined. For more information, refer to the Drop.rules example file.

```
{
  “RuleGroupName”: “suricata-drop-rules”,
  “Rules”: “./ruleGroups/drop.rules”,
  “Type”: “STATEFUL”.
  “Description”: “Suricata rule group”,
  “Type”: 100
}
```

Note
The `drop.rules` file must be added to the configuration package and only local path is allowed. Amazon S3 and HTTP links are not allowed.

Troubleshooting

Issue: The CloudFormation stack has completed successfully, but not all the AWS Network Firewall resources are created.

Solution: After the CloudFormation stack is complete, the CodePipeline stage created by the solution might still be in the **In-Progress** state. Once the CodePipeline stage is completed, all the AWS Network Firewall resources will be available in the AWS Network Firewall console.

Issue: AWS CodePipeline stage is failing.

Solution: If the AWS CodePipeline stage is in **Failed** state, it means this solution has not been able to complete the create or update network firewall resources operation. Refer to the logs in the AWS CodePipeline stages to ensure that the AWS CodeBuild stages are successful. If a JSON file is not valid or has incorrect information, the AWS CodeBuild stage that validates the files will list the errors along with the file names. For more information, refer to the AWS CodeBuild User Guide.

Uninstall the solution

You can uninstall the Firewall Automation for Network Traffic on AWS solution from the AWS Management Console, or by using the AWS Command Line Interface (AWS CLI).

Using the AWS Management Console

1. Sign in to the AWS CloudFormation console.
2. Select this solution's installation stack.
3. Choose **Delete**.
4. The following resources will be retained even after the solution is deleted. Refer to the following links to manually delete the resources:
   - AWS CodeCommit repository
   - Amazon CloudWatch log groups
   - Amazon S3 CodePipeline artifact bucket
   - Amazon S3 CodeBuild source code bucket
   - AWS Network Firewall
   - AWS Network Firewall firewall policy
   - AWS Network Firewall rule groups
   - Inspection VPC
   - AWS Transit Gateway attachment

Using AWS Command Line Interface

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

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

Collection of operational metrics

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

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each Firewall Automation for Network Traffic on AWS deployment
- **Timestamp:** Data-collection timestamp
- **Number of CloudFormation Stacks deployed in the account**
- **Number of Firewalls managed**
- **Number of Firewall Policies managed**
- **Number of stateful rule groups deployed**
- **Number of stateless rule groups deployed**
- **Number of Suricata rules deployed**
- **Network Firewall Destination Type**
- **Network Firewall Log Type**

AWS owns the data gathered through this survey. Data collection is subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section from:

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

to:

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

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others. The Firewall Automation for Network Traffic on AWS template is generated using the AWS Cloud Development Kit (CDK) (AWS CDK). Refer to the README.md file for additional information.

Revisions

Date	Change
February 2021	Initial release
April 2021	Release v1.0.1: Updated branch name. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Contributors

- Lalit Grover
- Nikhil Reddy

Notices

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

AWS glossary

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