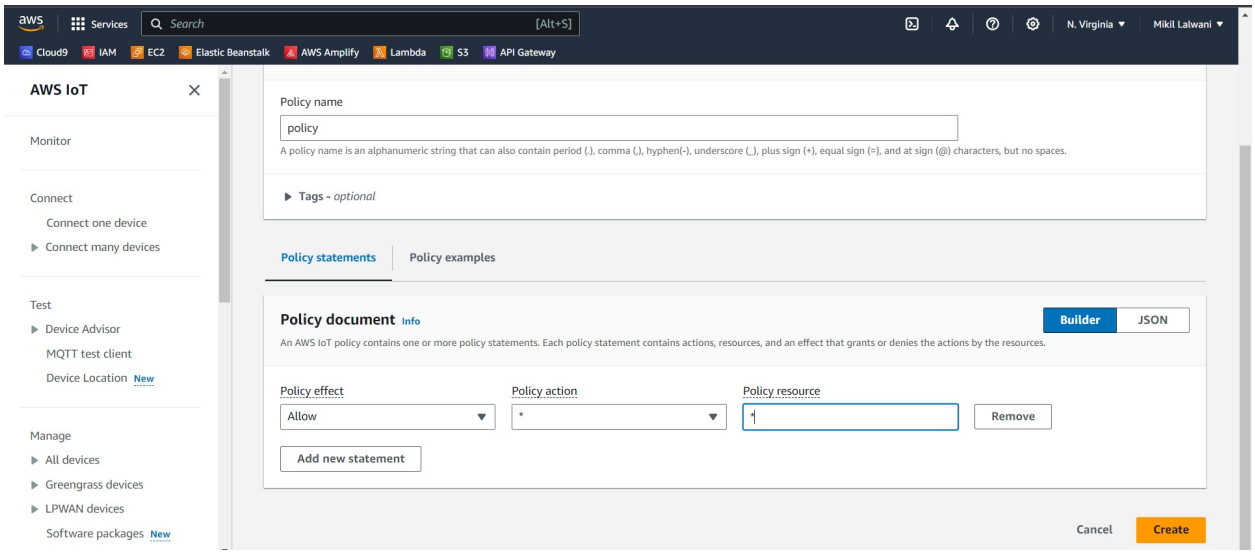
Group 13
37 Mikil Lalwani
56 Nilay Pophalkar
58 Sanskruti Punyarthi
61 Shree Samal

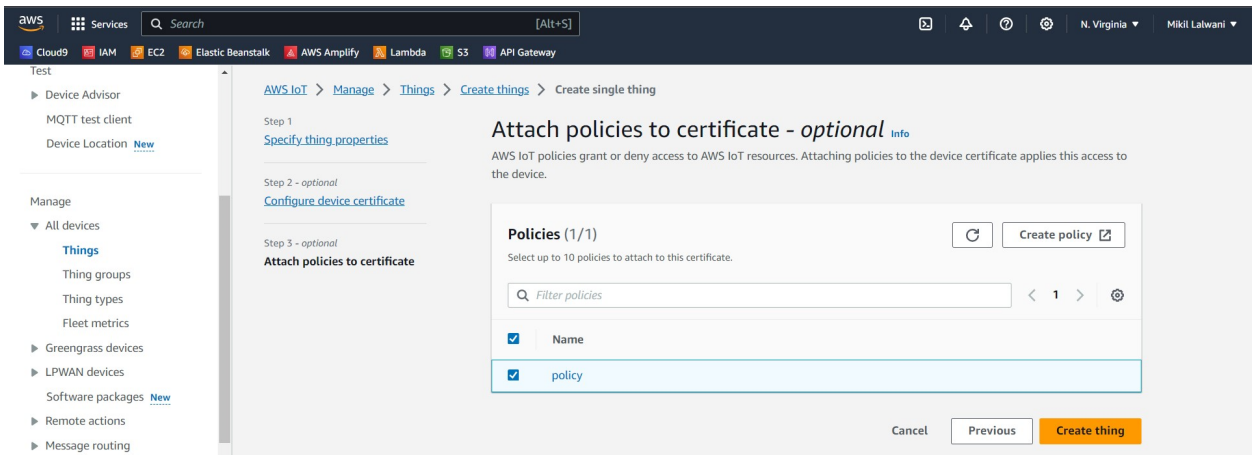
CA3 - Assignment 3 :
Implement Edge to cloud Protocols (MQTT, HTTP, Websockets) using a dummy data set.

MQTT

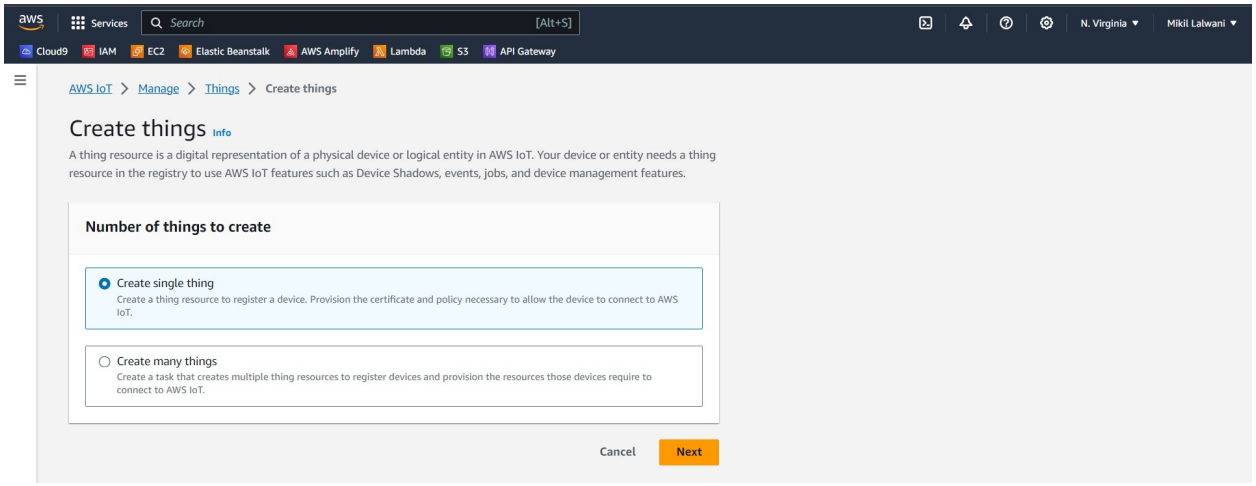
Step 1: Search for IOT Core

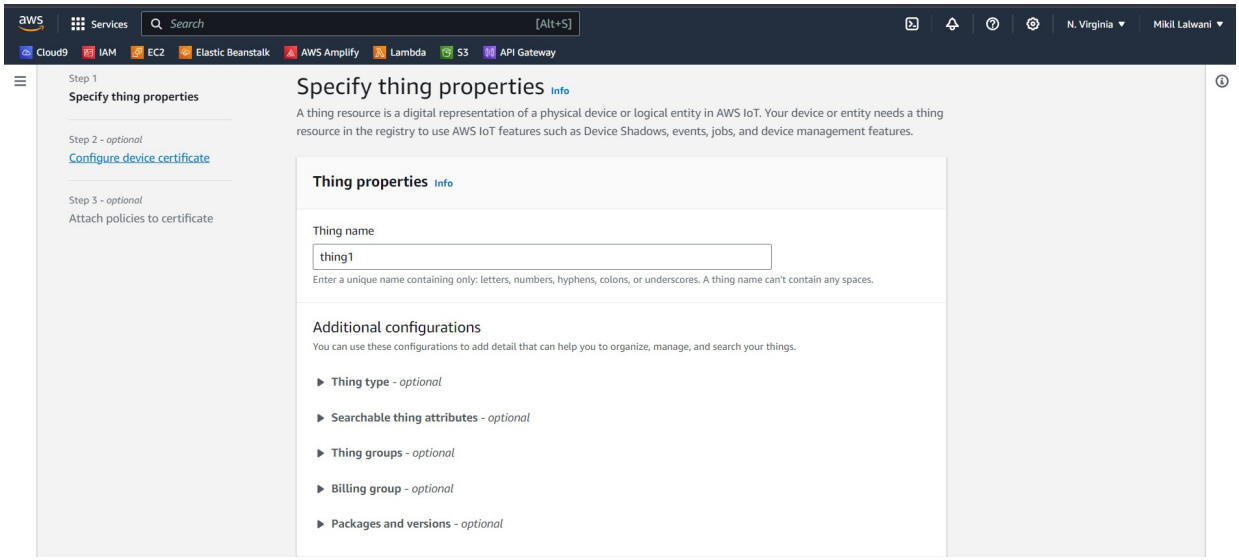
Step 2: Create a policy add add policy rule with
Policy effect = allow
Policy action = *
Policy Resource = *



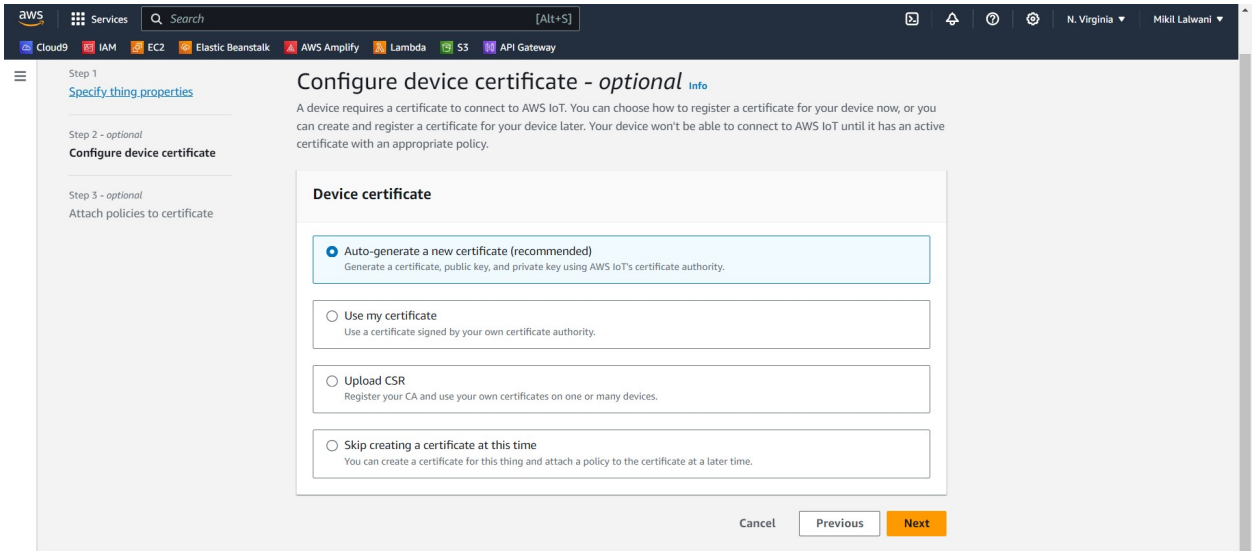


Step 3: Click on create thing and add the thing name

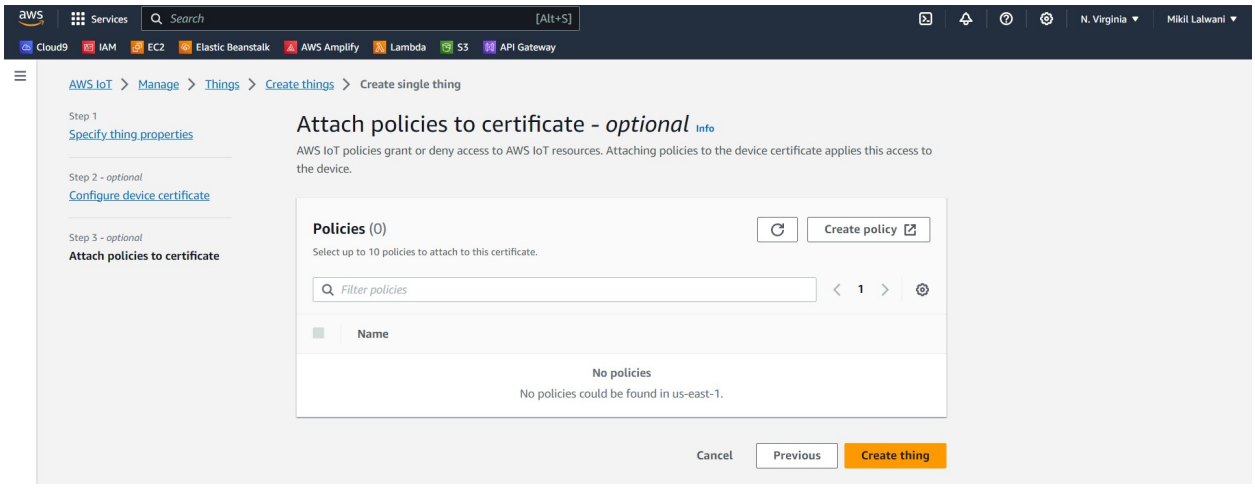


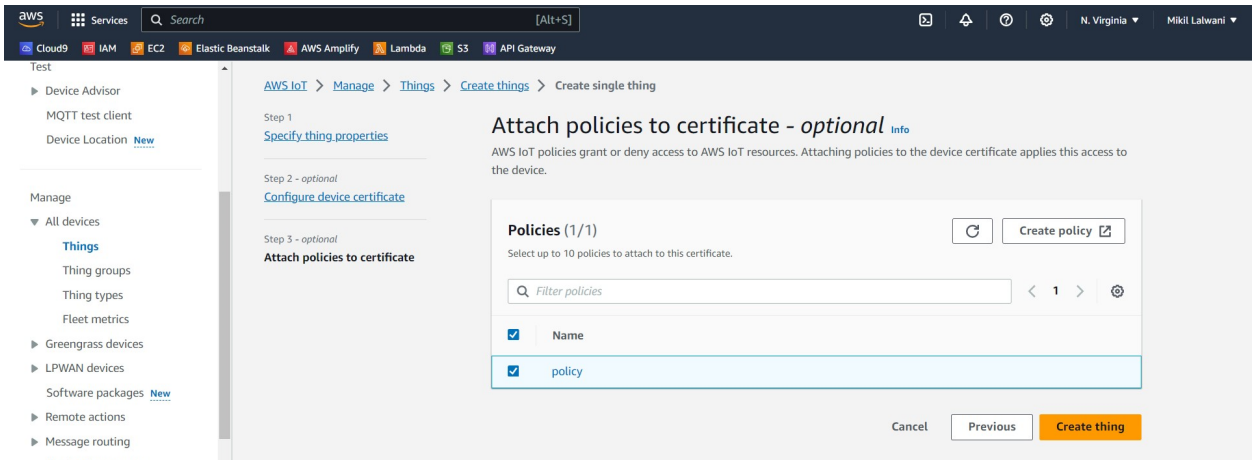


Step 4: Click on auto-generate a certificate

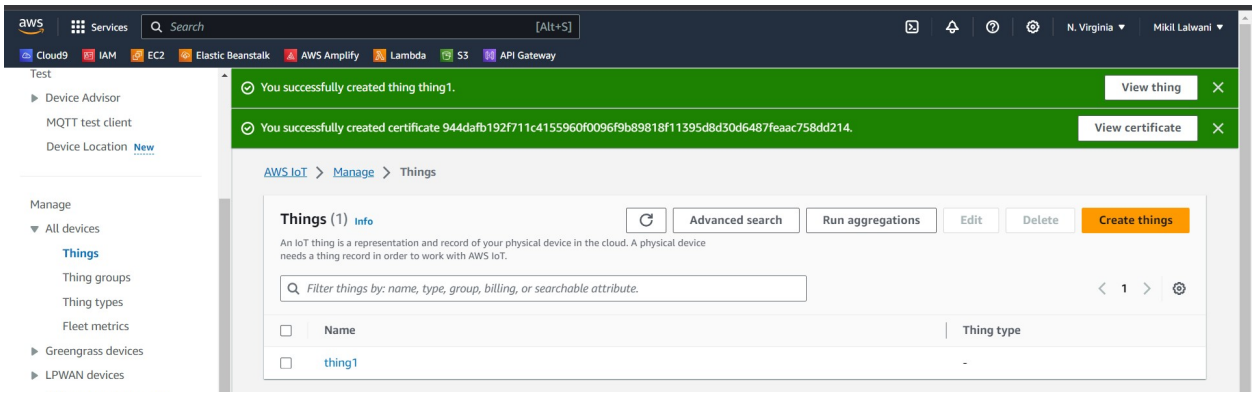


Step 5: Attach policy and click on create thing





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



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()
```

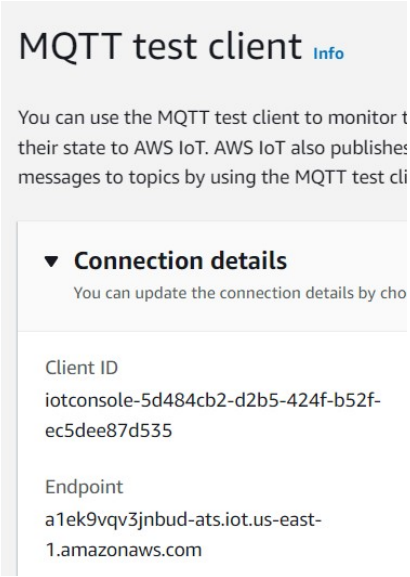
```
print("Client Connected")

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

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

Subscribe.py

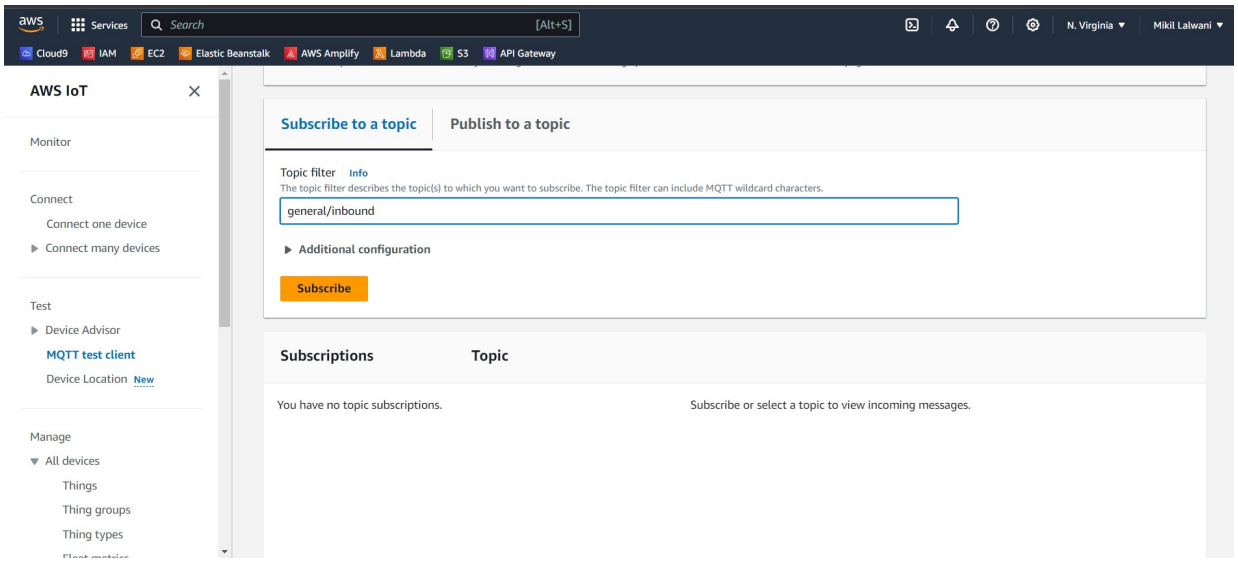
```
import time

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

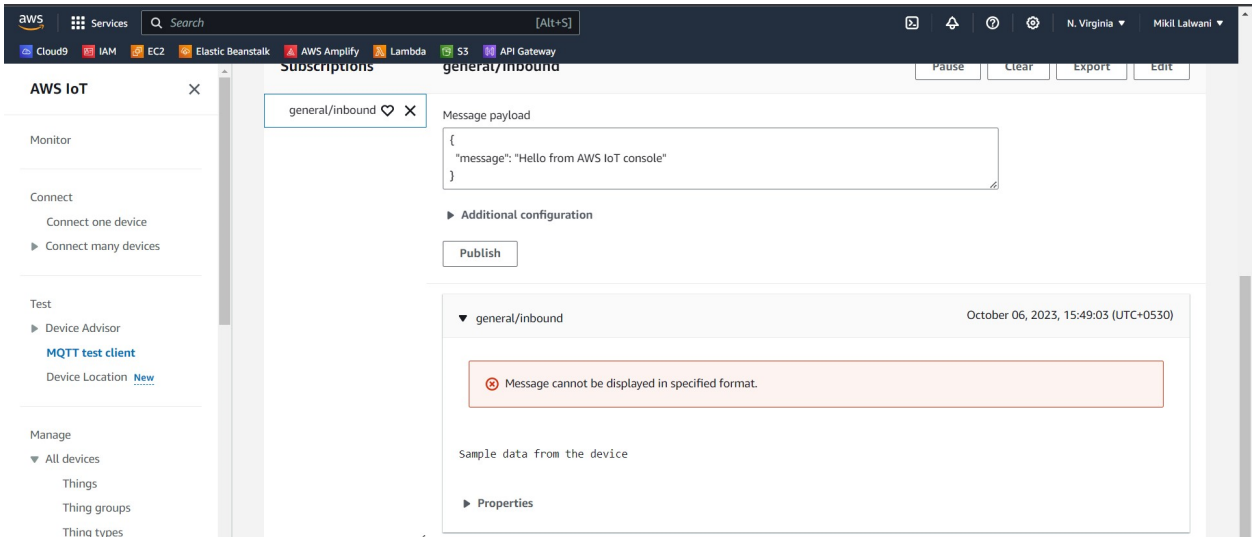


Step 11: In MQTT go to Subscribe to a topic and type ‘general/inbound’ and click subscribe don’t close this page

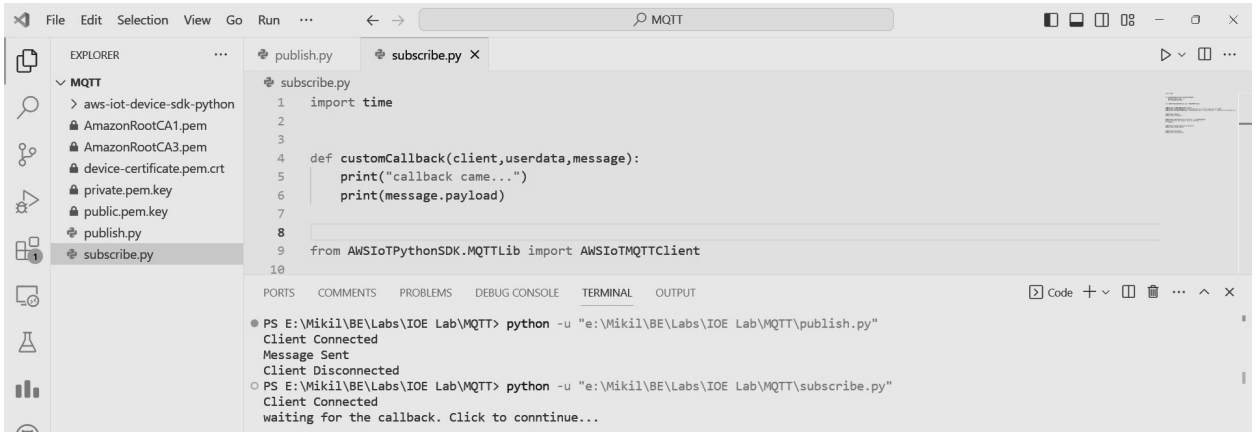


Step 12: After that run publish.py and go to subscribe page

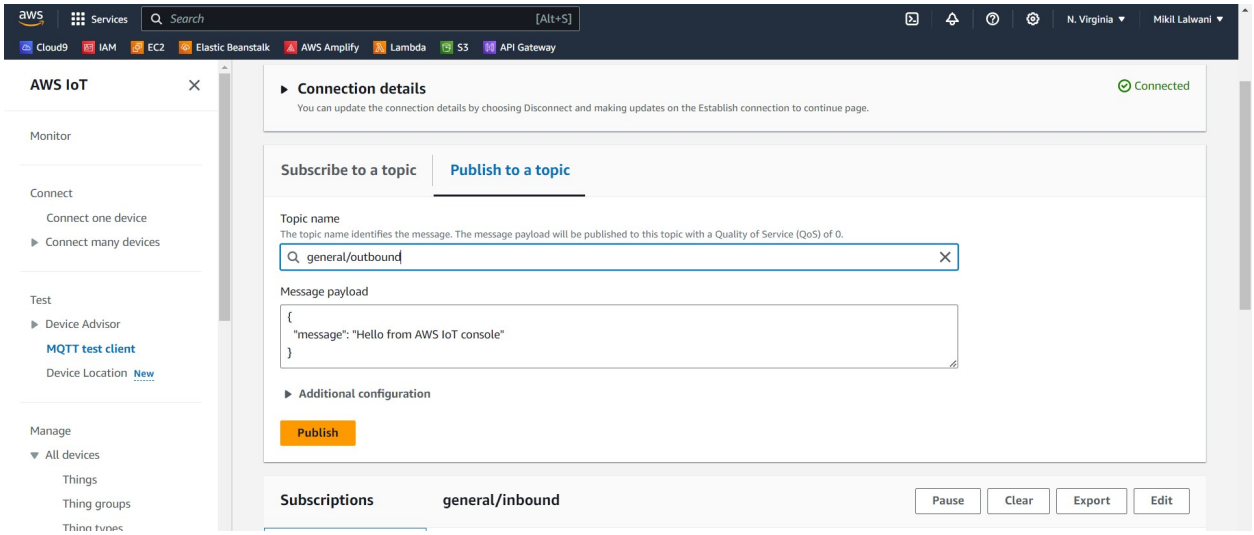
Step 13: You can see message on subscribe page



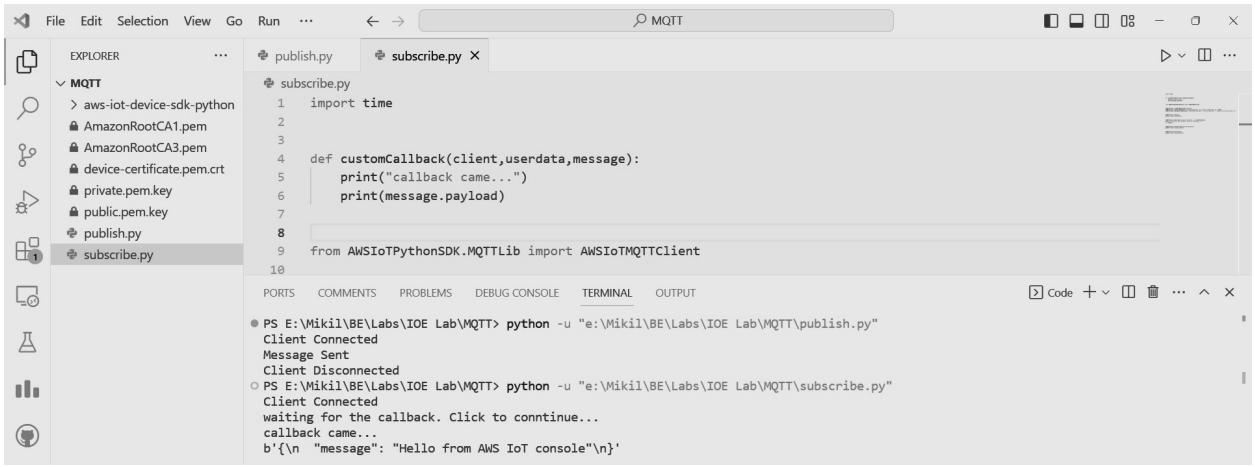
Step 14: Then run subscribe.py

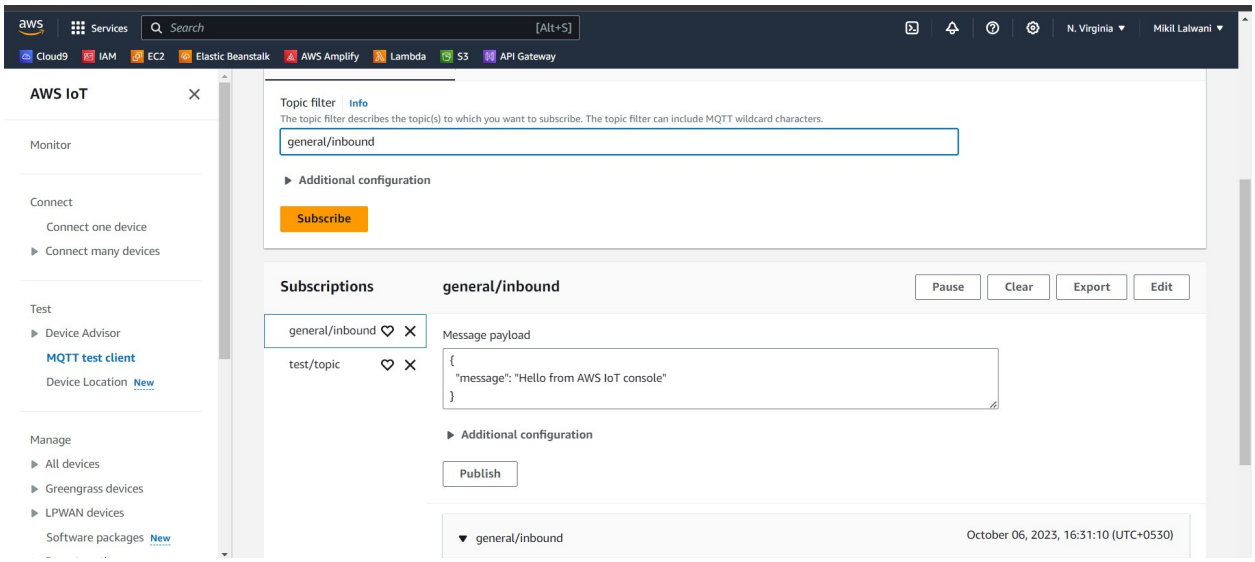


Step 15: Go to publish to topic and type general/outbound and click on publish



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





HTTP:

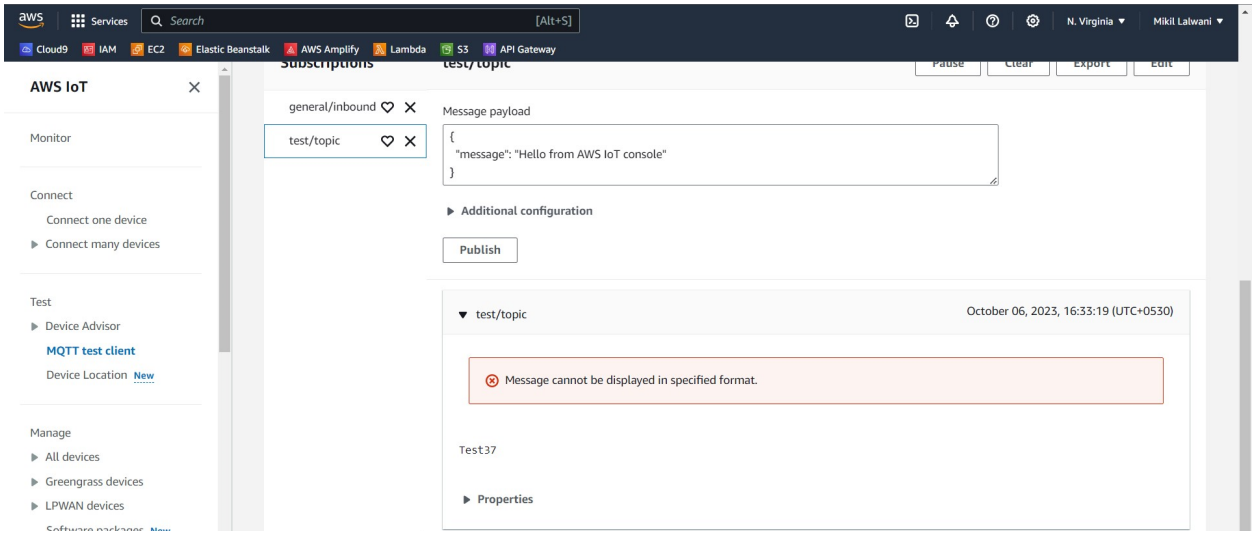
Code

```
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:
```

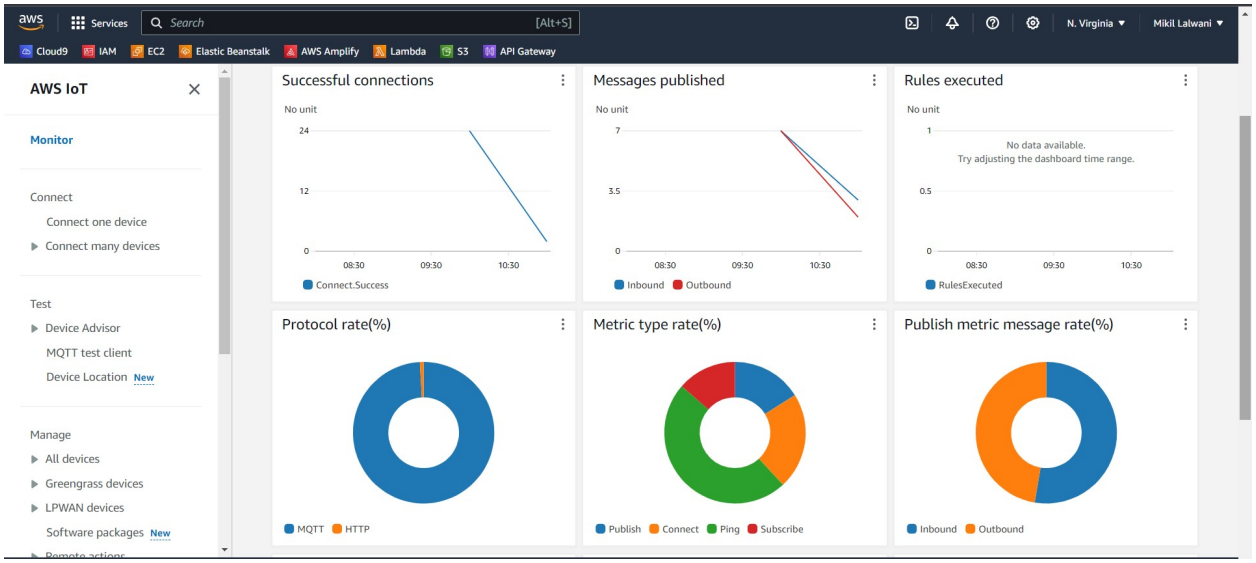
print("Response body:", publish.text)

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"**

Step 2: Check message received by subscribing to the topic

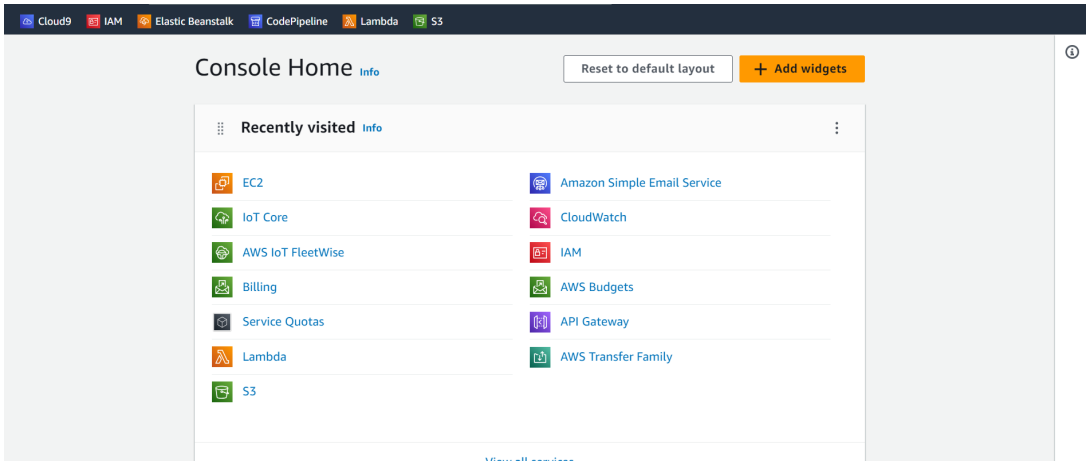


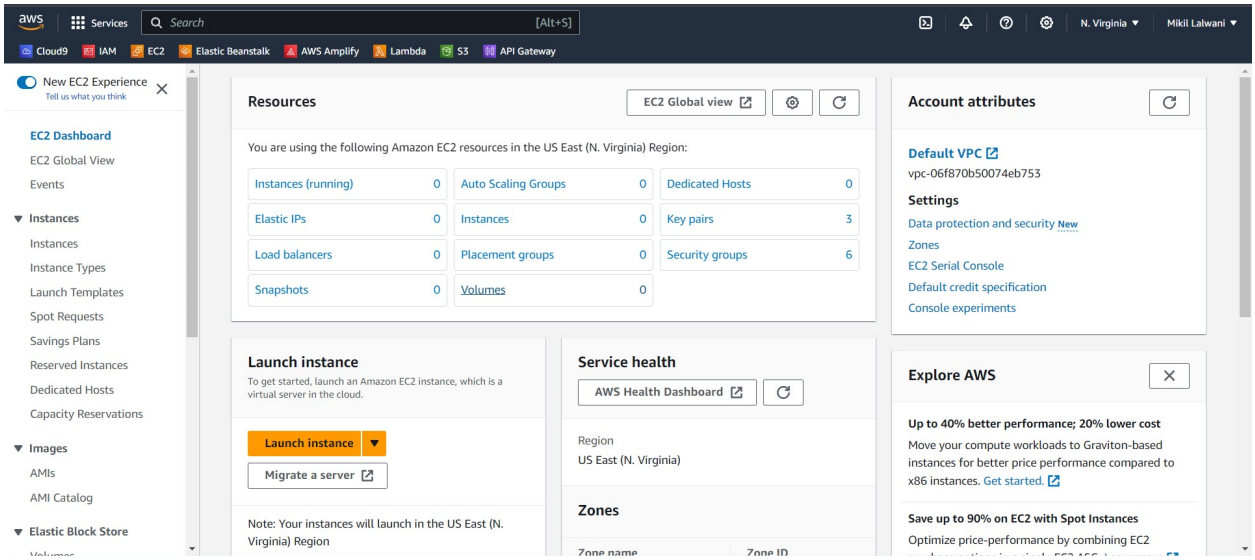
Step 3: On dashboard you can see both protocols has been implemented successfully



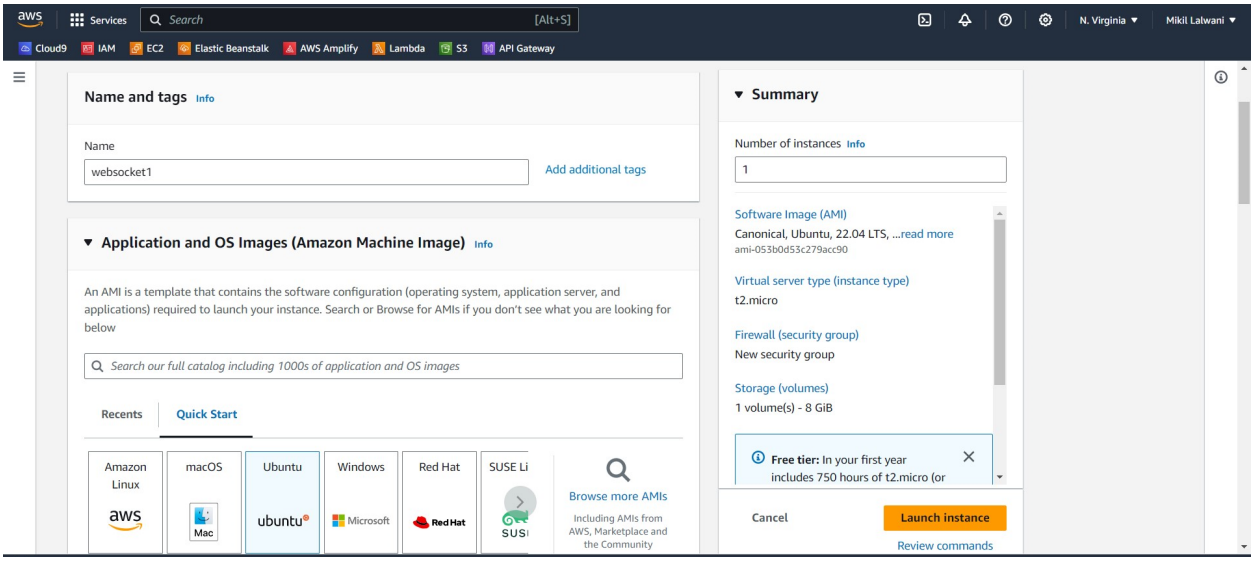
WebSockets:

Step 1: Go to the EC2 Console and create an EC2 instance.

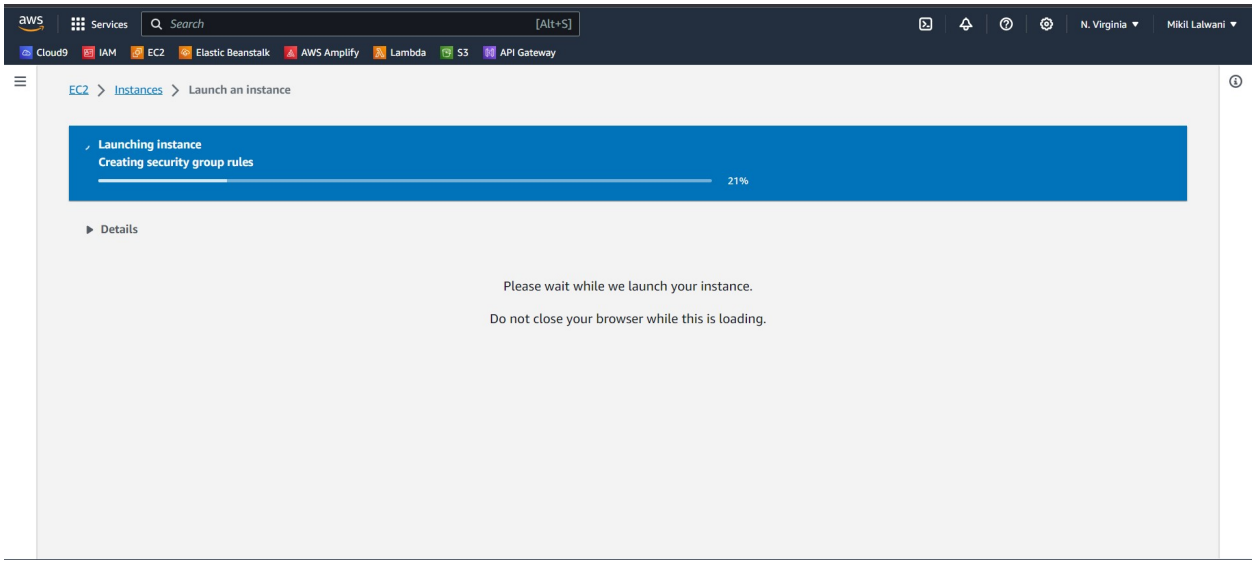


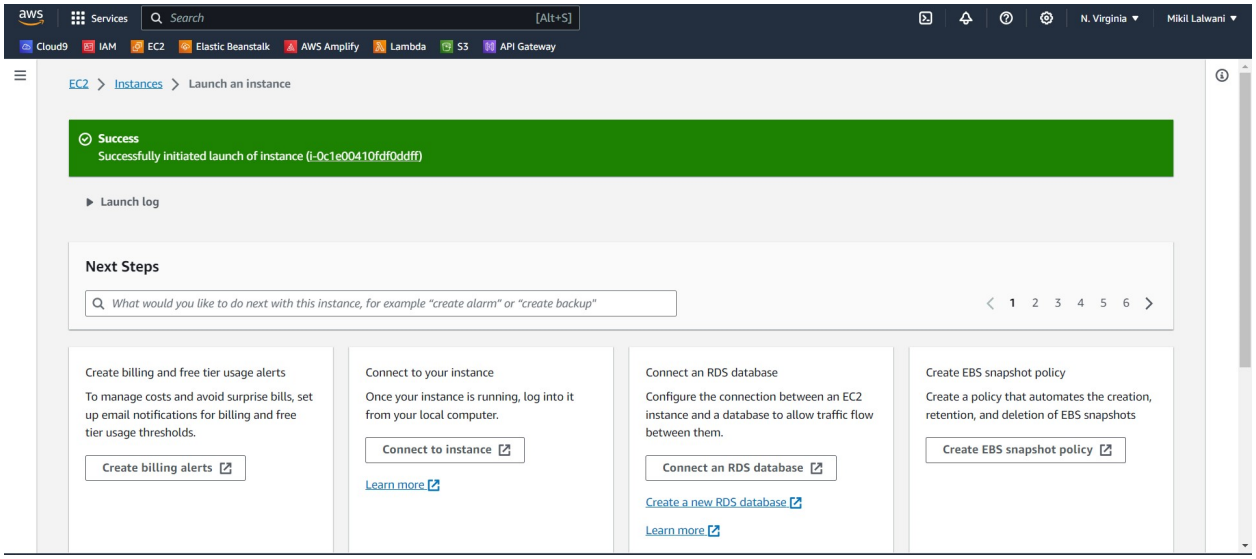


Step 2: An instance named websocket is created.

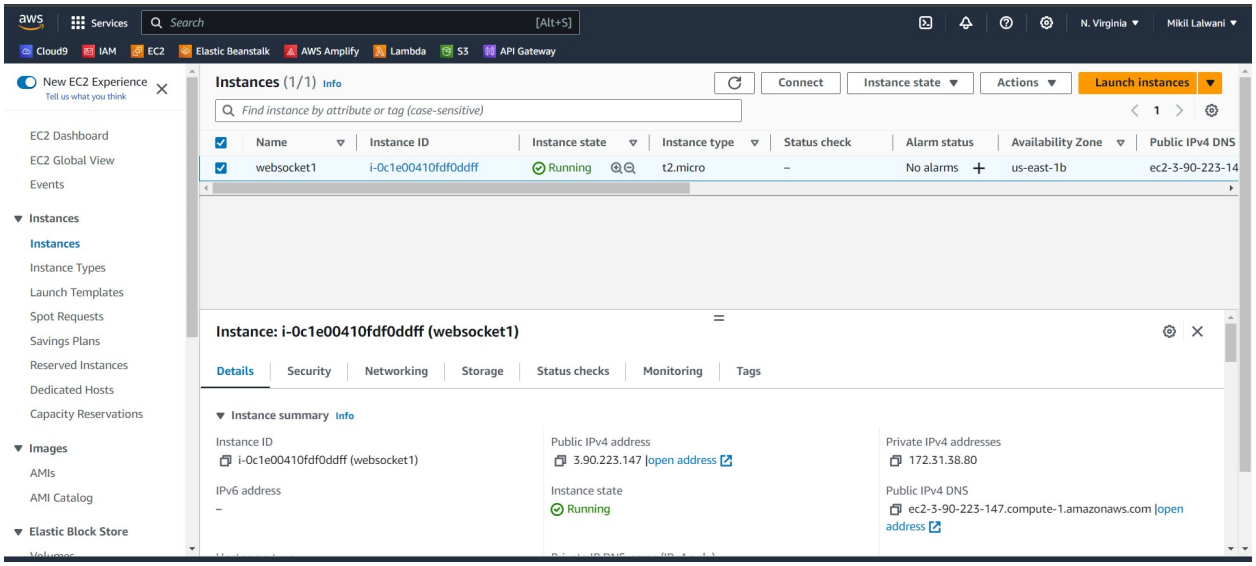


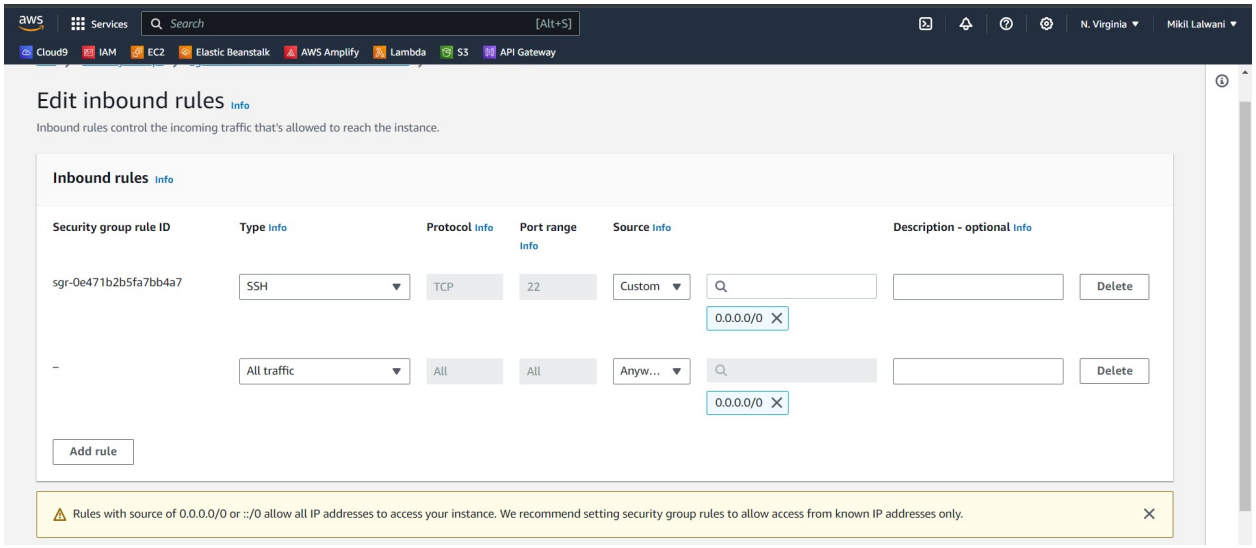
Step 3: Launch the instance.





Step 4: Check the inbound rules of the instance and edit the inbound rules to add the rule for All traffic.





Step 5: Open Ec2 instance and run the following commands:-
sudo su
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-venv

Create an environment:
mkdir ioe
cd ioe
python3 -m venv env

source env/bin/activate
pip install websockets

sudo nano websocket_cloud.py

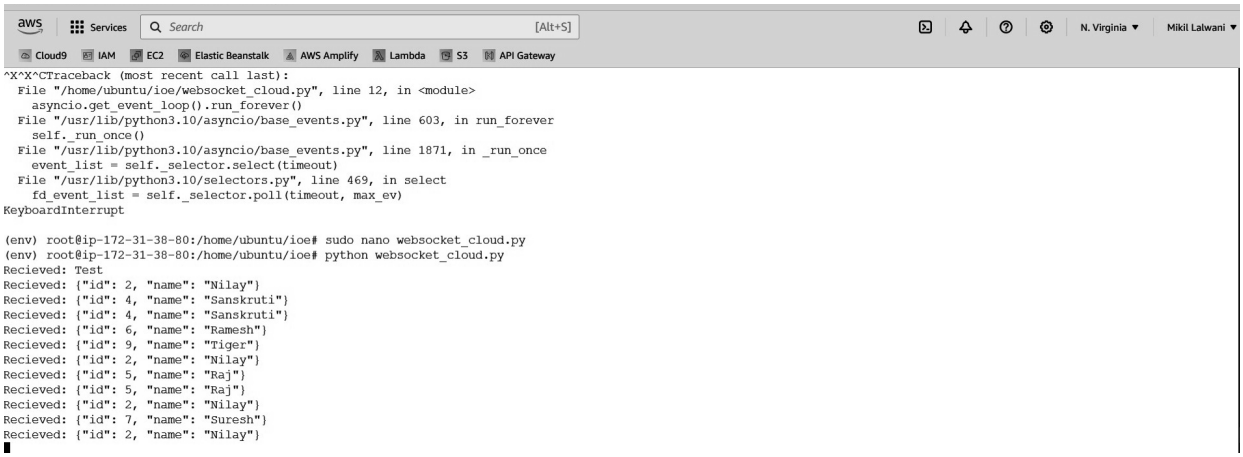
Step 6: Add this following code in the websocket_cloud.py.

Step 7: Add this following code in the edge_websocket.py.

edge_websocket.py

```python
import asyncio
import websockets
import random
import json
import pandas as pd
data = pd.read_csv('data.csv')

async def send_data():
    async with websockets.connect('ws://3.90.223.147:8080') as websocket:
        while True:
            random_index = random.randint(0,len(data)-1)
            random_data = data.iloc[random_index]
            data_dict = {
                "id": int(random_data["id"]),
                "name": random_data["name"]
            }

            await websocket.send(json.dumps(data_dict))
            print(f"Sent: {data_dict}")
            await asyncio.sleep(1)

if __name__ == '__main__':
    asyncio.run(send_data())
```

Step 8: Run the file websockety_cloud.py on the EC2 instance. The output is:



Step 9: Run the file of VSCode in the terminal. The output is: