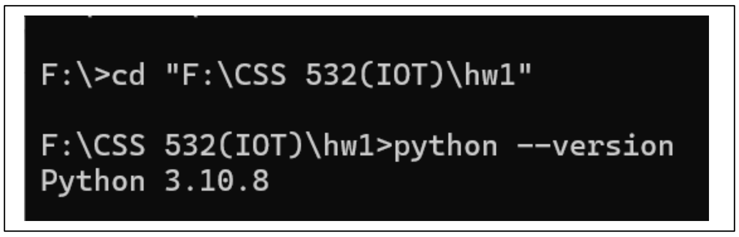
CSS 532 A (Internet of Things): HW 1

Name: Sahithi C
Student ID: 2303017

Online documentations referred:

1.) AWS Services documentation to install Boto 3(python SDK). (https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html)

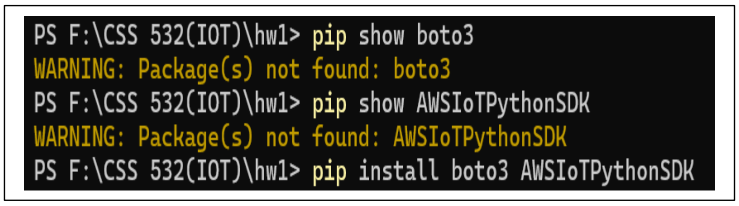
Step 1.) Install Python version greater than 3.8
Already have python version 3.10.8 installed in system.



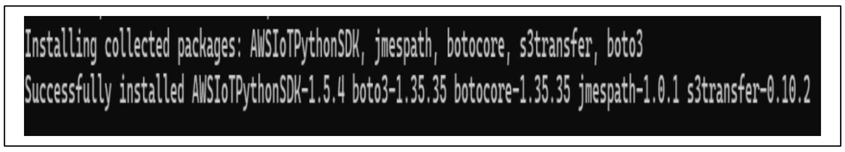
```
F:\>cd "F:\CSS 532(IOT)\hw1"

F:\CSS 532(IOT)\hw1>python --version
Python 3.10.8
```

Step 2.) Install boto3 & AWSIoTPythonSDK



```
PS F:\CSS 532(IOT)\hw1> pip show boto3
WARNING: Package(s) not found: boto3
PS F:\CSS 532(IOT)\hw1> pip show AWSIoTPythonSDK
WARNING: Package(s) not found: AWSIoTPythonSDK
PS F:\CSS 532(IOT)\hw1> pip install boto3 AWSIoTPythonSDK
```



```
Installing collected packages: AWSIoTPythonSDK, jmespath, botocore, s3transfer, boto3
Successfully installed AWSIoTPythonSDK-1.5.4 boto3-1.35.35 botocore-1.35.35 jmespath-1.0.1 s3transfer-0.10.2
```

Step3.) Created AWS free tier account for AWS IOT services

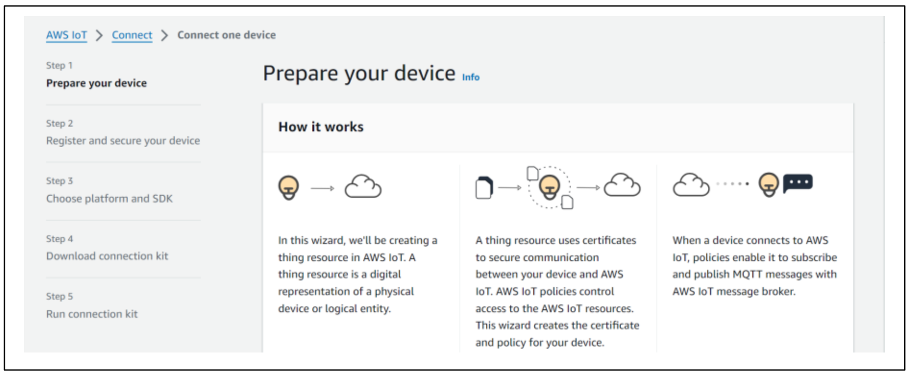
AWS Free Tier

Gain free, hands-on experience with the AWS products and services

Learn more about AWS Free Tier

Create a Free Account

Step4.) Go to Connect one device feature to create a thing resource in AWS IoT.



AWS IoT > Connect > Connect one device

Step 1
Prepare your device

Step 2
Register and secure your device

Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

Prepare your device Info

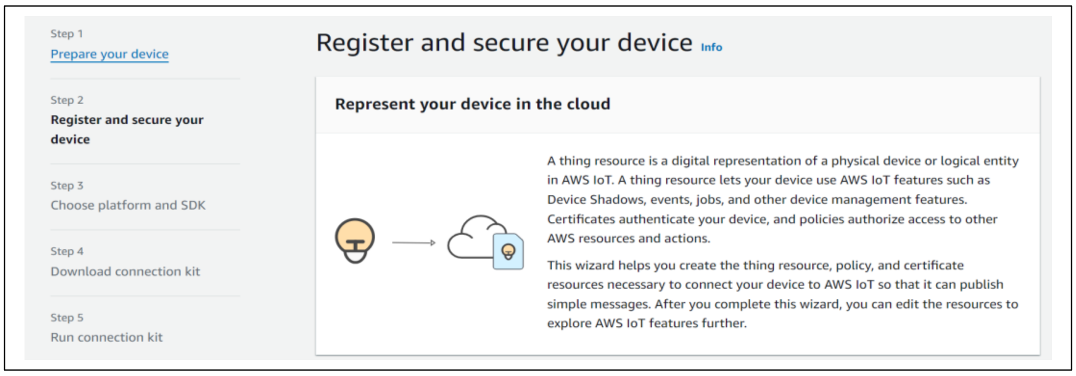
How it works

In this wizard, we'll be creating a thing resource in AWS IoT. A thing resource is a digital representation of a physical device or logical entity.

A thing resource uses certificates to secure communication between your device and AWS IoT. AWS IoT policies control access to the AWS IoT resources. This wizard creates the certificate and policy for your device.

When a device connects to AWS IoT, policies enable it to subscribe and publish MQTT messages with AWS IoT message broker.

Step5.) Created the thing resource, policy, and certificate resources necessary to connect my device to AWS IoT so that it can publish simple messages.



Step 1
Prepare your device

Step 2
Register and secure your device

Step 3
Choose platform and SDK

Step 4
Download connection kit

Step 5
Run connection kit

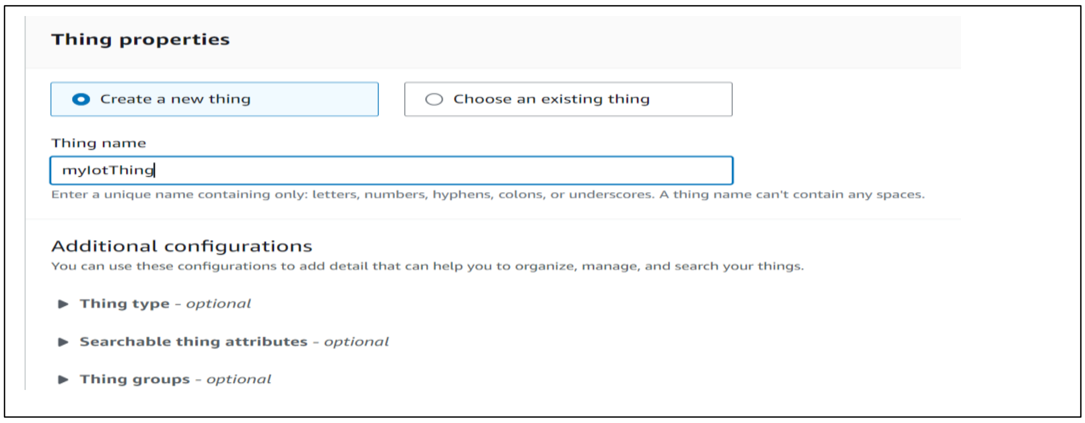
Register and secure your device Info

Represent your device in the cloud

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. A thing resource lets your device use AWS IoT features such as Device Shadows, events, jobs, and other device management features. Certificates authenticate your device, and policies authorize access to other AWS resources and actions.

This wizard helps you create the thing resource, policy, and certificate resources necessary to connect your device to AWS IoT so that it can publish simple messages. After you complete this wizard, you can edit the resources to explore AWS IoT features further.

6.) Created a new thing and named it as "**myIotThing**".



Thing properties

[✓ Create a new thing] Choose an existing thing

Thing name
myIotThing
Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

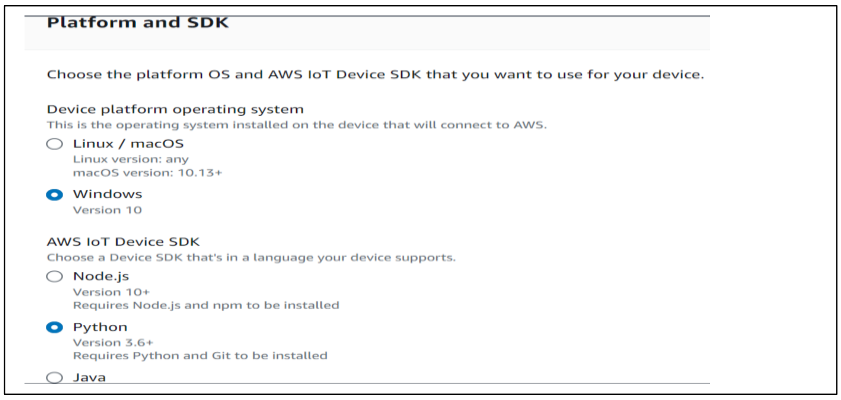
Additional configurations
You can use these configurations to add detail that can help you to organize, manage, and search your things.

▶ **Thing type** - *optional*

▶ **Searchable thing attributes** - *optional*

▶ **Thing groups** - *optional*

7.) I chose windows as OS platform and Python for AWS IoT device SDK for my device.



Platform and SDK

Choose the platform OS and AWS IoT Device SDK that you want to use for your device.

Device platform operating system
This is the operating system installed on the device that will connect to AWS.

Linux / macOS
Linux version: any
macOS version: 10.13+

[✓ Windows]
Version 10

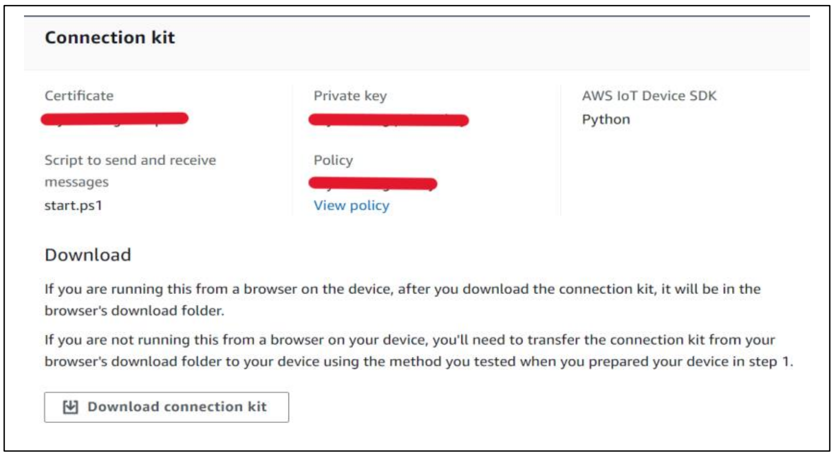
AWS IoT Device SDK
Choose a Device SDK that's in a language your device supports.

Node.js
Version 10+
Requires Node.js and npm to be installed

[✓ Python]
Version 3.6+
Requires Python and Git to be installed

Java

8.) Next, I downloaded the connection kit. Since, I am running this from a browser on my device, after downloading the connection kit was on the browser's downloaded folder. Then, unzipped the folder.



Connection kit

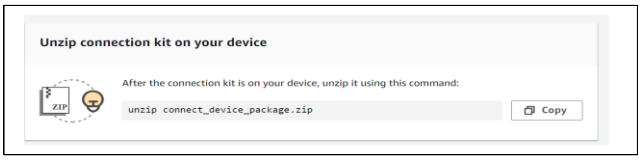
Certificate	Private key	AWS IoT Device SDK
[redacted]	[redacted]	Python
Script to send and receive messages	Policy	
start.ps1	[redacted] View policy	

Download

If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.

If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.

Download connection kit



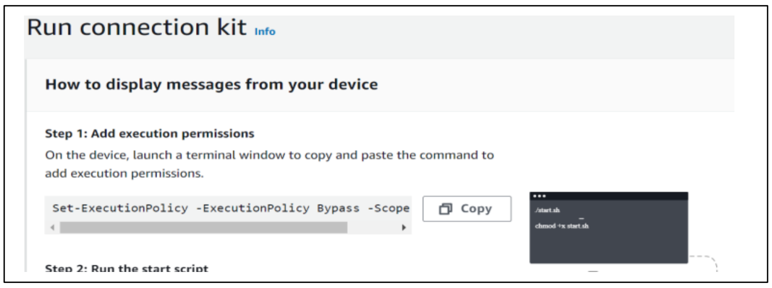
Unzip connection kit on your device

After the connection kit is on your device, unzip it using this command:

```
unzip connect_device_package.zip
```

Copy

9.) On the device, I opened the power shell and added execution permissions.



Run connection kit Info

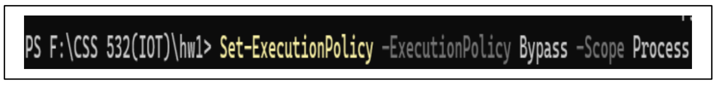
How to display messages from your device

Step 1: Add execution permissions

On the device, launch a terminal window to copy and paste the command to add execution permissions.

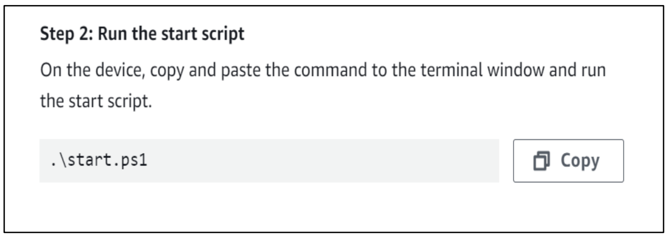
```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope
```

Step 2: Run the start script



```
PS F:\CSS 532(IOT)\hw1> Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process
```

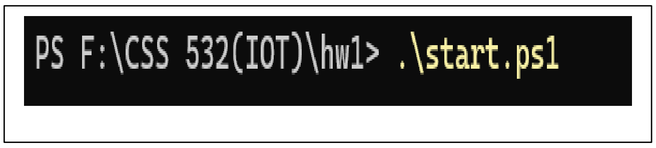
10.) Now, started running the script



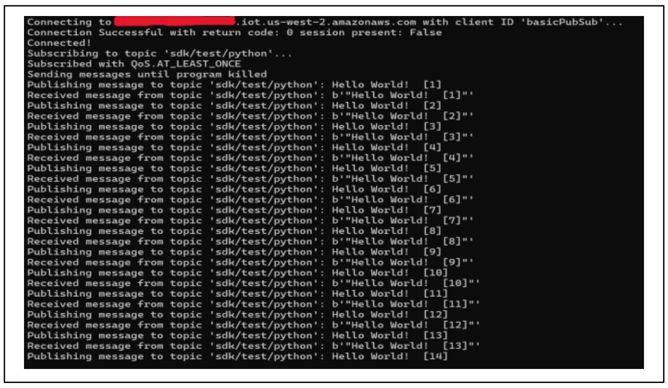
Step 2: Run the start script

On the device, copy and paste the command to the terminal window and run the start script.

```
.\start.ps1
```

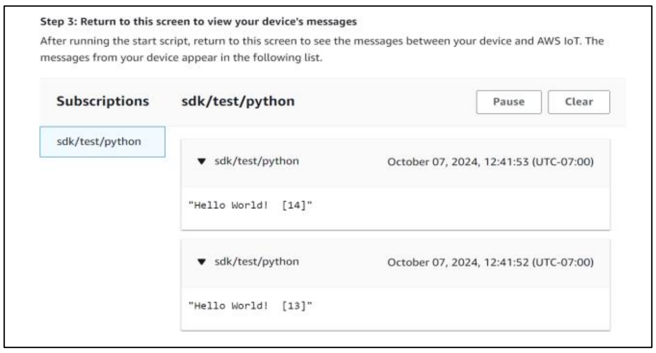


```
PS F:\CSS 532(IOT)\hw1> .\start.ps1
```



```
Connecting to [redacted].iot.us-west-2.amazonaws.com with client ID 'basicPubSub'...
Connection Successful with return code: 0 session present: False
Connected!
Subscribing to topic 'sdk/test/python'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Publishing message to topic 'sdk/test/python': Hello World!  [1]
Received message from topic 'sdk/test/python': b'"Hello World!  [1]"'
Publishing message to topic 'sdk/test/python': Hello World!  [2]
Received message from topic 'sdk/test/python': b'"Hello World!  [2]"'
Publishing message to topic 'sdk/test/python': Hello World!  [3]
Received message from topic 'sdk/test/python': b'"Hello World!  [3]"'
Publishing message to topic 'sdk/test/python': Hello World!  [4]
Received message from topic 'sdk/test/python': b'"Hello World!  [4]"'
Publishing message to topic 'sdk/test/python': Hello World!  [5]
Received message from topic 'sdk/test/python': b'"Hello World!  [5]"'
Publishing message to topic 'sdk/test/python': Hello World!  [6]
Received message from topic 'sdk/test/python': b'"Hello World!  [6]"'
Publishing message to topic 'sdk/test/python': Hello World!  [7]
Received message from topic 'sdk/test/python': b'"Hello World!  [7]"'
Publishing message to topic 'sdk/test/python': Hello World!  [8]
Received message from topic 'sdk/test/python': b'"Hello World!  [8]"'
Publishing message to topic 'sdk/test/python': Hello World!  [9]
Received message from topic 'sdk/test/python': b'"Hello World!  [9]"'
Publishing message to topic 'sdk/test/python': Hello World!  [10]
Received message from topic 'sdk/test/python': b'"Hello World!  [10]"'
Publishing message to topic 'sdk/test/python': Hello World!  [11]
Received message from topic 'sdk/test/python': b'"Hello World!  [11]"'
Publishing message to topic 'sdk/test/python': Hello World!  [12]
Received message from topic 'sdk/test/python': b'"Hello World!  [12]"'
Publishing message to topic 'sdk/test/python': Hello World!  [13]
Received message from topic 'sdk/test/python': b'"Hello World!  [13]"'
Publishing message to topic 'sdk/test/python': Hello World!  [14]
```

11.) After running the start script, I returned to the browser to check if messages from the device appear in the list.



Step 3: Return to this screen to view your device's messages

After running the start script, return to this screen to see the messages between your device and AWS IoT. The messages from your device appear in the following list.

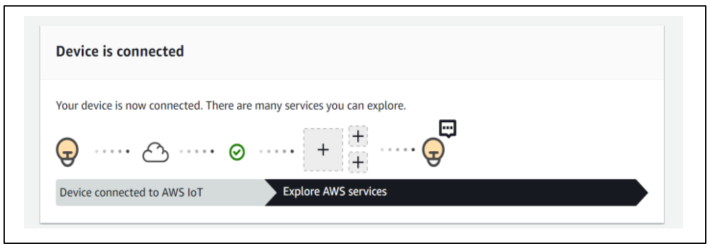
sdk/test/python October 07, 2024, 12:41:53 (UTC-07:00)

```
"Hello World!  [14]"
```

sdk/test/python October 07, 2024, 12:41:52 (UTC-07:00)

```
"Hello World!  [13]"
```

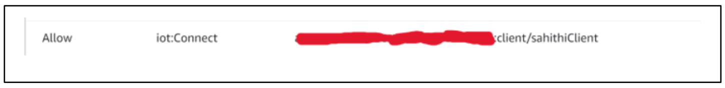
12.) Now, my device is successfully connected to AWS IoT.



Device is connected

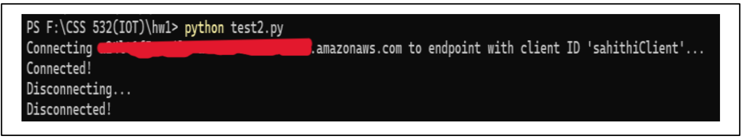
Your device is now connected. There are many services you can explore.

13.) Updated policies to allow/give permissions to connect to **my client Id**.



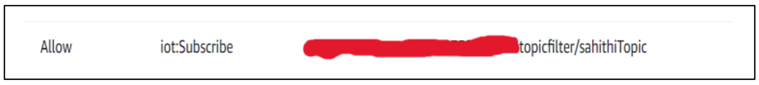
Allow	iot:Connect	[redacted]client/sahithiClient

14.) I successfully connected to AWS IoT by writing a Python script and executing it through the terminal.



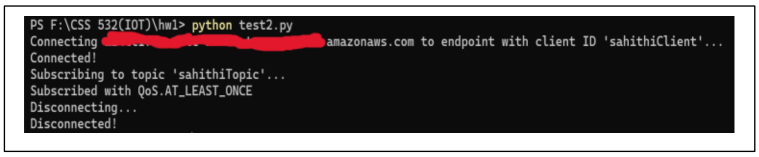
```
PS F:\CSS 532(IOT)\hw1> python test2.py
Connecting [redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Disconnecting...
Disconnected!
```

15.) Created a new policy for subscribing to "sahithiTopic".



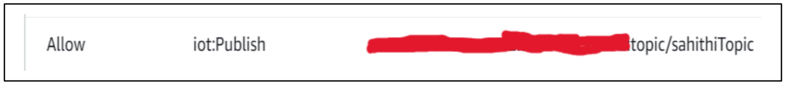
Allow	iot:Subscribe	[redacted]topicfilter/sahithiTopic

16.) I successfully subscribed to my topic by writing a Python script and executing it through the terminal.



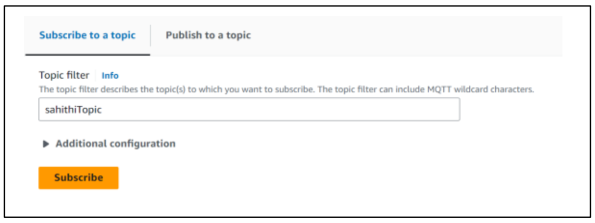
```
PS F:\CSS 532(IOT)\hw1> python test2.py
Connecting [redacted]amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Disconnecting...
Disconnected!
```

17.) Created a new policy for publishing to my topic.



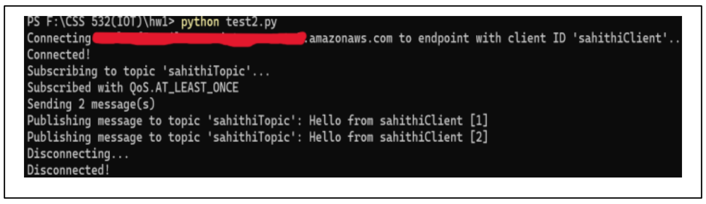
Allow   iot:Publish   [redacted]topic/sahithiTopic

Then, subscribed to a topic in AWS IoT to receive messages from console.

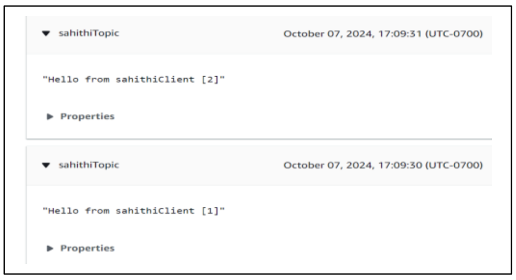


Topic filter Info
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.
sahithiTopic

Successfully published messages to the AWS IoT console by writing a Python script and executing it through the terminal and verified that the messages were reaching the AWS IoT console.



```
PS F:\CSS 532(IOT)\hw1> python test2.py
Connecting [redacted]amazonaws.com to endpoint with client ID 'sahithiClient'..
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [1]
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [2]
Disconnecting...
Disconnected!
```



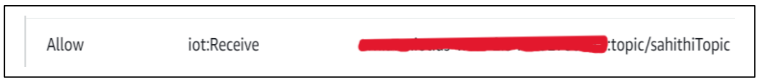
sahithiTopic   October 07, 2024, 17:09:31 (UTC-0700)

"Hello from sahithiClient [2]"

sahithiTopic   October 07, 2024, 17:09:30 (UTC-0700)

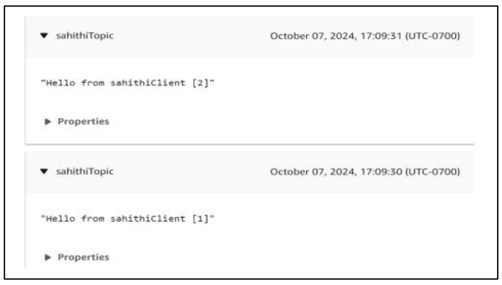
"Hello from sahithiClient [1]"

18.) Configured AWS IoT to automatically respond to messages sent from my device (laptop), and my device can display the received responses.
For this Initially, created a new policy to receive messages sent from AWS IoT



Allow   iot:Receive   [redacted]:topic/sahithiTopic

Next, I published the messages using the 'republish feature' provided by AWS IoT and verified that the messages were reaching my device terminal.

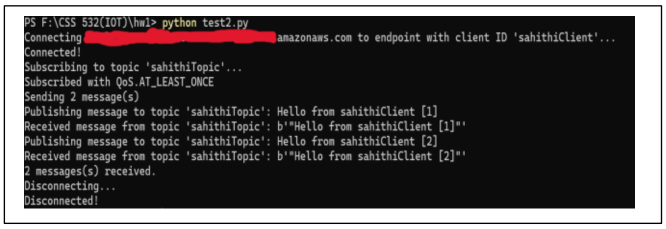


sahithiTopic   October 07, 2024, 17:09:31 (UTC-0700)

"Hello from sahithiClient [2]"

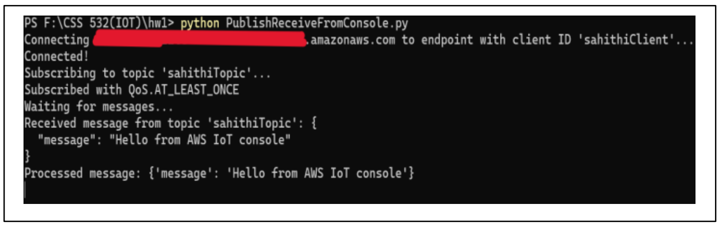
sahithiTopic   October 07, 2024, 17:09:30 (UTC-0700)

"Hello from sahithiClient [1]"

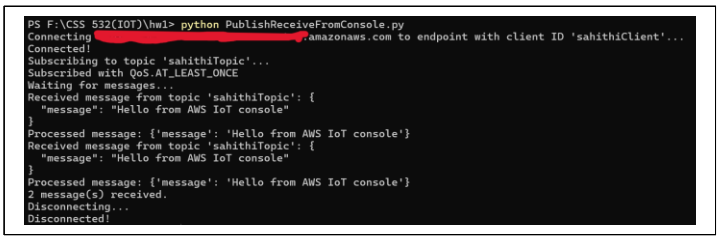


```
PS F:\CSS 532(IOT)\hw1> python test2.py
Connecting [redacted]amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [1]
Received message from topic 'sahithiTopic': b'"Hello from sahithiClient [1]"'
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [2]
Received message from topic 'sahithiTopic': b'"Hello from sahithiClient [2]"'
2 messages(s) received.
Disconnecting...
Disconnected!
```

19.) Configured AWS IoT to take user commands (input through cloud console) and send user commands to my device (laptop). For this, I wrote a Python script so that the AWS IoT takes user commands and send commands to my device, where my device processes the commands and receives it.



```
PS F:\CSS 532(IOT)\hw1> python PublishReceiveFromConsole.py
Connecting [redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for messages...
Received message from topic 'sahithiTopic': {
  "message": "Hello from AWS IoT console"
}
Processed message: {'message': 'Hello from AWS IoT console'}
```

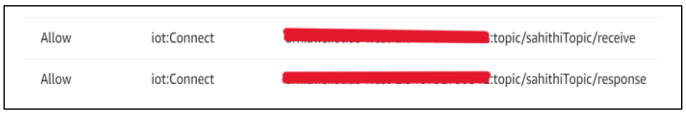


```
PS F:\CSS 532(IOT)\hw1> python PublishReceiveFromConsole.py
Connecting [redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for messages...
Received message from topic 'sahithiTopic': {
  "message": "Hello from AWS IoT console"
}
Processed message: {'message': 'Hello from AWS IoT console'}
Received message from topic 'sahithiTopic': {
  "message": "Hello from AWS IoT console"
}
Processed message: {'message': 'Hello from AWS IoT console'}
2 message(s) received.
Disconnecting...
Disconnected!
```

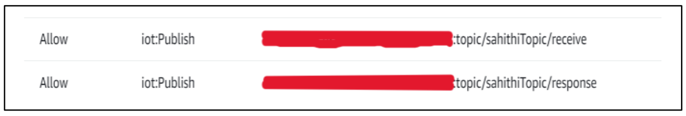
I wrote the code so that my terminal waits until it receives a specified number of messages, then disconnects from AWS IoT after reaching that count.

The, going further to implement my device to respond to the command by automatically sending some messages back to AWS, and the response from my device should be displayed in the AWS IoT console. I encountered an error where I had to create two topics for receiving and respond to messages.

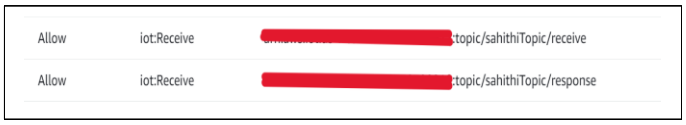
So, created new policies to connect, publish, receive and subscribe to "sahithiTopic/receive" and "sahithiTopic/response".



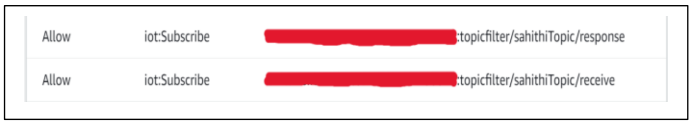
Allow	iot:Connect	[redacted]:topic/sahithiTopic/receive
Allow	iot:Connect	[redacted]:topic/sahithiTopic/response



Allow	iot:Publish	[redacted]:topic/sahithiTopic/receive
Allow	iot:Publish	[redacted]:topic/sahithiTopic/response

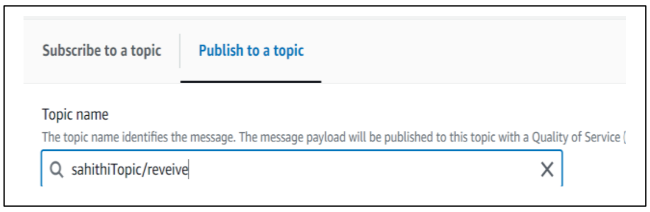


Allow	iot:Receive	[redacted]:topic/sahithiTopic/receive
Allow	iot:Receive	[redacted]:topic/sahithiTopic/response



Allow	iot:Subscribe	[redacted]:topicfilter/sahithiTopic/response
Allow	iot:Subscribe	[redacted]:topicfilter/sahithiTopic/receive

Then, added a topic to publish messages to the topic and subscribed to a topic to receive messages from the topic in AWS IoT.

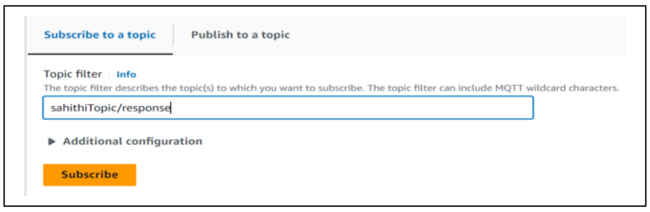


Subscribe to a topic | Publish to a topic

Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (

sahithiTopic/reveive



Subscribe to a topic | Publish to a topic

Topic filter Info

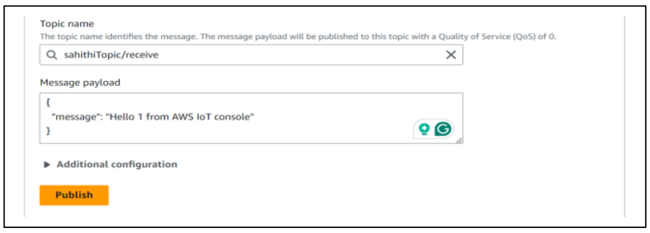
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

sahithiTopic/response

▶ Additional configuration

Subscribe

Published 1st message in AWS IoT console and verified that the messages were reaching the AWS IoT console.



Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

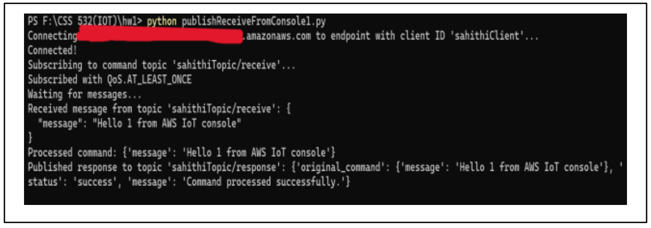
sahithiTopic/receive

Message payload

```
{
  "message": "Hello 1 from AWS IoT console"
}
```

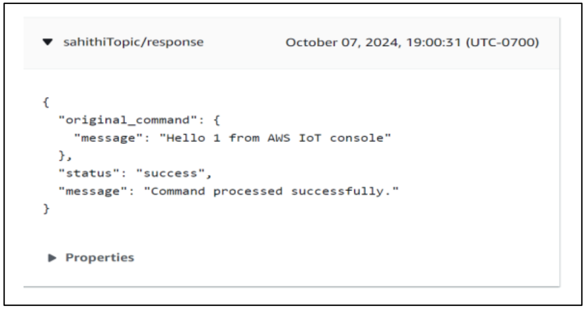
▶ Additional configuration

Publish



```
PS F:\CSS 532(IOT)\hw1> python publishReceiveFromConsole1.py
Connecting[redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to command topic 'sahithiTopic/receive'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for messages...
Received message from topic 'sahithiTopic/receive': {
  "message": "Hello 1 from AWS IoT console"
}
Processed command: {'message': 'Hello 1 from AWS IoT console'}
Published response to topic 'sahithiTopic/response': {'original_command': {'message': 'Hello 1 from AWS IoT console'}, '
status': 'success', 'message': 'Command processed successfully.'}
```

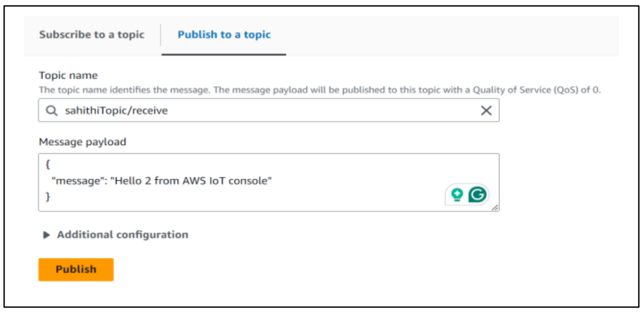
After processing the first message, my device automatically responds to AWS IoT by sending a success message and, my AWS IoT waits until it gets response for it's previous message and then publishes the second message for which it again waits for response until we reach the messages count limit which I have added in my python script and then, disconnects.



▼ sahithiTopic/response   October 07, 2024, 19:00:31 (UTC-0700)

```
{
  "original_command": {
    "message": "Hello 1 from AWS IoT console"
  },
  "status": "success",
  "message": "Command processed successfully."
}
```

▶ Properties



Subscribe to a topic | Publish to a topic

Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

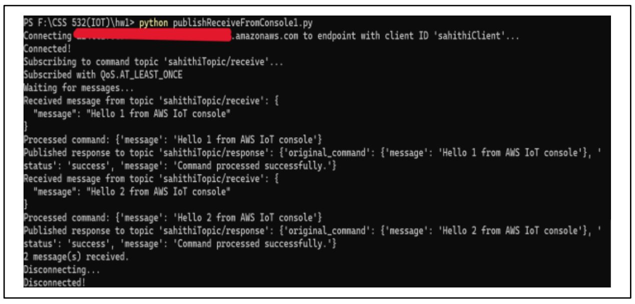
sahithiTopic/receive

Message payload

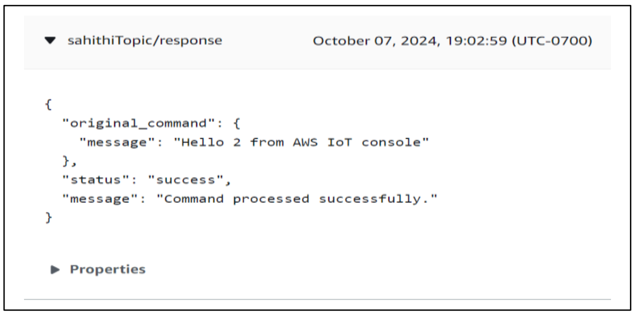
```
{
  "message": "Hello 2 from AWS IoT console"
}
```

▶ Additional configuration

Publish



```
PS F:\CSS 532(IOT)\hw1> python publishReceiveFromConsole1.py
Connecting [redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to command topic 'sahithiTopic/receive'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for messages...
Received message from topic 'sahithiTopic/receive': {
  "message": "Hello 1 from AWS IoT console"
}
Processed command: {'message': 'Hello 1 from AWS IoT console'}
Published response to topic 'sahithiTopic/response': {'original_command': {'message': 'Hello 1 from AWS IoT console'}, '
status': 'success', 'message': 'Command processed successfully.'}
Received message from topic 'sahithiTopic/receive': {
  "message": "Hello 2 from AWS IoT console"
}
Processed command: {'message': 'Hello 2 from AWS IoT console'}
Published response to topic 'sahithiTopic/response': {'original_command': {'message': 'Hello 2 from AWS IoT console'}, '
status': 'success', 'message': 'Command processed successfully.'}
2 message(s) received.
Disconnecting...
Disconnected!
```

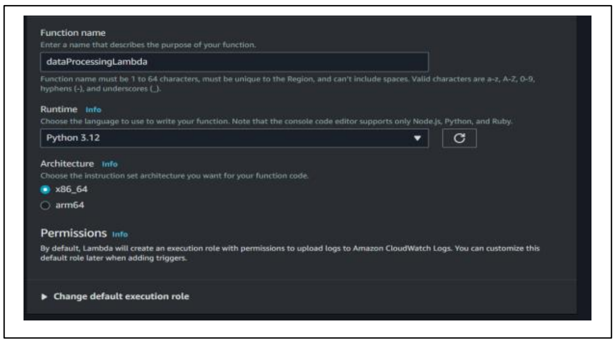


▼ sahithiTopic/response October 07, 2024, 19:02:59 (UTC-0700)

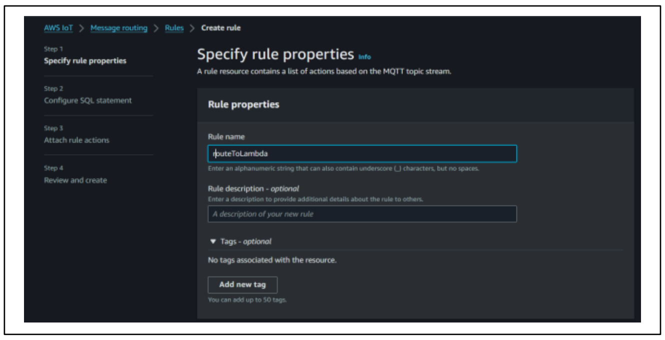
```
{
  "original_command": {
    "message": "Hello 2 from AWS IoT console"
  },
  "status": "success",
  "message": "Command processed successfully."
}
```

▶ Properties

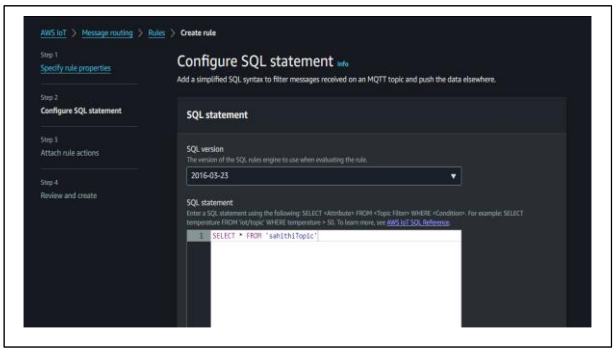
20.) Connected to AWS Lambda to my AWS IoT to process received from devices and save the raw data and the processed data into S3.
For this, I initially created a lambda function and named it as “dataProcessingLambda” and chose the language to python for writing the function. Created a rule for message routing.
Configured the SQL statement and set the rule actions.



Function name
Enter a name that describes the purpose of your function.
dataProcessingLambda
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12
Architecture Info
Choose the instruction set architecture you want for your function code.
● x86_64
○ arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
▶ Change default execution role

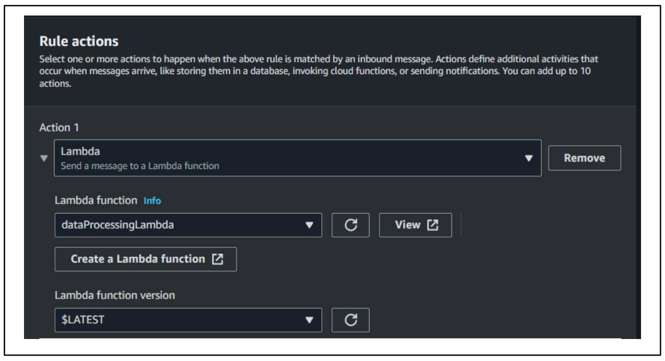


AWS IoT > Message routing > Rules > Create rule
Step 1 Specify rule properties
Step 2 Configure SQL statement
Step 3 Attach rule actions
Step 4 Review and create
Specify rule properties Info
A rule resource contains a list of actions based on the MQTT topic stream.
Rule properties
Rule name
routeToLambda
Enter an alphanumeric string that can also contain underscore (_) characters, but no spaces.
Rule description - *optional*
Enter a description to provide additional details about the rule to others.
A description of your new rule
▼ Tags - *optional*
No tags associated with the resource.
Add new tag
You can add up to 50 tags.



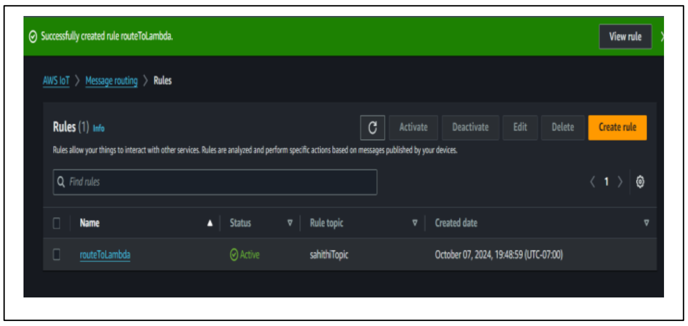
AWS IoT > Message routing > Rules > Create rule
Step 1 Specify rule properties
Step 2 Configure SQL statement
Step 3 Attach rule actions
Step 4 Review and create
Configure SQL statement Info
Add a simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere.
SQL statement
SQL version
The version of the SQL rules engine to use when evaluating the rule.
2016-03-23
SQL statement
Enter a SQL statement using the following: 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.

```
SELECT * FROM 'sahithiTopic'
```



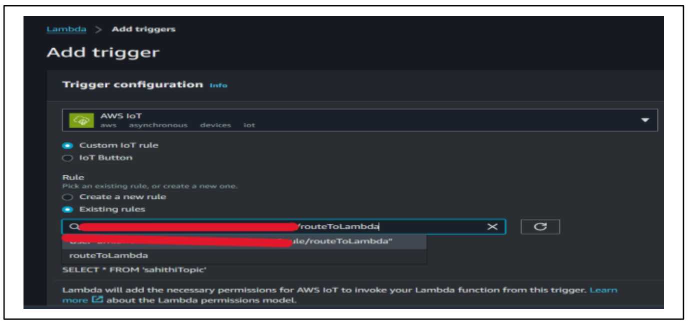
Rule actions
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.
Action 1
Lambda
Send a message to a Lambda function
Remove
Lambda function Info
dataProcessingLambda
View
Create a Lambda function
Lambda function version
$LATEST

Successfully, created rule to lambda for the topic “sahithiTopic” and added trigger to custom IoT rule for existing rule “rourToLambda”.



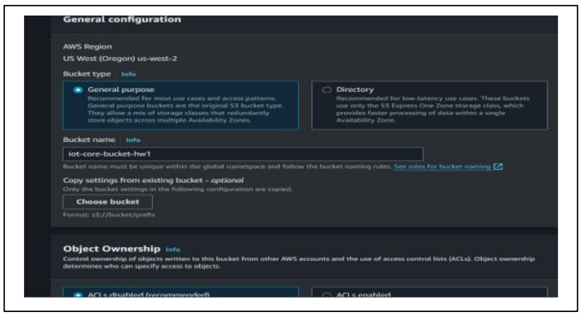
Successfully created rule routeToLambda. View rule
AWS IoT > Message routing > Rules
Rules (1) Info
Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

Name	Status	Rule topic	Created date
routeToLambda	Active	sahithiTopic	October 07, 2024, 19:48:59 (UTC-07:00)

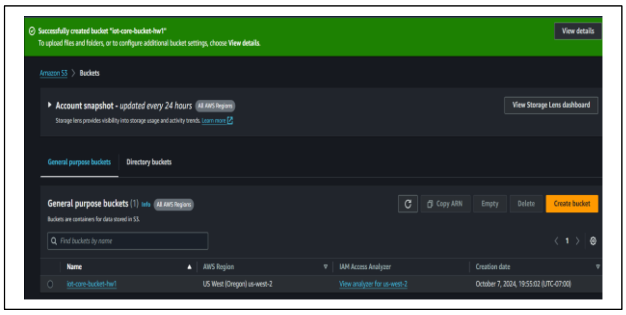


Lambda > Add triggers
Add trigger
Trigger configuration Info
AWS IoT
aws asynchronous devices iot
● Custom IoT rule
○ IoT Button
Rule
Pick an existing rule, or create a new one.
○ Create a new rule
● Existing rules
[redacted]/routeToLambda
[redacted]ule/routeToLambda"
routeToLambda
SELECT * FROM 'sahithiTopic'
Lambda will add the necessary permissions for AWS IoT to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Now, to save raw and processed data into S3 bucket created a new S3 bucket “iot-core-bucket-hw1”.



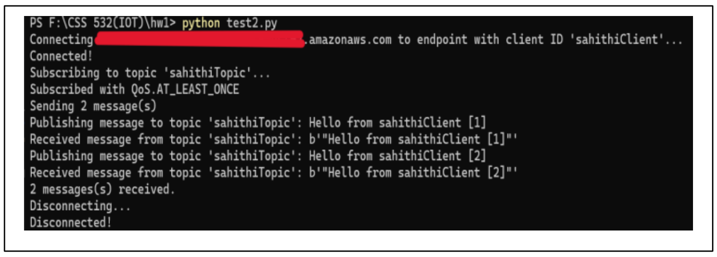
General configuration
AWS Region
US West (Oregon) us-west-2
Bucket type Info
● General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
○ Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
iot-core-bucket-hw1
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix
Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
● ACLs disabled (recommended) ○ ACLs enabled



Successfully created bucket "iot-core-bucket-hw1"
To upload files and folders, or to configure additional bucket settings, choose **View details**.
Amazon S3 > Buckets
▶ Account snapshot - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
General purpose buckets | Directory buckets
General purpose buckets (1) Info All AWS Regions
Buckets are containers for data stored in S3.

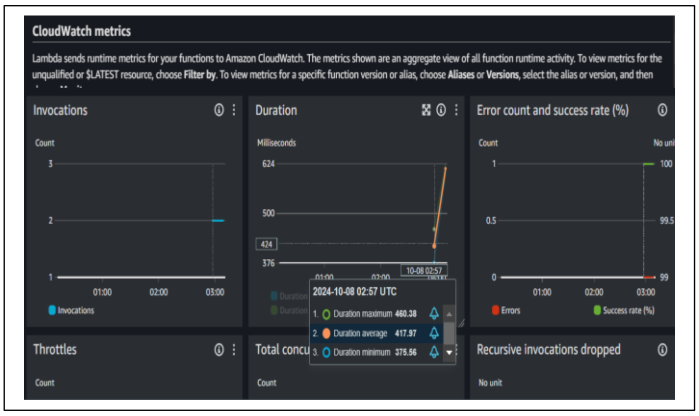
Name	AWS Region	IAM Access Analyzer	Creation date
iot-core-bucket-hw1	US West (Oregon) us-west-2	View analyzer for us-west-2	October 7, 2024, 19:55:02 (UTC-07:00)

21.) wrote the code for lambda function, deployed the code and executed sample script to check lambda function connection.

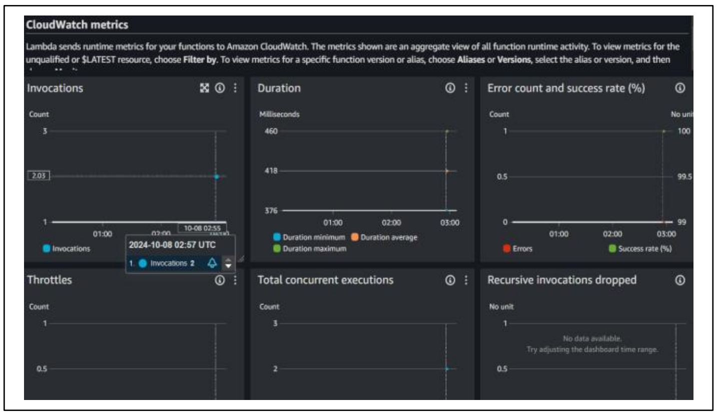


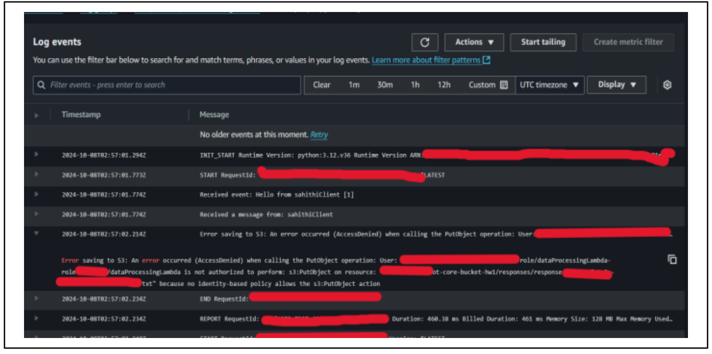
```
PS F:\CSS 532(IOT)\hw1> python test2.py
Connecting[redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [1]
Received message from topic 'sahithiTopic': b'"Hello from sahithiClient [1]"'
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [2]
Received message from topic 'sahithiTopic': b'"Hello from sahithiClient [2]"'
2 messages(s) received.
Disconnecting...
Disconnected!
```

Here I encountered an error while saving my data to s3 bucket. So, I updated the lambda policy to write to S3 bucket.



CloudWatch metrics
Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose Filter by. To view metrics for a specific function version or alias, choose Aliases or Versions, select the alias or version, and then
Invocations
Duration
Error count and success rate (%)
2024-10-08 02:57 UTC
Duration maximum 460.38
Duration average 417.97
Duration minimum 375.56
Throttles
Total concu
Recursive invocations dropped



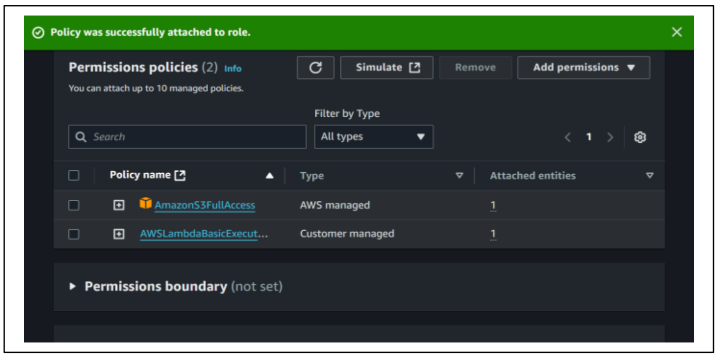


Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Timestamp	Message
	No older events at this moment. Retry
2024-10-08T02:57:01.294Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN:[redacted]
2024-10-08T02:57:01.773Z	START RequestId:[redacted]LATEST
2024-10-08T02:57:01.774Z	Received event: Hello from sahithiClient [1]
2024-10-08T02:57:01.774Z	Received a message from: sahithiClient
2024-10-08T02:57:02.214Z	Error saving to S3: An error occurred (AccessDenied) when calling the PutObject operation: User:[redacted]
2024-10-08T02:57:02.234Z	END RequestId:[redacted]
2024-10-08T02:57:02.234Z	REPORT RequestId:[redacted] Duration: 460.38 ms Billed Duration: 461 ms Memory Size: 128 MB Max Memory Used…

```
Error saving to S3: An error occurred (AccessDenied) when calling the PutObject operation: User:[redacted]role/dataProcessingLambda-
role[redacted]/dataProcessingLambda is not authorized to perform: s3:PutObject on resource:[redacted]ot-core-bucket-hw1/responses/response[redacted]
[redacted]txt" because no identity-based policy allows the s3:PutObject action
```



Policy was successfully attached to role.

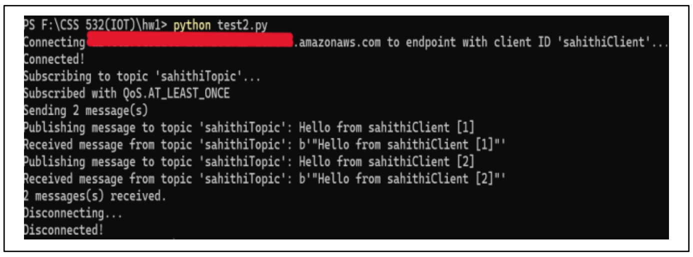
Permissions policies (2) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	1
AWSLambdaBasicExecut…	Customer managed	1

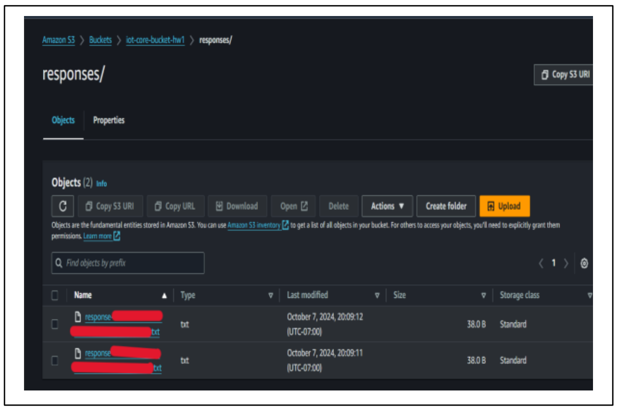
Permissions boundary (not set)

Again tested to check lambda function connection after updating the permissions to lambda policies.



```
PS F:\CSS 532(IOT)\hw1> python test2.py
Connecting[redacted].amazonaws.com to endpoint with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [1]
Received message from topic 'sahithiTopic': b'"Hello from sahithiClient [1]"'
Publishing message to topic 'sahithiTopic': Hello from sahithiClient [2]
Received message from topic 'sahithiTopic': b'"Hello from sahithiClient [2]"'
2 messages(s) received.
Disconnecting...
Disconnected!
```

Now, I was successfully able to deploy and run the lambda function and save the data to S3 bucket(by checking the log events & lambda runtime metrics for function).



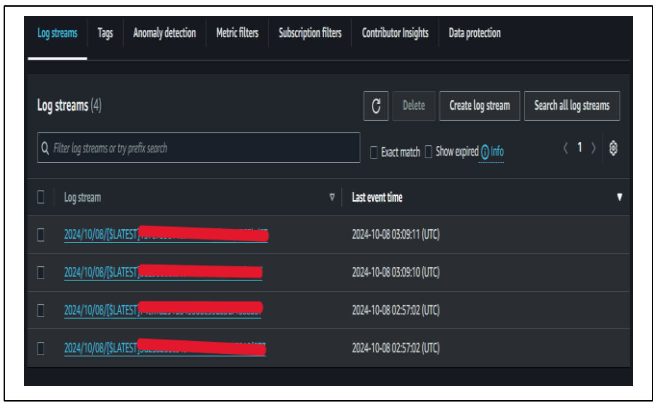
Amazon S3 > Buckets > iot-core-bucket-hw1 > responses/

responses/

Objects (2) Info

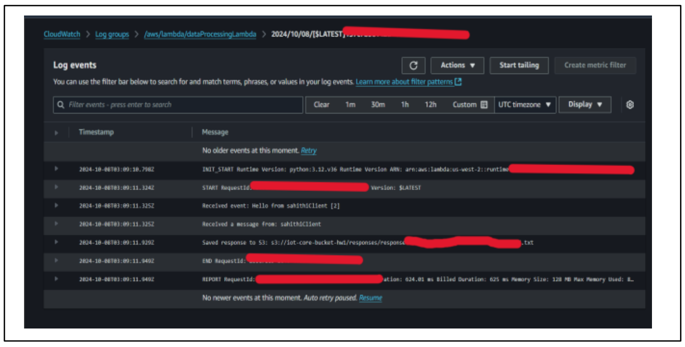
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
response-[redacted]txt	txt	October 7, 2024, 20:09:12 (UTC-07:00)	38.0 B	Standard
response-[redacted]txt	txt	October 7, 2024, 20:09:11 (UTC-07:00)	38.0 B	Standard



Log streams (4)

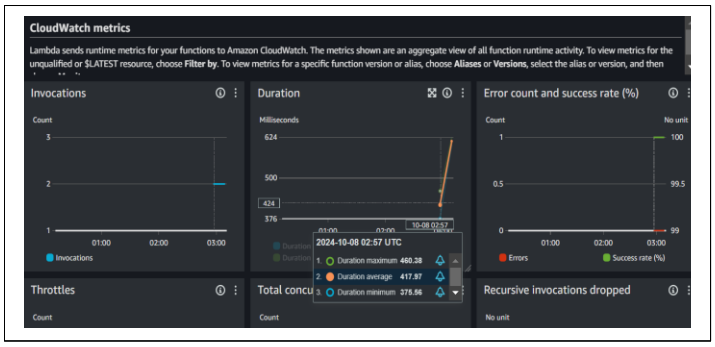
Log stream	Last event time
2024/10/08/[$LATEST][redacted]	2024-10-08 03:09:11 (UTC)
2024/10/08/[$LATEST][redacted]	2024-10-08 03:09:10 (UTC)
2024/10/08/[$LATEST][redacted]	2024-10-08 02:57:02 (UTC)
2024/10/08/[$LATEST][redacted]	2024-10-08 02:57:02 (UTC)


CloudWatch > Log groups > /aws/lambda/dataProcessingLambda > 2024/10/08/[$LATEST][redacted]

Log events

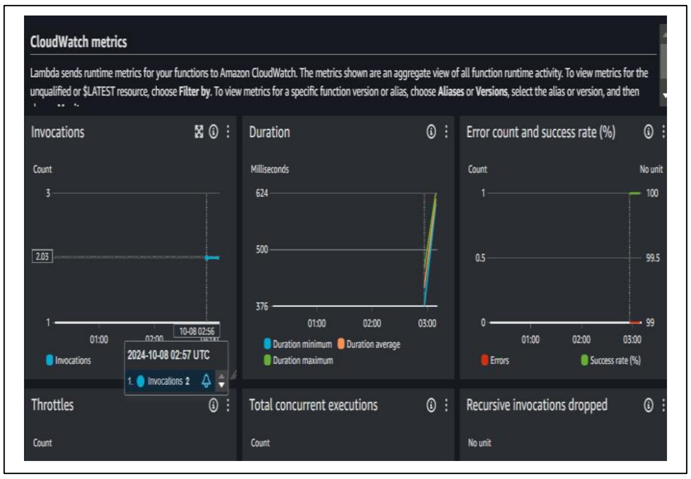
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

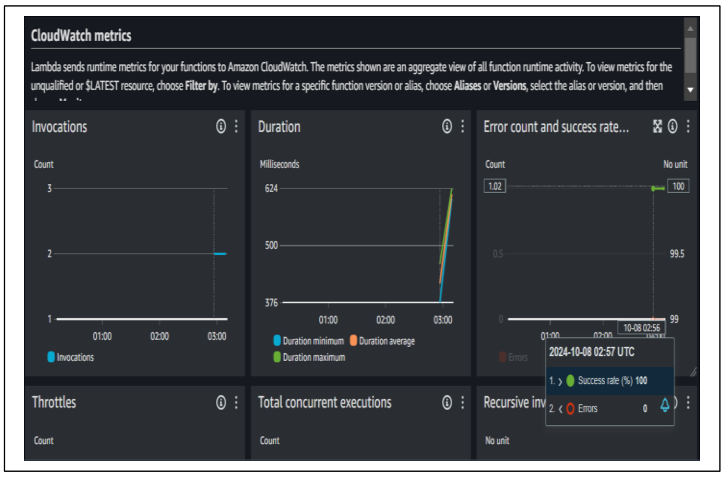
```
Timestamp                  Message
                           No older events at this moment. Retry
2024-10-08T03:09:10.798Z   INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:us-west-2::runtime[redacted]
2024-10-08T03:09:11.324Z   START RequestId:[redacted] Version: $LATEST
2024-10-08T03:09:11.325Z   Received event: Hello from sahithiClient [2]
2024-10-08T03:09:11.325Z   Received a message from: sahithiClient
2024-10-08T03:09:11.929Z   Saved response to S3: s3://iot-core-bucket-hw1/responses/response[redacted].txt
2024-10-08T03:09:11.949Z   END RequestId:[redacted]
2024-10-08T03:09:11.949Z   REPORT RequestId:[redacted]ation: 624.01 ms Billed Duration: 625 ms Memory Size: 128 MB Max Memory Used: 8…
                           No newer events at this moment. Auto retry paused. Resume
```


CloudWatch metrics

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose Filter by. To view metrics for a specific function version or alias, choose Aliases or Versions, select the alias or version, and then

2024-10-08 02:57 UTC
1. Duration maximum 460.38
2. Duration average 417.97
3. Duration minimum 375.56

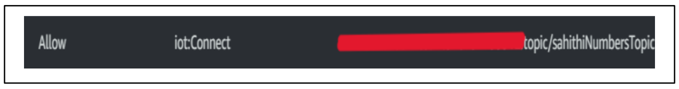

2024-10-08 02:57 UTC
1. Invocations 2


2024-10-08 02:57 UTC
1. Success rate (%) 100
2. Errors 0

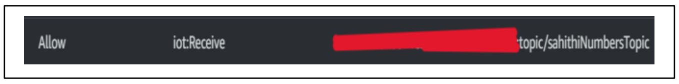
22.) Connected AWS Lambda to my AWS IoT to process received messages (e.g., modify the data, average the data, etc.) from device and save the raw data (the data that is abstracted from received messages) and the processed data (done by my Lambda) into S3.

For this, I wrote a new python script where it will publish a list of numbers to my AWS IoT. Then, created a new lambda function, connected it to my AWS IoT to process this data(calculate sum and take average of the sum) and save the data to S3 bucket.

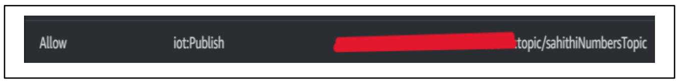
Initially I created new policies to use features like connect, subscribe, receive, and publish for “sahithiNumbersTopic”.



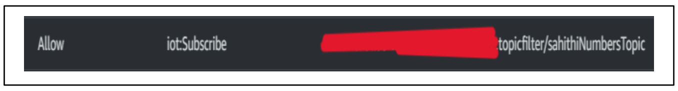
Allow	iot:Connect	[redacted]topic/sahithiNumbersTopic



Allow	iot:Receive	[redacted]topic/sahithiNumbersTopic

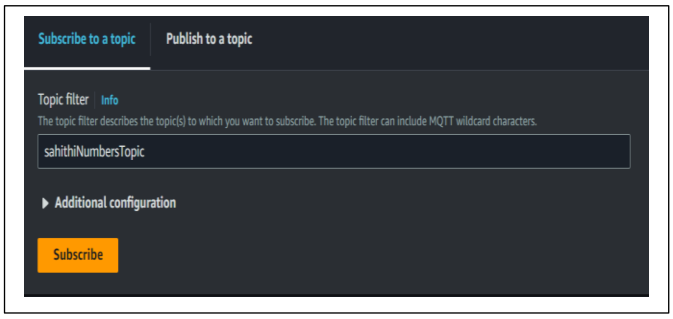


Allow	iot:Publish	[redacted]topic/sahithiNumbersTopic



Allow	iot:Subscribe	[redacted]topicfilter/sahithiNumbersTopic

Successfully published and republished messages to and from the AWS IoT console by writing a Python script and executing it through the terminal and verified that the messages were reaching the AWS IoT console by subscribing to the topic in AWS IoT console.


Subscribe to a topic | Publish to a topic

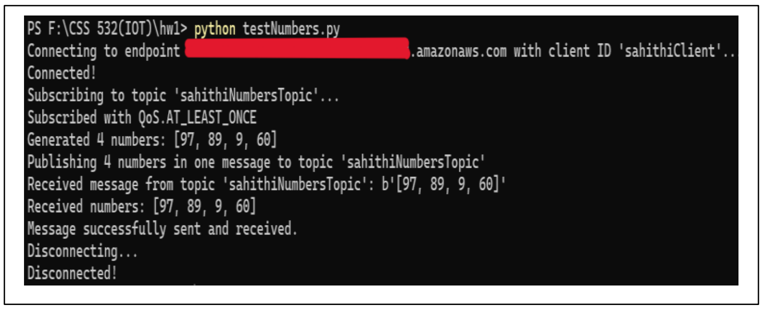
Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

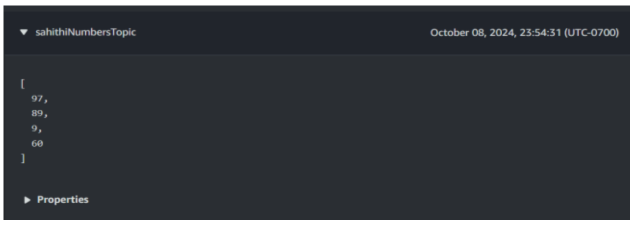
sahithiNumbersTopic

Additional configuration

Subscribe



```
PS F:\CSS 532(IOT)\hw1> python testNumbers.py
Connecting to endpoint [redacted].amazonaws.com with client ID 'sahithiClient'..
Connected!
Subscribing to topic 'sahithiNumbersTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Generated 4 numbers: [97, 89, 9, 60]
Publishing 4 numbers in one message to topic 'sahithiNumbersTopic'
Received message from topic 'sahithiNumbersTopic': b'[97, 89, 9, 60]'
Received numbers: [97, 89, 9, 60]
Message successfully sent and received.
Disconnecting...
Disconnected!
```

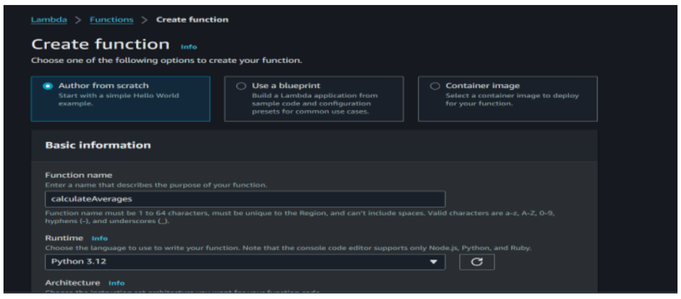


sahithiNumbersTopic October 08, 2024, 23:54:31 (UTC-0700)

```
[
  97,
  89,
  9,
  60
]
```

Properties

Then, created a new lambda function and set the rule for routing messages by co figuring the SQL statement and setting the rule actions.



Lambda > Functions > Create function

Create function

Choose one of the following options to create your function.

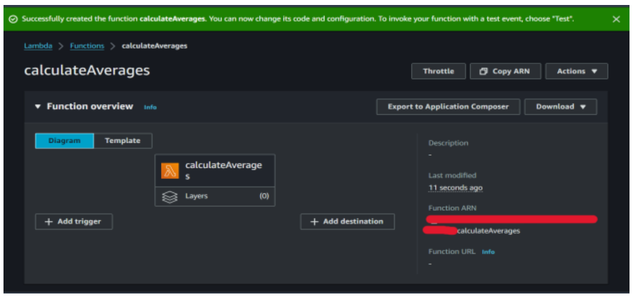
- [✓ Author from scratch] Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.

Basic information

Function name: Enter a name that describes the purpose of your function.
calculateAverages
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime: Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12

Architecture



Successfully created the function **calculateAverages**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

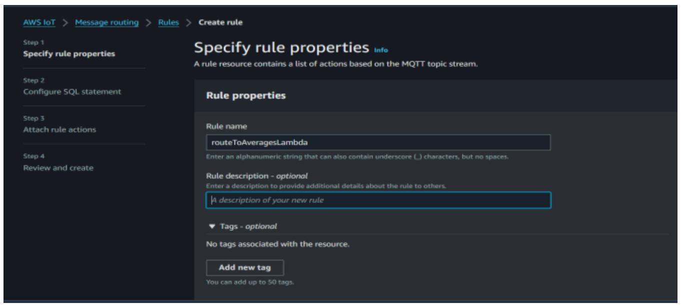
Lambda > Functions > calculateAverages

calculateAverages

Function overview

calculateAverages · Layers (0)

Description: -
Last modified: 11 seconds ago
Function ARN: [redacted]calculateAverages
Function URL: -



AWS IoT > Message routing > Rules > Create rule

Step 1 Specify rule properties
Step 2 Configure SQL statement
Step 3 Attach rule actions
Step 4 Review and create

Specify rule properties

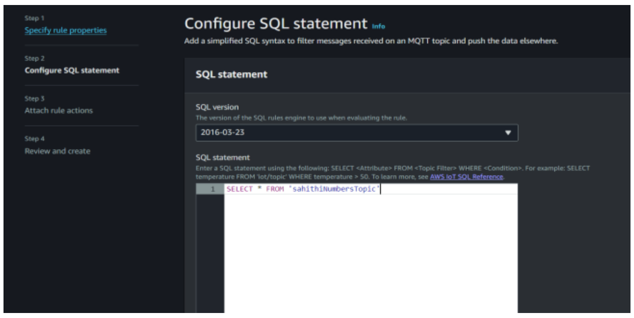
A rule resource contains a list of actions based on the MQTT topic stream.

Rule properties

Rule name: routeToAveragesLambda
Enter an alphanumeric string that can also contain underscore (_) characters, but no spaces.

Rule description - *optional*: Enter a description to provide additional details about the rule to others.
A description of your new rule

Tags - *optional*: No tags associated with the resource.
Add new tag
You can add up to 50 tags.



Step 1 Specify rule properties
Step 2 Configure SQL statement
Step 3 Attach rule actions
Step 4 Review and create

Configure SQL statement

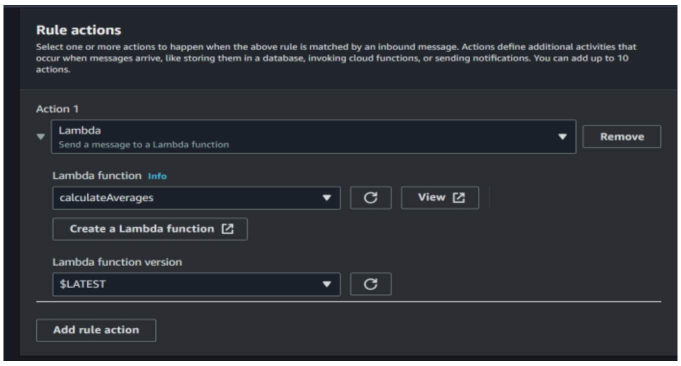
Add a simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere.

SQL statement

SQL version: The version of the SQL rules engine to use when evaluating the rule.
2016-03-23

SQL statement: Enter a SQL statement using the following: 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.

```
SELECT * FROM 'sahithiNumbersTopic'
```



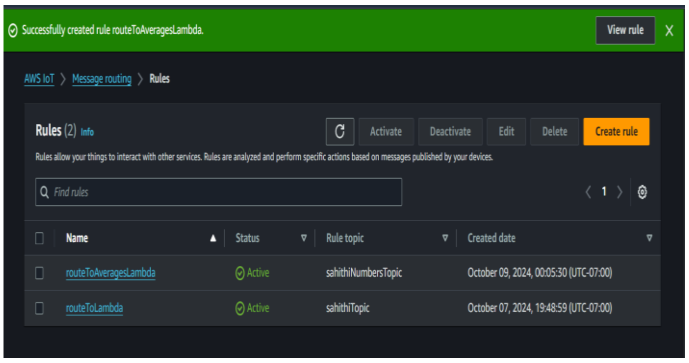
Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1: Lambda — Send a message to a Lambda function

Lambda function: calculateAverages

Lambda function version: $LATEST



Successfully created rule routeToAveragesLambda.

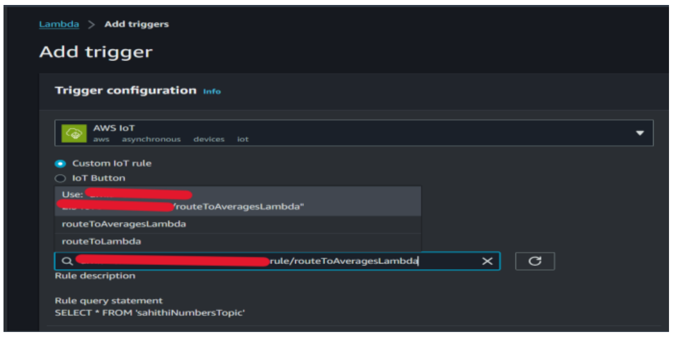
AWS IoT > Message routing > Rules

Rules (2)

Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

Name	Status	Rule topic	Created date
routeToAveragesLambda	Active	sahithiNumbersTopic	October 09, 2024, 00:05:30 (UTC-07:00)
routeToLambda	Active	sahithiTopic	October 07, 2024, 19:48:59 (UTC-07:00)

Successfully added trigger to the lambda function.



Lambda > Add triggers

Add trigger

Trigger configuration

AWS IoT (aws, asynchronous, devices, iot)

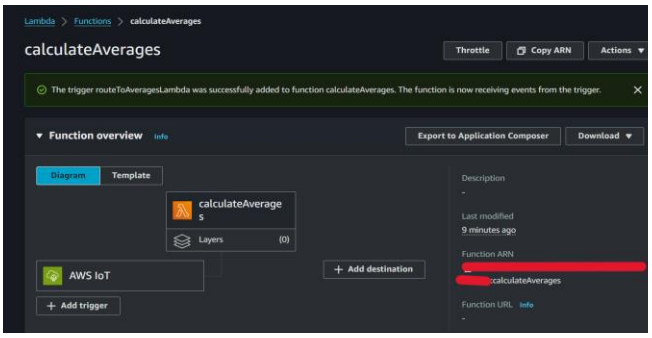
[✓ Custom IoT rule] IoT Button

Use: [redacted]/routeToAveragesLambda"
routeToAveragesLambda
routeToLambda

[redacted]rule/routeToAveragesLambda

Rule description

Rule query statement
SELECT * FROM 'sahithiNumbersTopic'



Lambda > Functions > calculateAverages

calculateAverages

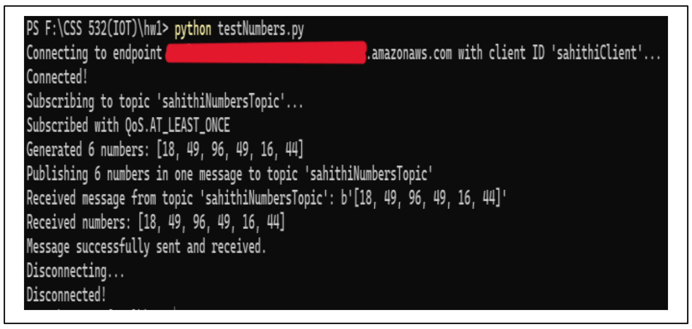
The trigger routeToAveragesLambda was successfully added to function calculateAverages. The function is now receiving events from the trigger.

Function overview

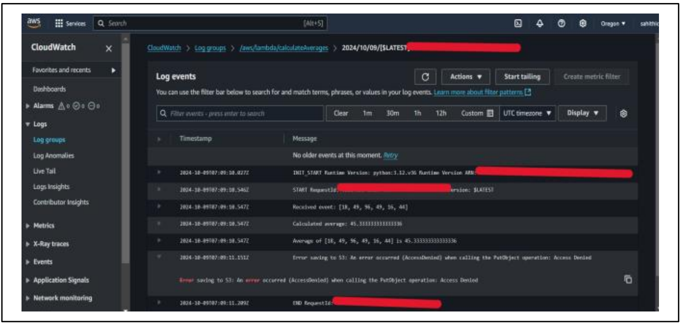
calculateAverages · Layers (0) · AWS IoT

Description: -
Last modified: 9 minutes ago
Function ARN: [redacted]calculateAverages
Function URL: -

Then, I ran the python script and got an error while saving the data to S3 buckte saying access denied. So, updated the S3 role policies to allow access.



```
PS F:\CSS 532(IOT)\hw1> python testNumbers.py
Connecting to endpoint [redacted].amazonaws.com with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiNumbersTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Generated 6 numbers: [18, 49, 96, 49, 16, 44]
Publishing 6 numbers in one message to topic 'sahithiNumbersTopic'
Received message from topic 'sahithiNumbersTopic': b'[18, 49, 96, 49, 16, 44]'
Received numbers: [18, 49, 96, 49, 16, 44]
Message successfully sent and received.
Disconnecting...
Disconnected!
```



CloudWatch > Log groups > /aws/lambda/calculateAverages > 2024/10/09/[$LATEST[redacted]

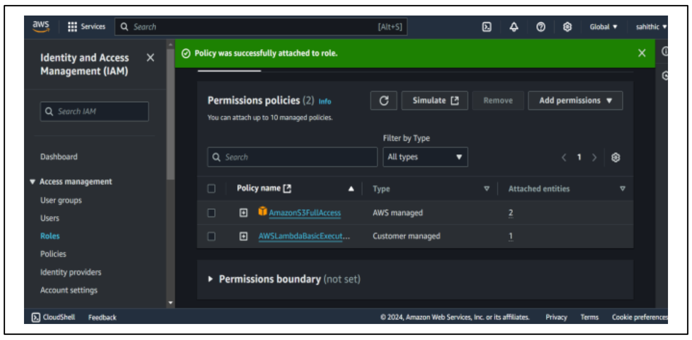
Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Clear 1m 30m 1h 12h Custom UTC timezone Display

```
Timestamp                  Message
                           No older events at this moment. Retry
2024-10-09T07:09:10.027Z   INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN:[redacted]
2024-10-09T07:09:10.546Z   START RequestId:[redacted]ersion: $LATEST
2024-10-09T07:09:10.547Z   Received event: [18, 49, 96, 49, 16, 44]
2024-10-09T07:09:10.547Z   Calculated average: 45.333333333333336
2024-10-09T07:09:10.547Z   Average of [18, 49, 96, 49, 16, 44] is 45.333333333333336
2024-10-09T07:09:11.151Z   Error saving to S3: An error occurred (AccessDenied) when calling the PutObject operation: Access Denied

                           Error saving to S3: An error occurred (AccessDenied) when calling the PutObject operation: Access Denied
2024-10-09T07:09:11.209Z   END RequestId:[redacted]
```



Policy was successfully attached to role.

Permissions policies (2) Info

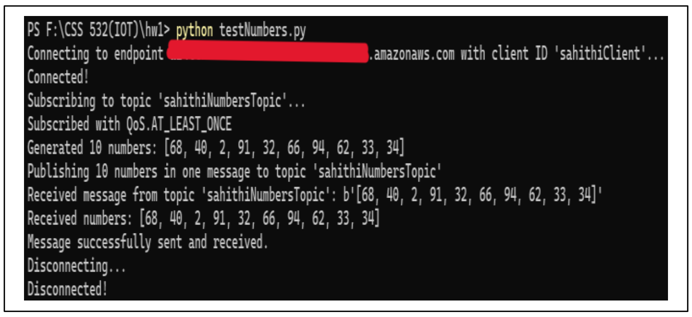
You can attach up to 10 managed policies.

Filter by Type: All types

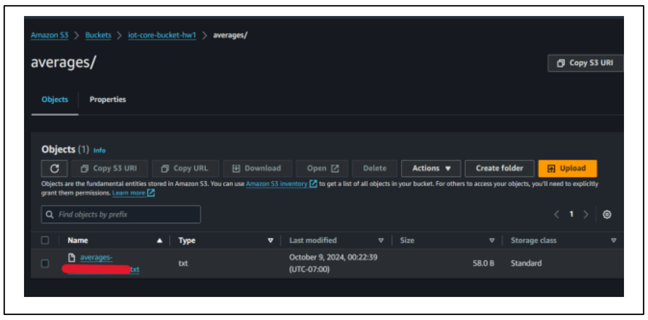
Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	2
AWSLambdaBasicExecut...	Customer managed	1

Permissions boundary (not set)

Again, ran the python script after updating the S3 role policy, now I was successfully able to publish and republish data to and from the AWS IoT console and was able to connect the lambda function to process the input data and store the data into S3 bucket.



```
PS F:\CSS 532(IOT)\hw1> python testNumbers.py
Connecting to endpoint [redacted].amazonaws.com with client ID 'sahithiClient'...
Connected!
Subscribing to topic 'sahithiNumbersTopic'...
Subscribed with QoS.AT_LEAST_ONCE
Generated 10 numbers: [68, 40, 2, 91, 32, 66, 94, 62, 33, 34]
Publishing 10 numbers in one message to topic 'sahithiNumbersTopic'
Received message from topic 'sahithiNumbersTopic': b'[68, 40, 2, 91, 32, 66, 94, 62, 33, 34]'
Received numbers: [68, 40, 2, 91, 32, 66, 94, 62, 33, 34]
Message successfully sent and received.
Disconnecting...
Disconnected!
```



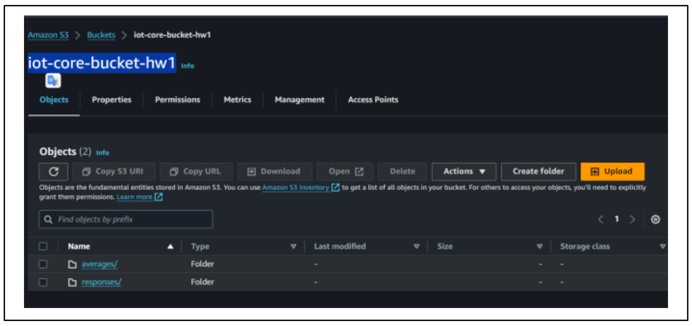
Amazon S3 > Buckets > iot-core-bucket-hw1 > averages/

averages/

Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
averages-[redacted]txt	txt	October 9, 2024, 00:22:39 (UTC-07:00)	58.0 B	Standard



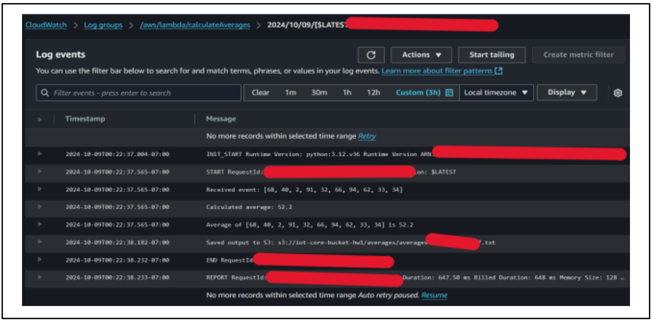
Amazon S3 > Buckets > iot-core-bucket-hw1

iot-core-bucket-hw1 Info

Objects (2) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
averages/	Folder	-	-	-
responses/	Folder	-	-	-



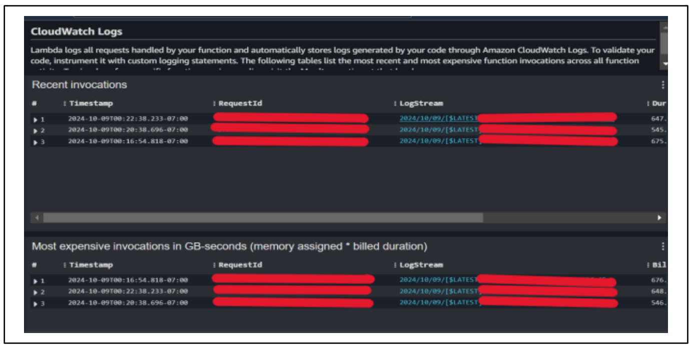
CloudWatch > Log groups > /aws/lambda/calculateAverages > 2024/10/09/[$LATES[redacted]

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Clear 1m 30m 1h 12h Custom (3h) Local timezone Display

```
Timestamp                       Message
                                No more records within selected time range Retry
2024-10-09T00:22:37.004-07:00   INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN:[redacted]
2024-10-09T00:22:37.565-07:00   START RequestId:[redacted]on: $LATEST
2024-10-09T00:22:37.565-07:00   Received event: [68, 40, 2, 91, 32, 66, 94, 62, 33, 34]
2024-10-09T00:22:37.565-07:00   Calculated average: 52.2
2024-10-09T00:22:37.565-07:00   Average of [68, 40, 2, 91, 32, 66, 94, 62, 33, 34] is 52.2
2024-10-09T00:22:38.182-07:00   Saved output to S3: s3://iot-core-bucket-hw1/averages/averages[redacted]7.txt
2024-10-09T00:22:38.232-07:00   END RequestId[redacted]
2024-10-09T00:22:38.233-07:00   REPORT RequestId:[redacted]Duration: 647.50 ms Billed Duration: 648 ms Memory Size: 128 …
                                No more records within selected time range Auto retry paused. Resume
```



CloudWatch Logs

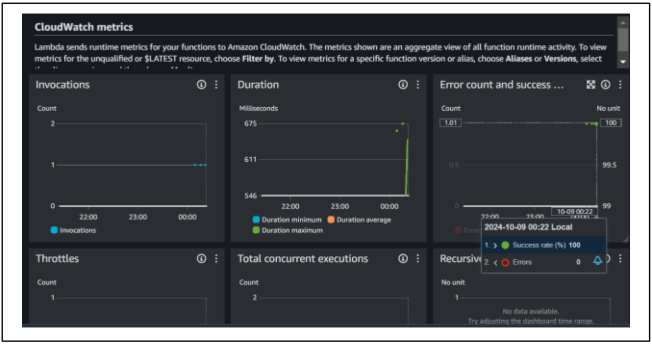
Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function

Recent invocations

#	Timestamp	RequestId	LogStream	Dur
1	2024-10-09T00:22:38.233-07:00	[redacted]	2024/10/09/[$LATEST[redacted]	647.
2	2024-10-09T00:20:38.696-07:00	[redacted]	2024/10/09/[$LATEST[redacted]	545.
3	2024-10-09T00:16:54.818-07:00	[redacted]	2024/10/09/[$LATEST[redacted]	675.

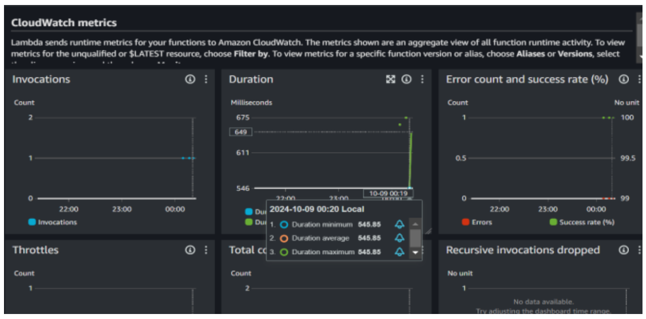
Most expensive invocations in GB-seconds (memory assigned * billed duration)

#	Timestamp	RequestId	LogStream	Bil
1	2024-10-09T00:16:54.818-07:00	[redacted]	2024/10/09/[$LATEST[redacted]	676.
2	2024-10-09T00:22:38.233-07:00	[redacted]	2024/10/09/[$LATEST[redacted]	648.
3	2024-10-09T00:20:38.696-07:00	[redacted]	2024/10/09/[$LATEST[redacted]	546.



CloudWatch metrics

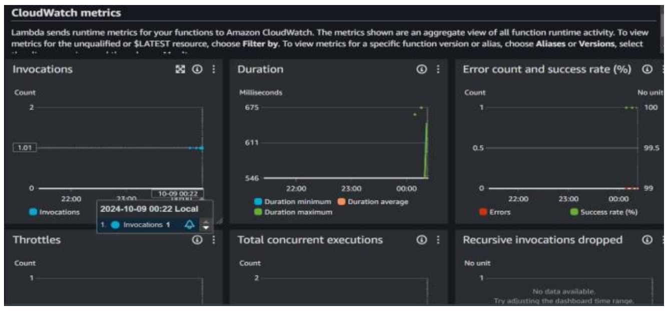
Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select



CloudWatch metrics

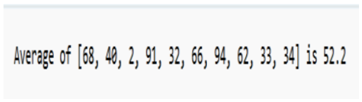
Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select

Invocations
Count
Invocations
Duration
Milliseconds
675
649
611
546
10-09 00:19
2024-10-09 00:20 Local
1. Duration minimum 545.85
2. Duration average 545.85
3. Duration maximum 545.85
Error count and success rate (%)
Count
No unit
Errors
Success rate (%)
22:00
23:00
00:00
Throttles
Recursive invocations dropped
No data available.
Try adjusting the dashboard time range.



CloudWatch metrics

Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select



```
Average of [68, 40, 2, 91, 32, 66, 94, 62, 33, 34] is 52.2
```

Errors / Difficulties faced while working on HW1:

1.) Initially, I faced difficulty as I didn't know how to update policies while experimenting with topic names and client IDs. However, as I continued to work on it, I was able to figure it out.

2.) Encountered an issue where I needed to create separate topics for receiving and responding to messages. This was necessary because I was using the same topic for both publishing and republishing, which caused the messages to be republished without waiting for a response. To resolve this, I created new policies for the sahithiTopic/receive and sahithiTopic/response topics to allow my device to successfully connect, publish, and subscribe to them.

3.) Forgot to update the policy to allow saving data to the S3 bucket while running the script.

Estimated time spent on this assignment:

I spent nearly 45 ~ 50 hours for this assignment in these two weeks.