Deploy the ASAv Auto Scale Solution on AWS

Auto Scale Solution for ASAv on AWS

The following sections describe how the components of the Auto Scale solution work for the ASAv on AWS.

About the Auto Scale Solution

Cisco provides CloudFormation Templates and scripts for deploying an auto-scaling group of ASAv firewalls using several AWS services, including Lambda, auto scaling groups, Elastic Load Balancing (ELB), Amazon S3 Buckets, SNS, and CloudWatch.

ASAv Auto Scale in AWS is a complete serverless implementation (i.e. no helper VMs involved in the automation of this feature) that adds horizontal auto scaling capability to ASAv instances in the AWS environment.

The ASAv Auto Scale solution is a CloudFormation template-based deployment that provides:

- Completely automated configuration automatically applied to scaled-out ASAv instances.
- Support for Load Balancers and multi-availability zones.
- Support for enabling and disabling the Auto Scale feature.

Auto Scale Use Case

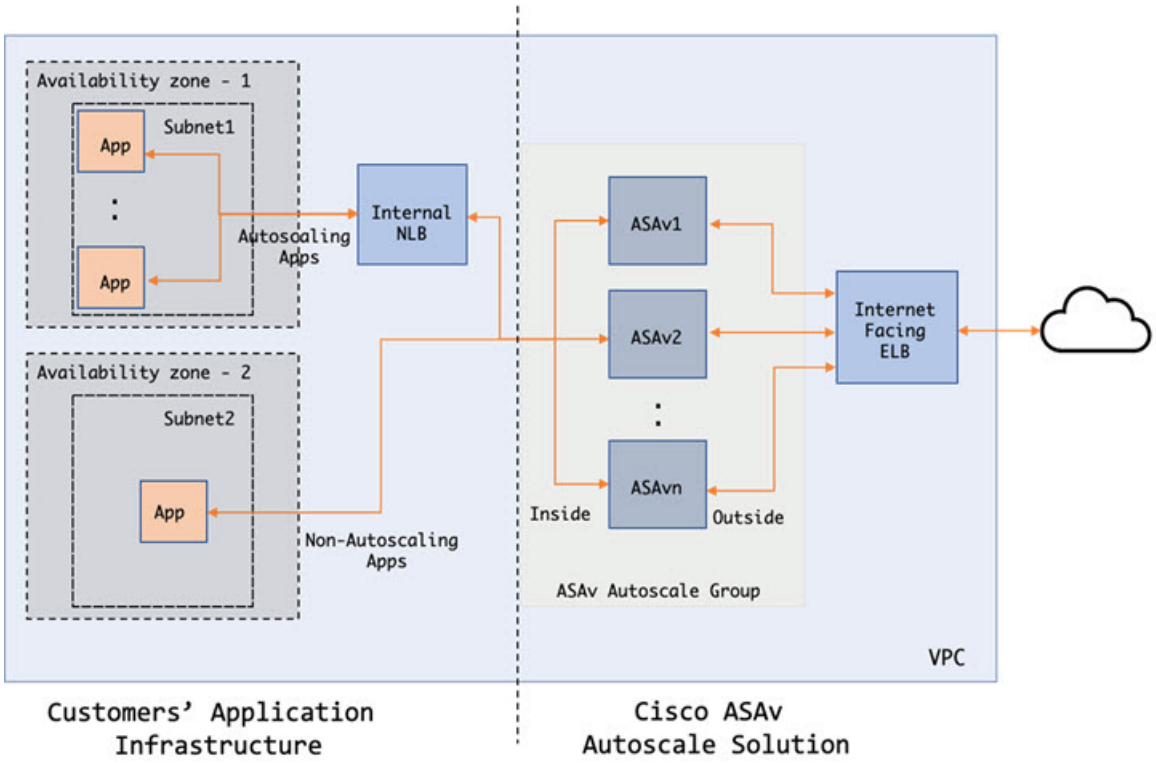
The Use Case for this ASAv AWS Auto Scale Solution is shown in the use case diagram. Because the AWS Load Balancer allows only Inbound-initiated connections, only externally generated traffic is allowed to pass inside via the Cisco ASAv firewall.

Note Secured ports need an SSL/TLS certificate, as described SSL Server Certificate, on page 6 in the Prerequisites.

The Internet-facing load balancer can be a Network Load Balancer or an Application Load Balancer. All of the AWS requirements and conditions hold true for either case. As indicated in the Use Case diagram, the right side of the dotted line is deployed via the ASAv templates. The left side is completely user-defined.

Note Application-initiated outbound traffic will not go through the ASAv.

Figure 1: ASAv Auto Scale Use Case Diagram



Port-based bifurcation for traffic is possible. This can be achieved via NAT rules. For example, traffic on Internet-facing LB DNS, Port: 80 can be routed to Application-1; Port: 88 traffic can be routed to Application-2.

How the Auto Scale Solution Works

To scale ASAv instances in and out, an external entity called the Auto Scale Manager monitors metrics, commands an auto scale group to add or delete ASAv instances, and configures ASAv instances.

The Auto Scale Manager is implemented using AWS Serverless architecture and communicates with AWS resources and the ASAv. We provide CloudFormation templates to automate the deployment of Auto Scale Manager components. The template also deploys other resources required for complete solution to work.

Note Serverless Auto Scale scripts are only invoked by CloudWatch events, hence they only run when an instance is launched.

Auto Scale Solution Components

The following components make up the Auto Scale solution.

CloudFormation Template

The CloudFormation template is used to deploy resources required by Auto Scale solution in AWS. The template consists of:

- Auto Scale Group, Load Balancer, Security Groups, and other miscellaneous components.
- The template takes user input to customize the deployment.

Note The template has limitations in validating user input, hence it is the user's responsibility to validate input during deployment.

Lambda Functions

The Auto Scale solution is a set of Lambda functions developed in Python, which gets triggered from Lifecycle hooks, SNS, CloudWatch event/alarm events. The basic functionality includes:

- Add/Remove Gig0/0, and Gig 0/1 interfaces to instance.
- Register Gig0/1 interface to Load Balancer's Target Groups.
- Configure and deploy a new ASAv with the ASA configuration file.

Lambda Functions are delivered to customer in the form of a Python package.

Lifecycle Hooks

- Lifecycle hooks are used to get lifecycle change notification about an instance.
- In the case of instance launch, a Lifecycle hook is used to trigger a Lambda function which can add interfaces to an ASAv instance, and register outside interface IPs to target groups.
- In the case of instance termination, a Lifecycle hook is used to trigger a Lambda function to deregister an ASAv instance from the target group.

Simple Notification Service (SNS)

- Simple Notification Service (SNS) from AWS is used to generate events.
- Due to the limitation that there is no suitable orchestrator for Serverless Lambda functions in AWS, the solution uses SNS as a kind of function chaining to orchestrate Lambda functions based on events.

Auto Scale Solution Prerequisites

Download Deployment Files

Download the files required to launch the ASAv Auto Scale for AWS solution. Deployment scripts and templates for your ASA version are available from the GitHub repository at:

- https://github.com/CiscoDevNet/cisco-asav

Attention Note that Cisco-provided deployment scripts and templates for auto scale are provided as open source examples, and are not covered within the regular Cisco TAC support scope. Check GitHub regularly for updates and ReadMe instructions.

Infrastructure Configuration

In a cloned/downloaded GitHub repository, the **infrastructure.yaml** file can be found in template folder. This CFT can be used to deploy VPCs, subnets, routes, ACLs, security groups, VPC end-points, and S3 buckets with bucket policies. This CFT can be modified to fit your requirements.

The following sections provide more information about these resources and their use in Auto Scale. You can manually deploy these resources and also use them in Auto Scale.

Note The **infrastructure.yaml** template deploys VPCs, subnets, ACLs, security groups, S3 buckets, and VPC end-points only. It does not create the SSL certificate, Lambda layer, or KMS key resources.

VPC

You should create the VPC as required for your application requirements. It is expected that the VPC have an Internet gateway with at least one subnet attached with a route to the Internet. Refer to the appropriate sections for the requirements for Security Groups, Subnets, etc.

Subnets

Subnets can be created as needed for the requirements of the application. The ASAv VM requires 3 subnets for operation as shown in the Use Case.

Note If multiple availability zone support is needed, then subnets are needed for each zone as subnets are zonal properties within the AWS Cloud

Outside Subnet

The Outside subnet should have route with '0.0.0.0/0' to the Internet gateway. This will contain the Outside interface of the ASAv, and also the Internet-facing NLB will be in this subnet.

Inside Subnet

This can be similar to the Application subnets, with or without NAT/Internet gateway. Please note that for ASAv health probes, it should be possible to reach the AWS Metadata Server (169.254.169.254) via port 80.

Note In this AutoScale solution, Load Balancer health probes are redirected to the AWS Metadata Server via inside/Gig0/0 interface. However, you can change this with your own application serving the health probe connections sent to ASAv from the Load Balancer. In that case, you need to replace the AWS Metadata Server object to the respective application IP address to provide the health probes response.

Management Subnet

This subnet is the ASAv Management interface. It's optional for you to have an Internet gateway (default route) or not.

Lambda Subnets

The AWS Lambda function requires two subnets having the NAT gateway as the default gateway. This makes the Lambda function private to the VPC. Lambda subnets do not need to be as wide as other subnets. Please refer to AWS documentation for best practices on Lambda subnets.

Application Subnets

There is no restriction imposed on this subnet from the Auto Scale solution, but in case the application needs Outbound connections outside the VPC, there should be respective routes configured on the subnet. This is because outbound-initiated traffic does not pass through Load Balancers. See the AWS Elastic Load Balancing User Guide.

Security Groups

All connections are allowed in the provided Auto Scale Group template. You need only the following connections for the Auto Scale Solution to work.

Table 1: Required Ports

Port	Usage	Subnet
Health Probe port (default: 8080)	Internet-facing Load Balancer health probes	Outside, Inside Subnets
Application ports	Application data traffic	Outside, Inside Subnets

Amazon S3 Bucket

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. You can place all the required files for both the firewall template and the application template in the S3 bucket.

When templates are deployed, Lambda functions get created referencing Zip files in the S3 bucket. Hence the S3 bucket should be accessible to the user account.

SSL Server Certificate

If the Internet-facing Load Balancer has to support TLS/SSL, a Certificate ARN is required. Refer to the following links for more information:

- Working with Server Certificates
- Create a Private Key and Self-Signed Certificate for Testing
- Create AWS ELB with Self-Signed SSL Cert (Third-party link)

Example of ARN: arn:aws:iam::[AWS Account]:server-certificate/[Certificate Name]

Lambda Layer

The ***autoscale_layer.zip*** can be created in a Linux environment, such as Ubuntu 18.04 with Python 3.6 installed.

```
#!/bin/bash
mkdir -p layer
virtualenv -p /usr/bin/python3.6 ./layer/
source ./layer/bin/activate
pip3 install pycrypto==2.6.1
pip3 install paramiko==2.7.1
pip3 install requests==2.23.0
pip3 install scp==0.13.2
pip3 install jsonschema==3.2.0
echo "Copy from ./layer directory to ./python\n"
mkdir -p ./python/.libs_cffi_backend/
cp -r ./layer/lib/python3.6/site-packages/* ./python/
cp -r ./layer/lib/python3.6/site-packages/.libs_cffi_backend/* ./python/.libs_cffi_backend/
zip -r autoscale_layer.zip ./python
```

The resultant ***autoscale_layer.zip*** file should be copied to the *lambda-python-files* folder.

KMS Master Key

This is required if the ASAv passwords are in encrypted format. Otherwise this component is not required. Passwords should be encrypted using only the KMS provided here. If KMS ARN is entered on CFT, then passwords have to be encrypted. Otherwise passwords should be plain text.

For more information about master keys and encryption, see the AWS document Creating keys and the AWS CLI Command Reference about password encryption and KMS.

Example:

```
$ aws kms encrypt --key-id <KMS-ARN> --plaintext 'MyC0mplIc@tedProtect1oN'
{
    "KeyId": "KMS-ARN",
    "CiphertextBlob":
"AQICAHgcQFAGtz/hvaxMtJvY/x/rfHnKI3clFPpSXUU7HQRnCAFwfXhXHJAHL8tcVmDqurALAAAAajBoBgkqhki
G9w0BBwagWzBZAgEAMFQGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQM45AIkTqjSekX2mniAgEQgCcOav6Hhol
+wxpWKtXY4y1Z1d0z1P4fx0jTdosfCbPnUExmNJ4zdx8="
}
$
```

The value of *CiphertextBlob* key should be used as a password.

Python 3 Environment

A ***make.py*** file can be found in the cloned repository top directory. This will Zip the python files into a Zip file and copy to a target folder. In order to do these tasks, the Python 3 environment should be available.

Auto Scale Deployment

Preparation

It is expected that the Application is either deployed or its deployment plan is available.

Input Parameters

The following input parameters should be collected prior to deployment.

Table 2: Auto Scale Input Parameters

Parameter	Allowed Values/Type	Description
PodNumber	String Allowed Pattern: '^\d{1,3}$'	This is the pod number. This will be suffixed to the Auto Scale Group name (ASAv-Group-Name). For example, if this value is '1', then the group name will be *ASAv-Group-Name-1*. It should be at least 1 numerical digit but not more than 3 digits. Default: 1
AutoscaleGrpNamePrefix	String	This is the Auto Scale Group Name Prefix. The pod number will be added as a suffix. Maximum: 18 characters Example: Cisco-ASAv-1
NotifyEmailID	String	Auto Scale events will be sent to this email address. You need to accept a subscription email request. Example: admin@company.com

Parameter	Allowed Values/Type	Description
VpcId	String	The VPC ID in which the device needs to be deployed. This should be configured as per AWS requirements. Type: AWS::EC2::VPC::Id If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
LambdaSubnets	List	The subnets where Lambda functions will be deployed. Type: List<AWS::EC2::Subnet::Id> If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
LambdaSG	List	The Security Groups for Lambda functions. Type: List<AWS::EC2::SecurityGroup::Id> If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
S3BktName	String	The S3 bucket name for files. This should be configured in your account as per AWS requirements. If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
LoadBalancerType	String	The type of Internet-facing Load Balancer, either “application” or “network”. Example: application
LoadBalancerSG	String	The Security Groups for the Load Balancer. In the case of a network load balancer, it won't be used. But you should provide a Security Group ID. Type: List<AWS::EC2::SecurityGroup::Id> If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
LoadBalancerPort	Integer	The Load Balancer port. This port will be opened on LB with either HTTP/HTTPS or TCP/TLS as the protocol, based on the chosen Load Balancer type. Make sure the port is a valid TCP port, it will be used to create the Load Balancer listener. Default: 80

Parameter	Allowed Values/Type	Description
SSLcertificate	String	The ARN for the SSL certificate for secured port connections. If not specified, a port opened on the Load Balancer will be TCP/HTTP. If specified, a port opened on the Load Balancer will be TLS/HTTPS.
TgHealthPort	Integer	This port is used by the Target group for health probes. Health probes arriving at this port on ASAv will be routed to the AWS Metadata server and should not be used for traffic. It should be a valid TCP port. If you want your application itself to reply to health probes, then accordingly NAT rules can be changed for the ASAv. In such a case, if the application does not respond, the ASAv will be marked as unhealthy and deleted due to the Unhealthy instance threshold alarm. Example: 8080
AssignPublicIP	Boolean	If selected as "true" then a public IP will be assigned. In case of a BYOL-type ASAv, this is required to connect to https://tools.cisco.com. Example: TRUE
ASAvInstanceType	String	The Amazon Machine Image (AMI) supports different instance types, which determine the size of the instance and the required amount of memory. Only AMI instance types that support ASAv should be used. See the ASA Release Notes. Example: c4.2xlarge
ASAvLicenseType	String	The ASAv license type, either BYOL or PAYG. Make sure the related AMI ID is of the same licensing type. Example: BYOL
ASAvAmiId	String	The ASAv AMI ID (a valid Cisco ASAv AMI ID). Type: AWS::EC2::Image::Id Please choose the correct AMI ID as per the region and desired version of the image.

Parameter	Allowed Values/Type	Description
ConfigFileURL	String	The HTTP URL for ASAv configuration files. Configuration files for each AZs should be available in the URL. The Lambda function will take care of choosing the correct file. You can deploy an HTTP server to host configuration files, or you can use the AWS S3 static web-hosting facility. **Note** The last "/" is also needed as configuration file names will be appeneded to the URL at the time of import. If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value. Example: https://myserver/asavconfig/asaconfig .txt/
NoOfAZs	Integer	The number of availability zones that ASAv should span across, between 1 and 3. In the case of an ALB deployment, the minimum value is 2, as required by AWS. Example: 2
ListOfAzs	Comma separated string	A comma-separated list of zones in order. **Note** The order in which these are listed matters. Subnet lists should be given in the same order. If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value. Example: us-east-1a, us-east-1b, us-east-1c
ASAvMgmtSubnetId	Comma separated list	A comma-separated list of management subnet-ids. The list should be in the same order as the corresponding availability zones. Type: List<AWS::EC2::SecurityGroup::Id> If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
ASAvInsideSubnetId	Comma separated list	A comma-separated list of inside/Gig0/0 subnet-ids. The list should be in the same order as the corresponding availability zones. Type: List<AWS::EC2::SecurityGroup::Id> If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.

Parameter	Allowed Values/Type	Description
ASAvOutsideSubnetId	Comma separated list	A comma-separated list of outside/Gig0/1 subnet-ids. The list should be in the same order as the corresponding availability zones. Type: List<AWS::EC2::SecurityGroup::Id> If the "*infrastructure.yaml*" file is used to deploy the infrastructure, the output section of the stack will have this value. Please use that value.
KmsArn	String	The ARN of an existing KMS (AWS KMS key to encrypt at rest). If specified, the ASAv passwords should be encrypted. The password encryption should be done using only the specified ARN. Generating Encrypted Password Example: " aws kms encrypt --key-id <KMS ARN> --plaintext <password> ". Please used such generated passwords as shown. Example: arn:aws:kms:us-east-1:[AWS Account]:key/7d586a25-5875-43b1-bb68-a452e2f6468e
CpuThresholds	Comma separated integers	The lower CPU threshold and the upper CPU threshold. The minimum value is 0 and maximum value is 99. Defaults: 10, 70 Please note that the lower threshold should be less than the upper threshold. Example: 30,70

Update the ASA Configuration Files

You prepare ASA configuration files and store them in a http/https server accessible by an ASAv instance. This is a standard ASA configuration file format. A scaled-out ASAv will download a configuration file and update its configuration.

The following sections provide examples on how the ASA configuration file could be modified for the Auto Scale solution.

Objects, Device Groups, NAT Rules, and Access Policies

See the following for an example of objects, route, and NAT rules for the Load Balancer health probes for the ASAv configuration.

```
! Load Balancer Health probe Configuration
object network aws-metadata-server
host 169.254.169.254
object service aws-health-port
service tcp destination eq 7777
object service aws-metadata-http-port
service tcp destination eq 80
route inside 169.254.169.254 255.255.255.255 10.0.100.1 1
```

```
nat (outside,inside) source static any interface destination static interface
aws-metadata-server service aws-health-port aws-metadata-http-port
!
```

Note The health probe connections above should be allowed on your access policy.

See the following for an example of the data-plane configuration for an ASAv configuration.

```
! Data Plane Configuration
route inside 10.0.0.0 255.255.0.0 10.0.100.1 1
object network http-server-80
host 10.0.50.40
object network file-server-8000
host 10.0.51.27
object service http-server-80-port
service tcp destination eq 80
nat (outside,inside) source static any interface destination static interface http-server-80
 service http-server-80-port http-server-80-port
object service file-server-8000-port
service tcp destination eq 8000
nat (outside,inside) source static any interface destination static interface file-server-8000
 service file-server-8000-port file-server-8000-port
object service https-server-443-port
service tcp destination eq 443
nat (outside,inside) source static any interface destination static interface http-server-80
 service https-server-443-port http-server-80-port
!
```

Configuration File Updates

The ASAv configuration should be updated in the *az1-connfiguration.txt*, *az2-configuration.txt*, and *az3-configuration.txt* files.

Note Having three configuration files allows you to modify the configuration based on the Availability Zone (AZ). For example, the static route to the aws-metadata-server will have a different gateway in each AZ.

Template Updates

The *deploy_autoscale.yaml* template should be modified carefully. You should modify the *UserData* field of the ***LaunchTemplate***. The *UserData* can be updated as needed. The *name-server* should be updated accordingly; for example, it can be the VPC DNS IP. Where your licensing is BYOL, the licensing *idtoken* should be shared here.

```
!
dns domain-lookup management
DNS server-group DefaultDNS
name-server <VPC DNS IP>
!
! License configuration
        call-home
        profile License
        destination transport-method http
        destination address http <url>
        license smart
```

```
feature tier standard
throughput level <entitlement>
license smart register idtoken <token>
```

Upload Files to Amazon Simple Storage Service (S3)

All the files in the *target* directory should be uploaded to the Amazon S3 bucket. Optionally, you can use the CLI to upload all of the files in the *target* directory to the Amazon S3 bucket.

```
$ cd ./target
$ aws s3 cp . s3://<bucket-name> --recursive
```

Deploy Stack

After all of the prerequisites are completed for deployment, you can create the AWS CloudFormation stack.

Use the *deploy_autoscale.yaml* file in the *target* directory.

Provide the parameters as collected in Input Parameters, on page 7.

Validate Deployments

Once the template deployment is successful, you should validate that the Lambda functions and the CloudWatch events are created. By default, the Auto Scale Group has the minimum and maximum number of instances as zero. You should edit the Auto Scale group in the AWS EC2 console with how many instances you want. This will trigger the new ASAv instances.

We recommend that you launch only one instance and check its workflow and validate its behavior as to whether it is working as expected. Post that actual requirements of the ASAv can be deployed, they can also be verified for the behavior. The minimum number of ASAv instances can be marked as Scale-In protected to avoid their removal by AWS Scaling policies.

Auto Scale Maintenance Tasks

Scaling Processes

This topic explains how to suspend and then resume one or more of the scaling processes for your Auto Scale group.

Start and Stop Scale Actions

To start and stop scale out/in actions, follow these steps.

- For AWS Dynamic Scaling—Refer to the following link for information to enable or disable scale out actions:

  Suspending and Resuming Scaling Processes

Health Monitor

Every 60 minutes, a CloudWatch Cron job triggers the Auto Scale Manager Lambda for the Health Doctor module:

- If there are unhealthy IPs which belong to a valid ASAv VM, that instance gets deleted if the ASAv is more than an hour old.
- If those IPs are not from a valid ASAv VM, then only IPs are removed from the Target Group.

Disable Health Monitor

To disable a health monitor, in ***constant.py*** make the constant as “True”.

Enable Health Monitor

To enable a health monitor, in ***constant.py*** make the constant as “False”.

Disable Lifecycle Hooks

In the unlikely event that Lifecycle hook needs to be disabled, if disabled it won’t add additional interfaces to Instances. It can also cause a series of failed deployment of ASAv instances.

Disable Auto Scale Manager

To disable Auto Scale Manager, respective CloudWatch Events “notify-instance-launch” and “notify-instance-terminate” should be disabled. Disabling this won’t trigger Lambda for any new events. But already executing Lambda actions will continue. There is no abrupt stop of Auto Scale Manager. Trying abrupt stopping by stack deletion or deleting resources can cause an indefinite state.

Load Balancer Targets

Because the AWS Load Balancer does not allow instance-type targets for instances having more than one network interface, the Gigabit0/1 interface IP is configured as a target on Target Groups. As of now however, the AWS Auto Scale health checks work only for instance-type targets, not IPs. Also, these IPs are not automatically added or removed from target groups. Hence our Auto Scale solution programmatically handles both of these tasks. But in the case of maintenance or troubleshooting, there could be a situation demanding manual effort to do so.

Register a Target to a Target Group

To register an ASAv instance to the Load Balancer, its Gigabit0/1 instance IP (outside subnet) should be added as a target in Target Group(s). See Register or Deregister Targets by IP Address.

Deregister a Target from a Target Group

To deregister an ASAv instance to the Load Balancer, its Gigabit0/1 instance IP (outside subnet) should be deleted as a target in Target Group(s). See Register or Deregister Targets by IP Address.

Instance Stand-by

AWS does not allow instance reboot in the Auto Scale group, but it does allow a user to put an instance in Stand-by and perform such actions. However, this works best when the Load Balancer targets are instance-type. However, ASAv VMs cannot be configured as instance-type targets, because of multiple network interfaces.

Put an Instance in Stand-by

If an instance is put into stand-by, its IP in Target Groups will still continue to be in the same state until the health probes fail. Because of this, it is recommended to deregister respective IPs from the Target Group before putting the instance into stand-by state; see Deregister a Target from a Target Group, on page 14 for more information.

Once the IPs are removed, see Temporarily Removing Instances from Your Auto Scaling Group.

Remove an Instance from Stand-by

Similarly you can move an instance from stand-by to running state. After removal from stand-by state, the instance's IP should be registered to Target Group targets. See Register a Target to a Target Group, on page 14.

For more information about how to put instances into stand-by state for troubleshooting or maintenance, see the AWS News Blog.

Remove/Detach Instance from Auto Scale Group

To remove an instance from the Auto Scale group, first it should be moved to stand-by state. See "Put Instances on Stand-by". Once the instance is in the stand-by state it can be removed or detached. See Detach EC2 Instances from Your Auto Scaling Group.

Terminate an Instance

To terminate an instance it should be put into stand-by state; see Instance Stand-by, on page 15. Once the instance is in stand-by, you can proceed to terminate.

Instance Scale-In Protection

To avoid an accidental removal of any particular instance from the Auto Scale group, it can be made as Scale-In protected. If an instance is Scale-In protected, it won't be terminated due to a Scale-In event.

Please refer to the following link to put an instance into Scale-In protected state.

https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-instance-termination.html

Important It is recommended to make the minimum number of instances which are healthy (the target IP should be healthy, not just the EC2 instance) as Scale-In protected.

Change Credentials and Registration IDs

Any changes in configuration won't be automatically reflected on already running instances. Changes will be reflected on upcoming devices only. Any such changes should be manually pushed to already existing devices.

Change the ASAv Admin Password

A change to the ASAv password requires the user to change it on each device manually for running instances. For new ASAv devices to be onboarded, the ASAv password will be taken from the Lambda environment variables. See Using AWS Lambda Environment Variables.

Changes to AWS Resources

You can change many things in AWS post deployment, such as the Auto Scale Group, Launch Configuration, CloudWatch events, Scaling Policies etc. You can import your resources into a CloudFormation stack or create a new stack from your existing resources.

See Bringing Existing Resources Into CloudFormation Management for more information about how to manage changes performed on AWS resources.

Collect and Analyze CloudWatch Logs

In order to export CloudWatch logs please refer to Export Log Data to Amazon S3 Using the AWS CLI.

Auto Scale Troubleshooting and Debugging

AWS CloudFormation Console

You can verify the input parameters to your CloudFormation stack in the AWS CloudFormation Console, which allows you to create, monitor, update and delete stacks directly from your web browser.

Navigate to the required stack and check the parameter tab. You can also check inputs to Lambda Functions on the Lambda Functions environment variables tab.

To learn more about the AWS CloudFormation console, see the *AWS CloudFormation User Guide*.

Amazon CloudWatch Logs

You can view logs of individual Lambda functions. AWS Lambda automatically monitors Lambda functions on your behalf, reporting metrics through Amazon CloudWatch. To help you troubleshoot failures in a function, Lambda logs all requests handled by your function and also automatically stores logs generated by your code through Amazon CloudWatch Logs.

You can view logs for Lambda by using the Lambda console, the CloudWatch console, the AWS CLI, or the CloudWatch API. To learn more about log groups and accessing them through the CloudWatch console, see the Monitoring system, application, and custom log files in the *Amazon CloudWatch User Guide*.

Load Balancer Health Check Failure

The load balancer health check contains information such as the protocol, ping port, ping path, response timeout, and health check interval. An instance is considered healthy if it returns a 200 response code within the health check interval.

If the current state of some or all your instances is `OutOfService` and the description field displays the message that the `Instance has failed at least the Unhealthy Threshold number of health checks consecutively`, the instances have failed the load balancer health check.

You should check the health probe NAT rule in the ASA configuration. For more information, see Troubleshoot a Classic Load Balancer: Health checks.

Traffic Issues

To troubleshoot traffic issues with your ASAv instances, you should check the Load Balancer rules, the NAT rules, and the static routes configured in the ASAv instances.

You should also check the AWS virtual network/subnets/gateway details provided in the deployment template, including security group rules, etc. You can also refer to AWS documentation, for example, Troubleshooting EC2 instances.

ASAv Failed to Configure

If the ASAv fails to configure, you should check the connectivity to the Amazon S3 static HTTP webserver hosting configuration. See Hosting a static website on Amazon S3 for more information.

ASAv Failed to License

If the ASAv fails to license, you should check the connectivity to the CSSM server, check the ASAv Security Group configuration, and check the Access Control Lists.

Unable to SSH into the ASAv

If you are unable to SSH into the ASAv, check to see if the complex password was passed to ASAv via the template.