```
#To enable mqtt connection from any device that runs python the AWS iot core for two way
#communication
import ssl
import paho.mqtt.client as mqtt
import json

# MQTT server details
mqtt_broker = 'a3lbqy8xbd64zo-ats.iot.ap-south-1.amazonaws.com'
mqtt_port = 8883  # Default MQTT over TLS port

# TLS/SSL configuration
#Enter the path to the certificates
tls_cert =
'93d425ad0b89762d5b2815ea904ccb6389a028cae242daca92126680a45e7b00-certificate.pem
.crt'
tls_key =
'93d425ad0b89762d5b2815ea904ccb6389a028cae242daca92126680a45e7b00-private.pem.ke
y'
ca_cert = 'AmazonRootCA1.pem'

# Callback functions for MQTT events


def on_connect(client, userdata, flags, rc):
    print("Connected with result code: " + str(rc))
    # Subscribe to a topic
    client.subscribe("esp32/pub")

    # Publish a test message
    topic = 'esp32/sub'
    message = {'message': 'Tdevice connected to network'}
    #message is being converted into json and sent to aws iot core using esp32/sub topic
    message_json = json.dumps(message)
    client.publish(topic, message_json)

#when msg is received
def on_message(client, userdata, msg):
    print("Received message: " + msg.payload.decode())

#when we are publishing
def on_publish(client, userdata, mid):
    print("Message published")

#if there is a disconnect with the server
```

```
def on_disconnect(client, userdata, rc):
    print("Disconnected with result code: " + str(rc))
    # Perform any cleanup or reconnection tasks

# Create an MQTT client instance with client ID
client_id = 'connectPubSub'  # Set your desired client ID here
client = mqtt.Client(client_id=client_id)

# Set TLS/SSL options this ensures that the mqtt messages are encrypted
client.tls_set(ca_certs=ca_cert, certfile=tls_cert, keyfile=tls_key,
cert_reqs=ssl.CERT_REQUIRED,
               tls_version=ssl.PROTOCOL_TLS)

# Set callback functions
client.on_connect = on_connect
client.on_message = on_message
client.on_publish = on_publish
client.on_disconnect = on_disconnect

# Connect to the MQTT broker
client.connect(mqtt_broker, mqtt_port)

# Start the MQTT client loop (manually or automatically)
client.loop_start()

# Following code ensures that we are able to continously publish and receive messages using
the terminal
try:
    while True:
        # Prompt the user for input
        message = input("Enter a message: ")

        # Publish the input message which was entered on the command line
        topic = 'esp32/sub'
        message = {'message': message}
        message_json = json.dumps(message)
        client.publish(topic, message_json)
except KeyboardInterrupt:
    pass

# Disconnect the MQTT client
client.loop_stop()
client.disconnect()
```