Universidade Federal de Uberlândia

FEELT – Faculdade de Engenharia Elétrica
Engenharia de Controle e Automação

LASEC – Laboratório de Automação, Sistemas Eletrônicos e Controle

Guia prático para comunicação entre Raspberry Pi e AWS IoT Core através do protocolo MQTT, utilizando Python

Autor: Guilherme Balduino Lopes - guilhermebalopes@ufu.br

Introdução

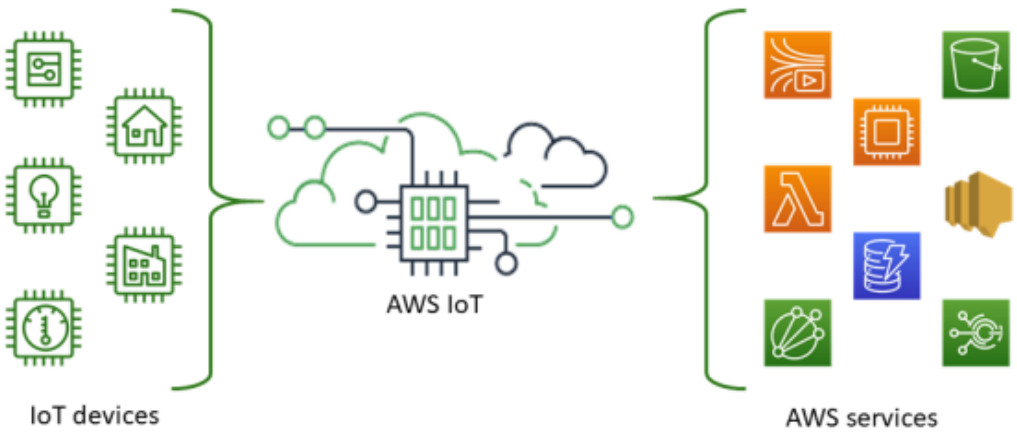
O termo IoT (*Internet of Things,* ou em português "Internet das Coisas") foi cunhado por um britânico chamado Kevin Ashton, que criou um conjunto de sensores conectados à internet. Atualmente temos uma quantidade cada vez maior de dispositivos enviando dados e/ou executando tarefas através de comandos recebidos via internet.

No entanto, para que dispositivos possam trocar dados entre si, é necessária a existência de uma estrutura que permita que a troca dessas informações aconteça de forma segura e confiável.

Neste contexto, existe um serviço na plataforma de Cloud Computing AWS chamado AWS IoT Core, o qual permite que dispositivos possam interagir com aplicativos na nuvem e até mesmo com outros dispositivos, oferecendo todos os benefícios de escalabilidade, podendo suportar bilhões de dispositivos e trilhões de mensagens, tudo isto de forma segura e simples.

Outra grande vantagem de se utilizar o serviço de IoT da AWS para este tipo de comunicação é o fato de este facilitar muito a conexão com outros serviços dentro da própria AWS como Amazon DynamoDB, AWS Lambda, Amazon S3, entre outros. A Figura 1 é uma imagem retirada do site da AWS que exemplifica este serviço.

Figura 1 – Comunicação AWS IoT Core



Além disso, o AWS IoT Core suporta atualmente 4 protocolos:

- MQTT (Message Queuing and Telemetry Transport)
- MQTT over WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol – Secure)
- LoRaWAN (Long Range Wide Area Network)

Para facilitar a comunicação entre dispositivos físicos com a AWS IoT, neste serviço existem os AWS IoT Device Softwares, tal como AWS IoT Greengrass, AWS IoT Device Tester e AWS IoT Device SDKs. O AWS IoT Device (and Mobile) SDKs conta com bibliotecas de código aberto, guias do desenvolvedor com exemplos e guias de portabilidade para que seja possível criar produtos ou soluções inovadoras de IoT em várias plataformas de hardware diferentes. E possui suporte para Python, C++, JavaScript, Java e C embarcado.

Processo de comunicação Passo a Passo

1. Acessar a Console do AWS IoT

Acesse sua conta da AWS na região de sua preferência e busque pelo serviço de IoT Core. Caso ainda não tenha uma conta na AWS, é possível criar uma no nível gratuito clicando aqui.

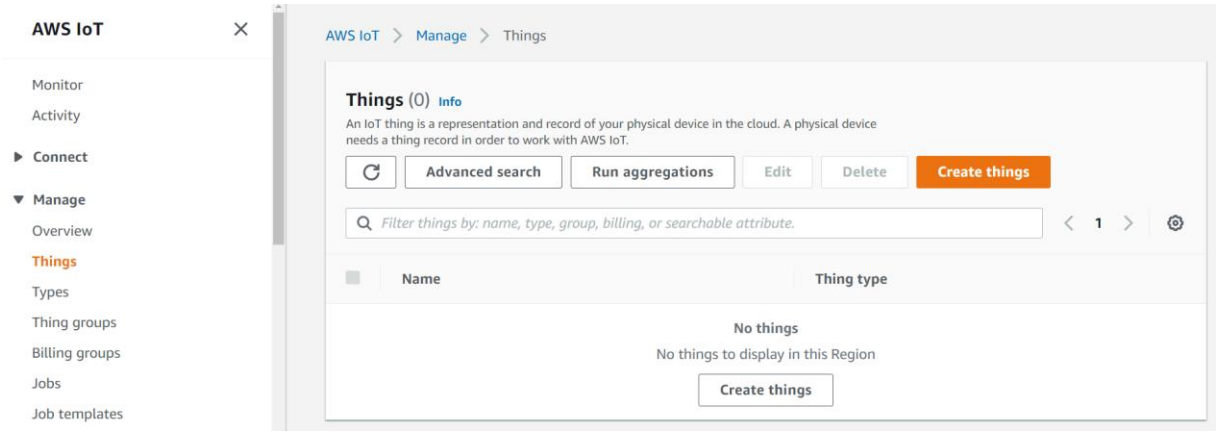
2. Registrar um device

O primeiro passo é registrar o seu device como uma "Coisa" no AWS IoT Core. Isto fará com que exista uma representação digital do seu dispositivo (físico ou não) dentro da plataforma da AWS. Para isso deve-se seguir os seguintes passos.

2.1 Criar uma "Coisa"

Dentro do console AWS IoT Core, abra a aba **Manage**, clique em **Things** e depois clique em **Create things.**

Figura 2 – Tela de criação das Coisas



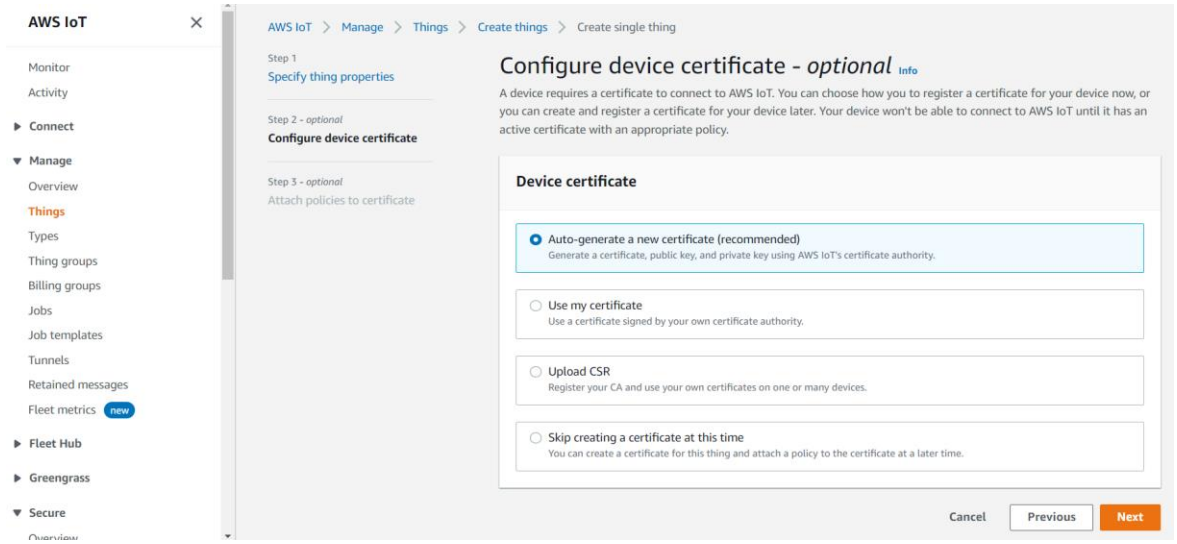
Na tela de **Create things**, selecione **Create single thing** e clique em **Next.**

Na próxima tela, adicione um nome a sua Coisa (“device01” por exemplo) e clique em **Next**. Note que existem algumas configurações adicionais onde é possível adicionar atributos e agrupar as Coisas para auxiliar e simplificar o processo de gerenciamento dos dispositivos, porém, estas configurações não se fazem necessárias no momento.

2.2 Criar um Certificado

Após criada a Coisa, é necessário se criar um **Device certificate**. Para isso, selecione **Auto-generate a new certificate** e clique em **Next.**

Figura 3 – Tela de criação de certificado para a Coisa

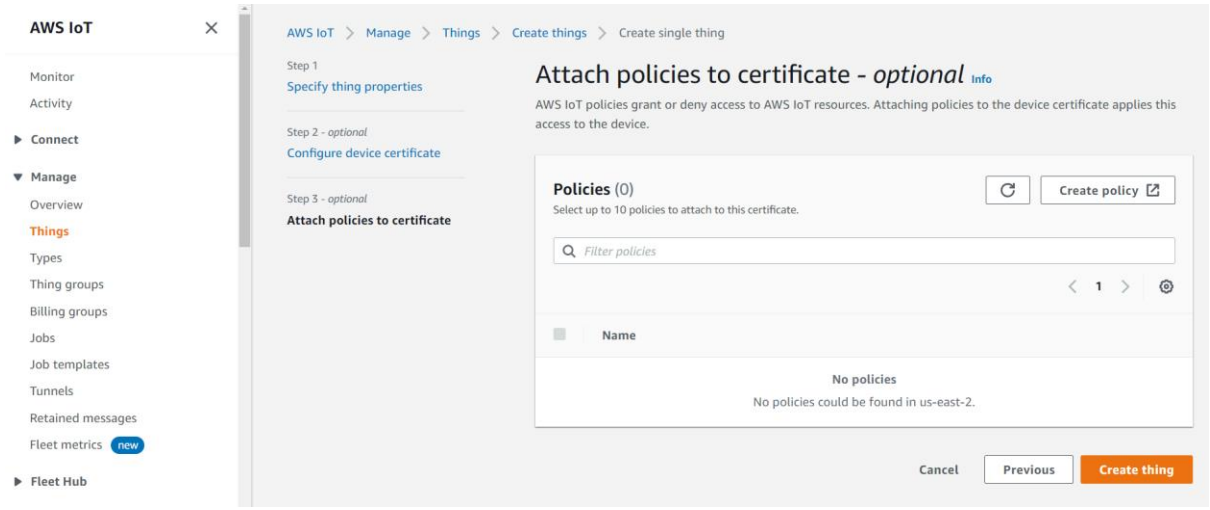


Obs.: Caso esta tela não tenha aparecido, é possível criar um certificado de forma independente indo em **Secure > Certificates,** clicar em **Add certificate > Create certificate,** selecionar **Auto-generate a new certificate,** logo abaixo selecionar **Activate,** clicar em **Create** e continuar seguindo com os próximos passos.

2.3 Criar uma Política

Criado o certificado, deve-se então criar uma política e vincula-la ao certificado recém criado. Para isso, clique em **Create policy.**

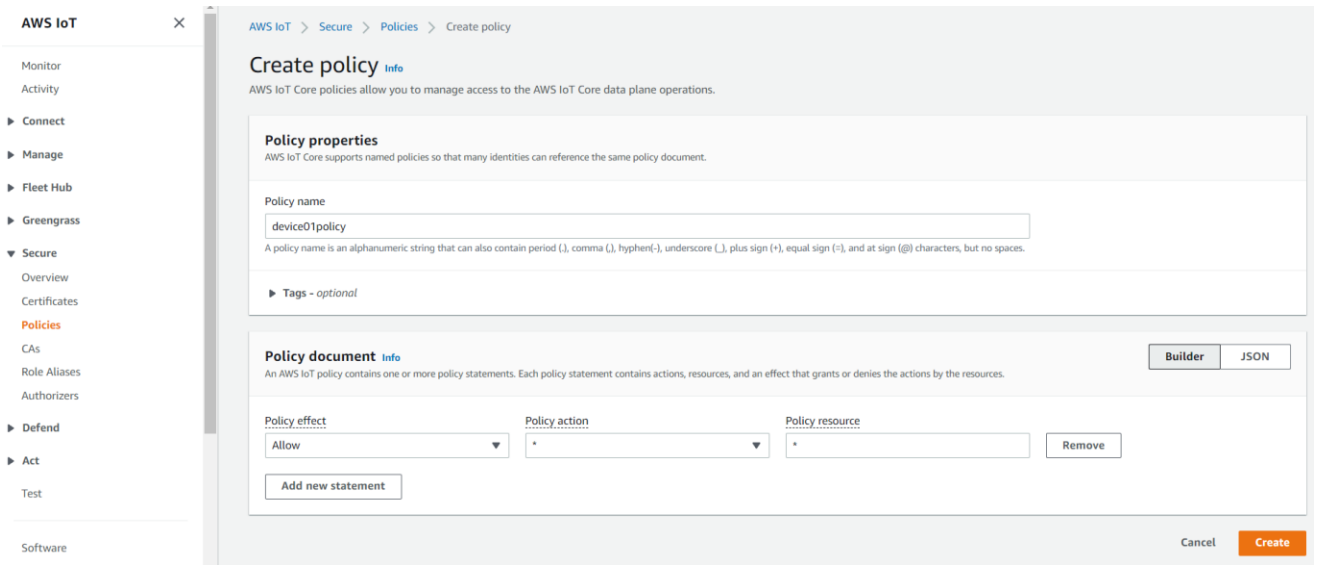
Figura 4 – Tela de vinculação de política ao certificado



Obs.: Caso esta tela não tenha aparecido, é possível criar uma política de forma independente indo em **Secure > Policies,** clicar em **Create policy** e continuar seguindo os próximos passos.

Abrirá então uma nova janela para a criação da política.

Figura 5 – Tela de criação de política

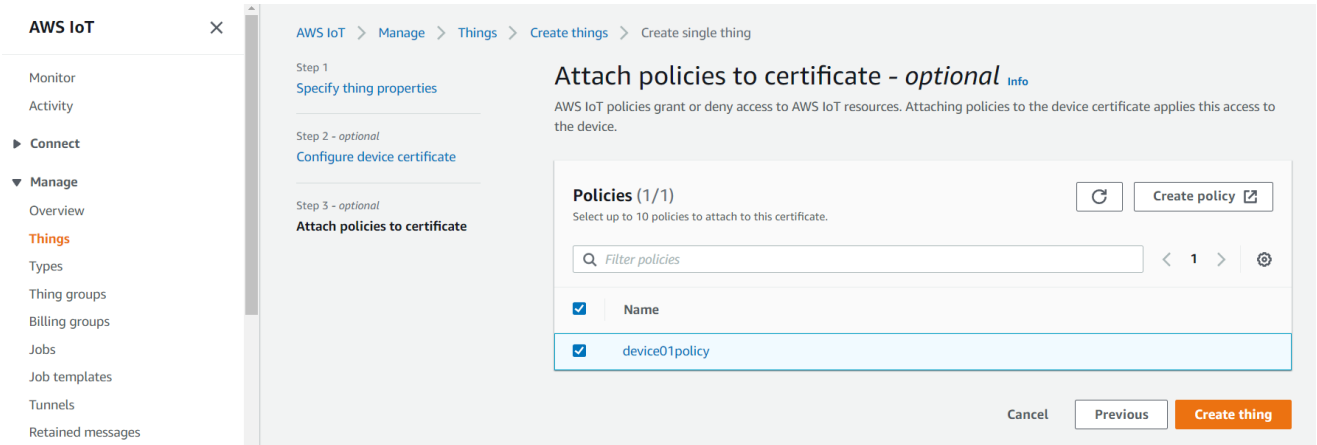


- Em **Policy name** dê um nome à Política (“device01policy” por exemplo).
- Em **Policy effect** mantenha “Allow”.
- Em **Policy action** e em **Policy resource** coloque um asterisco (*) para que permita todos os tipos de acesso.

2.4 Vincular Coisa, Política e Certificado

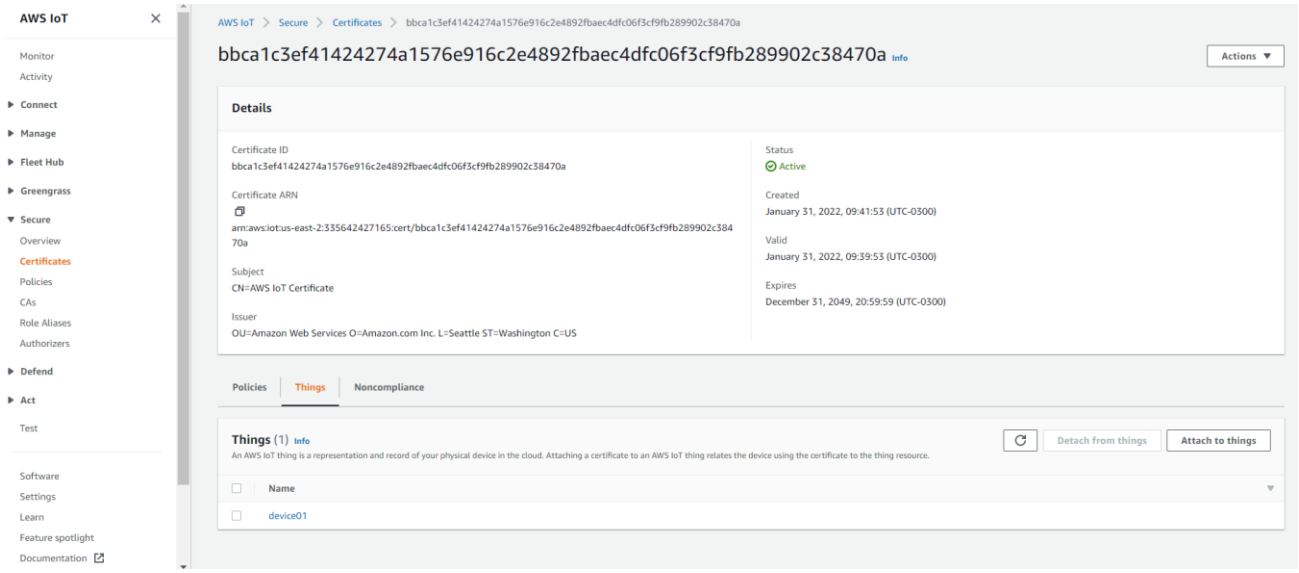
Voltando à tela de criação da Coisa, selecione a política recém criada e clique em **Create thing.**

Figura 6 – Tela de vinculação de política ao certificado



Obs.: Esta ação automaticamente vinculará a Coisa criada ao certificado e à política. Caso queira se certificar disso, após completar o item 2.5, é possível ir em **Certificates** na aba **Secure,** clicar no certificado criado e conferir em **Things** se a Coisa está vinculada, como mostrado na próxima Figura.

Figura 7 – Tela com detalhes do certificado

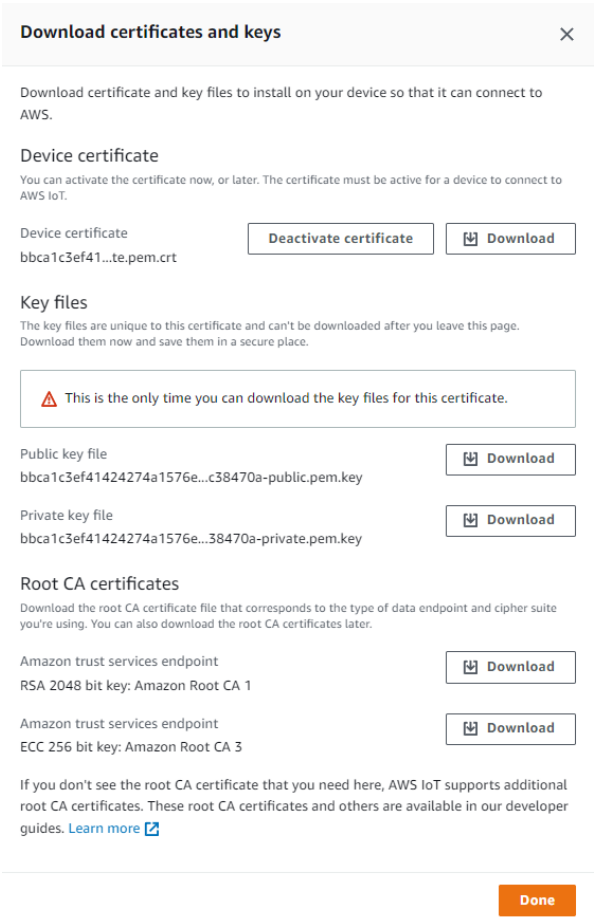


Obs.: Caso não esteja, é possível vincular clicando em **Attach policies** e em **Attach to things** e selecionar a política e a Coisa respectivamente**.**

2.5 Fazer download dos certificados e das chaves

Faça o download dos certificados e das chaves clicando em cada **Download** e salve-os em uma pasta específica. É importante saber onde foram salvos pois serão necessários para a autenticação do Device posteriormente.

Figura 8 – Tela de download de certificados e chaves

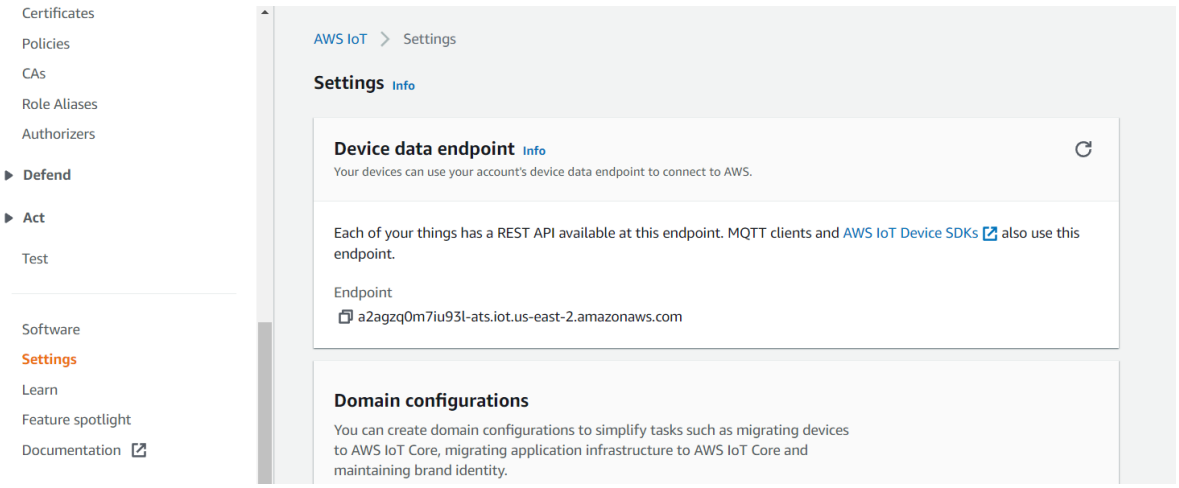


3. Obter o Endpoint

Todas as conexões realizadas entre uma coisa e o serviço de IoT são realizados através de um endpoint. Um endpoint é basicamente uma URL única fornecida a você automaticamente pela AWS. Para obter o seu endpoint, siga os passos a seguir.

- Clique em **Settings** no canto inferior esquerdo da tela;
- Localize a caixa **Device data endpoint**;
- Copie e guarde esta informação, ela será usada pelo Device para acessar o AWS IoT;

Figura 9 – Tela de obtenção de Endpoint



4. Preparar o Publisher Device (programa Python)

A Figura 10 mostra um exemplo simples de Publisher desenvolvido em Python. A partir desta estrutura básica é possível desenvolver vários tipos de programas complexos para comunicação em nuvem através do conceito de Internet of Things. Não se esqueça de instalar a biblioteca AWSIoTPythonSDK, para isso utilize o comando “pip install AWSIoTPythonSDK” no terminal de comando.

Figura 10 – Código Publisher Device Python no VSCode

device01_publisher.py > ...

```python
from time import sleep
import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT

endpoint = "a2agzq0m7iu93l-ats.iot.us-east-2.amazonaws.com"
client_id = "device01"
path_certificate = "D:\\AWS\\device01\\bbca1c3ef41424274a1576e916c2e4892fbaec4dfc06f3cf9fb289902c38470a-certificate.pem.crt"
path_privatekey = "D:\\AWS\\device01\\bbca1c3ef41424274a1576e916c2e4892fbaec4dfc06f3cf9fb289902c38470a-private.pem.key"
path_rootca1 = "D:\\AWS\\device01\\AmazonRootCA1.pem"

my_awsmqtt_client = AWSIoTPyMQTT.AWSIoTMQTTClient(client_id)
my_awsmqtt_client.configureEndpoint(endpoint, 8883)
my_awsmqtt_client.configureCredentials(path_rootca1, path_privatekey, path_certificate)

my_awsmqtt_client.connect()

topic = "test/status"
msg = "Hello World"

print('Inicio de publicação')

for i in range(5):
    my_awsmqtt_client.publish(topic, msg, 1)
    print(f"Publicação 0{i+1}: {msg} ao tópico [{topic}]")
    sleep(0.5)

print('Fim de publicação')
my_awsmqtt_client.disconnect()
```

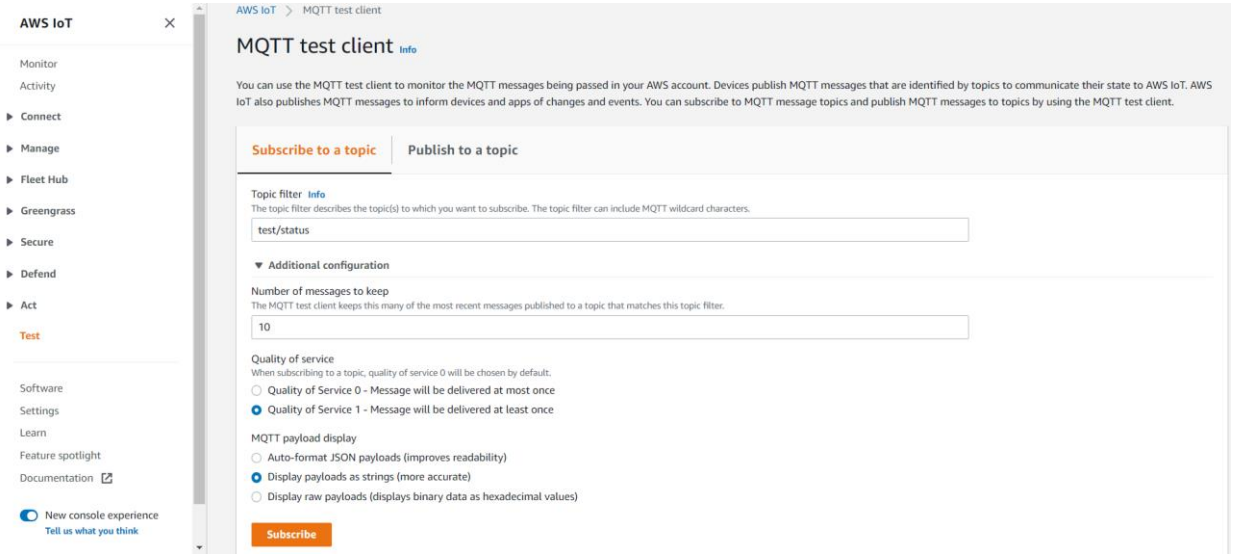
Clique na imagem caso queira ser redirecionado ao código no GitHub.

Caso utilize o código Python acima, substitua o **endpoint** (linha 4) pelo seu, o **cliente_id** por um de sua preferência, e em **path_certificate**, **path_privatekey** e **path_rootca1** altere para o caminho dos arquivos baixados no item 2.5. Em **topic** e **msg** coloque o tópico e a mensagem que preferir.

5. **Testar a comunicação IoT/MQTT**

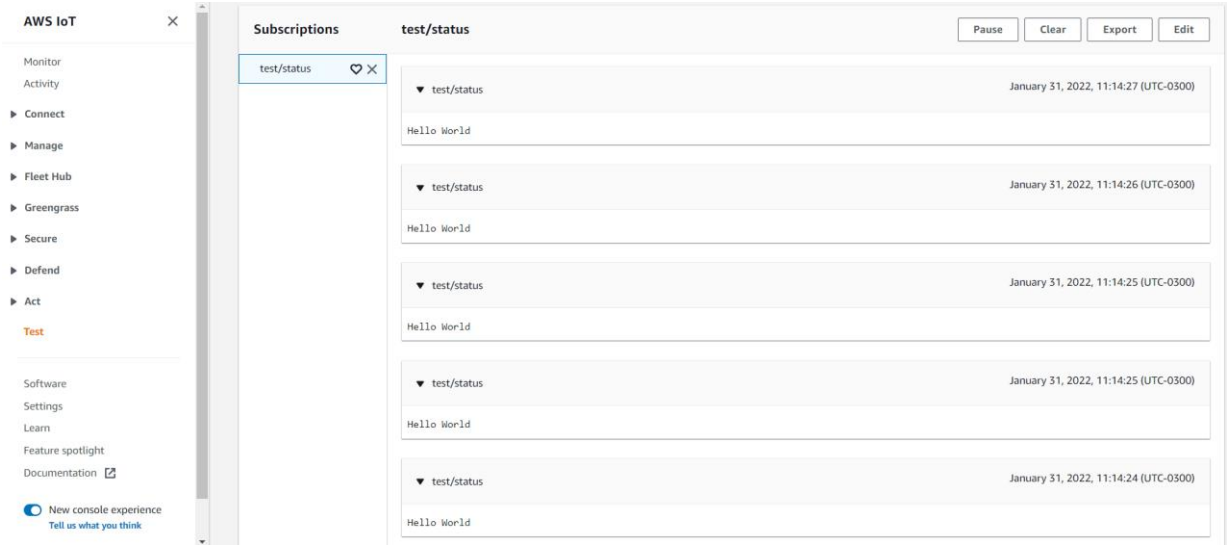
Com o código do Publisher Device pronto, é possível testar a comunicação no próprio console da AWS IoT. Para isso clique em **Test** no menu de navegação da esquerda. Em **Subscribe to a topic** digite o tópico em que o seu Device publicará, escolha o nível de QoS (**Quality of Service)** que melhor te servir, e em **MQTT payload display** selecione **"Display payloads as strings"**, visto que o Publisher Device está publicando mensagens no formato *string.*

Figura 11 – Tela de teste de comunicação MQTT no IoT Core



Após se inscrever em seu tópico no console da AWS IoT, execute o programa do Publisher Device. Volte à tela de teste no console da AWS e verifique se as mensagens foram recebidas corretamente.

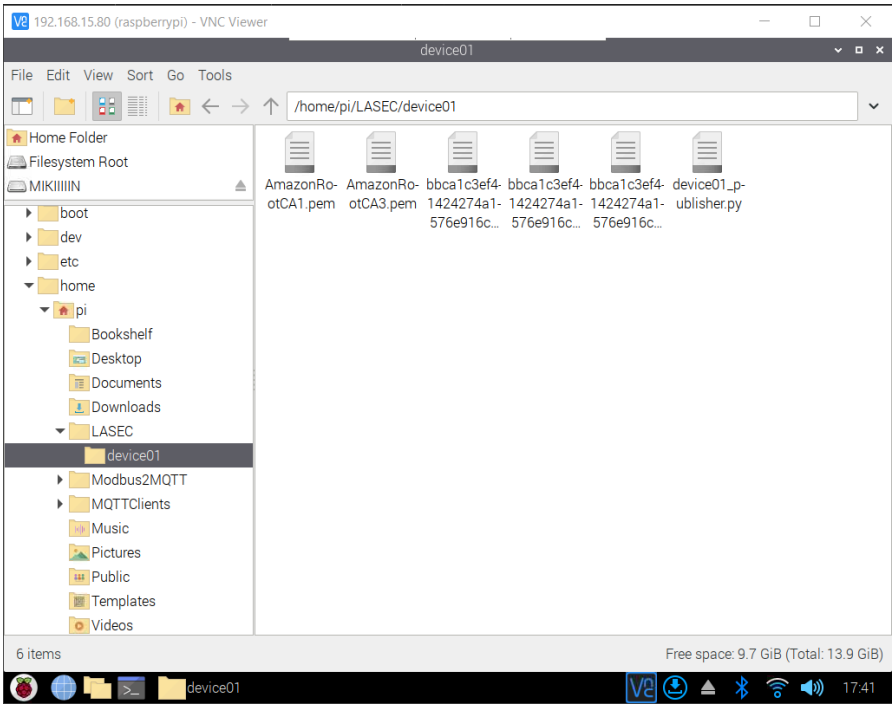
Figura 12 – Tela de teste com publicações recebidas do código em Python



6. Transferir o Device para a Raspberry Pi

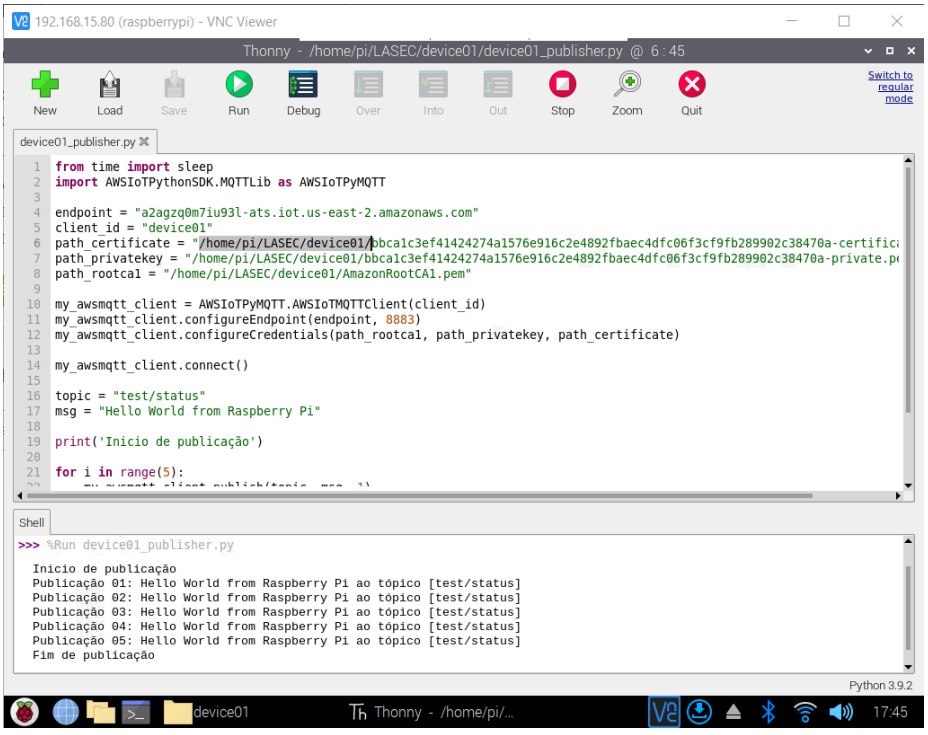
Caso queira fazer esta comunicação IoT entre uma Raspberry Pi e a AWS IoT basta transferir o código, os certificados e as chaves para uma pasta específica em sua Raspberry Pi utilizando o Software FileZilla ou então um Pen Drive.

Figura 13 – Pasta na Raspberry Pi com código Python e certificados



Feito isto, basta adaptar o código alterando as variáveis **path** com os novos caminhos e instalar a biblioteca AWSIoTPythonSDK utilizando o pip no terminal de comando.

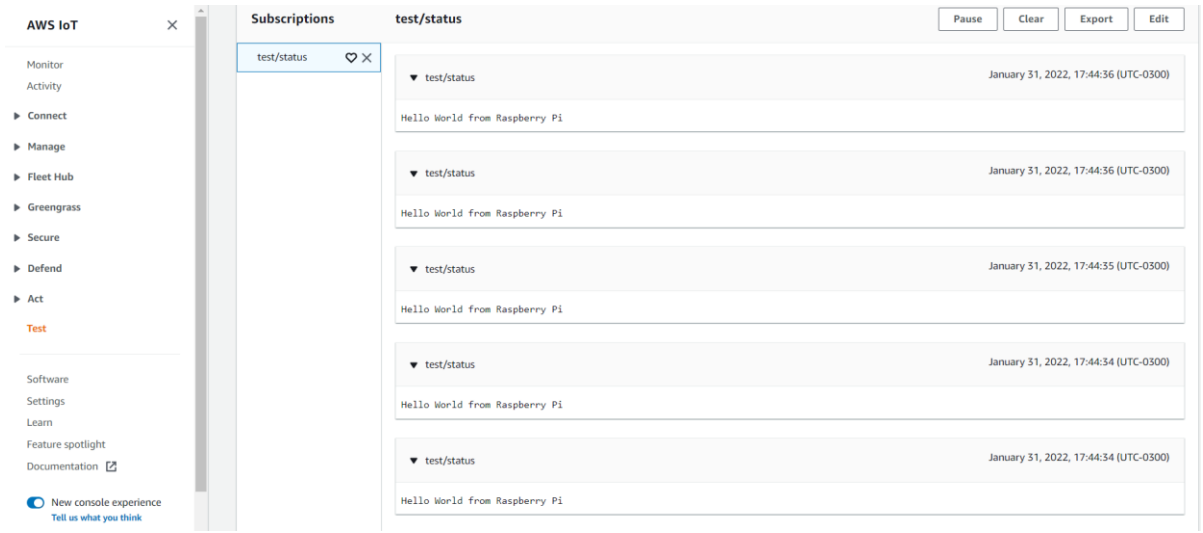
Figura 14 – Código Publisher Device adaptado para a Raspberry Pi



7. Validar o funcionamento

Da mesma forma que foi feito no item 5, é possível testar a comunicação com o IoT Core através do painel **Test.**

Figura 15 – Tela de teste com publicações recebidas da Raspberry Pi



8. Subscriber Device

Vale ressaltar que, além de um Publisher, também é possível fazer um Device com a função de Subscriber. A Figura 16 mostra um exemplo de Subscriber Device.

Figura 16 – Código Subscriber Device Python no VSCode

```python
import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT

endpoint = "a2agzq0m7iu93l-ats.iot.us-east-2.amazonaws.com"
client_id = "device01"
path_certificate = "D:\\AWS\\device01\\bbca1c3ef41424274a1576e916c2e4892fbaec4dfc06f3cf9fb289902c38470a-certificate.pem.crt"
path_privatekey = "D:\\AWS\\device01\\bbca1c3ef41424274a1576e916c2e4892fbaec4dfc06f3cf9fb289902c38470a-private.pem.key"
path_rootca1 = "D:\\AWS\\device01\\AmazonRootCA1.pem"

myAWSIoTMQTTClient = AWSIoTPyMQTT.AWSIoTMQTTClient(client_id)
myAWSIoTMQTTClient.configureEndpoint(endpoint, 8883)
myAWSIoTMQTTClient.configureCredentials(path_rootca1, path_privatekey, path_certificate)

print('-' * 100)
print('MQTT Subscriber'.center(100))
topic = input('\nInscrever-se no tópico: ')

myAWSIoTMQTTClient.connect()


def customCallback(client, userdata, message):
    print(message.payload.decode("utf-8"))

myAWSIoTMQTTClient.subscribe(topic, 1, customCallback)
print(f'\nInscrito com sucesso no tópico [{topic}]! Pressione Enter para sair...\n')
x = input()

myAWSIoTMQTTClient.unsubscribe(topic)
print("Cliente não mais inscrito...")

myAWSIoTMQTTClient.disconnect()
print("Cliente desconectado!")
```

Clique na imagem caso queira ser redirecionado ao código no GitHub.

Adicionalmente, informações sobre as conexões e mensagens postadas serão atualizadas no Dashboard do serviço de IoT da AWS.

Clique em Dashboard na parte esquerda da tela para que sejam exibidas informações como quantidade de conexões, protocolo utilizado para a postagem de mensagens, direção das mensagens, entre outros itens.

Outro grande componente dentro do serviço AWS IoT é o Motor de Regras (**Rules**). Este componente permite que as mensagens recebidas pelo AWS IoT sejam analisadas de maneira simples e disparem ações se um determinado critério for atendido. O motor de regras também pode ser usado para transformar as mensagens antes de chegar ao destino.

Bons estudos!

Material complementar

AWS IoT Core – Developer Guide
MQTT – Protocolos para IoT – Marcelo Barros

Agradecimentos

Prof. Dr. Renato Ferreira Fernandes
Prof. Dr. Márcio José da Cunha