

Step by step guide on IoT data synchronization using MQTT

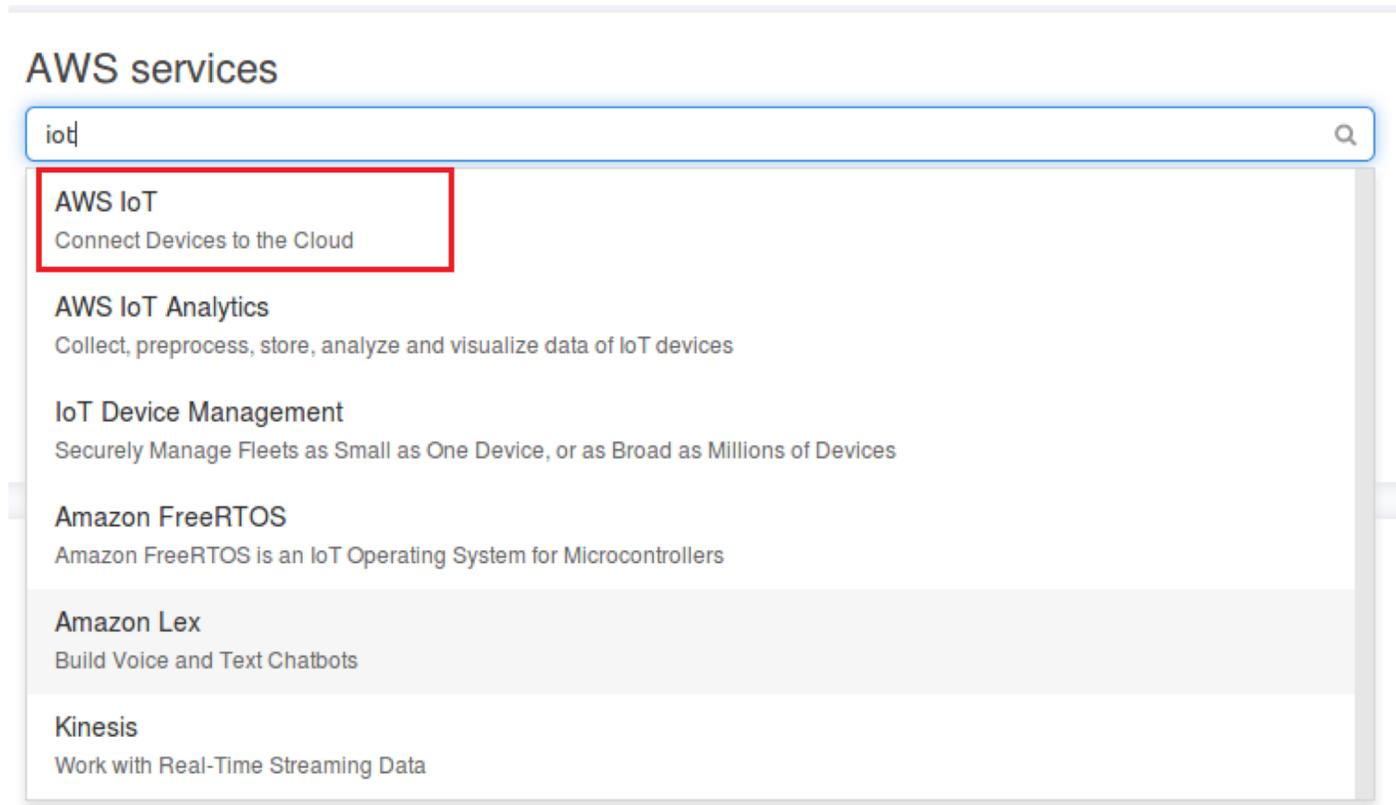
APIIT Education Group
Center of Technology and Innovation
Mustafa Mabrook

Topics & Structure of the report

- ▶ Create an IoT thing
- ▶ Add a certificate to the created thing
- ▶ Add a new policy and attach it to the created certificate
- ▶ Subscribe to an IoT topic
- ▶ Connect to the thing via a python script
- ▶ Publish data to the subscribed topic
- ▶ Add an action on receiving an event (Insert to DynamoDB)

Creating an IoT Thing

1. Navigate to the AWS IoT from AWS console



Creating an IoT Thing

2. Navigate to Manage Things option and then click on create button

The screenshot displays the AWS IoT console interface. On the left, a navigation sidebar includes icons and labels for 'Monitor', 'Onboard', 'Manage Things' (which is highlighted with a blue bar), 'Types', 'Groups', 'Jobs', 'Secure', 'Act', and 'Test'. The main content area is titled 'Things' and features a search bar and a 'Create' button in the top right corner. Below the header, a single IoT thing card is visible, labeled 'qms-thing' with the type 'NO TYPE' and a three-dot menu icon in the top right corner of the card.

Creating an IoT Thing

3. Choose “Create a single thing” option

Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more.](#)

Register a single AWS IoT thing

Create a thing in your registry

Create a single thing

Bulk register many AWS IoT things

Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

Creating an IoT Thing

4. Key in the thing name, choose the thing group if any and then click next

Name

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected ▼ [Create a type](#)

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using Jobs.

Thing Group

Groups / [Create group](#) [Change](#)

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key	Value	
<input type="text" value="Provide an attribute key, e.g. Manufacturer"/>	<input type="text" value="Provide an attribute value, e.g. Acme-Corporation"/>	Clear
Add another		

Show thing shadow ▼

[Back](#) [Next](#)

Add a certificate to the created thing

1. Choose “Create certificate” option

CREATE A THING

Add a certificate for your thing STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority. [Create certificate](#)

Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own. [Create with CSR](#)

Use my certificate
Register your CA certificate and use your own certificates for one or many devices. [Get started](#)

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT. [Create thing without certificate](#)

Add a certificate to the created thing

2. Download the created certificate, the public key, the private key, and root CA

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	aebf5c1e9f.cert.pem	Download
A public key	aebf5c1e9f.public.key	Download
A private key	aebf5c1e9f.private.key	Download

You also need to download a root CA for AWS IoT from Symantec:
A root CA for AWS IoT [Download](#)

[Activate](#)

[Done](#) [Attach a policy](#)

Add a certificate to the created thing

3. Activate the root CA and then click Done

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	aebf5c1e9f.cert.pem	Download
A public key	aebf5c1e9f.public.key	Download
A private key	aebf5c1e9f.private.key	Download

You also need to download a root CA for AWS IoT from Symantec:
A root CA for AWS IoT [Download](#)

[Activate](#)

[Done](#) [Attach a policy](#)

Add a new policy

1. Navigate to “Secure” menu option and then select “Policies”
2. Click on the “Create” button to create a new policy

The screenshot displays the AWS IoT console interface. On the left, a navigation sidebar includes icons and labels for 'Monitor', 'Onboard', 'Manage', 'Secure' (with sub-items: 'Certificates', 'Policies', 'CAs', 'Role Aliases', 'Authorizers'), 'Act', and 'Test'. The 'Secure' section is highlighted. The main content area is titled 'Policies' and contains a 'Card' dropdown menu, a search bar with the placeholder text 'Search policies', and a blue 'Create' button. Below these elements, a single policy card is shown with the name 'qms-policy' and a three-dot menu icon in the top right corner.

Add a new policy

3. Key in the policy name and specify the action and the resource ARN
4. Choose the appropriate effect and click create

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name

Add statements

Policy statements define the types of actions that can be performed by a resource. **Advanced mode**

Action

 *Make sure to only allow what you need (avoid using the * as possible)*

Resource ARN

 *Key in the exact ARN instead of the **

Effect

Allow Deny

Attach the created policy to the certificate

1. Navigate to “Secure” menu option and then select “Certificates”
2. Select the created certificate and navigate to “Policies”
3. From the “Actions” option, select the “Attach policy” option

The screenshot displays a certificate management interface. At the top, a dark header shows the word "CERTIFICATE" in small letters, followed by a long alphanumeric string: "a1b9077c4b2ab1446932d35b145203da00d972f2af57d4a2dc986f6971640b93". Below this, the word "ACTIVE" is visible. On the right side of the header, there is an "Actions" dropdown menu. The main content area is divided into a left sidebar with tabs for "Details", "Policies", and "Things". The "Policies" tab is currently selected, and its content area displays the message: "There are no policies attached to this certificate." The "Actions" dropdown menu is open, showing a list of options: "Activate", "Deactivate", "Revoke", "Accept transfer", "Reject transfer", "Revoke transfer", "Start transfer", "Attach policy", "Attach thing", "Download", and "Delete". The "Attach policy" option is highlighted in bold.

Attach the created policy to the certificate

4. Select the created policy which is “test” in this case and click “Attach”

Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
a1b9077c4b2ab1446932d35b145203da00d972f2af57d4a2dc986f6971640b93

Choose one or more policies

<input type="checkbox"/> qms-policy	View
<input checked="" type="checkbox"/> test	View

1 policy selected

[Cancel](#) [Attach](#)

Subscribe to an IoT topic

1. Key in the topic name, choose a QoS, choose the MQTT payload format and then Click “Subscribe to topic”



- Monitor
- Onboard
- Manage
- Secure
- Act
- Test**

- Software
- Settings
- Learn

MQTT client ?

Connected as `iotconsole-1517987737508-4`

Subscriptions

- [Subscribe to a topic](#)
- [Publish to a topic](#)

Subscribe

Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic

Subscribe to topic

Max message capture ?

Quality of Service ?

0 - This client will not acknowledge to the Device Gateway that messages are received

1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

Auto-format JSON payloads (improves readability)

Display payloads as strings (more accurate)

Display raw payloads (in hexadecimal)

Publish

Specify a topic and a message to publish with a QoS of 0.

Publish to topic

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

Subscribe to an IoT topic

2. Keep the window open and move on to the next step which is connection to the thing via a python script

The screenshot displays the AWS IoT console interface. On the left, a sidebar titled 'Subscriptions' shows a list of topics, with 'test' selected. The main area is titled 'test' and contains a 'Publish' section. Below the 'Publish' heading, there is a text input field containing 'test' and a 'Publish to topic' button. A code editor below the input field shows the following JSON message:

```
1 {  
2   "message": "Hello from AWS IoT console"  
3 }
```

Connect to the thing via a python script

1. Use the following script to connect to the created thing using the appropriate account's endpoint and the downloaded certificates/keys
2. Call the sync function and pass the topic name and the message to be sent

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import json

# Initialization
client = AWSIoTMQTTClient('id')
client.configureEndpoint('██████████.iot.ap-southeast-1.amazonaws.com', 8883)
client.configureCredentials('root.pem', 'private.pem.key', 'cert.pem.crt')
client.configureOfflinePublishQueueing(-1) # Infinite offline publish queuing

def sync(topic, payload):
    try:
        client.connect()
        client.publish(topic, json.dumps(payload), 1)
        client.disconnect()
    except:
        return

sync("test", "khachapuri")
```

AWS account endpoint

Downloaded certificates/keys

topic name

message to be sent

Note: The endpoint can be found in the “Settings” menu option in the bottom left corner

Connect to the thing via a python script

3. Go back to the AWS console to see the sent message

The screenshot displays the AWS IoT console's MQTT client interface. At the top, it shows 'MQTT client' and 'Connected as iotconsole-1517987737508-4'. The main area is divided into a left sidebar and a main content area. The sidebar has a 'Subscriptions' section with a 'test' topic selected. The main content area has a 'Publish' section with a text input field containing 'test' and a 'Publish to topic' button. Below this is a code editor showing a JSON message:

```
1 {  
2   "message": "Hello from AWS IoT console"  
3 }
```

. At the bottom, a message log shows a message received on the 'test' topic at 'Feb 7, 2018 3:17:04 PM +0800' with the content '"khachapuri"'. Red boxes highlight the 'test' topic name and the '"khachapuri"' message content in the log.

MQTT client [?](#) Connected as iotconsole-1517987737508-4

Subscriptions test Export Clear Pause

[Subscribe to a topic](#)

[Publish to a topic](#)

test ✕

Publish
Specify a topic and a message to publish with a QoS of 0.

Publish to topic

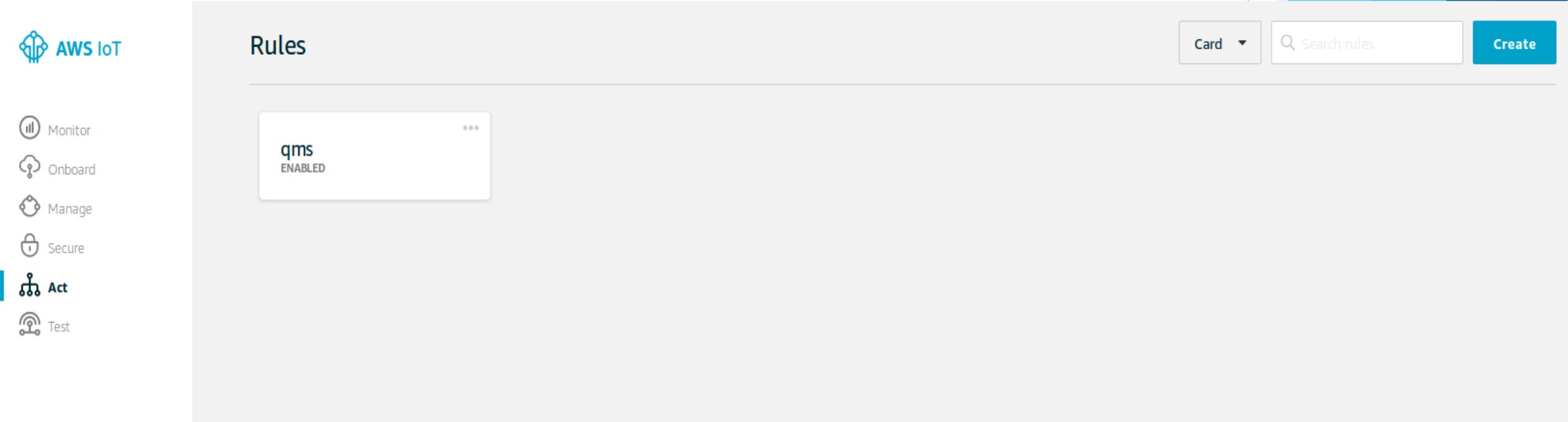
```
1 {  
2   "message": "Hello from AWS IoT console"  
3 }
```

topic name: test Feb 7, 2018 3:17:04 PM +0800 Export Hide

sent message: "khachapuri"

Add an action on receiving an event

1. Navigate to the “Act” menu option and then click “Create”



Add an action on receiving an event

2. Key in the rule name, the attribute, and the topic name (Make sure the topic name is the same one we published to)

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

Rule name

Description

Message source

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

Using SQL version [?](#)

Rule query statement

Attribute

Topic filter

Subscribed topic name

Add an action on receiving an event

3. Click on “Add action” and choose (Invoke a Lambda function option)

Set one or more 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. (*.required)

Add action

Select an action

Select an action.



Insert a message into a DynamoDB table

DYNAMODB



Split message into multiple columns of a database table (DynamoDBv2)

DYNAMODBv2



Invoke a Lambda function passing the message data

LAMBDA

Add an action on receiving an event

4. Choose the lambda function name to be executed from the dropdown list and click on “Add action”

Configure action

 Invoke a Lambda function passing the message data
LAMBDA

We'll set [the permissions](#) on the Lambda function for you.

***Function name**

insert_to_storage   [Create a new resource](#)

[Add action](#)

Lambda function code sample

insert_to_storage

Qualifiers ▼

Actions ▼

twes ▼

Test

Save

Function code [Info](#)

Code entry type

Edit code inline ▼

Runtime

Python 3.6 ▼

Handler [Info](#)

lambda_function.lambda_handler

```
▲ AWS Cloud9 File Edit Find View Goto Tools Window
Environment
  ▼ insert_to_storage
    ◀ lambda_function.py
  lambda_function × (+)
1 import boto3
2 import json
3
4 def lambda_handler(event, context):
5     dynamodb = boto3.resource('dynamodb', region_name='ap-southeast-1')
6     table = dynamodb.Table('ad-qms-storage')
7     event = json.dumps(event)
8     event = json.loads(event)
9     req_time = event['req_time']
10    student_id = event['student_id']
11    if event['remove'] == "0":
12        del event['remove']
13        table.put_item(Item=event)
14    else:
15        table.delete_item(Key={
16            'req_time': req_time,
17            'student_id': student_id
18        })
```

View the DynamoDB table



Scan: [Table] ad-qms-storage: req_time, student_id ^

Viewing 1 to 1 items

Scan ^

+ Add filter

Start search

<input type="checkbox"/>	req_time	student_id	canceled	counter	employee_username	end_serving_time	services	serving_time	start_serving_time
<input type="checkbox"/>	07/02/18 04:58:	TP040902	0	Null	Null	Null	5	Null	Null



Q&A

The background features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the left and right sides of the frame, creating a modern, dynamic feel. The central area is a clean, white space where the text is placed.

Thank You