TMA Innovation Center

AWS IOT QUICK START

Contents

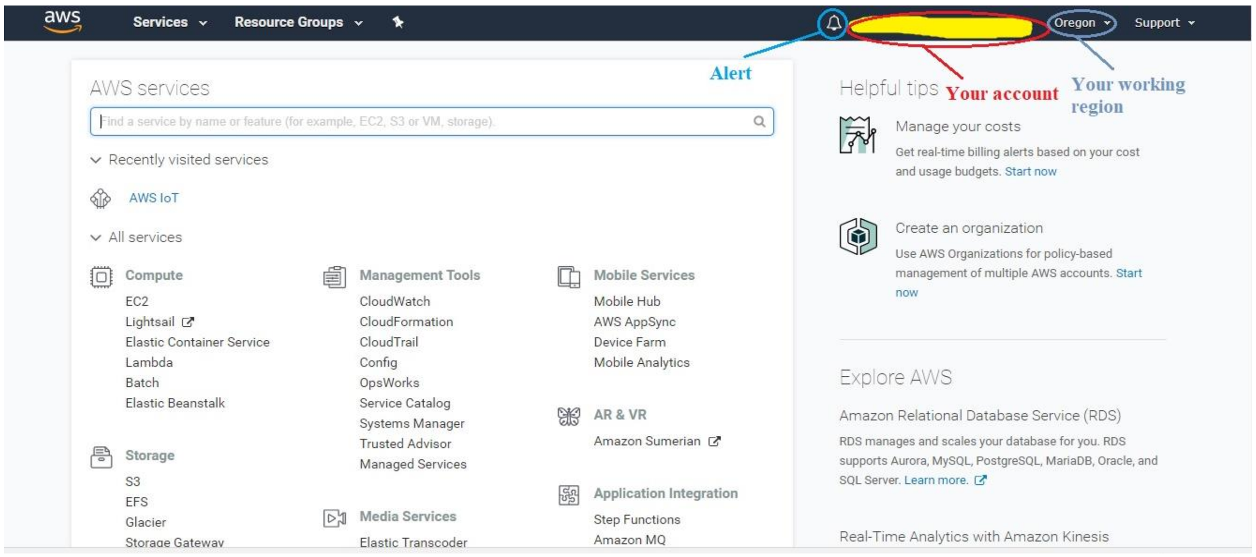
1. **SETUP** 3
    1. **Software setup** 3
    2. **Hardware setup** 4
2. **AWS with ESP32** 4
    1. **Register a device** 4
    2. **Run the device** 9
3. **AWS with Rasberry Pi** 12
    1. **NodeJS** 12
    2. **Embedded C** 14
4. **REFERENCE** 16

1. SETUP

1. Software setup

ACCOUNT SETUP

- ➢ Ask your manager for an account because it is not free
- ➢ In case you already have an account, login
- ➢ Login AWS IoT with your account



Manage your costs

Get real-time billing alerts based on your cost and usage budgets. Start now

Create an organization

Use AWS Organizations for policy-based management of multiple AWS accounts. Start now

Amazon Relational Database Service (RDS)

RDS manages and scales your database for you. RDS supports Aurora, MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. Learn more.

Real-Time Analytics with Amazon Kinesis

Figure 1 When you login successfully

IDE & LIBRARY SETUP

Here are some software and library setup before you can run the basic configuration and examples. It just only a few minutes to download all of them

ESP32 software setup

- ➢ Install Arduino core for ESP32 from https://github.com/espressif/arduino-esp32
- ➢ Download and install the AWS_IOT library for ESP32 https://github.com/ExploreEmbedded/Hornbill-Examples/tree/master/arduino-esp32/AWS_IOT

Raspberry Pi software setup

- ➢ Clone the AWS IoT Embedded C SDK

```
$ git clone https://github.com/aws/aws-iot-device-sdk-embedded-C
```

- ➢ Clone some external libraries, then copy their source code inside the *external_libs* directory of the Embedded C SDK

```
$ git clone https://github.com/cpputest/cpputest
$ git clone https://github.com/ARMmbed/mbedtls
```

- Clone the prepared NodeJS demo (This one is like a separated SDK)

```
$ git clone https://github.com/nbxtruong/AWS-IoT-Demo
```

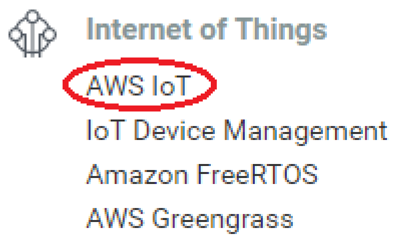
2. Hardware setup

- An ESP32
- A Raspberry Pi 3 board
- DHT22 (optional – in case you want to understand deeper)
- Some LEDs (optional – in case you want to understand deeper)
- Jumper Wires (optional – in case you want to understand deeper)
- Some Buttons or Switches (optional – in case you want to understand deeper)
- Resistor 330 Ohm and 1k (optional – in case you want to understand deeper)
- A breadboard (optional – in case you want to understand deeper)

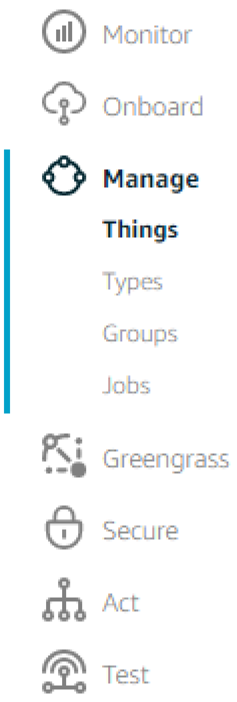
2. AWS with ESP32

1. Register a device

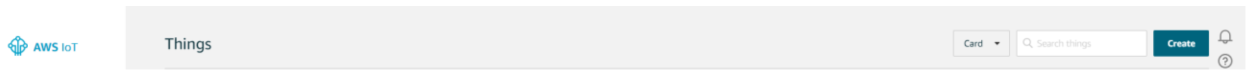
- From your console, find **Internet of Things** and choose **AWS IoT**



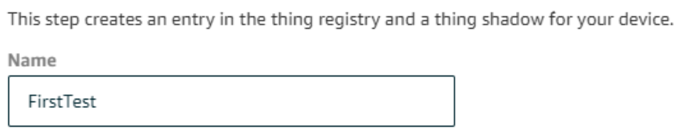
- Choose **Manage -> Things**



- ➢ Click on **Create** to create a new **Thing**



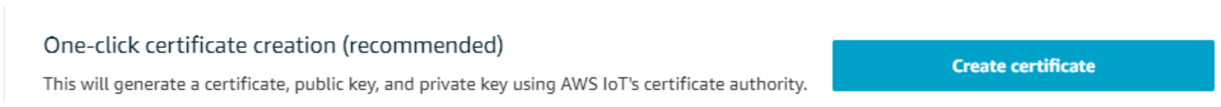
- ➢ Select **Create a single thing**
- ➢ Name your **Thing** then select **Next** (I named my **Thing** *FirstTest*)


This step creates an entry in the thing registry and a thing shadow for your device.

Name

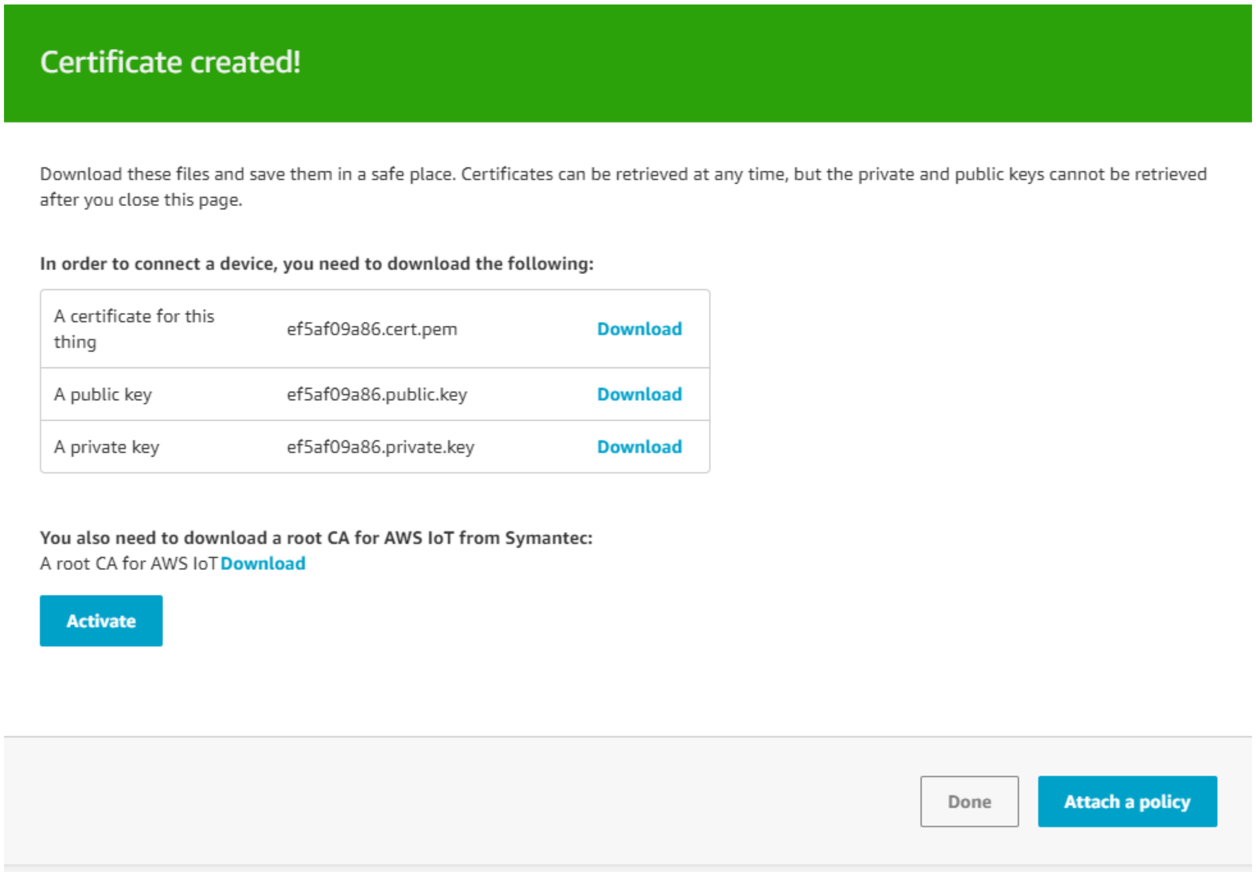
FirstTest

- ➢ Create a certificate by clicking **Create certificate**


One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

- ➢ Download all the certificates (the root CA should be saved in Notepad under the name **aws-root-ca.pem**) then activate all. Remember not to give these keys to anyone.



Certificate created!

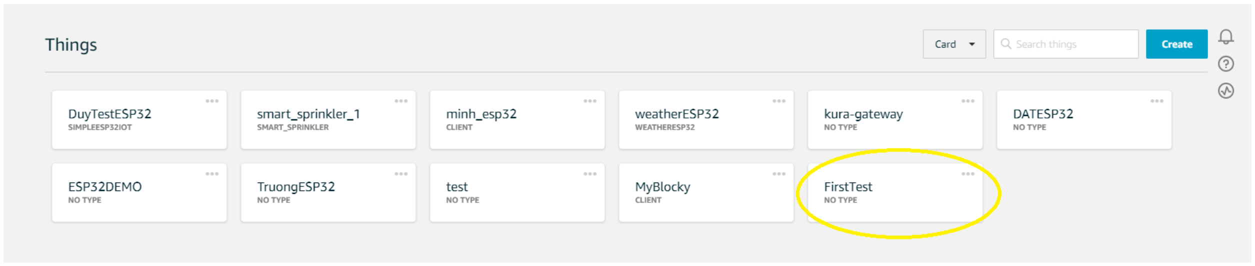
Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	ef5af09a86.cert.pem	Download
A public key	ef5af09a86.public.key	Download
A private key	ef5af09a86.private.key	Download

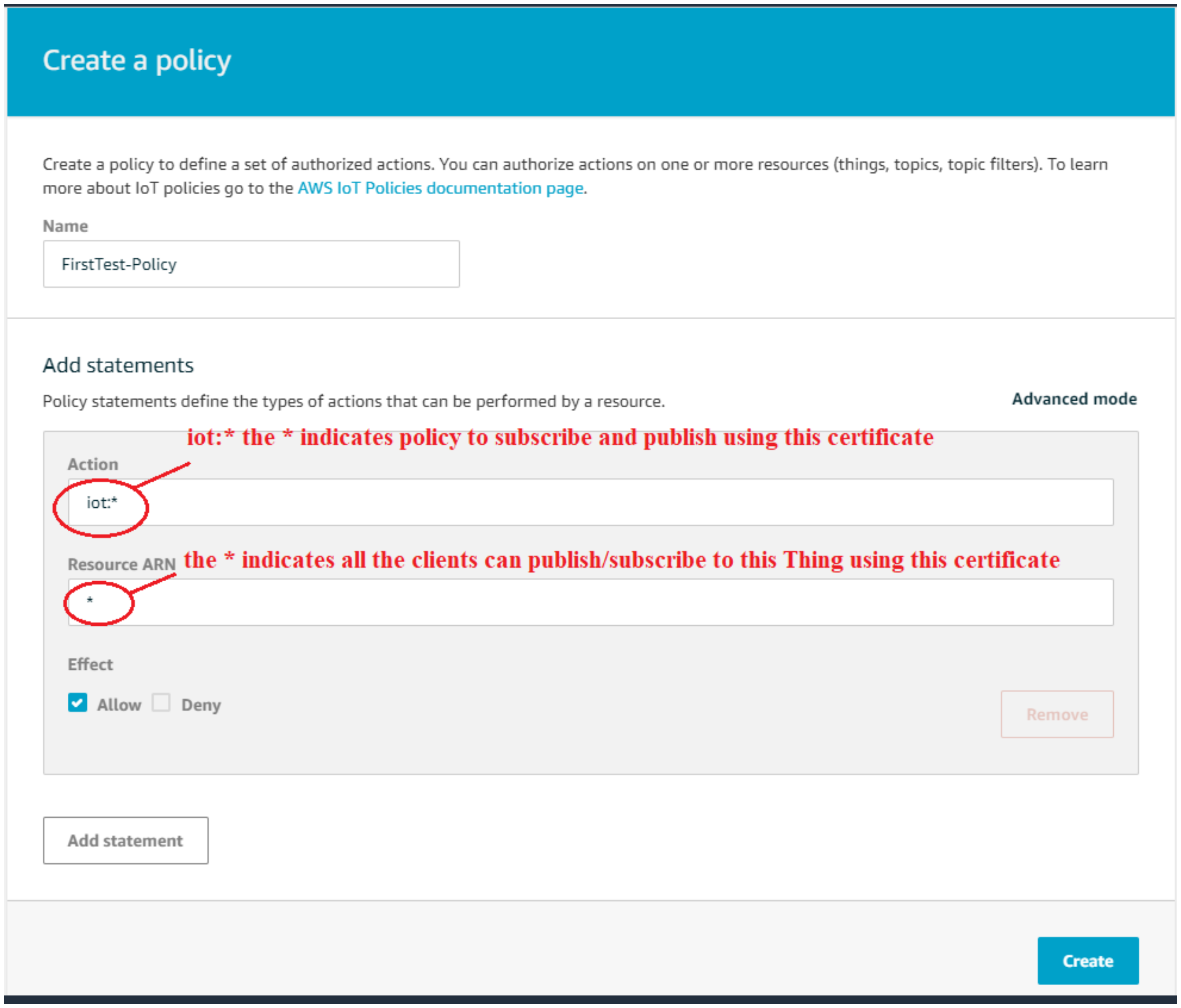
You also need to download a root CA for AWS IoT from Symantec:
A root CA for AWS IoT Download

➢ Your **Thing** is created



➢ Go to **Security->Policies->Create** to create a policy for it

➢ Name your policy and add statements for it (I named mine *FirstTest-Policy*)



Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: FirstTest-Policy

Add statements

Policy statements define the types of actions that can be performed by a resource.

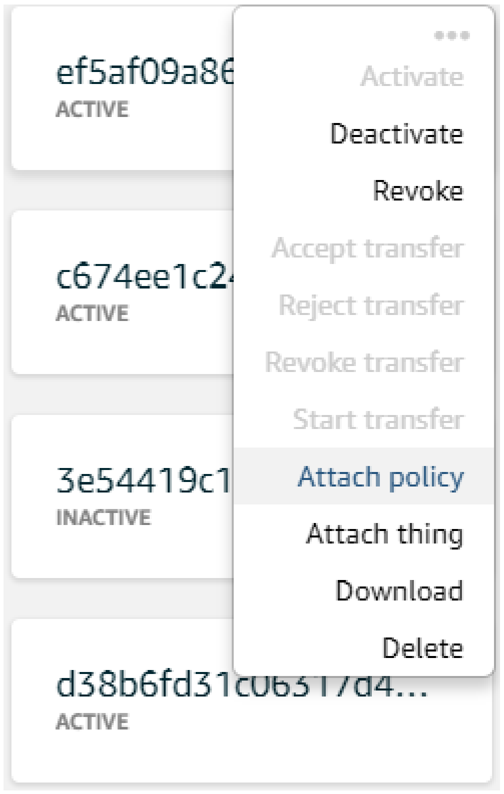
Action: iot:*

Resource ARN: *

Effect: [✓ Allow] Deny

Advanced mode
iot:* the * indicates policy to subscribe and publish using this certificate
the * indicates all the clients can publish/subscribe to this Thing using this certificate
Remove
Add statement
Create

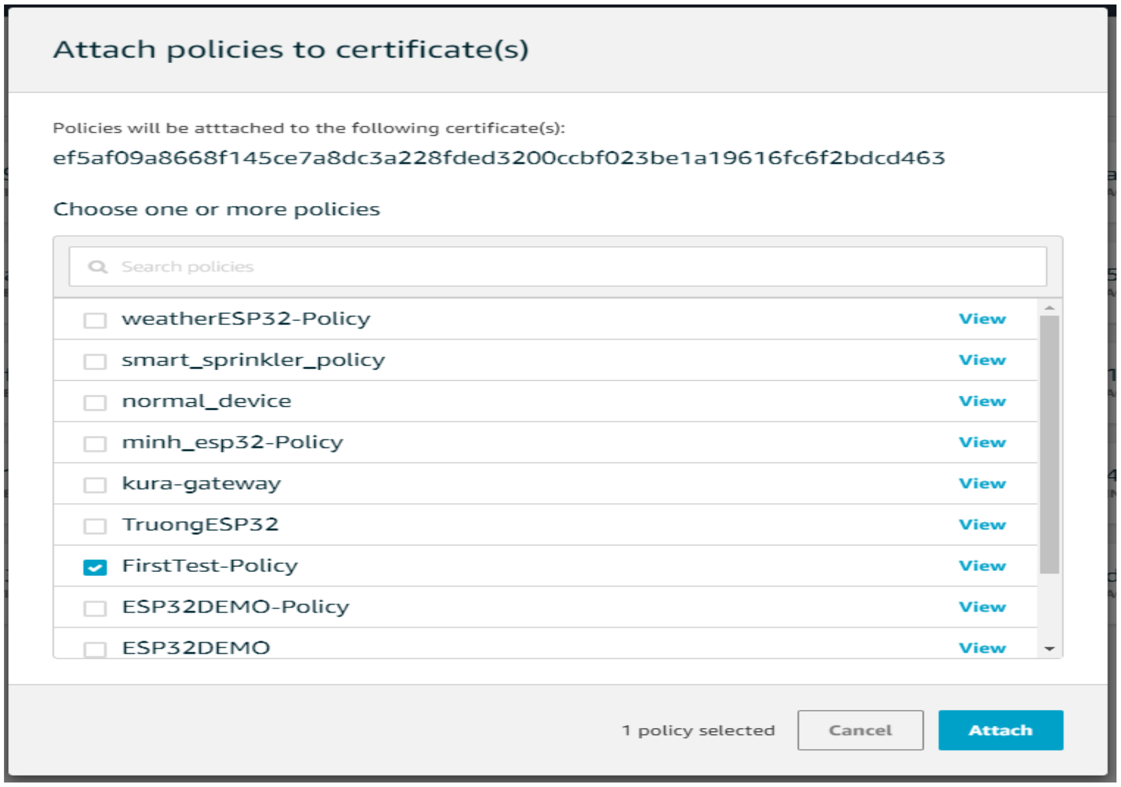
- Now go back to **Certificates** and attach the **Policy** that was defined above, you can view your **Things Certificates** in the **Security** section if there are many certificates


ef5af09a86
ACTIVE

c674ee1c2
ACTIVE

3e54419c1
INACTIVE

d38b6fd31c06317d4...
ACTIVE


Attach policies to certificate(s)

Policies will be atttached to the following certificate(s):
ef5af09a8668f145ce7a8dc3a228fded3200ccbf023be1a19616fc6f2bdcd463

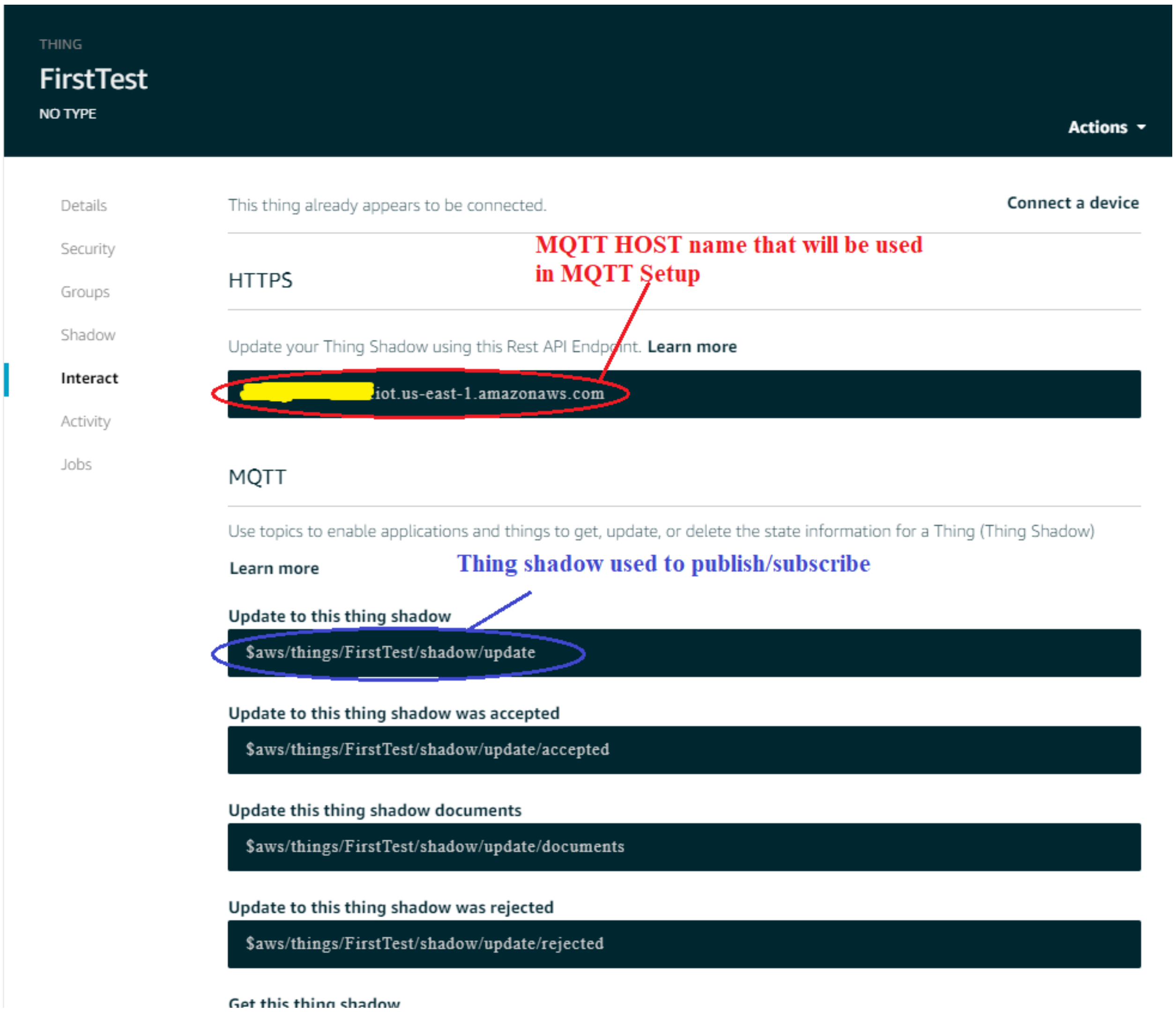
Choose one or more policies

Search policies

- [ ] weatherESP32-Policy
- [ ] smart_sprinkler_policy
- [ ] normal_device
- [ ] minh_esp32-Policy
- [ ] kura-gateway
- [ ] TruongESP32
- [x] FirstTest-Policy
- [ ] ESP32DEMO-Policy
- [ ] ESP32DEMO

1 policy selected

➢ In the **Interact** section of your **Thing**, please mind that



THING

FirstTest

NO TYPE

Actions

Details

Security

Groups

Shadow

Interact

Activity

Jobs

This thing already appears to be connected.

Connect a device

HTTPS

MQTT HOST name that will be used in MQTT Setup

Update your Thing Shadow using this Rest API Endpoint. **Learn more**

```
[redacted]iot.us-east-1.amazonaws.com
```

MQTT

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

Learn more

Thing shadow used to publish/subscribe

Update to this thing shadow

```
$aws/things/FirstTest/shadow/update
```

Update to this thing shadow was accepted

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

Update this thing shadow documents

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

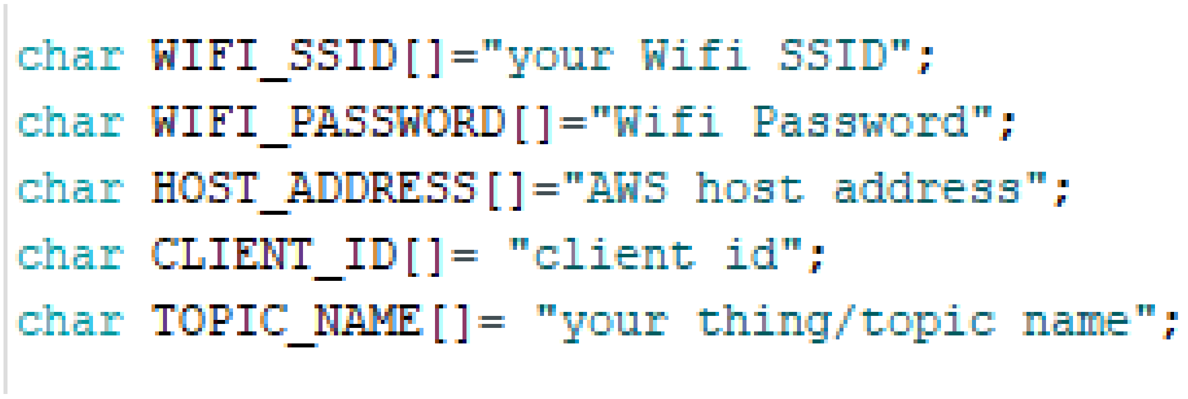
Update to this thing shadow was rejected

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

Get this thing shadow

2. Run the device

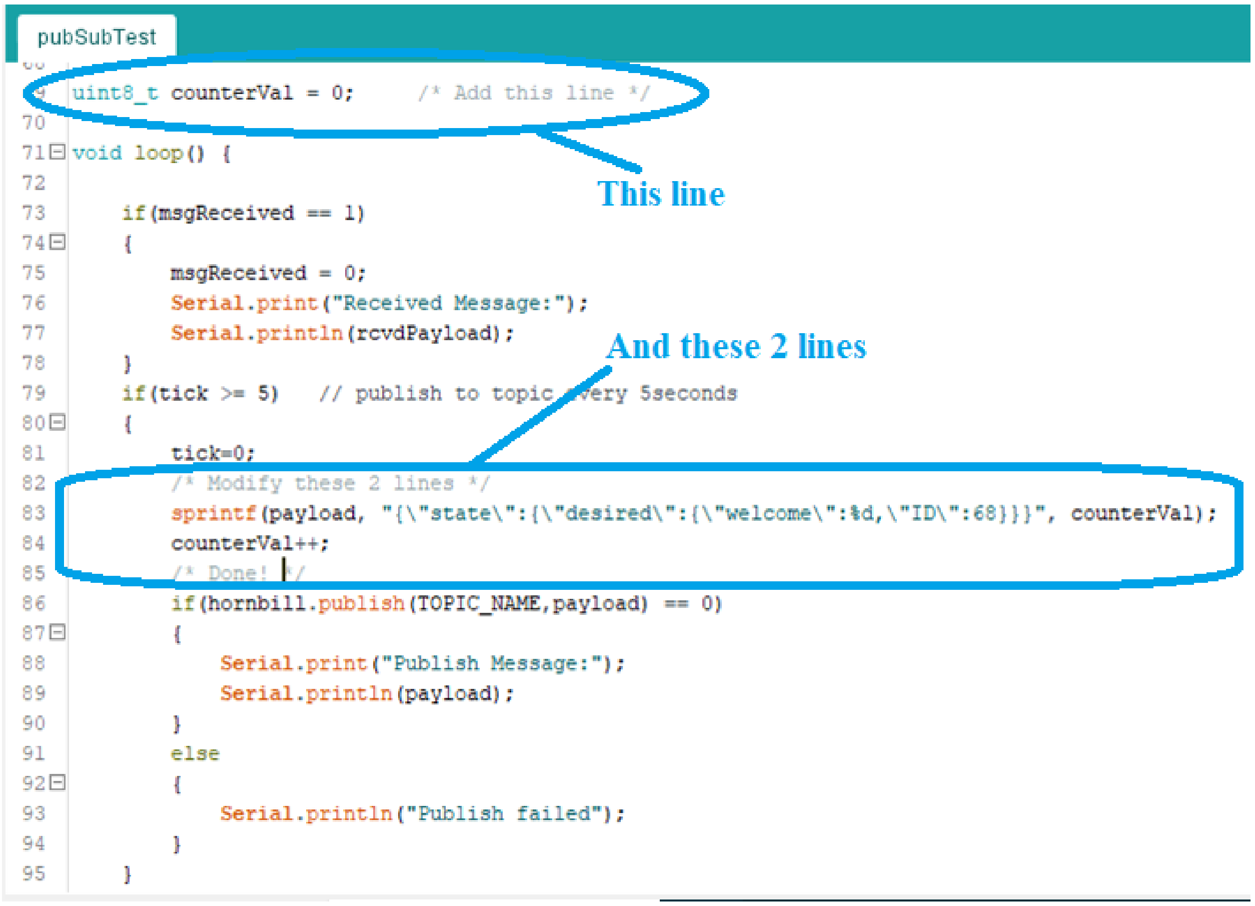
- Open the pubSubTest example of AWS_IOT library
- Change these with yours, the **CLIENT_ID** can be named what ever you want, for the **TOPIC_NAME**, I chose the Thing Shadow update: **$aws/things/<your-thing-name>/shadow/update**



```
char WIFI_SSID[]="your Wifi SSID";
char WIFI_PASSWORD[]="Wifi Password";
char HOST_ADDRESS[]="AWS host address";
char CLIENT_ID[]= "client id";
char TOPIC_NAME[]= "your thing/topic name";
```

➢ Copy the containing of certificates **(root-CA, certificate, private-key)** into **ws_iot_certificate.c** file (**don't delete the \n**\).

➢ Add some lines of code like the image below



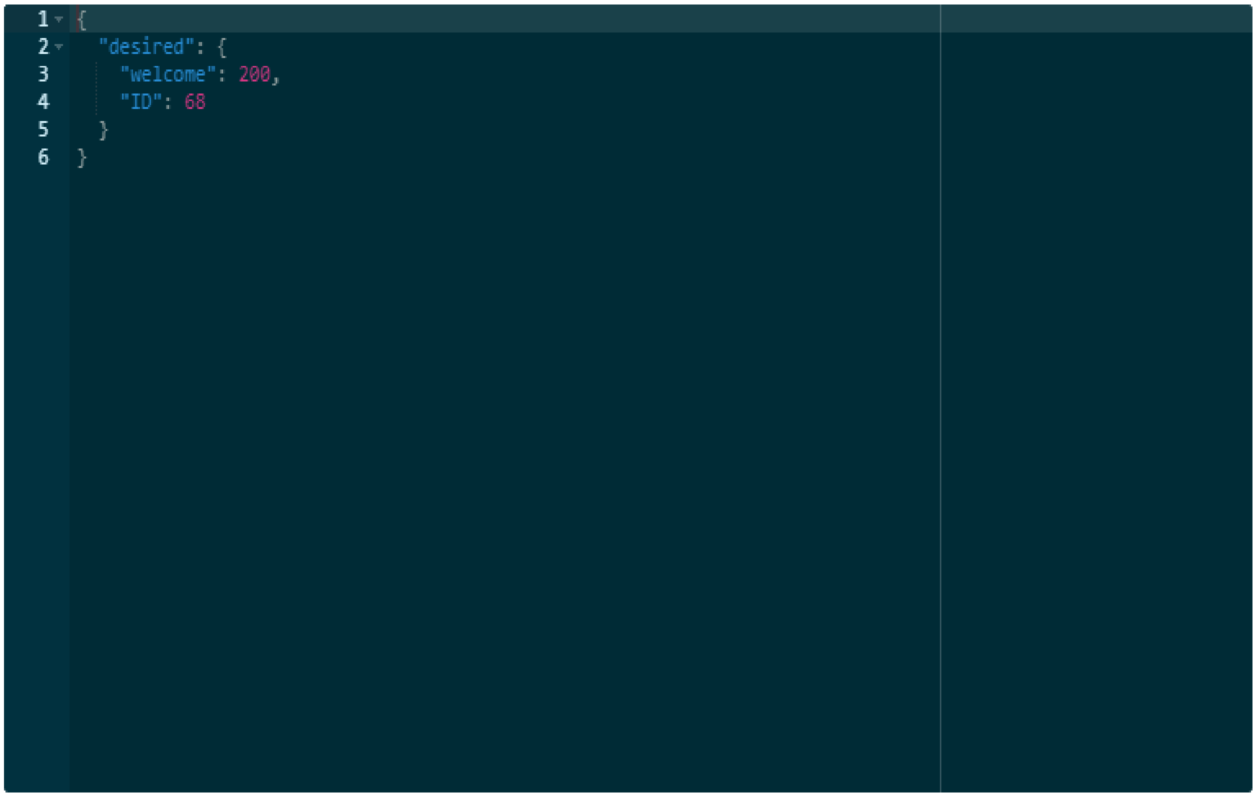
```
uint8_t counterVal = 0;     /* Add this line */

void loop() {

    if(msgReceived == 1)
    {
        msgReceived = 0;
        Serial.print("Received Message:");
        Serial.println(rcvdPayload);
    }
    if(tick >= 5)   // publish to topic every 5seconds
    {
        tick=0;
        /* Modify these 2 lines */
        sprintf(payload, "{\"state\":{\"desired\":{\"welcome\":%d,\"ID\":68}}}", counterVal);
        counterVal++;
        /* Done! */
        if(hornbill.publish(TOPIC_NAME,payload) == 0)
        {
            Serial.print("Publish Message:");
            Serial.println(payload);
        }
        else
        {
            Serial.println("Publish failed");
        }
    }
```

➢ Upload the code

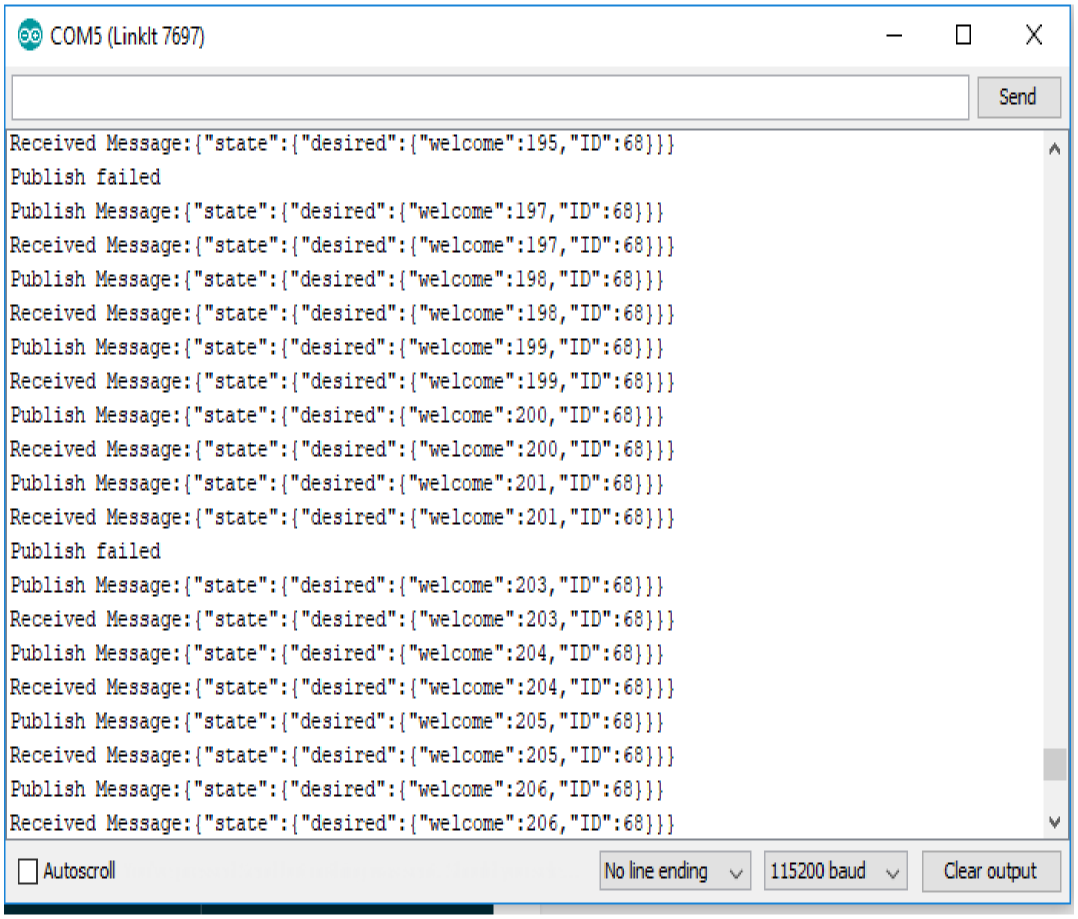
Result: **Thing Shadow** updates the counter value continuously and the device also receives the value in the subscribe message.

Last update: Jan 15, 2018 4:16:41 PM +0700

Shadow state:



```
{
  "desired": {
    "welcome": 200,
    "ID": 68
  }
}
```



```
Received Message:{"state":{"desired":{"welcome":195,"ID":68}}}
Publish failed
Publish Message:{"state":{"desired":{"welcome":197,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":197,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":198,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":198,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":199,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":199,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":200,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":200,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":201,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":201,"ID":68}}}
Publish failed
Publish Message:{"state":{"desired":{"welcome":203,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":203,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":204,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":204,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":205,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":205,"ID":68}}}
Publish Message:{"state":{"desired":{"welcome":206,"ID":68}}}
Received Message:{"state":{"desired":{"welcome":206,"ID":68}}}
```

3. AWS with Rasberry Pi

1. NodeJS

This section will show you how to update data to AWS Thing Shadow using Node JS. A repository has been created, on your Raspberry Pi terminal

```
$ git clone https://github.com/nbxtruong/AWS-IoT-Demo
$ cd AWS-IoT-Demo
$ npm install
```

Then create a folder call "**cert**" and copy all your certificates in it **(root-CA, certificate, private-key)**

Open ThingShadow.js file and do the following steps

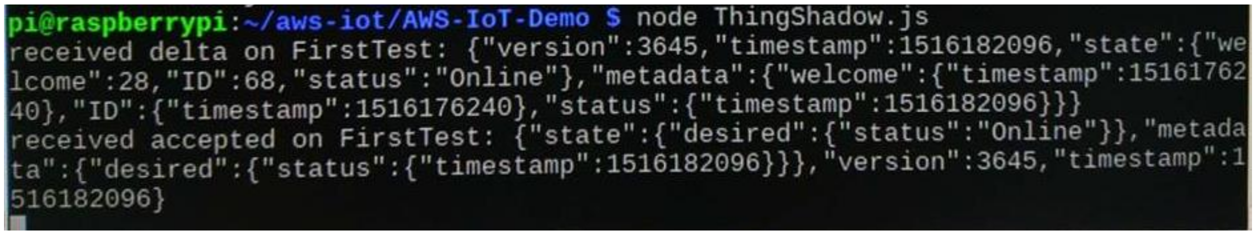
- edit the all certificate names in these lines

```
var thingShadows = awsIot.thingShadow({
        keyPath: './cert/<your-private-key>.<key-format>',
        caPath: './cert/<your-ca>.<ca-key-format>',
        clientId: '<your-client-ID>',
        host: '<your-host-URL>'
});
```

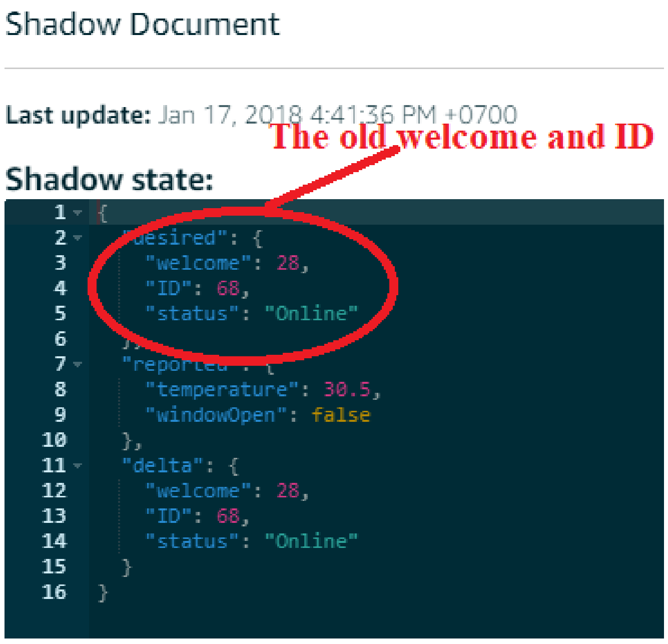
➢ edit your device name in **thingShadow.register** and **thingShadow.update** function

```
$ node ThingShadow.js
```

Result: **Thing Shadow** updates the counter value continuously…now get to the Shadow section on AWS and enter the new value for **welcome** field and **ID** field



```
pi@raspberrypi:~/aws-iot/AWS-IoT-Demo $ node ThingShadow.js
received delta on FirstTest: {"version":3645,"timestamp":1516182096,"state":{"we
lcome":28,"ID":68,"status":"Online"},"metadata":{"welcome":{"timestamp":15161762
40},"ID":{"timestamp":1516176240},"status":{"timestamp":1516182096}}}
received accepted on FirstTest: {"state":{"desired":{"status":"Online"}},"metada
ta":{"desired":{"status":{"timestamp":1516182096}}},"version":3645,"timestamp":1
516182096}
```

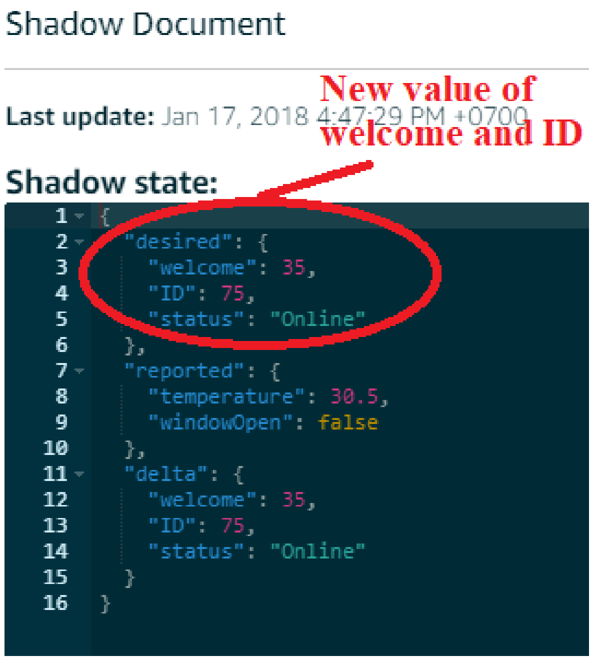


Shadow Document

Last update: Jan 17, 2018 4:41:36 PM +0700

Shadow state:

```
{
  "desired": {
    "welcome": 28,
    "ID": 68,
    "status": "Online"
  },
  "reported": {
    "temperature": 30.5,
    "windowOpen": false
  },
  "delta": {
    "welcome": 28,
    "ID": 68,
    "status": "Online"
  }
}
```



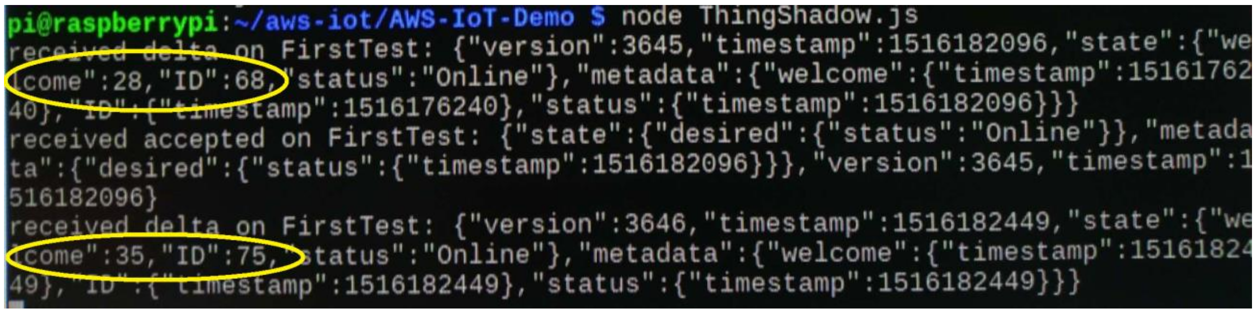
Shadow Document

Last update: Jan 17, 2018 4:47:29 PM +0700

Shadow state:

```
{
  "desired": {
    "welcome": 35,
    "ID": 75,
    "status": "Online"
  },
  "reported": {
    "temperature": 30.5,
    "windowOpen": false
  },
  "delta": {
    "welcome": 35,
    "ID": 75,
    "status": "Online"
  }
}
```

➢ check your result on your Raspberry terminal, the 2 yellow circles prove that the values of **welcome** and **ID** field are also updated by the device



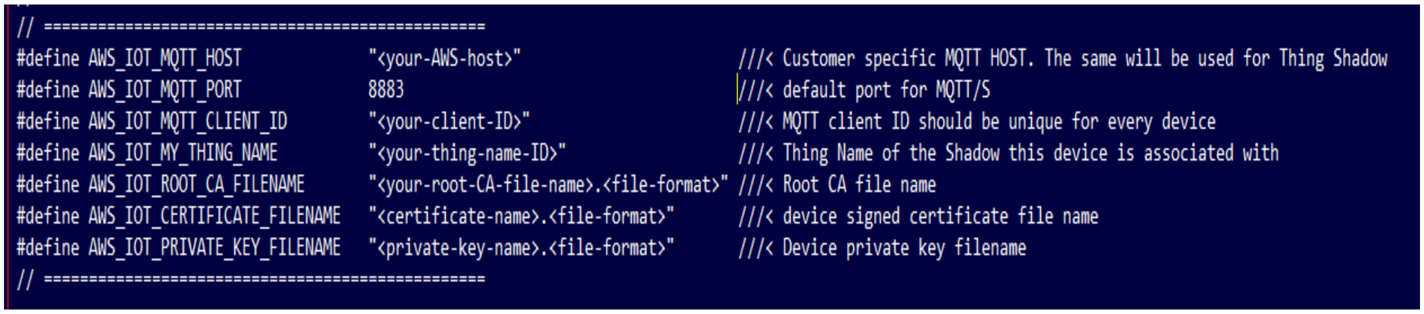
```
pi@raspberrypi:~/aws-iot/AWS-IoT-Demo $ node ThingShadow.js
received delta on FirstTest: {"version":3645,"timestamp":1516182096,"state":{"we
lcome":28,"ID":68,"status":"Online"},"metadata":{"welcome":{"timestamp":15161762
40},"ID":{"timestamp":1516176240},"status":{"timestamp":1516182096}}}
received accepted on FirstTest: {"state":{"desired":{"status":"Online"}},"metada
ta":{"desired":{"status":{"timestamp":1516182096}}},"version":3645,"timestamp":1
516182096}
received delta on FirstTest: {"version":3646,"timestamp":1516182449,"state":{"we
lcome":35,"ID":75,"status":"Online"},"metadata":{"welcome":{"timestamp":15161824
49},"ID":{"timestamp":1516182449},"status":{"timestamp":1516182449}}}
```

2. Embedded C

This section will show you how to publish/subsribe and update data to Thing Shadow using the Embedded C SDK

```
$ cd aws-iot-device-sdk-embedded-C
```

- ➢ Copy your certificate, private key, and root CA certificate into the certs directory
- ➢ Go to **sample_apps/subscribe_publish_sample** directory and config the **aws_iot_config.h** as follow



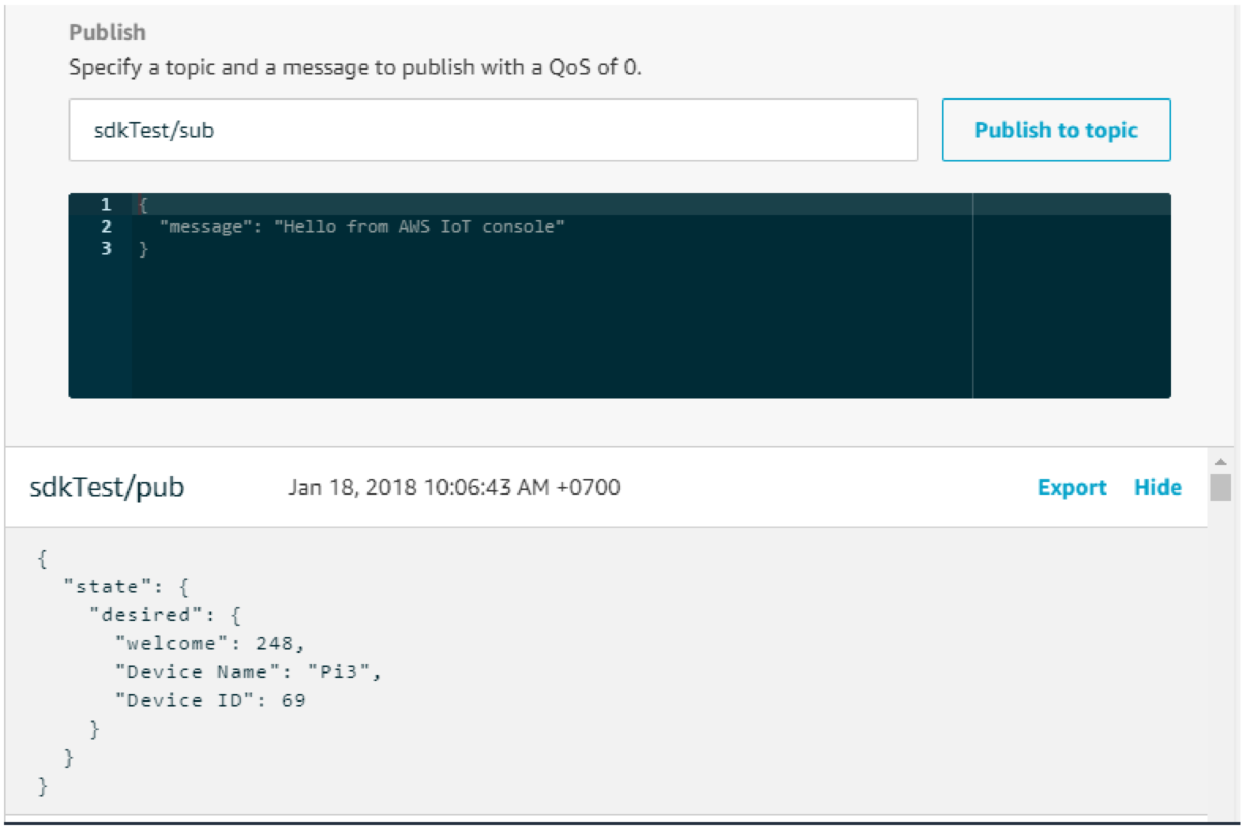
```
// =================================================
#define AWS_IOT_MQTT_HOST              "<your-AWS-host>"                     ///< Customer specific MQTT HOST. The same will be used for Thing Shadow
#define AWS_IOT_MQTT_PORT              8883                                  ///< default port for MQTT/S
#define AWS_IOT_MQTT_CLIENT_ID         "<your-client-ID>"                    ///< MQTT client ID should be unique for every device
#define AWS_IOT_MY_THING_NAME          "<your-thing-name-ID>"                ///< Thing Name of the Shadow this device is associated with
#define AWS_IOT_ROOT_CA_FILENAME       "<your-root-CA-file-name>.<file-format>" ///< Root CA file name
#define AWS_IOT_CERTIFICATE_FILENAME   "<certificate-name>.<file-format>"    ///< device signed certificate file name
#define AWS_IOT_PRIVATE_KEY_FILENAME   "<private-key-name>.<file-format>"    ///< Device private key filename
// =================================================
```

- ➢ Replace the subscribe_publish_sample.c file with the subscribe_publish_sample.c attached with this document

```
$ cd aws-iot-device-sdk-embedded-C

$ ./subscribe_publish_sample
```

- ➢ Go to the *Test* section of your AWS Thing online, subscribe to topic *sdkTest/pub* and…here is what you have

Result: your message has been uploaded successfully. If you press ***Publish to topic***, go to your Raspberry terminal



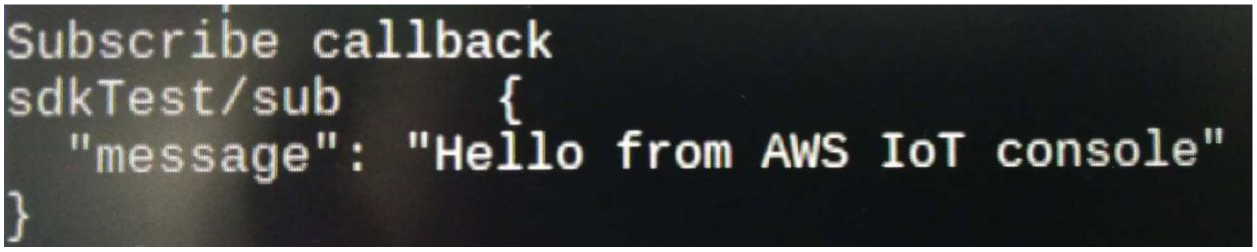
```
sdkTest/sub
```

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

sdkTest/pub Jan 18, 2018 10:06:43 AM +0700

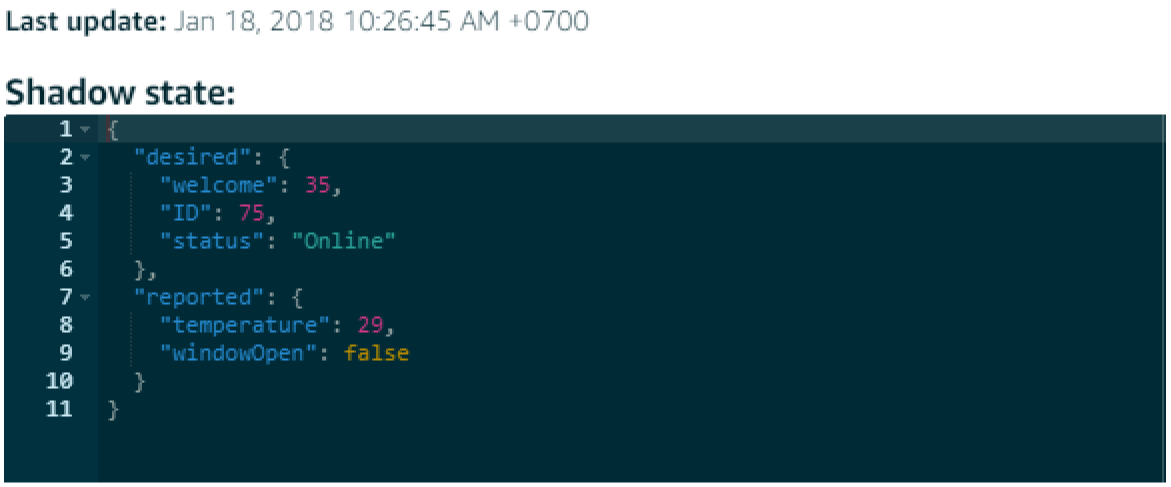
```
{
  "state": {
    "desired": {
      "welcome": 248,
      "Device Name": "Pi3",
      "Device ID": 69
    }
  }
}
```

<u>Result</u>: your message is received by the Raspberry Pi



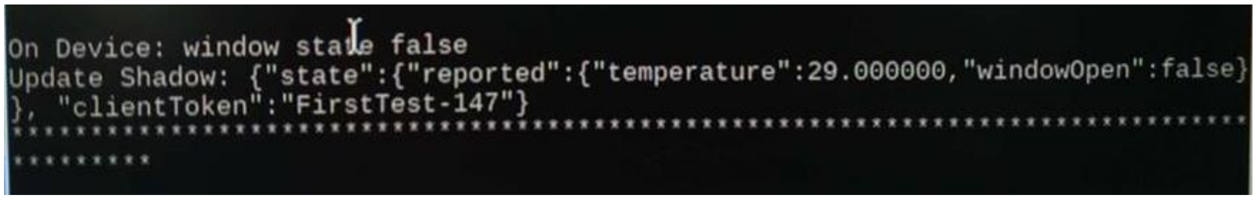
```
Subscribe callback
sdkTest/sub     {
  "message": "Hello from AWS IoT console"
}
```

- ➢ In case you want to publish/subscribe to **Thing Shadow**, go to the ***shadow_sample*** directory, do the same configuration and run the sample
- ➢ Check your **Thing Shadow** and Raspberry Terminal


Last update: Jan 18, 2018 10:26:45 AM +0700

Shadow state:

```
{
  "desired": {
    "welcome": 35,
    "ID": 75,
    "status": "Online"
  },
  "reported": {
    "temperature": 29,
    "windowOpen": false
  }
}
```



```
On Device: window state false
Update Shadow: {"state":{"reported":{"temperature":29.000000,"windowOpen":false}
}, "clientToken":"FirstTest-147"}
*****************************************************************************
*********
```

4. REFERENCE

For more details, please have a look at

- https://docs.aws.amazon.com/iot/latest/developerguide/iot-embedded-c-sdk.html
- http://exploreembedded.com/wiki/AWS_IOT_with_Arduino_ESP32
- https://aws.amazon.com/documentation/iot/

Last but not least: please read the official document about AWS IoT Developer Guide for an ***extremely detailed guidance***