

IoT Security and Privacy

Assignment 7 – AWS IoT

10 points

Team Members: Rajib Dey, Debashri Roy, Cody Baker, Kennedy Vrutaal

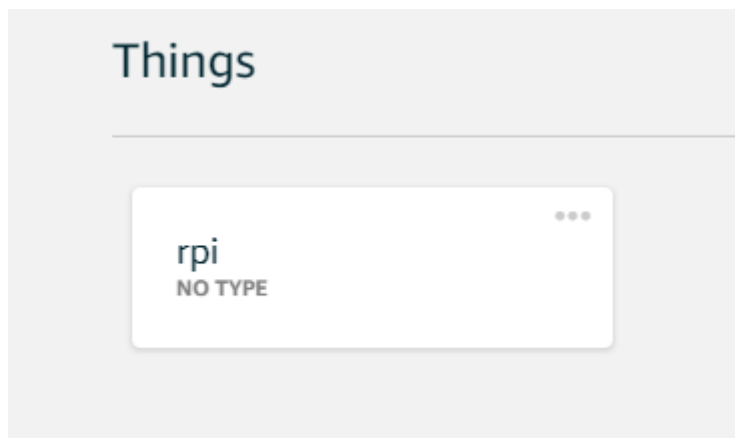
Assignment:

In this assignment, students are required to connect Raspberry Pi to Amazon AWS IoT and update the state of the connected sensor continuously with the AWS IoT thing shadow. The data such as the state generated by the sensor should be written into Amazon DynamoDB. Note: be careful not to exceed the free account quota of data for AWS IoT. Please refer to the references [1][3][4][5][6][7][8][8], particularly [1], if necessary for this assignment.

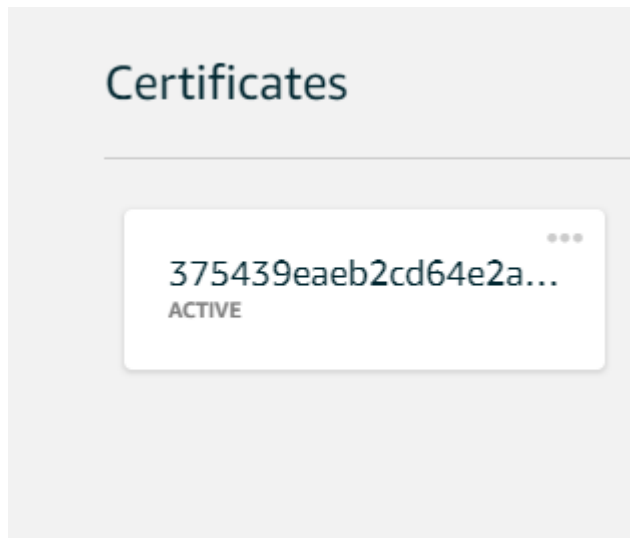
Requirements:

1. Document in detail the procedures connecting Raspberry Pi to Amazon AWS IoT. (3 points)

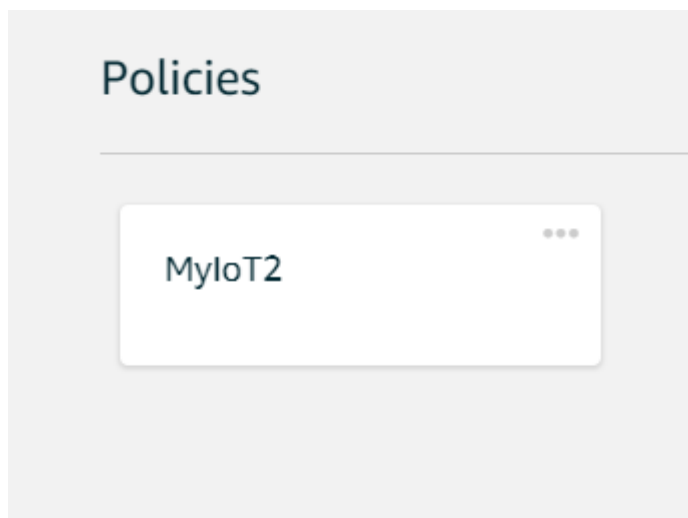
To connect the Raspberry Pi to Amazon AWS IoT, we had to create a thing shadow on AWS.



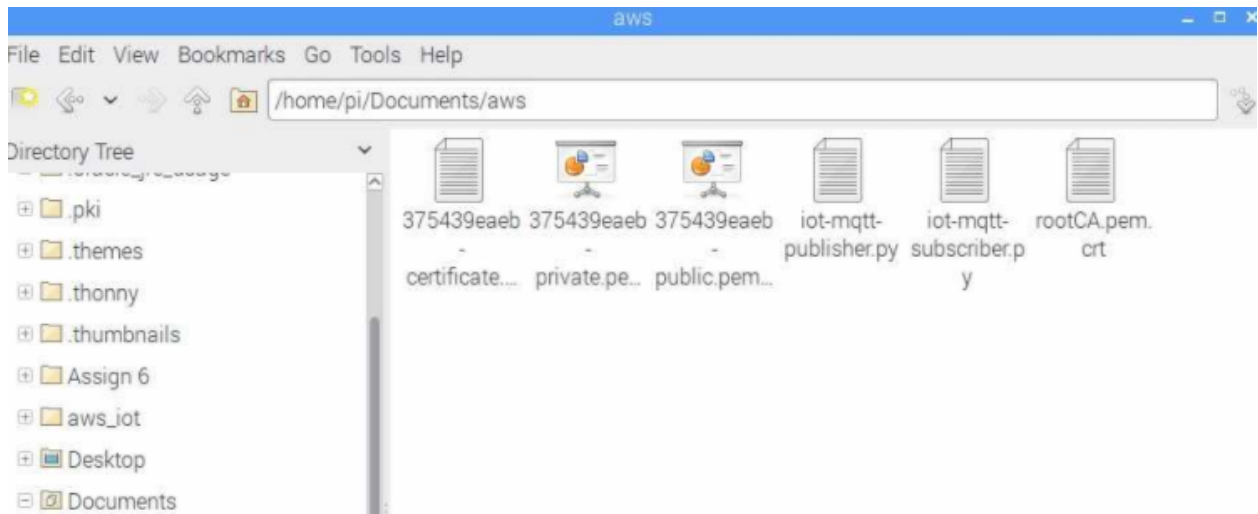
Then we had to create a certificate, which provided us with the necessary public and private keys, and certificate.



We then created a policy and attached our IoT thing and policy to our certificate we had created and then activated our certificate.



We then proceed to copy all the necessary files over in the same directory on our Raspberry pi.



We then made sure we had the latest stable version of the paho-mqtt library installed. We then proceed to use the two python files provided by Ms. Chao to subscribe and publish to AWS IoT. We modified the python files to use our IoT endpoint information and certificate and keys.

2. Document in detail the procedures updating the state of the connected sensor continuously with the AWS IoT thing shadow. (3 points)

With the raspberry pi connected to AWS IoT, updating the state of the connected sensor continuously was just a matter of running a python script that would send data to the thing shadow in AWS IoT.

To update the thing shadow, a mqtt message must be sent to the “update” topic for the specific ARN that was assigned to our thing shadow. Below is a screenshot of the different topics that data can be published to.

MQTT

Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)

Learn more

Update to this thing shadow

```
$aws/things/rpi/shadow/update
```

Update to this thing shadow was accepted

```
$aws/things/rpi/shadow/update/accepted
```

Update this thing shadow documents

```
$aws/things/rpi/shadow/update/documents
```

Update to this thing shadow was rejected

```
$aws/things/rpi/shadow/update/rejected
```

The publisher python code (shown in section 4) simply packs data into a json object and publishes it to the “\$aws/things/rpi/shadow/update” topic which then gets sent to AWS IoT and updates the thing shadow with the new data. The updates to the thing shadow can be viewed under the “Shadow” section of AWS IoT as shown in the screenshot below.

Shadow Document

Last update: Nov 17, 2017 4:46:55 PM -0500

Shadow state:

```
1 {  
2   "reported": {  
3     "color": "Fu",  
4     "time": "2017/11/17 16:46:1510955215",  
5     "motion": 0  
6   }  
7 }
```

3. Document in detail the procedures writing sensor data into Amazon DynamoDB. (3 points)

To write sensor data into our Amazon DynamoDB we created a rule, which had the action to save the data into a table we had created into DynamoDB.

RULE

save

ENABLED

Actions ▾

Overview

Description

Edit

save gps data

Rule query statement

Edit


The source of the messages you want to process with this rule.

SELECT * FROM '#'

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. [Learn more](#)

 Insert a message into a DynamoDB table

test

Remove Edit ▸

Add action

When creating the table, we had to also create a role in the IAM console that would give the raspberry pi permission to write into our table.

*IAM role name

[Update role](#)

[Create a new role](#)

Scan: [Table] test: topic, timestamp ^

Scan ▾

[Table] test: topic, timestamp

+ Add filter

Start search

<input type="checkbox"/>	topic	timestamp	payload
<input type="checkbox"/>	\$aws/things/rpi/:	15109533487...	{ "metadata" :...
<input type="checkbox"/>	\$aws/things/rpi/:	15109539691...	{ "metadata" :...
<input type="checkbox"/>	\$aws/things/rpi/:	15109546952...	{ "metadata" :...
<input type="checkbox"/>	\$aws/things/rpi/:	15109546966...	{ "metadata" :...
<input type="checkbox"/>	\$aws/things/rpi/:	15109546981...	{ "metadata" :...
<input type="checkbox"/>	\$aws/things/rpi/:	15109546997...	{ "metadata" :...

4. Post the code for this project into this report. Students should try to write the code for the proposed project in this assignment. (1 point)

The code for the subscriber part:

```
#!/usr/bin/python3
```

```
#required libraries
```

```
import sys
```

```
import ssl
```

```
import json
```

```
import paho.mqtt.client as mqtt
```

```

#called while client tries to establish connection with the server
def on_connect(mqttc, obj, flags, rc):
    if rc==0:
        print ("Subscriber Connection status code: "+str(rc)+" | Connection status: successful")
        mqttc.subscribe("$aws/things/rpi/shadow/update/accepted", qos=0)

mqttc.publish("$aws/things/rpi/shadow/update",{"state":{"reported":{"color":"Fu"}}})#The
names of these topics start with $aws/things/thingName/shadow."
    elif rc==1:
        print ("Subscriber Connection status code: "+str(rc)+" | Connection status: Connection
refused")

#called when a topic is successfully subscribed to
def on_subscribe(mqttc, obj, mid, granted_qos):
    print("Subscribed: "+str(mid)+" "+str(granted_qos)+"data"+str(obj))

#called when a message is received by a topic
def on_message(mqttc, obj, msg):
    print("Received message from topic: "+msg.topic+" | QoS: "+str(msg.qos)+" | Data
Received: "+str(msg.payload))

#creating a client with client-id=mqtt-test
mqttc = mqtt.Client(client_id="cgao")

mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe
mqttc.on_message = on_message

#Configure network encryption and authentication options. Enables SSL/TLS support.
#adding client-side certificates and enabling tlsv1.2 support as required by aws-iot service
mqttc.tls_set(ca_certs="/home/pi/Documents/aws/rootCA.pem.crt",
              certfile="/home/pi/Documents/aws/375439eae-certificat.pem.crt",
              keyfile="/home/pi/Documents/aws/375439eae-private.pem.key",
              tls_version=ssl.PROTOCOL_TLSv1_2,
              ciphers=None)

#connecting to aws-account-specific-iot-endpoint
mqttc.connect("a2cy0km9why8d6.iot.us-east-1.amazonaws.com", port=8883) #AWS IoT
service hostname and portno

#automatically handles reconnecting
mqttc.loop_forever()

```

And the code for the publisher part:

```

#!/usr/bin/python3

#required libraries
import sys
import ssl
import json
import paho.mqtt.client as mqtt

# for motion sensor
import RPi.GPIO as GPIO
import time
from datetime import datetime

#called while client tries to establish connection with the server
def on_connect(mqttc, obj, flags, rc):
    if rc==0:
        print ("Subscriber Connection status code: "+str(rc)+" | Connection status: successful")
        mqttc.subscribe("$aws/things/rpi/shadow/update/accepted", qos=0)
    elif rc==1:
        print ("Subscriber Connection status code: "+str(rc)+" | Connection status: Connection
refused")

#called when a topic is successfully subscribed to
def on_subscribe(mqttc, obj, mid, granted_qos):
    print("Subscribed: "+str(mid)+" "+str(granted_qos)+"data"+str(obj))

#called when a message is received by a topic
def on_message(mqttc, obj, msg):
    print("Received message from topic: "+msg.topic+" | QoS: "+str(msg.qos)+" | Data
Received: "+str(msg.payload))

#creating a client with client-id=mqtt-test
mqttc = mqtt.Client(client_id="rpi")

mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe
mqttc.on_message = on_message

#Configure network encryption and authentication options. Enables SSL/TLS support.
#adding client-side certificates and enabling tlsv1.2 support as required by aws-iot service
mqttc.tls_set(ca_certs="/home/pi/Documents/aws/rootCA.pem.crt",
              certfile="/home/pi/Documents/aws/375439eae-certificcate.pem.crt",

```

```

    keyfile="/home/pi/Documents/aws/375439eae-b-private.pem.key",
    tls_version=ssl.PROTOCOL_TLSv1_2,
    ciphers=None)

#connecting to aws-account-specific-iot-endpoint
mqttc.connect("a2cy0km9why8d6.iot.us-east-1.amazonaws.com", port=8883) #AWS IoT
service hostname and portno

#automatically handles reconnecting
#start a new thread handling communication with AWS IoT
mqttc.loop_start()

sensor = 12

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(sensor,GPIO.IN)

rc=0
try:
    while rc == 0:
        i = GPIO.input(sensor)
        print(i)    # i = 1: Motion detected; i = 0: No Motion
        data={}
        data['motion']=i
        data['time']=datetime.now().strftime('%Y/%m/%d %H:%M:%s')
        payload = '{"state":{"reported":'+json.dumps(data)+'}}'
        print(payload)

        #the topic to publish to
        #the names of these topics start with $aws/things/thingName/shadow.
        msg_info = mqttc.publish("$aws/things/rpi/shadow/update", payload, qos=1)

        time.sleep(1.5)

except KeyboardInterrupt:
    pass

GPIO.cleanup()

```

References:

[1] Chao Gao, [Amazon AWS IoT](#), 2016

- [2] [AWS IoT developer guide](#), 2016
- [3] Onur ŞALK, [Amazon Web Services IoT](#), November 02, 2015
- [4] [Get Started with AWS IoT and Raspberry Pi](#), Oct. 18, 2015
- [5] [AWS January 2016 Webinar Series - Getting Started with AWS IoT](#), Youtube, Jan 26, 2016
- [6] [AWS Identity and Access Management User Guide](#), 2016
- [7] [paho-mqtt 1.1](#), 2016
- [8] [Introducing JSON](#), 2016