

## Group 13

37 Mikil Lalwani  
56 Nilay Pophalkar  
58 Sanskruti Punyarthi  
61 Shree Samal

### CA3 - Assignment 3 :

**Implement Edge to cloud Protocols (MQTT and COAP) using a dummy data set.**

## MQTT

### Step 1: Search for IOT Core

### Step 2: Create a policy add add policy rule with

**Policy effect = allow**

**Policy action = \***

**Policy Resource = \***

The screenshot shows the AWS IoT Policy creation interface. On the left, there's a sidebar with navigation links like Monitor, Connect, Test, and Manage. The main area has a 'Policy name' input field containing 'policy'. Below it is a 'Policy statements' tab. Under the 'Policy document' section, there's a table with columns for 'Policy effect' (set to 'Allow'), 'Policy action' (set to '\*'), and 'Policy resource' (set to a wildcard resource). A 'Builder' button is available for creating the policy document. At the bottom right, there are 'Cancel' and 'Create' buttons.

The screenshot shows the AWS IoT Things interface. The left sidebar includes links for Device Advisor, MQTT test client, and various management options like All devices, Greengrass devices, LPWAN devices, and Software packages. The main area shows a breadcrumb path: AWS IoT > Manage > Things > Create things > Create single thing. It's step 3 of 3, titled 'Attach policies to certificate - optional'. A table lists a single policy named 'policy'. At the bottom right, there are 'Cancel', 'Previous', and 'Create thing' buttons.

### Step 3: Click on create thing and add the thing name

The screenshot shows the 'Create things' step 1 interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, and search bar. Below it, a horizontal menu bar includes Cloud9, IAM, EC2, Elastic Beanstalk, AWS Amplify, Lambda, S3, and API Gateway. On the right, it shows 'N. Virginia' and 'Mikil Lalwani'. The main content area has a title 'Create things' with an 'Info' link. A descriptive text explains that a thing resource is a digital representation of a physical device or logical entity in AWS IoT. Below this, a section titled 'Number of things to create' contains two options: 'Create single thing' (selected) and 'Create many things'. The 'Create single thing' option includes a sub-instruction: 'Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.' At the bottom right are 'Cancel' and 'Next' buttons.

The screenshot shows the 'Specify thing properties' step 2 interface. At the top, it has the same navigation and menu bar as the previous screen. The main content area is titled 'Specify thing properties' with an 'Info' link. It includes a descriptive text about thing resources. On the left, there's a sidebar with three sections: 'Step 1 Specify thing properties' (selected), 'Step 2 - optional Configure device certificate' (disabled), and 'Step 3 - optional Attach policies to certificate' (disabled). The main panel is titled 'Thing properties' with an 'Info' link. It has a 'Thing name' input field containing 'thing1'. Below it is a note: 'Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.' Under 'Additional configurations', there's a list of optional items: 'Thing type - optional', 'Searchable thing attributes - optional', 'Thing groups - optional', 'Billing group - optional', and 'Packages and versions - optional'. The right side of the screen shows a light gray sidebar with a help icon.

## Step 4: Click on auto-generate a certificate

The screenshot shows the 'Configure device certificate - optional' step. It includes a note about device requirements and three options: 'Auto-generate a new certificate (recommended)', 'Use my certificate', 'Upload CSR', and 'Skip creating a certificate at this time'. The 'Auto-generate a new certificate' option is selected. Navigation buttons 'Cancel', 'Previous', and 'Next' are at the bottom.

## Step 5: Attach policy and click on create thing

The screenshot shows the 'Attach policies to certificate - optional' step. It notes that AWS IoT policies grant or deny access to resources. A 'Policies (0)' section shows a search bar and a 'Create policy' button. Below it, a message says 'No policies' and 'No policies could be found in us-east-1'. Navigation buttons 'Cancel', 'Previous', and 'Create thing' are at the bottom.

This screenshot shows the same 'Attach policies to certificate - optional' step, but with a policy attached. The 'Policies (1/1)' section shows a single policy named 'policy' with a checked checkbox. Navigation buttons 'Cancel', 'Previous', and 'Create thing' are at the bottom.

## Step 6: Download all certificates and keys at the same location

The screenshot shows the AWS IoT Things console. At the top, there is a file download dialog with the following contents:

File	Created	Type
private.pem.key	06-10-2023 13:15	KEY File
public.pem.key	06-10-2023 13:15	KEY File
device-certificate.pem.crt	06-10-2023 13:15	Security Certificate
AmazonRootCA1.pem	06-10-2023 13:15	PEM File
AmazonRootCA3.pem	06-10-2023 13:15	PEM File

Below the download dialog, the AWS IoT Things console interface is visible. It shows two success messages in the notification bar:

- You successfully created thing thing1.
- You successfully created certificate 944dafb192f711c4155960f0096f9b89818f11395d8d30d6487feaac758dd214.

The main pane displays the 'Things' list with one item named 'thing1'. The sidebar on the left shows navigation options like 'All devices', 'Things', 'Thing groups', etc.

## Step 7: Clone AWS-MQTT-SDK git repo using the below link in the same folder where you have downloaded the keys and certificate

```
-git clone https://github.com/aws/aws-iot-device-sdk-python.git
```

## Step 8: After cloning Run below commands

```
-cd aws-iot-device-sdk-python  
-python setup.py install (2)
```

## Step 9: In main directory save two codes one to publish message to MQTT server and one to subscribe

### Publish.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  
import sys
```

```
myMQTTClient = AWSIoTMQTTClient("thing1") #Enter your things name
```

```
myMQTTClient.configureEndpoint("a1ek9vqv3jnbud-ats.iot.us-east-1.amazonaws.com", 8883)  
myMQTTClient.configureCredentials("./AmazonRootCA1.pem","./private.pem.key",  
"./device-certificate.pem.crt")
```

```
myMQTTClient.connect()
```

```
print("Client Connected")

msg = "Sample data from the device"
topic = "general/inbound"
myMQTTClient.publish(topic, msg, 0)
print("Message Sent")

myMQTTClient.disconnect()
print("Client Disconnected")
```

### Subscribe.py

```
import time

def customCallback(client,userdata,message):
    print("callback came... ")
    print(message.payload)

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

myMQTTClient = AWSIoTMQTTClient("thing1")
myMQTTClient.configureEndpoint("a1ek9vqv3jnbud-ats.iot.us-east-1.amazonaws.com", 8883)
myMQTTClient.configureCredentials("./AmazonRootCA1.pem","./private.pem.key",
                                 "./device-certificate.pem.crt")

myMQTTClient.connect()
print("Client Connected")

myMQTTClient.subscribe("general/outbound", 1, customCallback)
print('waiting for the callback. Click to conntinue... ')
x = input()

myMQTTClient.unsubscribe("general/outbound")
print("Client unsubscribed")

myMQTTClient.disconnect()
```

```
print("Client Disconnected")
```

**Step 10:** In both of above codes replace endpoint with your MQTT test client endpoint

The screenshot shows the 'MQTT test client' interface. At the top, it says 'MQTT test client [Info](#)'. Below that, a message states: 'You can use the MQTT test client to monitor their state to AWS IoT. AWS IoT also publishes messages to topics by using the MQTT test cli'. Under the heading 'Connection details', it says 'You can update the connection details by choosing the connection profile'. It displays the following information:

- Client ID:** iotconsole-5d484cb2-d2b5-424f-b52f-ec5dee87d535
- Endpoint:** a1ek9vqv3jnbud-ats.iot.us-east-1.amazonaws.com

**Step 11:** In MQTT go to Subscribe to a topic and type ‘general/inbound’ and click subscribe don’t close this page

The screenshot shows the 'AWS IoT' service in the AWS console. The left sidebar has sections for 'Monitor', 'Connect' (with 'Connect one device' and 'Connect many devices'), 'Test' (with 'Device Advisor' and 'MQTT test client' which is selected), and 'Manage' (with 'All devices', 'Things', 'Thing groups', 'Thing types', and 'CloudWatch Metrics'). The main area is titled 'Subscribe to a topic' and 'Publish to a topic'. It includes a 'Topic filter' input field containing 'general/inbound', an 'Additional configuration' section, and a large orange 'Subscribe' button. Below this, there's a 'Subscriptions' table with a single row: 'You have no topic subscriptions.' and 'Subscribe or select a topic to view incoming messages.'

## Step 12: After that run publish.py and go to subscribe page

A screenshot of a code editor window titled "MQTT". The left sidebar shows an "EXPLORER" view with files: aws-iot-device-sdk-python, publish.py, and subscribe.py. The main area shows the content of publish.py:

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import sys

myMQTTClient = AWSIoTMQTTClient("thing1") #Enter your things name
myMQTTClient.configureEndpoint("a1ek9vqv3jnbud-ats.iot.us-east-1.amazonaws.com", 8883)
myMQTTClient.configureCredentials("./AmazonRootCA1.pem","./private.pem.key", "./device-certificate.pem.crt")

```

The terminal tab at the bottom shows the output of running the script:

```
PS E:\Mikil\BE\Lab\IOE Lab\MQTT> python -u "e:\Mikil\BE\Lab\IOE Lab\MQTT\publish.py"
Client Connected
Message Sent
Client Disconnected
PS E:\Mikil\BE\Lab\IOE Lab\MQTT>
```

## Step 13: You can see message on subscribe page

A screenshot of the AWS IoT Subscriptions page. The top navigation bar shows "Services" and "Subscriptions". The main area shows a subscription named "general/inbound". The "Message payload" section contains the following JSON:

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

The "Additional configuration" section has a "Publish" button. Below the subscription details, there is a message box stating "Message cannot be displayed in specified format." and a "Properties" section.

## Step 14: Then run subscribe.py

A screenshot of a code editor window titled "MQTT". The left sidebar shows an "EXPLORER" view with files: aws-iot-device-sdk-python, publish.py, and subscribe.py. The main area shows the content of subscribe.py:

```
import time

def customCallback(client, userdata, message):
    print("callback came...")
    print(message.payload)

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

```

The terminal tab at the bottom shows the output of running the script:

```
PS E:\Mikil\BE\Lab\IOE Lab\MQTT> python -u "e:\Mikil\BE\Lab\IOE Lab\MQTT\publish.py"
Client Connected
Message Sent
Client Disconnected
PS E:\Mikil\BE\Lab\IOE Lab\MQTT> python -u "e:\Mikil\BE\Lab\IOE Lab\MQTT\subscribe.py"
Client Connected
waiting for the callback. Click to continue...
```

## Step 15: Go to publish to topic and type general/outbound and click on publish

The screenshot shows the AWS IoT Publish interface. On the left sidebar, under the 'Test' section, 'MQTT test client' is selected. In the main area, the 'Publish to a topic' tab is active. The 'Topic name' field contains 'general/outbound'. The 'Message payload' field contains the JSON message: { "message": "Hello from AWS IoT console" }. Below the message payload is an 'Additional configuration' section with a 'Publish' button. At the bottom, there is a 'Subscriptions' section for 'general/inbound' with buttons for Pause, Clear, Export, and Edit.

## Step 16: After clicking on publish go to terminal and you can see the message

The screenshot shows a terminal window with the title bar 'MQTT'. The terminal displays the following Python code and its execution:

```
publish.py
import time
def customCallback(client, userdata, message):
    print("callback came...")
    print(message.payload)
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
PORTS COMMENTS PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT
PS E:\Mikil\BE\Lab\IOE Lab\MQTT> python -u "e:\Mikil\BE\Lab\IOE Lab\MQTT\publish.py"
Client Connected
Message Sent
Client Disconnected
PS E:\Mikil\BE\Lab\IOE Lab\MQTT> python -u "e:\Mikil\BE\Lab\IOE Lab\MQTT\subscribe.py"
Client Connected
waiting for the callback. Click to continue...
callback came...
b'\n "message": "Hello from AWS IoT console"\n'
```

The screenshot shows the AWS IoT Publish interface. Under the 'Test' section, 'MQTT test client' is selected. In the main area, the 'Subscribe' tab is active. The 'Topic filter' field contains 'general/inbound'. Below it is an 'Additional configuration' section with a 'Subscribe' button. At the bottom, there is a 'Subscriptions' section for 'general/inbound' with a 'general/inbound' entry, and buttons for Pause, Clear, Export, and Edit. A timestamp at the bottom right indicates the message was received on October 06, 2023, at 16:31:10 (UTC+0530).

## HTTP:

### Code

```
import requests  
import argparse
```

```
# define command-line parameters  
parser = argparse.ArgumentParser(description="Send messages through an HTTPS connection.")  
parser.add_argument('--endpoint', required=True, help="Your AWS IoT data custom endpoint,  
not including a port. " + "Ex: \"abcdEXAMPLExyz-ats.iot.us-east-1.amazonaws.com\"")  
parser.add_argument('--cert', required=True, help="File path to your client certificate, in PEM  
format.")  
parser.add_argument('--key', required=True, help="File path to your private key, in PEM  
format.")  
parser.add_argument('--topic', required=True, default="test/topic", help="Topic to publish  
messages to.")  
parser.add_argument('--message', default="Hello World!", help="Message to publish. " +  
"Specify empty string to publish nothing.")  
  
# parse and load command-line parameter values  
args = parser.parse_args()  
  
# create and format values for HTTPS request  
publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'  
publish_msg = args.message.encode('utf-8')  
  
# make request  
publish = requests.request('POST',  
    publish_url,  
    data=publish_msg,  
    cert=[args.cert, args.key])  
  
# print results  
print("Response status: ", str(publish.status_code))  
if publish.status_code == 200:
```

```
print("Response body:", publish.text)
```

**Step 1:** Change your endpoint and device certificate and private key file name in the below code  
`python http_code.py --endpoint "a1ek9vqv3jnbud-ats.iot.us-east-1.amazonaws.com" --cert  
"./device-certificate.pem.crt" --key "./private.pem.key" --topic "test/topic" --message  
"Test37"`

The screenshot shows a code editor interface with several tabs open. The current tab is `http_code.py`, which contains the following code:

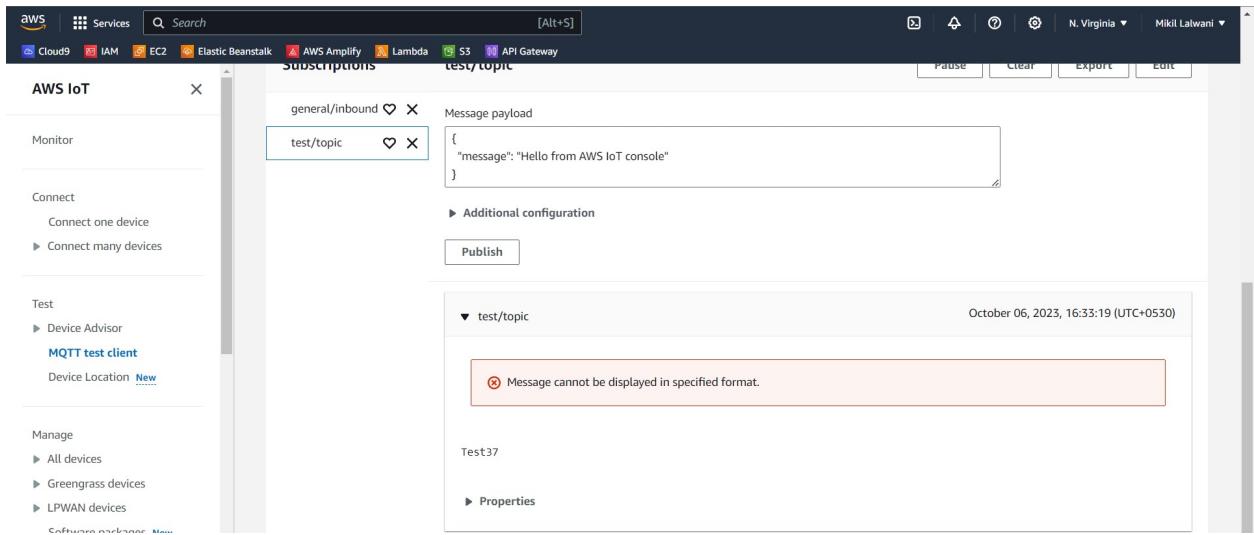
```
publish.py subscribe.py http_code.py X

14
15
16 # parse and load command-line parameter values
17 args = parser.parse_args()
18
19
20 # create and format values for HTTPS request
21 publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'
22 publish_msg = args.message.encode('utf-8')
23
24
```

Below the code editor, there are several icons for file operations like copy, paste, and search. At the bottom, there are tabs for PORTS, COMMENTS, PROBLEMS, DEBUG CONSOLE, TERMINAL, and OUTPUT. The TERMINAL tab is active, showing the following command and its output:

```
PS E:\Mikil\BE\Lab\IOE Lab\MQTT> python http_code.py --endpoint "a1ek9vqv3jnbud-ats.iot.us-east-1.amazonaws.com" --cert "./device-certificate.pem.crt" --key "./private.pem.key" --topic "test/topic" --message "Test37"
Response status: 200
Response body: {"message": "OK", "traceId": "ecd885d9-3329-6123-7f06-3e06f870c9e9"}
```

### **Step 2:** Check message received by subscribing to the topic



**Step 3:** On dashboard you can see both protocols has been implemented successfully

