

Singapore Institute of Technology

BEng (Hons) Information and Communications Technology majoring in Software Engineering

INF2009 Edge Computing and Analytics

Academic Year 2024/2025 Trimester 2

Week 10 Lab – AWS IoT Core

Name: Lim Chee Hean

Student ID: 2201529

2. Setup IoT Thing aka Device in IoT Core

Specify thing properties [Info](#)

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties [Info](#)

Thing name

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

Thing types are an optional way to store description and configuration information that is common to things that have the same thing type.

Thing type



Add searchable attributes to allow your thing to be grouped and searched without using fleet indexing.

No searchable attributes are associated with the selected thing type.

Configure device certificate - *optional* [Info](#)

A device requires a certificate to connect to AWS IoT. You can choose how to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

☒ **Auto-generate a new certificate (recommended)**
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

☐ **Use my certificate**
Use a certificate signed by your own certificate authority.

☐ **Upload CSR**
Register your CA and use your own certificates on one or many devices.

☐ **Skip creating a certificate at this time**
You can create a certificate for this thing and attach a policy to the certificate at a later time.

[Cancel](#)

[Previous](#)

[Next](#)

Create policy [Info](#)

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

Policy properties

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name

raspberry_pi_policy

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

► [Tags - optional](#)

[Policy statements](#)

[Policy examples](#)

Policy document [Info](#)

[Builder](#)

[JSON](#)

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect

Allow ▼

Policy action

* ▼

Policy resource

* ▼

[Remove](#)

[Add new statement](#)

[Cancel](#)

[Create](#)

Attach policies to certificate - *optional* [Info](#)

AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

Policies (1/1)

[Create policy](#)

Select up to 10 policies to attach to this certificate.

< 1 >



Name



[raspberry_pi_policy](#)

[Cancel](#)[Previous](#)[Create thing](#)

Download certificates and keys



Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate

552a471dc7e...te.pem.crt

[Deactivate certificate](#)[Download](#)

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.



This is the only time you can download the key files for this certificate.

Public key file

552a471dc7e72b62884aa83...3b6b40a-public.pem.key

[Download](#)

Key downloaded

Private key file

552a471dc7e72b62884aa83...b6b40a-private.pem.key

[Download](#)

Key downloaded

Things (1) Info

Advanced search

Run aggregations

Run connectivity status query

Edit

Delete

Create things

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Q Filter things by: name, type, group, billing, or searchable attribute.

< 1 > ⚙

<input type="checkbox"/>	Name	Thing type
<input type="checkbox"/>	raspberrypi_1	raspberrypi

Certificates Info

X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with AWS IoT.

Certificates

Certificates you've transferred

Certificates (1)

Actions ▾

Add certificate ▾

Q Find certificates

< 1 > ⚙

<input type="checkbox"/>	Certificate ID	Status	Created
<input type="checkbox"/>	552a471dc7e72b62884aa839efacf602e1d8e41f8edae14e289d129423b6b40a	Active	March 15, 2025, 22:38:56 (UTC+08:00)

AWS IoT > Security > Certificates > 552a471dc7e72b62884aa839efacf602e1d8e41f8edae14e289d129423b6b40a

552a471dc7e72b62884aa839efacf602e1d8e41f8edae14e289d129423b6b40a Info

Actions ▾

Details

Certificate ID

552a471dc7e72b62884aa839efacf602e1d8e41f8edae14e289d129423b6b40a

Certificate ARN

arn:aws:iot:us-east-1:129888882199:cert/552a471dc7e72b62884aa839efacf602e1d8e41f8edae14e289d129423b6b40a

Subject

CN=AWS IoT Certificate

Issuer

OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US

Status

Active

Created

March 15, 2025, 22:38:56 (UTC+08:00)

Valid

March 15, 2025, 22:36:56 (UTC+08:00)

Expires

January 01, 2050, 07:59:59 (UTC+08:00)

Policies

Things

Noncompliance

Policies (1) Info

Detach policies

Attach policies

Q

< 1 > ⚙

<input type="checkbox"/>	Name
<input type="checkbox"/>	raspberrypi_policy

3. Setup Raspberry Pi to send any data over MQTT Core

Create virtual environment and install dependencies.

```
cheehean@raspberrypi:~/labs/awsiotcore $ python -m venv awsiotcore
cheehean@raspberrypi:~/labs/awsiotcore $ source awsiotcore/bin/activate
(awsiotcore) cheehean@raspberrypi:~/labs/awsiotcore $ pip install paho-mqtt
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Using cached https://www.piwheels.org/simple/paho-mqtt/paho-mqtt-2.1.0-py3-none-any.whl (67 kB)
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-2.1.0
(awsiotcore) cheehean@raspberrypi:~/labs/awsiotcore $ pip install --upgrade psutil
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting psutil
  Using cached psutil-7.0.0-cp36-abi3-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (279 kB)
Installing collected packages: psutil
Successfully installed psutil-7.0.0
(awsiotcore) cheehean@raspberrypi:~/labs/awsiotcore $ |
```

Copy files from local machine to Raspberry Pi.

```
C:\Users\Lim Chee Hean\Desktop\awsiotcore>scp * cheehean@192.168.1.10:/home/cheehean/labs/awsiotcore
cheehean@192.168.1.10's password:
aws-certificate.pem.crt                                100% 1220   198.6KB/s   00:00
aws-private.pem.key                                  100% 1679   205.0KB/s   00:00
aws-public.pem.key                                   100% 451    110.1KB/s   00:00
pipython.py                                          100% 1104   359.4KB/s   00:00
rootCA.pem                                          100% 1188   386.7KB/s   00:00
```

Update endpoint domain name.

```
GNU nano 7.2                                pipython.py *
import time
import paho.mqtt.client as mqtt
import ssl
import json
import _thread as thread
import psutil
import json

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

client = mqtt.Client()
client.on_connect = on_connect
client.tls_set(ca_certs='./rootCA.pem', certfile='./aws-certificate.pem.crt', keyfile='./aws-private.pem.key', tls_version=ssl.PROTOCOL_TLS)
client.tls_insecure_set(True)
client.connect("a2s61kijkg8e-ats.iot.us-east-1.amazonaws.com", 8883, 60) #Copy end point from your AWS IoT Core Console

def justADummyFunction(Dummy):
    while (1):
        # This is where you can put your edge analytics to generate data from your sensors
        # processed/raw data can be sent to AWS IoT core for further analytics/processing on the cloud
        message = "Hello from INF2009 RaspberryPi Device#1"
        print(message)
        client.publish("device/data", payload=message , qos=0, retain=False)

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-C Copy
```

Subscribe to topic.

MQTT test client [Info](#)

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

▶ Connection details ✔ Connected

Disconnect

To disconnect from the MQTT test client, choose Disconnect. To re-establish a connection, you can update the details and choose Connect.

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.

device/data

▶ Additional configuration

Subscribe

Receive messages from Raspberry Pi.

Subscriptions

device/data

Pause

Clear

Export

Edit

device/data ♥ ✕

Message payload

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

▶ Additional configuration

Publish

▼ device/data

March 16, 2025, 15:34:24 (UTC+0800)

✖ Message cannot be displayed in specified format.

Hello from INF2009 RaspberryPi Device#1

▶ Properties

4. Ingest Real-time Data into DynamoDB via IoT Rule

Update sample code to send CPU utilisation and other information.

```
GNU nano 7.2 pipython.py *
import json

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

client = mqtt.Client()
client.on_connect = on_connect
client.tls_set(ca_certs='./rootCA.pem', certfile='./aws-certificate.pem.crt', keyfile='./aws-private.pem.key', tls_version=ssl.PROTOCOL_SSLv23)
client.tls_insecure_set(True)
client.connect("a2s6lkijkghd8e-ats.iot.us-east-1.amazonaws.com", 8883, 60) #Copy end point from your AWS IoT Core Console

def justADummyFunction(Dummy):
    while (1):
        # This is where you can put your edge analytics to generate data from your sensors
        # processed/raw data can be sent to AWS IoT core for further analytics/processing on the cloud
        # message = "Hello from INF2009 RaspberryPi Device#1"
        message = json.dumps({"time": int(time.time()), "quality": "GOOD", "hostname": "rpiedge", "value": psutil.cpu_percent()}, indent=2)
        print(message)
        client.publish("device/data", payload=message, qos=0, retain=False)
        time.sleep(5)

thread.start_new_thread(justADummyFunction, ("Create Thread",))

client.loop_forever()
```

Test publish and receive message.

```
(awsiotcore) cheehean@raspberrypi:~/labs/awsiotcore $ python pipython.py
/home/cheehean/labs/awsiotcore/pipython.py:13: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
client = mqtt.Client()
{
  "time": 1742110955,
  "quality": "GOOD",
  "hostname": "rpiedge",
  "value": 0.0
}
Connected with result code 0
{
  "time": 1742110960,
  "quality": "GOOD",
  "hostname": "rpiedge",
  "value": 0.1
}
{
  "time": 1742110965,
  "quality": "GOOD",
  "hostname": "rpiedge",
  "value": 0.0
}
{
  "time": 1742110970,
  "quality": "GOOD",
  "hostname": "rpiedge",
  "value": 0.0
}
```

▼ device/data	March 16, 2025, 15:45:40 (UTC+0800)
<pre>{ "time": 1742111140, "quality": "GOOD", "hostname": "rpiedge", "value": 0.0 }</pre>	
► Properties	

▼ device/data	March 16, 2025, 15:45:35 (UTC+0800)
<pre>{ "time": 1742111135, "quality": "GOOD", "hostname": "rpiedge", "value": 0.0 }</pre>	
► Properties	

Create rule.

Specify rule properties [Info](#)

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

Rule properties

Rule name

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.

▼ **Tags - optional**

No tags associated with the resource.

Add new tag

You can add up to 50 tags.

CancelNext

Configure SQL statement.

Configure SQL statement [Info](#)

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

SQL statement [Info](#)

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](#).

1

`SELECT * FROM 'device/data'`

SQLLn 1, Col 28⊗ Errors: 0⚠ Warnings: 0⚙

CancelPreviousNext

Create DynamoDB table.

Create table

Table details

Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

raspberry_pi_device_demo

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

hostnameString

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

timestampNumber

1 to 255 characters and case sensitive.

Table settings

☐ Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

☒ Customize settings

Use these advanced features to make DynamoDB work better for your needs.

Configure rule actions.

Rule actions

Info

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

DynamoDB

▼

Insert a message into a DynamoDB table

Remove

Table name

Info

raspberry_pi_device_demo

▼

↺

View

↗

Create DynamoDB table

↗

Partition key

The partition key (also called hash key) must match the partition key of the DynamoDB table that you created.

hostname

Partition key type

The partition key (also called hash key) type can be STRING or NUMBER. The default value is STRING.

STRING

▼

Partition key value

The partition key (also called hash key) value supports substitution templates that provide data at runtime.

\${hostname}

Sort key - optional

The sort key (also called range key) must match the sort key of the DynamoDB table that you created.

timestamp

Sort key type

The sort key (also called range key) type can be STRING or NUMBER. The default value is STRING.

NUMBER

▼

Sort key value

The sort key (also called range key) value supports substitution templates that provide data at runtime.

\${time}

Assign existing IAM role. Unable to create new role due to AWS Academy account limitations.

IAM role
Choose a role to grant AWS IoT access to your endpoint.

LabRole ▼

↺

View ↗

Create new role

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

Review configuration and create rule.

Review and create Info

Step 1: Rule properties

Edit

Rule properties

Name

raspberrypi_device_rule

Description

-

Step 2: SQL statement

Edit

SQL statement

SQL version

2016-03-23

SQL query

SELECT * FROM 'device/data'

Step 3: Rule actions

Edit

Actions

DynamoDB
Insert a message into a DynamoDB table

Table name	Partition key	Partition key type
raspberrypi_device_demo	hostname	STRING
Partition key value	Sort key - optional	Sort key type
\${hostname}	timestamp	NUMBER
Sort key value	Write message data to this column	Operation - optional
\${time}	- optional	-
	-	

IAM role

arn:aws:iam::129888882199:role/voclabs ↗

View messages inserted into DynamoDB.

Tables (1)

Any tag key

Any tag value

Find tables

< 1 >

raspberry_pi_device_demo

▼ Scan or query items

Scan

Query

Select a table or index

Table - raspberry_pi_device_demo

Select attribute projection

All attributes

► Filters

Run

Reset

Completed. Read capacity units consumed: 2

Items returned (8)

Actions

Create item

< 1 >

	hostname (String)	timestamp (Number)	payload
<input type="checkbox"/>	rpiedge	1742115930	{ "hostname": { "S": "rpiedge" }, "time": { "N": "1742115930" }, "...
<input type="checkbox"/>	rpiedge	1742115935	{ "hostname": { "S": "rpiedge" }, "time": { "N": "1742115935" }, "...
<input type="checkbox"/>	rpiedge	1742115940	{ "hostname": { "S": "rpiedge" }, "time": { "N": "1742115940" }, "...
<input type="checkbox"/>	rpiedge	1742115945	{ "hostname": { "S": "rpiedge" }, "time": { "N": "1742115945" }, "...
<input type="checkbox"/>	rpiedge	1742115950	{ "hostname": { "S": "rpiedge" }, "time": { "N": "1742115950" }, "...

Attributes

View DynamoDB JSON

Copy

1

{

2

"hostname": "rpiedge",

3

"timestamp": 1742115930,

4

"payload": {

5

"hostname": "rpiedge",

6

"quality": "GOOD",

7

"time": 1742115930,

8

"value": 0

9

}

10

}