

Week 4 Sunday – Taibah Valley – Task 7 – Lambda Function
S. Y. Al-Kafrawi
28th of June 2020

1 – Creating a lambda function [Choosing the hello-world-python blueprint]

Create function [Info](#)

Choose one of the following options to create your function.

Author from scratch ☐

Start with a simple Hello World example.

Use a blueprint ☒

Build a Lambda application from sample code and configuration presets for common use cases.

Browse serverless app repository ☐

Deploy a sample Lambda application from the AWS Serverless Application Repository.

Blueprints [Info](#) Export

? < 1 >

Keyword : ⊕

hello-world ☐

A starter AWS Lambda function.

nodejs

hello-world-python ☒

A starter AWS Lambda function.

python3.7

Cancel Configure

2 - Creating a new role from AWS policy templates. Choosing a name for the role. From Policy templates, choosing Amazon SNS publish policy.

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☐ Use an existing role

☒ Create a new role from AWS policy templates

Role name

Enter a name for your new role.

Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - optional [Info](#)

Choose one or more policy templates.

▼ ↺

Amazon SNS publish policy ✕

SNS

3 – Applying the provided example code

```
from __future__ import print_function

import json
import boto3

print('Loading function')

def lambda_handler(event, context):

    # Parse the JSON message
    eventText = json.dumps(event)

    # Print the parsed JSON message to the console. You can view this text in the Monitoring
    # tab in the AWS Lambda console or in the Amazon CloudWatch Logs console.
    print('Received event: ', eventText)

    # Create an SNS client
    sns = boto3.client('sns')

    # Publish a message to the specified topic
    response = sns.publish (
        TopicArn = 'arn:aws:iam::123456789012:My_IoT_SNS_Topic',
        Message = eventText
    )

    print(response)
```

The code highlighted in red will be changed

4 – Creating an SNS topic

Create topic

Details

Name

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional*

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. [Info](#)

Maximum 100 characters, including hyphens (-) and underscores (_).

5 – Subscribing to an SNS topic and choosing Email as a service

Details

Topic ARN

Protocol
The type of endpoint to subscribe

Endpoint
An email address that can receive notifications from Amazon SNS.

6 – Creating A rule

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

Description

9 – Entering In the Rule query statement field, [SELECT * FROM 'my/topic']

“ SELECT * specifies that you want to send the entire MQTT message that triggered the rule. FROM 'my/topic' is the topic filter. The rules engine uses the topic filter to determine which rules to trigger when an MQTT message is received.”

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * FROM 'my/topic'
```

10 – Configure the test event

Configure test event ✕

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☐ Create new test event
☒ Edit saved test events

Saved test event

Myevent ▼

↺

```
1 {  
2   "message" : "Hello, This is Salah testing AWS sns."  
3 }
```

11 – Receiving the test message via E-mail

