AppSentinels.ai
Application Security Re-invented

AppSentinels API Security Platform

Lambda Layer Deployment

Revision	Date Modified	Author	Comments
1.0	14-Aug-24	Arun	Initial spec for lambda layer deployment
1.1	05-Nov-24	Arun	Removed redundancy IAM permissions
1.2	16-jan-26	Sagar	Added Docker based lambda application deployment

Contents

AppSentinels Layers for AWS Lambda Functions 4
Deploying Layers 4
Deploying Layers Manually 4
Update Layers 5
Automated Deployment of Layers 5
Deployment Of Layers for Docker Based Lambda Application 5
Example of Docker Based Application Integration 6
Manage Layers 7
Roles and permission requirements 8
Layers configuration 8

AppSentinels Layers for AWS Lambda Functions

This document covers the steps to deploy the AppSentinels wrapper with AWS Lambda function. AppSentinels wrapper captures the HTTP request and response headers and payload and forwards that to AppSentinels OnPrem Controller. AppSentinels wrapper consists of two components - a tech stack dependent component, called *appsentinels-wrapper*, and an external lambda extension, called *appsentinels-log-server*. Following points summarize the operations done by the Appsentinels wrapper:

- The *appsentinels-wrapper* wraps the lambda function calls to capture HTTP request and response. It depends on the tech stack used by the lambda function - python wrapper is used if lambda function is implemented using python and a Nodejs wrapper is required if the lambda function is implemented using Nodejs.
- AWS provides the wrapper mechanism with the environment variable *AWS_LAMBDA_EXEC_WRAPPER*, which can be used to define the wrapper function that is invoked instead of the actual lambda hander.
- The *appsentinels-wrapper* bundles the HTTP request and response headers and payload in the form of a message and sends that message over UDP or TCP to *appsentinels-log-server*, running as lambda extension running in the same environment.
- The log server pushes the message received in a queue.
- A concurrent task (Go routine) picks the message from queue, converts that into a format expected by the controller and post that to controller.

Please note

- A layer version is always accessed using layer ARN.
- Contact support@appsentinels.ai for the latest ARN and the regions in which it is available.
- Current ARN:
    - arn:aws:lambda:ap-south-1:488922454646:layer:appsentinels-wrapper:53
    - arn:aws:lambda:ap-south-1:488922454646:layer:appsentinels-log-server:59

Deploying Layers

Deploying Layers Manually

Please execute following steps for both *appsentinels-wrapper* and *appsentinels-log-server*. After running these steps for both layers, lambda function should display both layers.

- Go to *Lambda > Functions > YourLambdaFunction*.
- Scroll down and click on **Add a layer**.
- Select **Specify an ARN**.
- Specify the ARN of the published AppSentinels layer (*appsentinels-wrapper* and *appsentinels-log-server*).
- Click on **Add**.

Update Layers

If a newer version of AppSentinels wrapper (*appsentinels-wrapper* or *appsentinels-log-server*) is available, following steps can be used to upgrade to the newer version of that component.

- Go to *Lambda > Functions > YourLambdaFunction*.
- Click on **Edit Layers**.
- Check the **Layer version** displayed along with the AppSentinels layer (*appsentinels-wrapper* or *appsentinels-log-server*).
- Click on the version number and select the new version to deploy from the drop-down.
- Click on **Save**.

Automated Deployment of Layers

- Customer can deploy AppSentinels wrapper on single or all Lambda function using the script deploy_wrapper.sh https://sample-config.appsentinels.ai/appsentinels-deployment/aws-lambda/deploy_wrapper.sh
- This script requires following command line arguments:
    - AppSentinels wrapper layer appsentinels-wrapper ARN
    - AppSentinels log-server layer appsentinels-log-server ARN
    - Region (AppSentinels has to publish the layers in this region before customer can deploy it).
    - Lambda Function: all or name of the lambda function.
        - all: deploy layer on all the Lambda function in the given account in the given region.
        - lambda function name: deploy layer on specific lambda function with given name.
    - Controller URL (AppSentinels edge controller URL).
- Example: Deploy lambda wrapper on testHttpServer function.
  ./deploy_wrapper.sh "arn:aws:lambda:ap-south-1:488922454646:layer:appsentinels-wrapper:53" "arn:aws:lambda:ap-south-1:488922454646:layer:appsentinels-log-server:59" ap-south-1 testHttpServer Appsentinels-Controller-NLB-14d455ada1d88f50.elb.ap-south-1.amazonaws.com

Deployment Of Layers for Docker Based Lambda Application

- To deploy the appsentinels-wrapper and appsentinels-log-server for the docker based lambda application Please follow the below steps:
    - Download and place **go_extension.zip** and **python_wrapper.zip** from below location in the same directory where the application **Dockerfile** is present
        - https://downloads.appsentinels.ai/appsentinels-deployment/aws-lambda/go_extension.zip
        - https://downloads.appsentinels.ai/appsentinels-deployment/aws-lambda/python_wrapper.zip
    - Add the below section in lambda application Dockerfile before actual application application build stage:

-

```
FROM alpine:latest AS extract
RUN apk add --no-cache unzip

# Copy your zip files into the build context
COPY go_extension.zip /tmp/
COPY python_wrapper.zip /tmp/

# Unzip them into a temporary opt structure
RUN mkdir -p /opt/extensions && \
unzip /tmp/go_extension.zip -d /opt/ && \
unzip /tmp/python_wrapper.zip -d /opt/
```

- Build the Application and deploy the image to aws lambda

Example of Docker Based Application Integration

- Below is the actual lambda application Dockerfile snapshot

```
FROM public.ecr.aws/lambda/python:3.10

# Copy requirements.txt
COPY requirements.txt ${LAMBDA_TASK_ROOT}

# Install dependencies
RUN pip install -r requirements.txt

# Copy the extracted layer content from the first stage
COPY --from=extract /opt /opt

# Copy your application code
COPY app.py ${LAMBDA_TASK_ROOT}

# Set the CMD to your handler
CMD [ "app.handler" ]
```

- After adding the AppSentinels block to add the layer


```
# --- STEP 1:  AppSentinels STAGE (The "Layer" Stage) ---
# Stage 1: Unzip the layers
FROM alpine:latest AS extract
RUN apk add --no-cache unzip

# Copy your zip files into the build context
COPY go_extension.zip /tmp/
COPY python_wrapper.zip /tmp/

# Unzip them into a temporary opt structure
RUN mkdir -p /opt/extensions && \
    unzip /tmp/go_extension.zip -d /opt/ && \
    unzip /tmp/python_wrapper.zip -d /opt/


# --- STEP 2: Customer STAGE ---
# Stage 2: Build the final Lambda image

FROM public.ecr.aws/lambda/python:3.10

# Copy requirements.txt
COPY requirements.txt ${LAMBDA_TASK_ROOT}

# Install dependencies
RUN pip install -r requirements.txt

# Copy the extracted layer content from the first stage
COPY --from=extract /opt /opt

# Copy your application code
COPY app.py ${LAMBDA_TASK_ROOT}

# Set the CMD to your handler
CMD [ "app.handler" ]
```

Manage Layers

- The script manage_extensions.sh https://sample-config.appsentinels.ai/appsentinels-deployment/aws-lambda/manage_extensions.sh can be used to perform following operations:
    - List all the extension or layer across all Lambda functions.
    - List all functions where an extension or layer with given ARN is deployed.
    - Remove the extension or layer with given ARN from all the functions.
- Please note that appsentinels-wrapper and appsentinels-log-server are separate layers and need to be managed separately, so this command needs to be executed separately to manage the appsentinels-wrapper and appsentinels-log-server layers.
- This script can be used to disable all instances of AppSentinels wrapper by running the command twice, once for each layer.

Roles and permission requirements

- Lambda permissions
  - lambda:PublishLayerVersion
  - lambda:AddLayerVersionPermission
  - lambda:GetLayerVersion
  - lambda:UpdateFunctionConfiguration
  - lambda:GetFunction
  - lambda:DeleteLayerVersion
  - lambda:GetFunctionConfiguration
- Lambda function CloudWatch logs
  - logs:CreateLogGroup
  - logs:CreateLogStream
  - logs:PutLogEvents
  - logs:DescribeLogStreams
  - logs:GetLogEvents

Layers configuration

Following table lists the environment variables used to configure the AppSentinels extension (mandatory marked in green)

Environment Variable	**Description**
AS_EXTENSION_LOG_LEVEL	Log level - Debug/Info/Warn/Error/Fatal, default Info
AS_SERVER_ADDR	Address of TCP/UDP log server (default 127.0.0.1)
AS_SERVER_PORT	Log server port number (default 9015)
AS_SERVER_PROTOCOL	Transport protocol to post messages to log server (udp)
AS_LOGGING_MODE	Logging mode - default/metadata/nologging
AS_CONTROLLER_ENDPOINT	AppSentinels Controller URL or IP address
AS_CONTROLLER_PORT	AppSentinels Controller port number
AS_MAX_BATCH_SIZE	Maximum number of logs in a batch (default 16)
AS_MAX_BODY_SIZE	Maximum request/response body size sent in the logs (60K)
AS_RELAY_PROTOCOL	Protocol use to relay the logs (http/https)
AS_SKIP_METHODS	HTTP methods skipped (default HEAD and OPTIONS)

Environment Variable	Description
AS_SKIP_URIS	Comma separated list of URIs to be skipped
AS_CONTENT_TYPE_REGEX	Regex defining content types process (json/xml/graphql/form)
AS_SERVER_CERT_VERIFY	Enable server certificate verification in HTTPS (default: no)

- Please note that AS_CONTROLLER_ENDPOINT is mandatory configuration parameter which needs to be specified during extension deployment.
- AS_LOGGING_MODE is set automatically based on the configuration.