Singapore Institute of Technology

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

INF2009 Edge Computing and Analytics

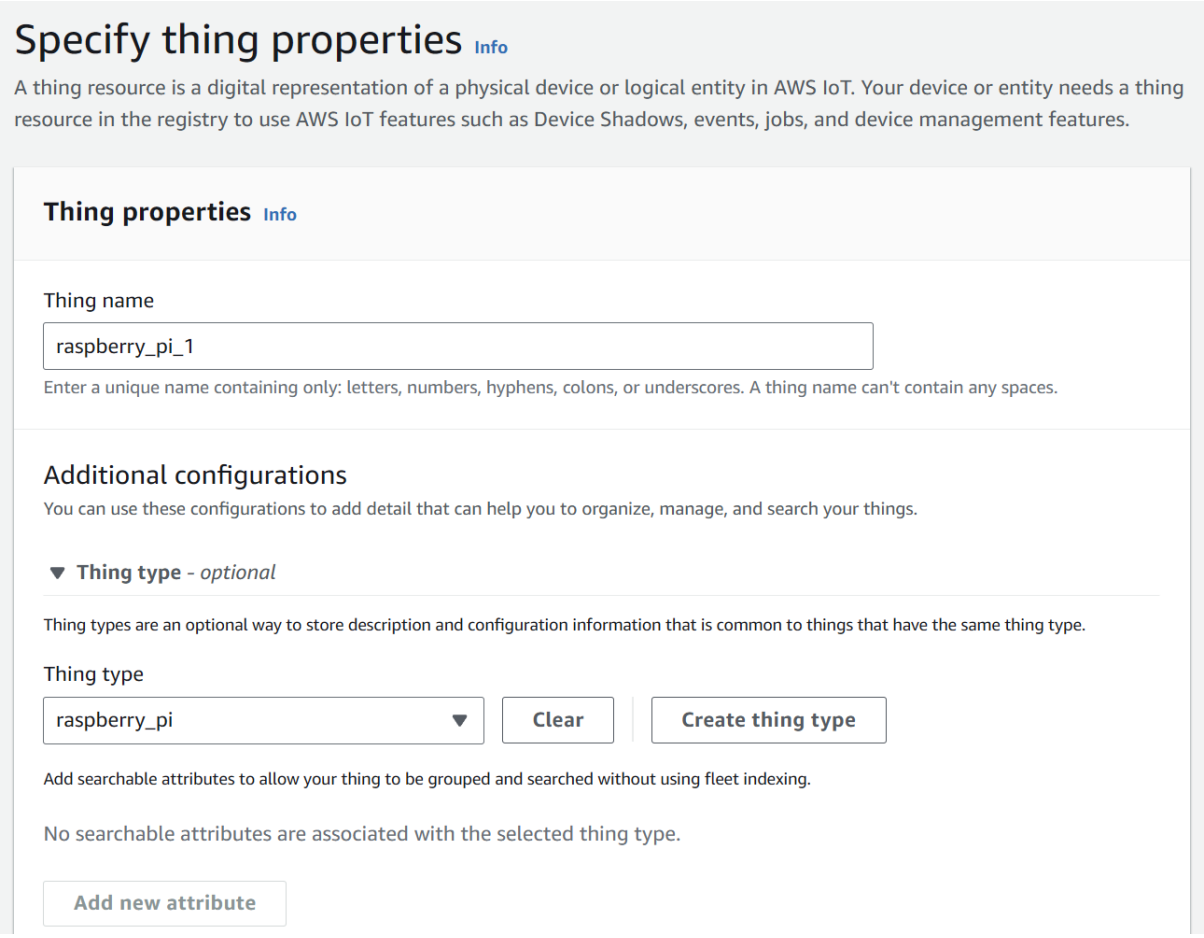
Academic Year 2024/2025 Trimester 2

Week 10 Lab – AWS IoT Core

Name: Lim Chee Hean

Student ID: 2201529

2. Setup IoT Thing aka Device in IoT Core



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	Policy action	Policy resource	
Allow	*	*	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)

Select up to 10 policies to attach to this certificate.

Create policy

Filter policies

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

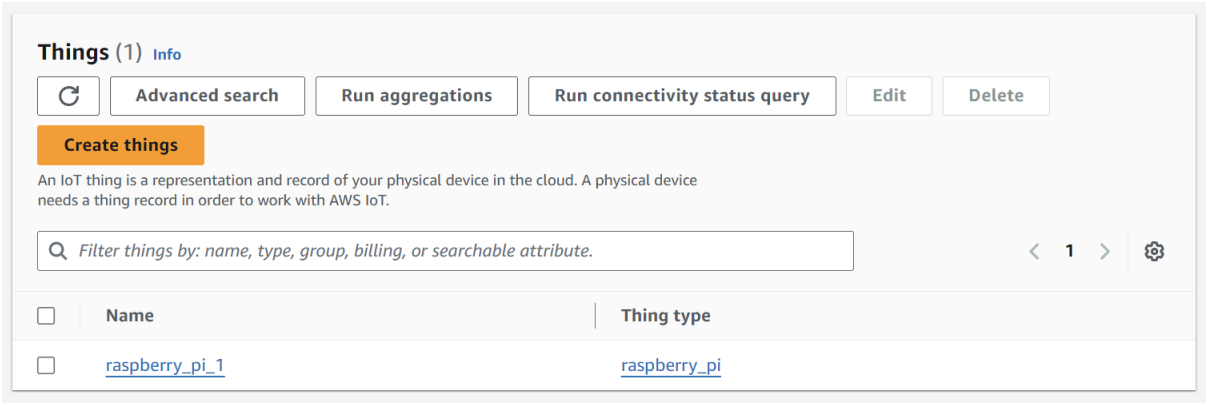
Key downloaded

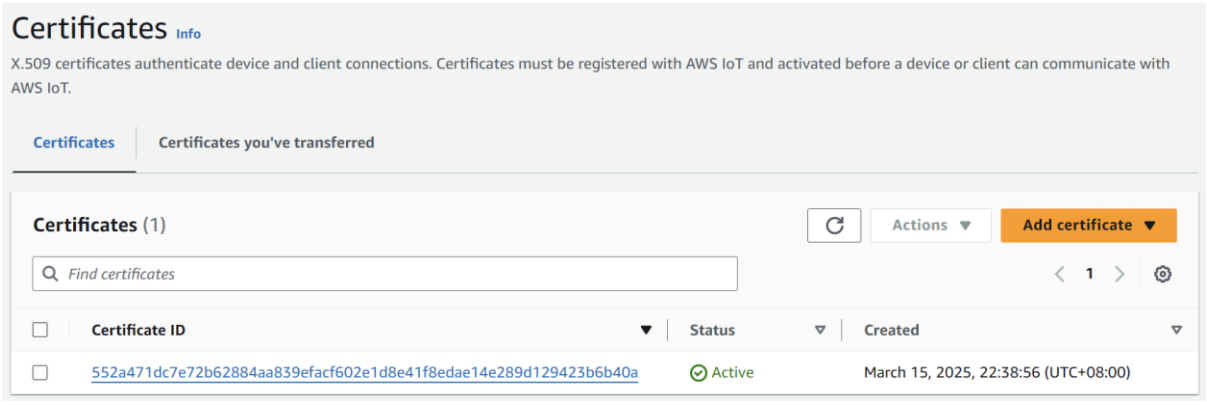
Private key file

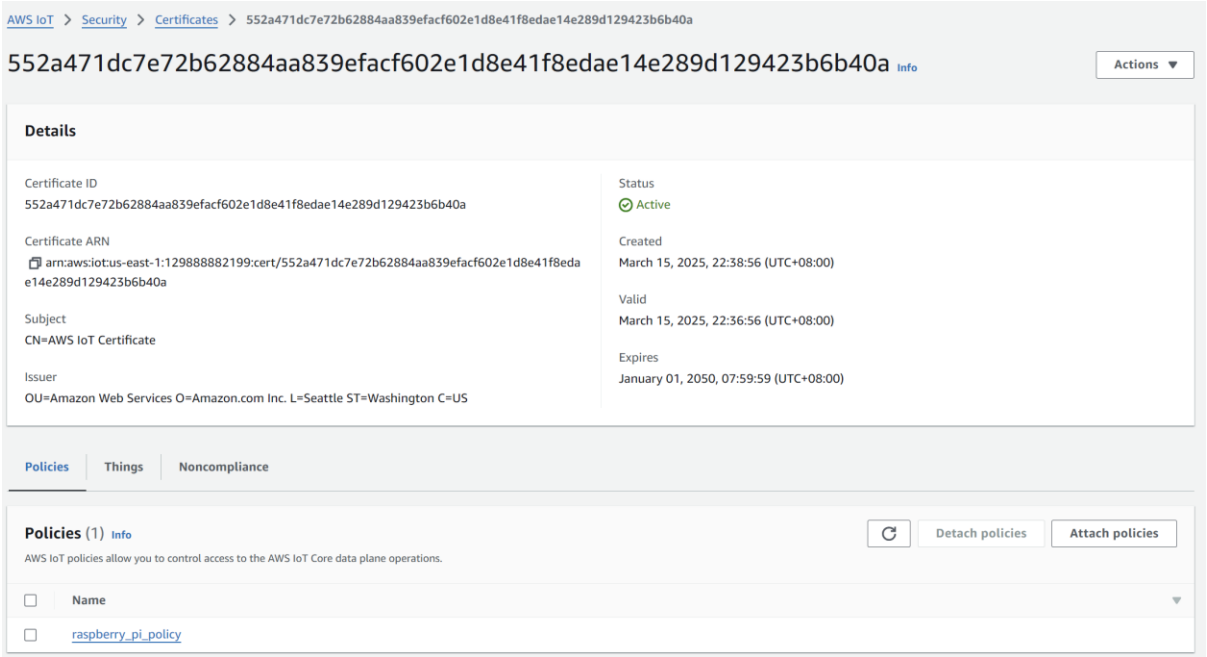
552a471dc7e72b62884aa83...b6b40a-private.pem.key

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.
Filter things by: name, type, group, billing, or searchable attribute.
1
Name
Thing type
raspberry_pi_1
raspberry_pi


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
Find certificates
1
Certificate ID
Status
Created
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
AWS IoT policies allow you to control access to the AWS IoT Core data plane operations.
Detach policies
Attach policies
Name
raspberry_pi_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 $
```

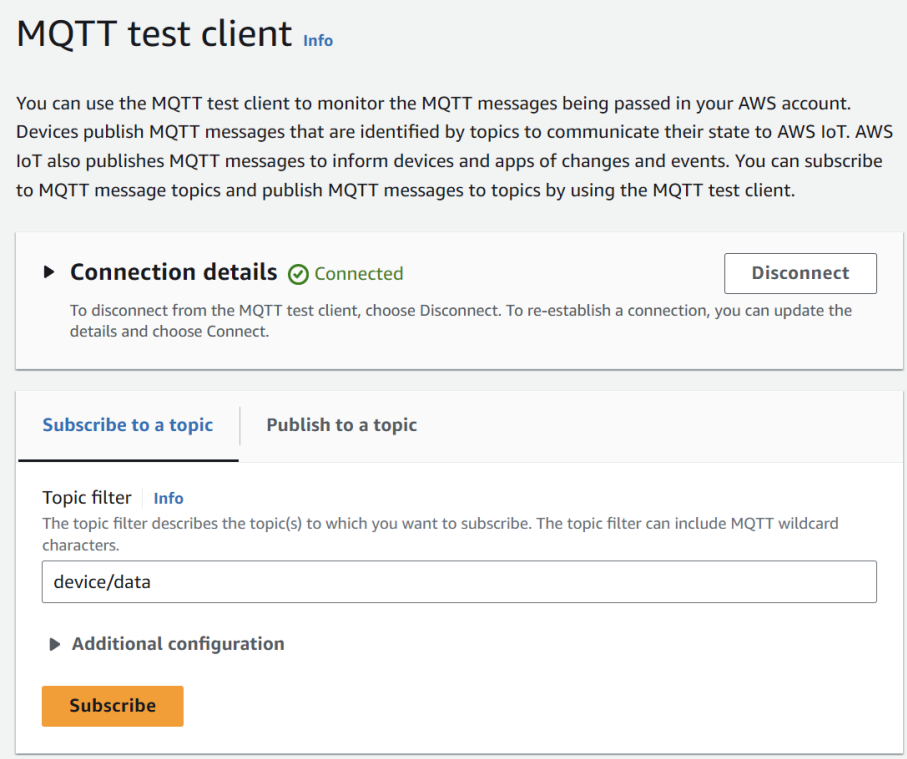
Copy files from local machine to Raspberry Pi.

```
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
```

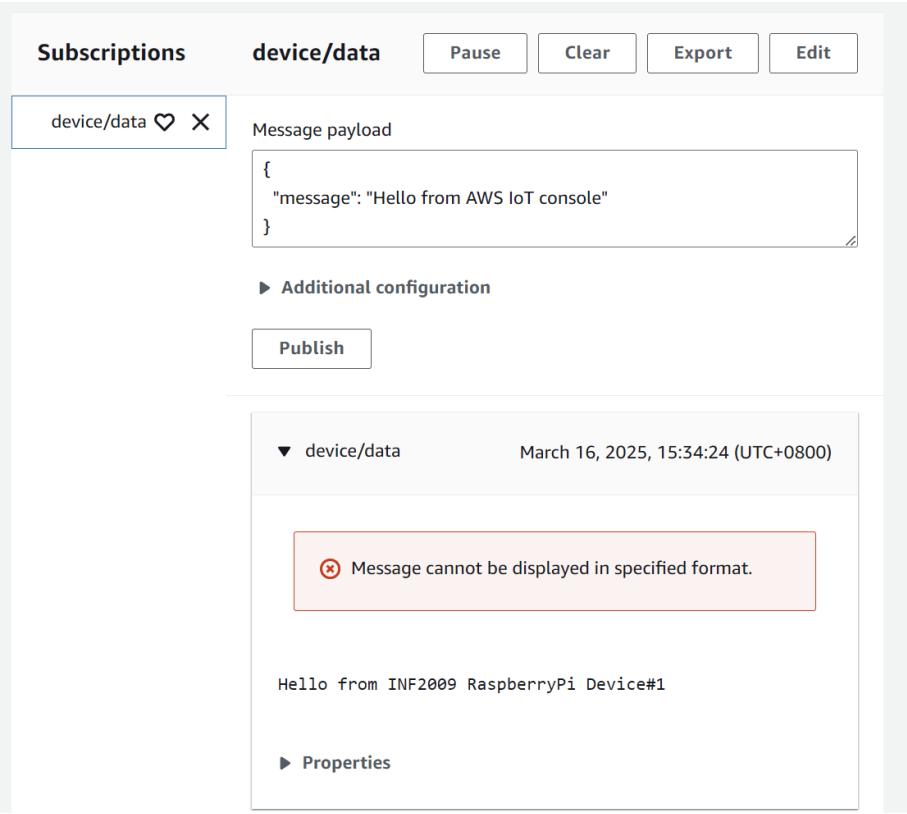
Update endpoint domain name.

```
  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_vers>
client.tls_insecure_set(True)
client.connect("a2s61kijkghd8e-ats.iot.us-east-1.amazonaws.com", 8883, 60) #Copy end point from your AWS IoT Core Conso>


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-6 Copy
```

Subscribe to topic.



Receive messages from Raspberry Pi.



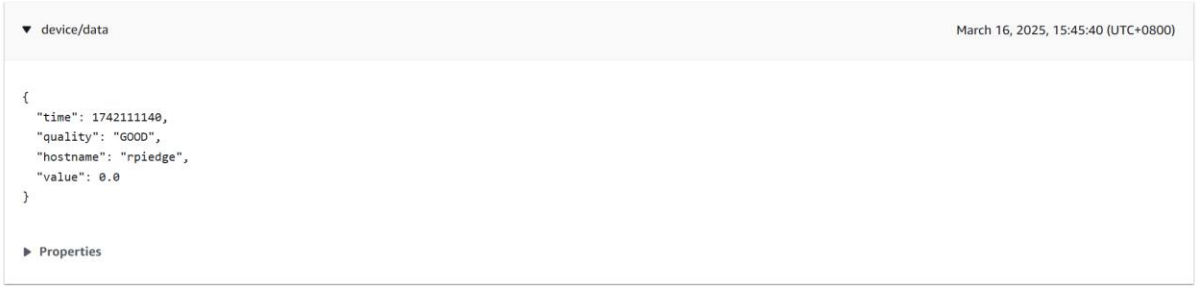
4. Ingest Reat-time Data into DynamoDB via IoT Rule

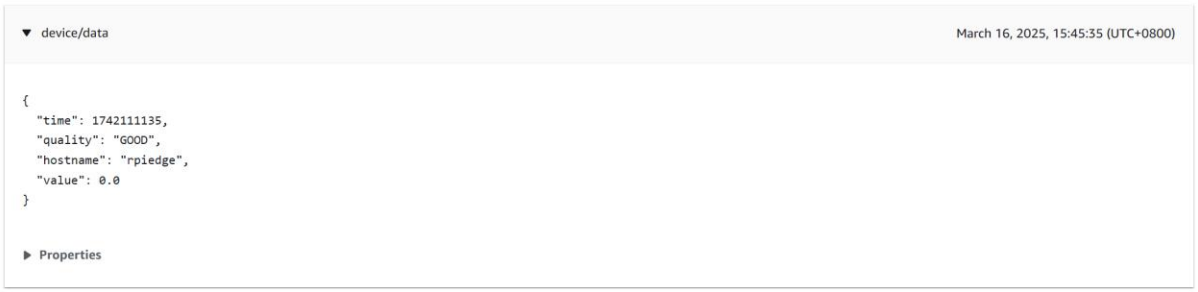
Update sample code to send CPU utilisation and other information.

```
 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("a2s61kijkghd8e-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()
```

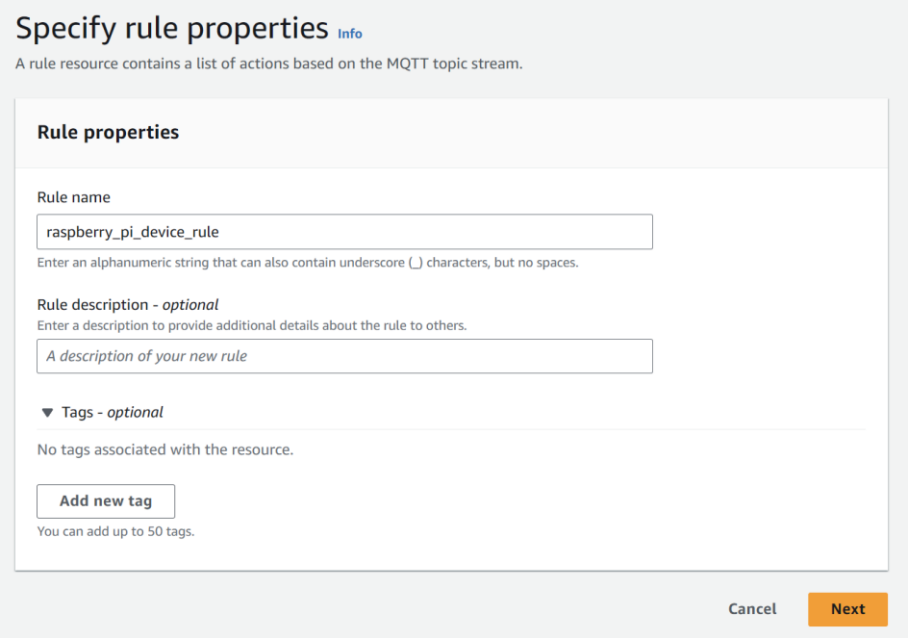
Test publish and receive message.

```
(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
}
```

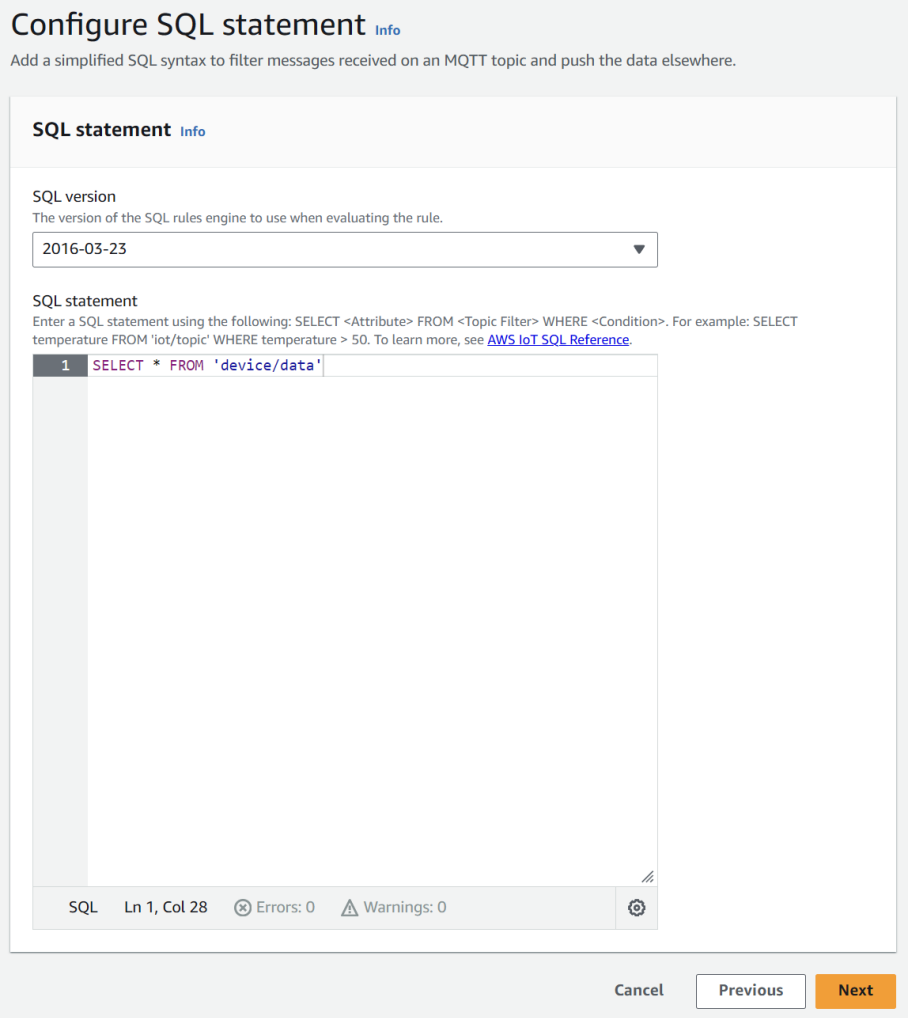




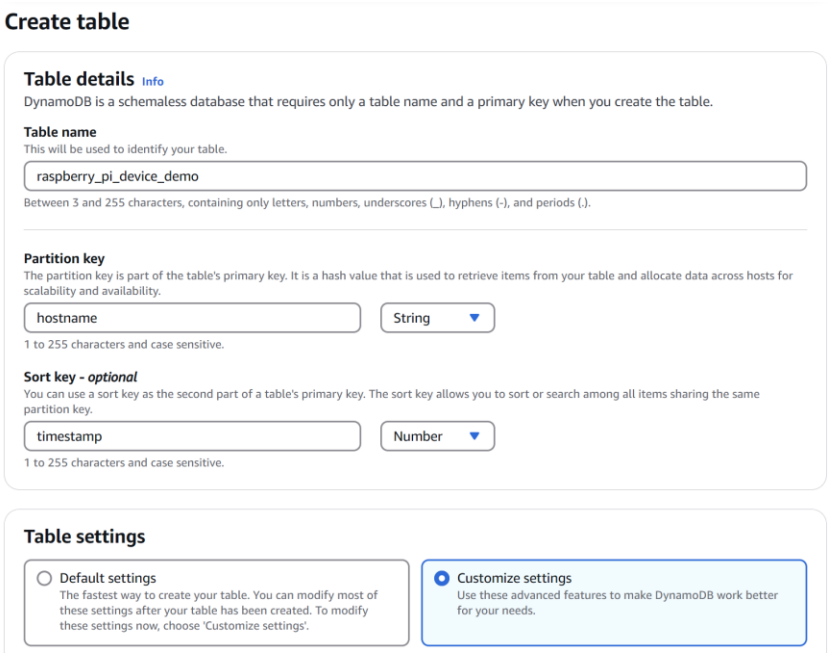
Create rule.



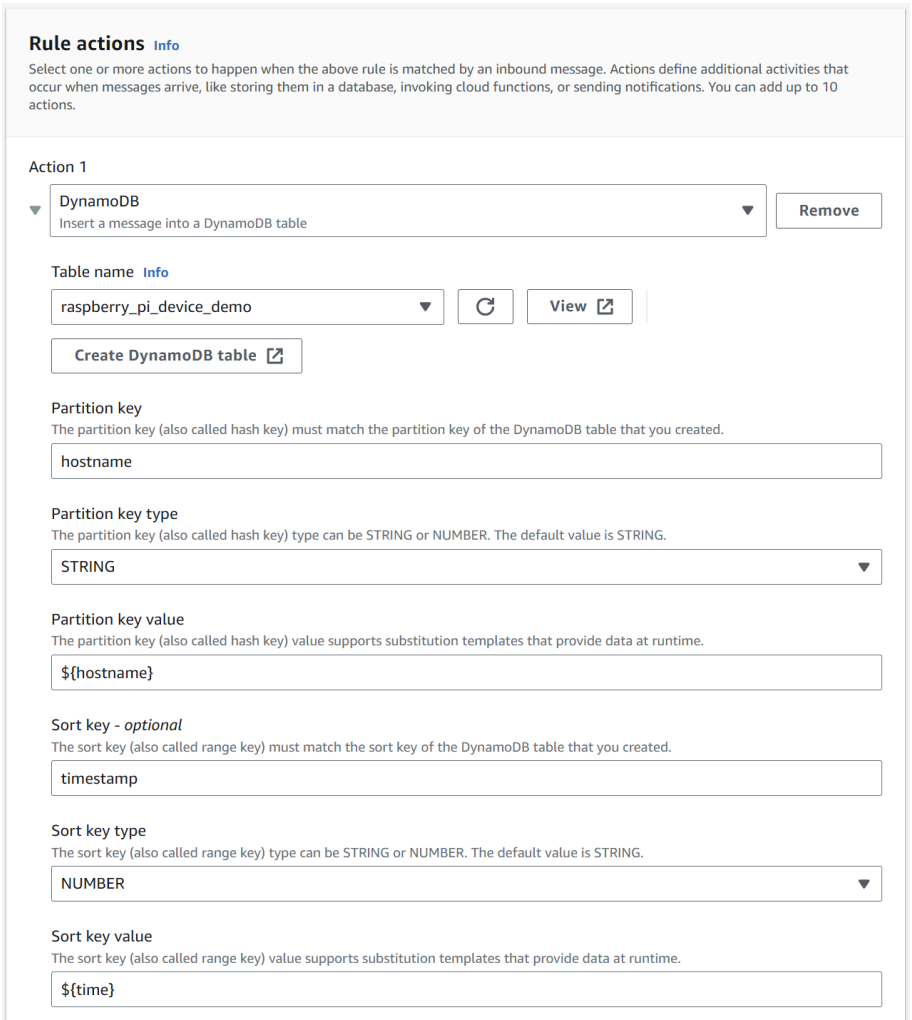
Configure SQL statement.



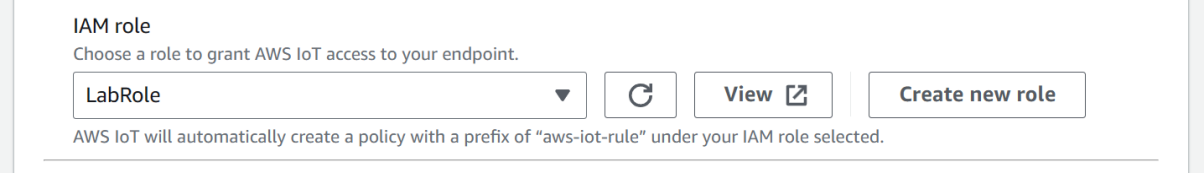
Create DynamoDB table.



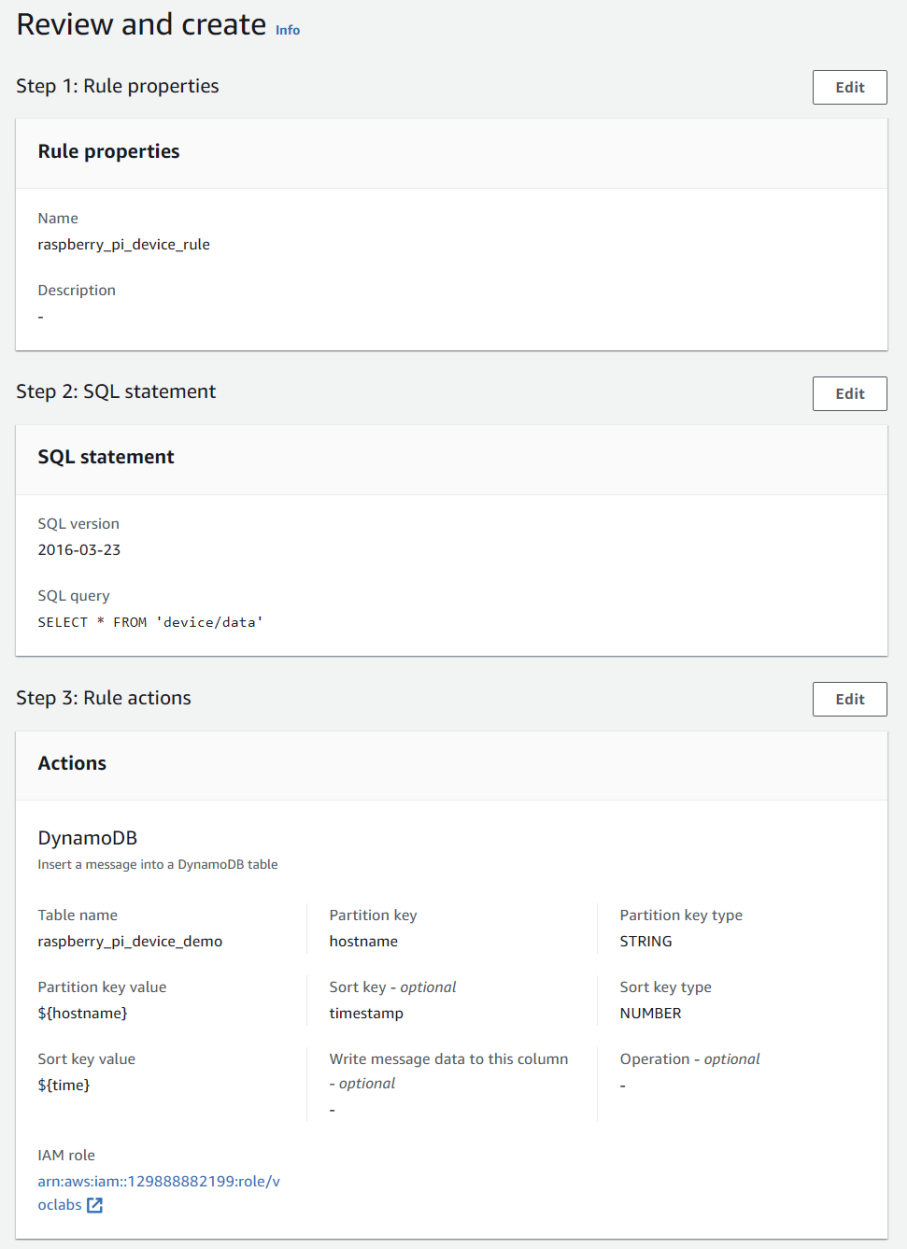
Configure rule actions.



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



Review configuration and create rule.



View messages inserted into DynamoDB.