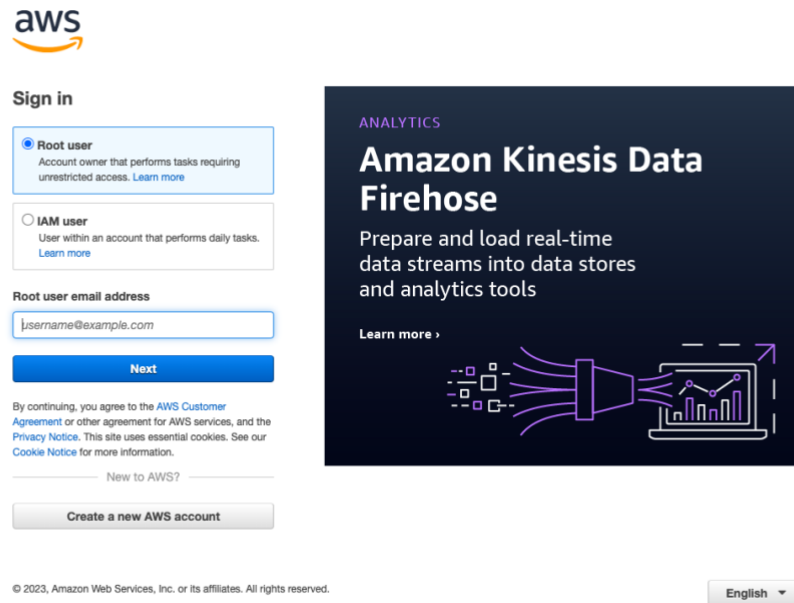
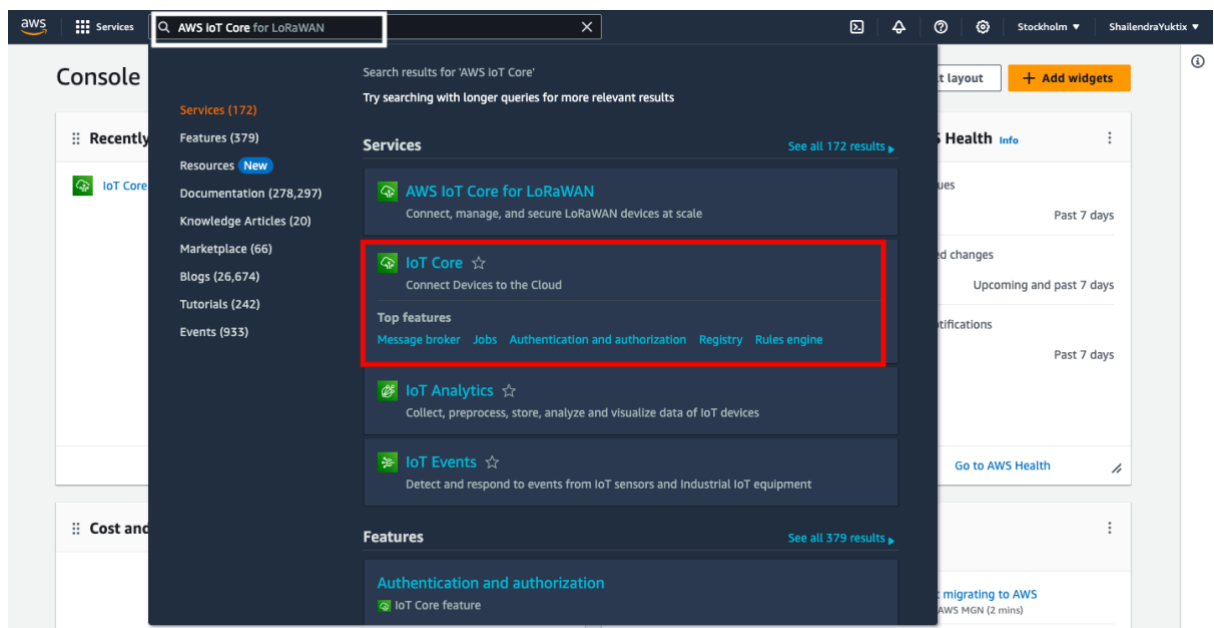


## Account Creation in AWS

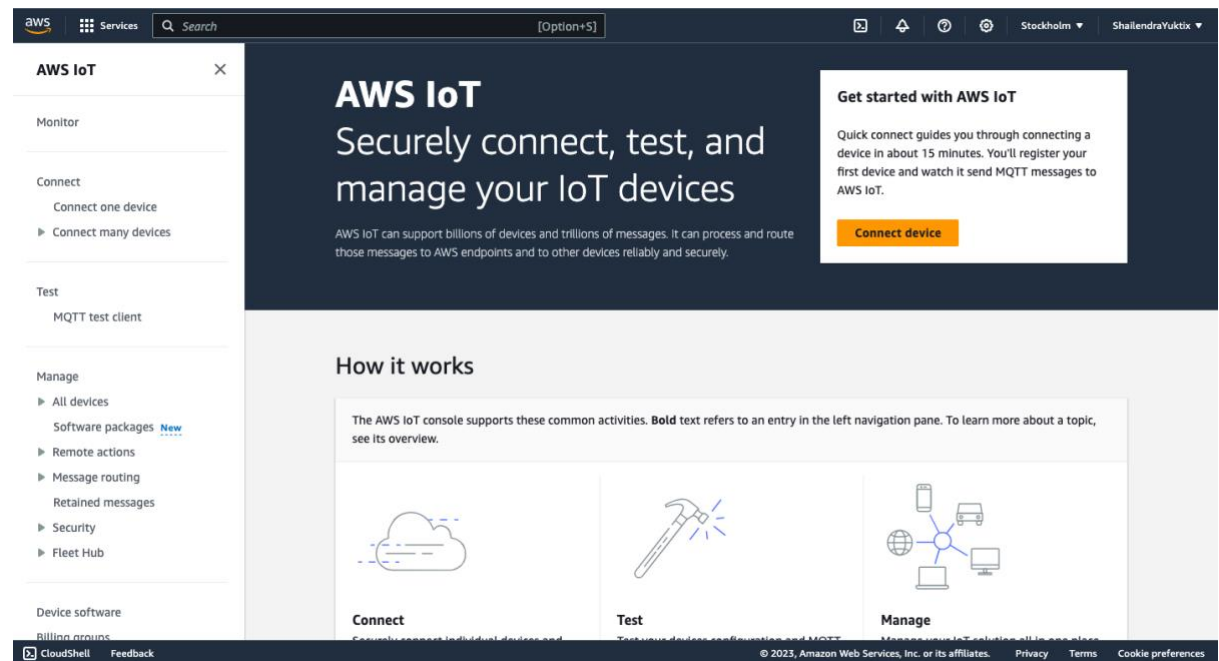
1. First of all you have to create an account in AWS. The account is free for the first year and it will need your Credit card for the purpose of verification.
2. As we are interested in AWS IoT Core, after verification and account creation, sign in to the AWS console as a root user.



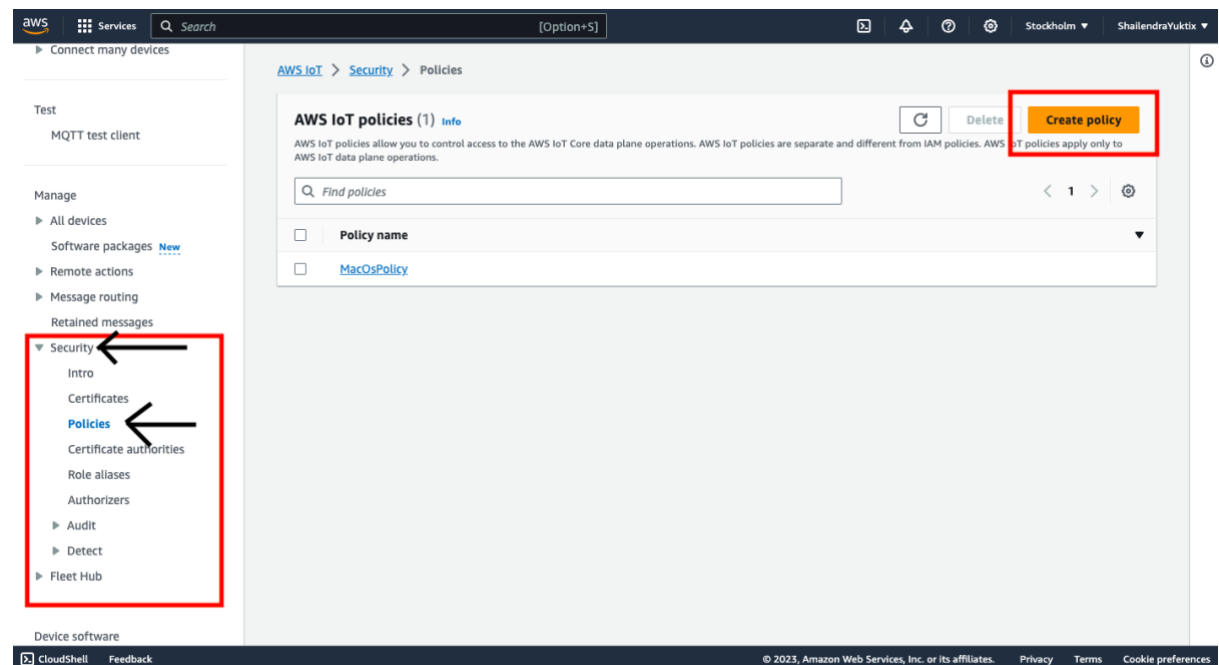
3. Put in your email id and password.
4. After the login, you will see AWS console. In the search bar, search for AWS IOT Core and select IOT Core from the drop down.



- Once you click on IoT Core, you would be redirected to IoT core dashboard.



- The first thing, we are going to do here is create a security policy that we are going to attach to the device that we will create. The policy is created to make sure that we give sufficient right to the device to send the data to the IoT core.
- On the left hand side menu, click on **security** and then click on **Policies**.



- The click on Create Policy button on the top, right corner.
- Once you click on create policy, it will ask to provide a policy name. Use any human readable name. Remember, this policy, we would have to attach to the device that we will create down the line.

Connect many devices

Test

MQTT test client

Manage

- All devices
- Software packages **New**
- Remote actions
- Message routing
- Retained messages

**Create policy** Info

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

**Policy properties**

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name

AWSIOTTest

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (\_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

10. After the Policy name, scroll down. In the policy document, click on Add new statement.

**Policy properties**

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name

AWSIOTTest

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (\_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

Tags - optional

**Policy statements** | Policy examples

**Policy document** Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Builder JSON

No policy statements

Add new statement

Cancel Create

11. Now after clicking on create new statement, select Allow under Policy Effect. Then click on arrow under the Policy action and select IoT connect. Under the Policy resource, put \*.

12. Repeat step 11, 4 time, till you have chosen all the highlighted Policy actions as show in image below.

Connect many devices

Test

MQTT test client

Manage

- All devices
- Software packages **New**
- Remote actions
- Message routing
- Retained messages
- Security
  - Intro
  - Certificates
  - Policies**
  - Certificate authorities
  - Role aliases
  - Authorizers
- Audit
- Detect
- Fleet Hub

**Create policy** Info

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

**Policy properties**

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name

AWSIOTTest

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (\_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

Tags - optional

**Policy statements** | Policy examples

**Policy document** Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect

Allow

Add new statement

Builder JSON

iot:Connect  
Permission to connect to the AWS IoT Core message broker.

iot:Publish  
Permission to publish an MQTT topic.

iot:Receive  
Permission to receive a message from AWS IoT Core.

iot:GetRetainedMessage  
Permission to get the contents of a retained message.

iot:ListRetainedMessages  
Permission to retrieve a summary list of retained messages.

iot:RetainPublish  
Permission to publish an MQTT message with the RETAIN flag set.

iot:Subscribe  
Permission to subscribe to a topic filter.

Device Shadow policy actions

iot:DeleteThingShadow

iot:Connect

arn:aws:iot:region:account:resource/\*

Remove

13. Please make sure that once you choose all the 4 Policy effect, Policy action and put in a \* in Policy resource, your current screen look like below image. Once verified, click on **create**.

**Policy statements** | Policy examples

**Policy document** [Info](#)

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect	Policy action	Policy resource	
Allow	iot:Connect	*	Remove
Allow	iot:Publish	*	Remove
Allow	iot:Receive	*	Remove
Allow	iot:Subscribe	*	Remove

[Add new statement](#)

Cancel [Create](#)

14. You will see your newly created policy in the next page.

Connect many devices

Test

MQTT test client

Manage

All devices

Software packages [New](#)

Remote actions

Message routing

Retained messages

Security

Intern

Successfully created policy AWSIoTTest.

[View policy](#)

[AWS IoT](#) > [Security](#) > Policies

**AWS IoT policies (2)** [Info](#)

AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

☐ Policy name

☐ MacOsPolicy

☐ AWSIoTTest

15. Now next step would be to create, one device ( there is an option to create many devices and attach the same policy to all of them, but we will start with just 1 device. On the left hand side, click on **Connect one device**.

**AWS IoT**

Monitor

Connect

Connect one device

Connect many devices

Test

MQTT test client

Successfully created policy AWSIoTTest.

[View policy](#)

[AWS IoT](#) > [Security](#) > Policies

**AWS IoT policies (2)** [Info](#)

AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

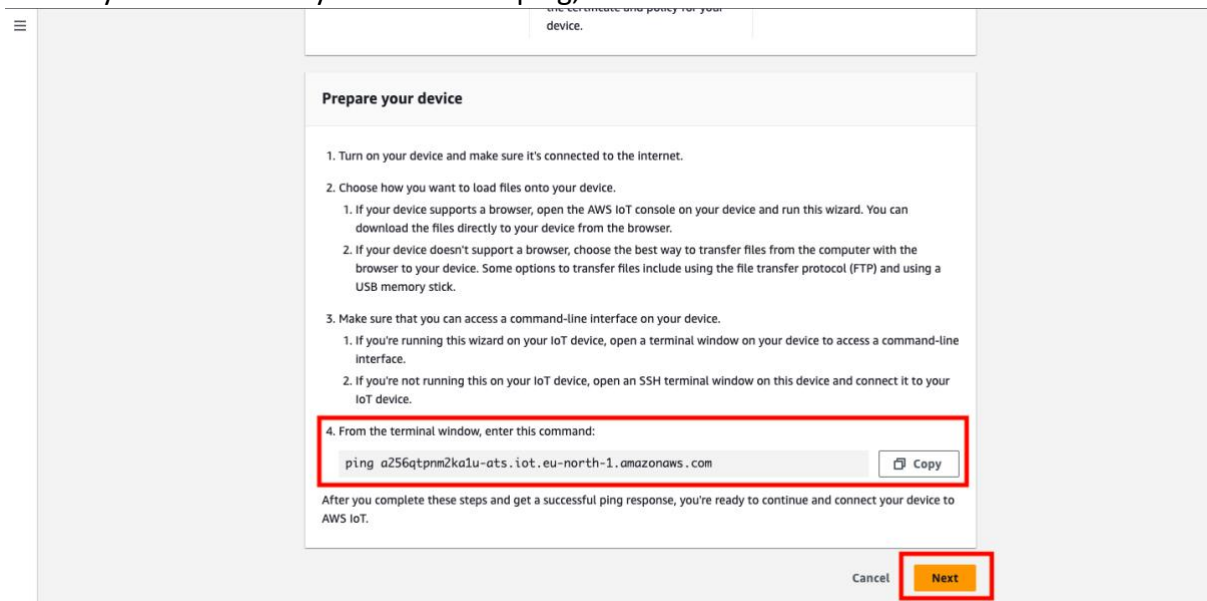
☐ Policy name

☐ MacOsPolicy

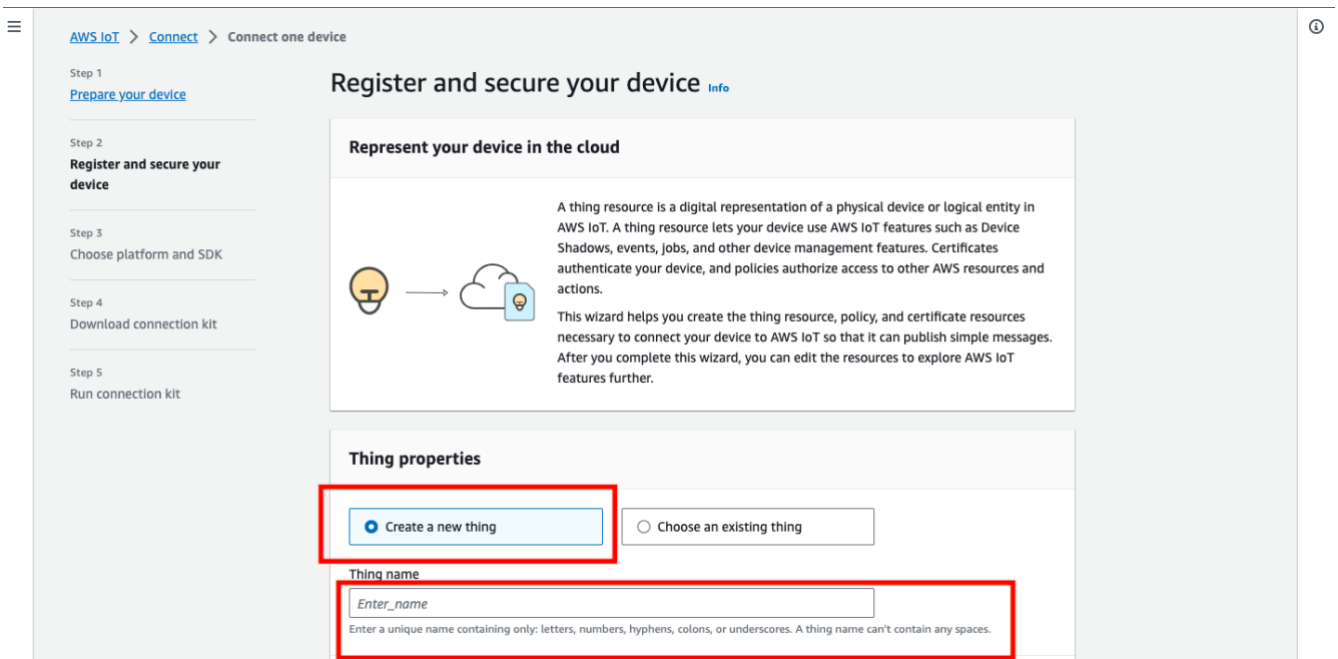
☐ AWSIoTTest

16. Once, you click on Connect one device, a new page will open, follow the steps mentioned there. Just to test, if your device is able to communicate with IoT core end point, in the bottom of the page, use the command to use the **PING Test**. You

can run the command on the terminal ( if you are using RPI or any other SBC ). Once you do that and you are able to ping, click on **Next**



17. On the next screen, select **Create a New thing** Or you can choose the existing thing. Thing here is the device. Once you choose, Create a New thing, provide the thing name. Make sure, you follow all the rules of naming convention.



18. Leave all the other options and click on Next.

**Thing properties**

☒ Create a new thing ☐ Choose an existing thing

Thing name  
awslootttestdevice1  
Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

**Additional configurations**  
You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ Thing type - optional
- ▶ Searchable thing attributes - optional
- ▶ Thing groups - optional
- ▶ Billing group - optional

**Certificate and policy for your device**  
Your device requires a unique device certificate to securely authenticate its identity to AWS IoT, and an AWS IoT policy that authorizes it to send and receive messages. We'll create these resources for your device automatically. You can review and edit their properties later, if necessary.

Cancel Previous **Next**

19. In the next screen, you would be asked to choose the OS and Language. Choose Linux / macOS and Python and then click on next.

**Platform and SDK**

Choose the platform OS and AWS IoT Device SDK that you want to use for your device.

**Device platform operating system**  
This is the operating system installed on the device that will connect to AWS.

☒ Linux / macOS  
Linux version: any  
macOS version: 10.13+

☐ Windows  
Version 10

**AWS IoT Device SDK**  
Choose a Device SDK that's in a language your device supports.

☐ Node.js  
Version 10+  
Requires Node.js and npm to be installed

☒ Python  
Version 3.6+  
Requires Python and Git to be installed

☐ Java  
Version 8  
Requires Java JDK, Maven, and Git to be installed

Cancel Previous **Next**

20. In the next screen, download the connection kit. Once downloaded, unzip using the unzip command given in the same page. Then click on next.

**Download connection kit**

**Unzip connection kit on your device**

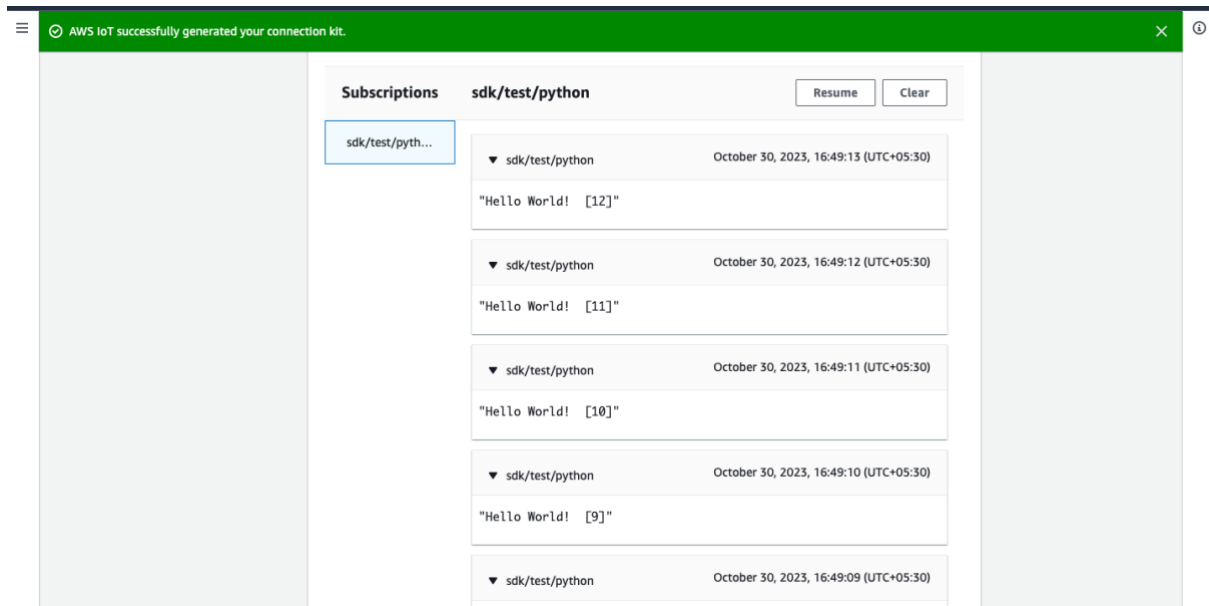
After the connection kit is on your device, unzip it using this command:

```
unzip connect_device_package.zip
```

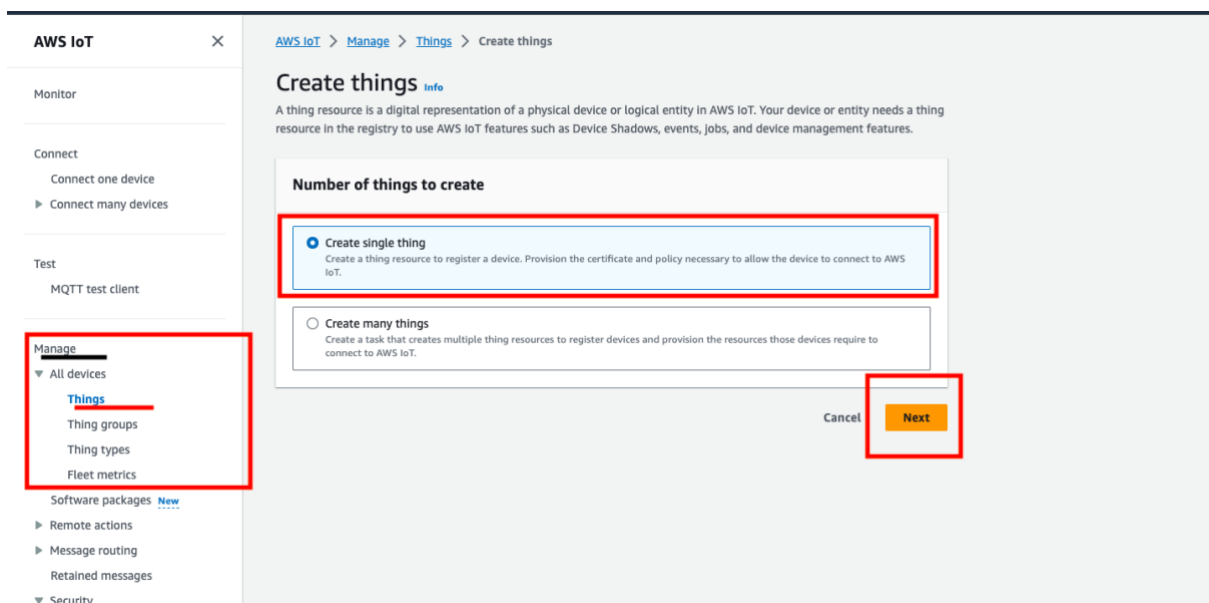
Copy

Cancel Previous **Next**

21. Once you click on next, follow the steps given in page to run the start.sh file.
22. Once you run the file, if you see the messages posted from the device on this page, that means, you are able to successfully connect a device to IoT core.



23. Now as you are able to connect one thing ( for test purpose ), now lets **Create one thing**. Click on the left hand side menu option on the top ( three horizontal lines ). And then Under Manage, click on Things and a new screen will come. Now select create Single thing and then click on next



24. In next screen, under thing properties provide a name to your thing, below that, select no shadow ( default ) and then click on next.

The screenshot shows the AWS IoT console interface. On the left is a navigation sidebar with sections: Monitor, Connect (with sub-items 'Connect one device' and 'Connect many devices'), Test (with 'MQTT test client'), and Manage (with 'All devices' expanded, showing 'Things', 'Thing groups', 'Thing types', 'Fleet metrics', 'Software packages', 'Remote actions', 'Message routing', and 'Retained messages'). The main content area is titled 'Specify thing properties' and includes a breadcrumb trail: 'AWS IoT > Manage > Things > Create things > Create single thing'. It shows three steps: Step 1 (current), Step 2 (optional), and Step 3 (optional). The 'Thing properties' section has a 'Thing name' input field containing 'IoTTest1'. Below this is an 'Additional configurations' section with expandable options for 'Thing type', 'Searchable thing attributes', 'Thing groups', 'Billing group', and 'Packages and versions'.

**Specify thing properties** [Info](#)

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

**Thing properties** [Info](#)

Thing name

IoTTest1

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

**Additional configurations**

You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ Thing type - optional
- ▶ Searchable thing attributes - optional
- ▶ Thing groups - optional
- ▶ Billing group - optional
- ▶ Packages and versions - optional

The screenshot shows the 'Device Shadow' configuration step in the AWS IoT console. It includes a description of Device Shadows and three radio button options: 'No shadow' (selected), 'Named shadow', and 'Unnamed shadow (classic)'. The 'Next' button is highlighted with a red box.

**Device Shadow** [Info](#)

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

☒ No shadow

☐ Named shadow

Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

☐ Unnamed shadow (classic)

A thing can have only one unnamed shadow.

Cancel **Next**

25. In the new screen, go with default option – Auto Generate a new certificate and click on next.

The screenshot shows the 'Configure device certificate' step in the AWS IoT console. It includes a description of device certificates and three radio button options: 'Auto-generate a new certificate (recommended)' (selected), 'Use my certificate', and 'Upload CSR'. There is also a 'Skip creating a certificate at this time' option. The 'Next' button is highlighted with a red box.

**Configure device certificate - optional** [Info](#)

A device requires a certificate to connect to AWS IoT. You can choose how to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

**Device certificate**

☒ Auto-generate a new certificate (recommended)

Generate a certificate, public key, and private key using AWS IoT's certificate authority.

☐ Use my certificate

Use a certificate signed by your own certificate authority.

☐ Upload CSR

Register your CA and use your own certificates on one or many devices.

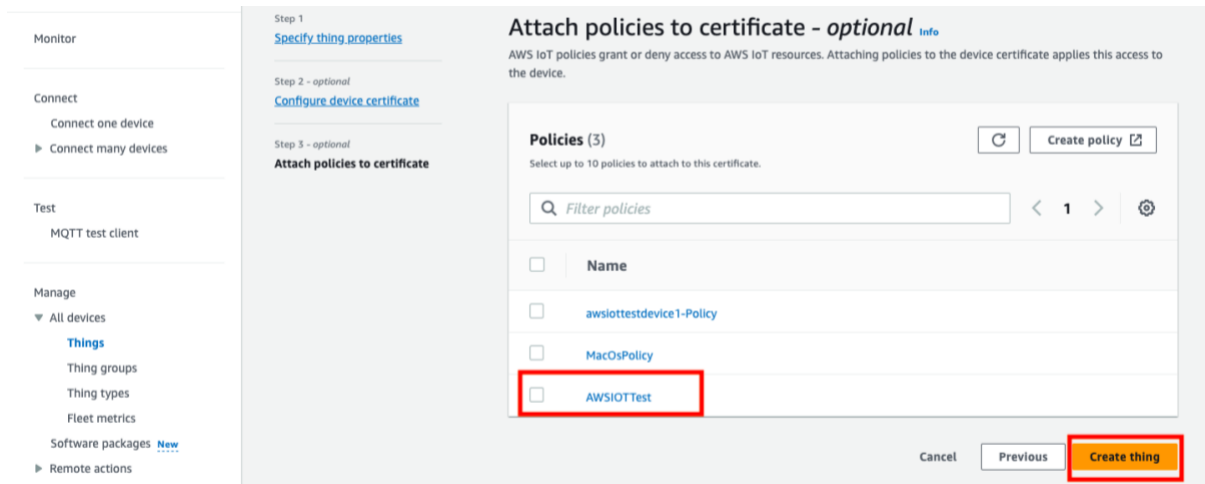
☐ Skip creating a certificate at this time

You can create a certificate for this thing and attach a policy to the certificate at a later time.

Cancel Previous **Next**

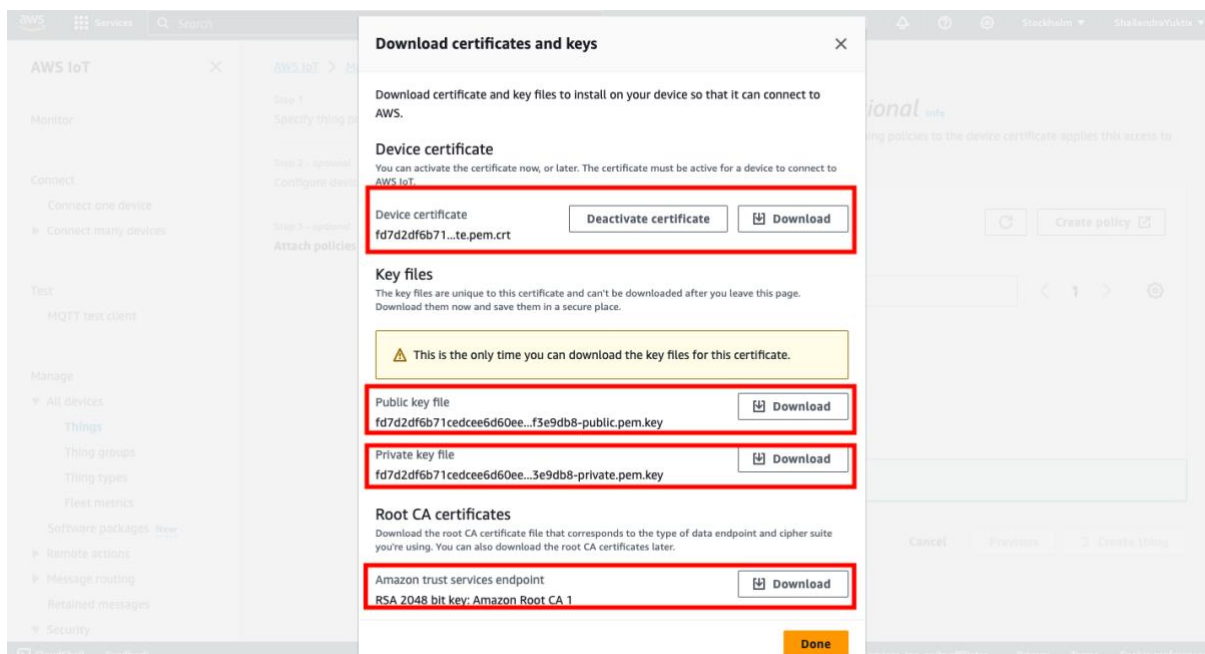


26. In the next screen, select the policy that you created in step 14 and click on **create thing**.



27. Now in the next screen, you would be show certain certificates. Please note, that you have to download all of them and rename them as show below (for ease of use in the code ). These certificates will be in the same directory as that of our code.

Certificate File Names	
File	File Path and name
Private Key	private.pem.key
Public Key	Not required for now ( public.pem.key )
Device Certificate	Device.pem.cert
Root CA certificate	Amazon-root-CA-1.pem



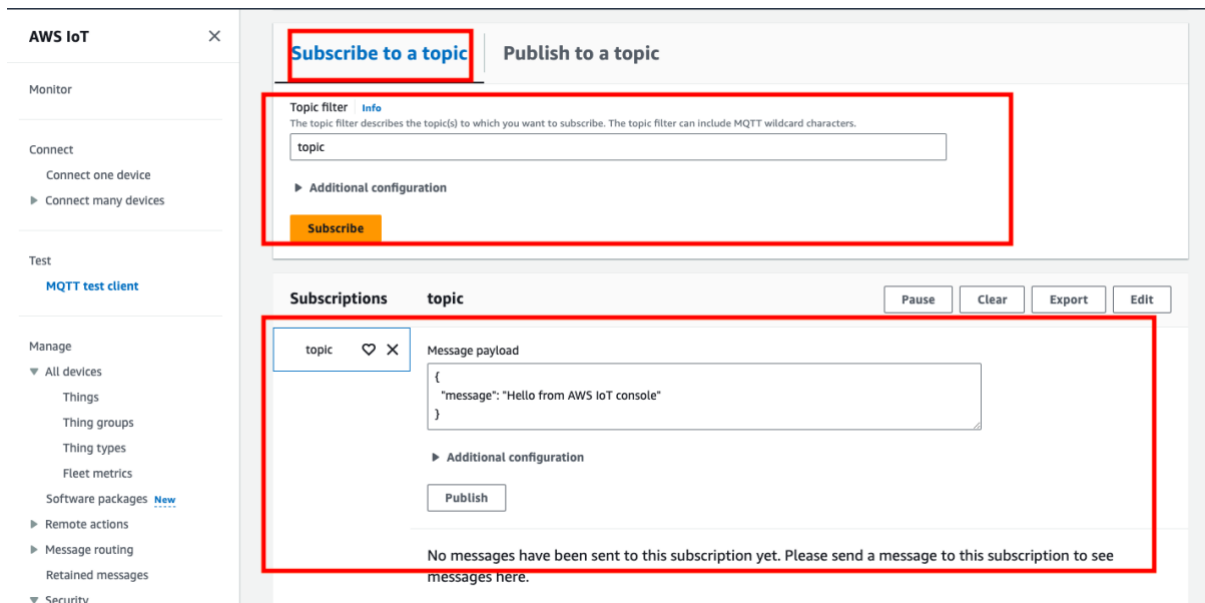
28. In case if you are using any Linux system or macOS for testing purpose, better is to create a virtual environment so that multiple version of Python mess with your code.
29. I am using Python3.7 for this project.
30. Once you run the code and you have the right certificates in place. You should see following in the your linux or RPI machine.

```
(awsiot) Shailendras-MacBook-Air:AWSIoTCore shailendra0408$ python3 TempSensor.py
26.375310839752338
{"message": "OK", "traceId": "4061a0bc-7c16-9282-72b9-bb8c64c0b333"}27.39171710081452
{"message": "OK", "traceId": "102c3f6c-79a6-f021-9bda-3d2367004a8c"}32.811024725857195
{"message": "OK", "traceId": "805a57af-ccbc-d275-baf9-170a44fd7222"}20.969604349927092
{"message": "OK", "traceId": "75bcce2a-7380-59a8-0e3f-05876fe2755b"}34.50321472313371
{"message": "OK", "traceId": "29c90731-4c8c-c8a5-8d29-bda5bbe801b8"}32.43374653323447
{"message": "OK", "traceId": "80d5922a-1dce-0e7e-45c3-d868fbc27393"}22.749645855866994
{"message": "OK", "traceId": "3428af6c-7da5-ed8d-1834-d6a117cda26e"}30.855992526200556
{"message": "OK", "traceId": "5eababdc-4859-8048-2515-b5b570b1d485"}25.243679186571857
{"message": "OK", "traceId": "5cb7e288-202e-1bc2-ad24-930ae428bca6"}22.39276796251556
{"message": "OK", "traceId": "265d6c89-932f-ae42-5b28-c3c94a6f29b8"}33.589881140721495
{"message": "OK", "traceId": "8968c392-7b22-2bdc-178d-61fea51256a9"}33.309791099863595
{"message": "OK", "traceId": "73ff8c45-3915-4798-193b-9529b4c9f005"}21.221091338618866
{"message": "OK", "traceId": "db7be886-b8bb-d44c-01f3-de95d3a70829"}29.755330807414396
{"message": "OK", "traceId": "f0f5993a-5fd4-de6d-2fdb-af6049deb6e8"}34.7000065801024
{"message": "OK", "traceId": "3bea3502-6e3a-6272-589e-253690c0afaa"}28.59896154280363
{"message": "OK", "traceId": "30364669-4ebc-a93c-d3c3-64d47668003f"}27.404390310426766
{"message": "OK", "traceId": "93ae48b5-e65d-7060-eafd-b87206758ecb"}34.63134995607386
{"message": "OK", "traceId": "5604a026-b4e5-9a94-3cbe-c2112215f900"}23.895844306461623
{"message": "OK", "traceId": "23db2412-9465-39bb-a15f-c4602bace6a7"}31.913427585151048
{"message": "OK", "traceId": "6eb140ed-e6ed-7aaa-f434-ce2d38fd9903"}23.583658896674855
{"message": "OK", "traceId": "2a1e26a8-1e13-80d4-244b-d8001064beb7"}26.358638656012907
{"message": "OK", "traceId": "92e8bc29-87b8-13c0-af85-828b4f4fde75"}22.958987945676206
{"message": "OK", "traceId": "c6b3b4da-71a1-a0e2-0612-54ce828eeaa1"}25.276902262207425
{"message": "OK", "traceId": "af940cdc-a799-312b-33db-0b8932f32844"}20.25925683147578
{"message": "OK", "traceId": "53a328dd-c6bf-746f-ba82-a0c8396b727d"}31.010885717022305
{"message": "OK", "traceId": "29216b75-383d-e462-3fc6-c50fe6ed0253"}[]
```

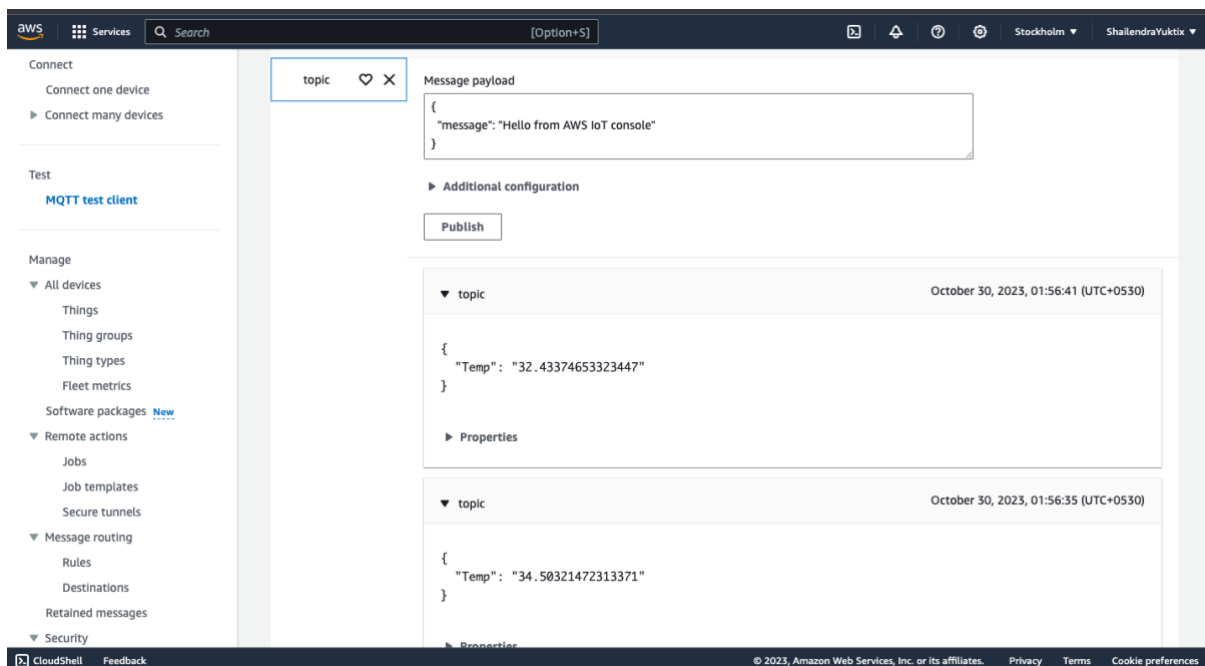
31. On the AWS IoT Core, on the left hand side menu, search for Test and then below that MQTT test client ( in this, we are going to use HTTPS to send MQTT message to a kind of broker over a topic ).

The screenshot shows the AWS IoT console interface. On the left-hand side, there is a navigation menu with the following items: Monitor, Connect, Test (highlighted with a red box), and Manage. Under the 'Test' menu, the 'MQTT test client' option is visible. At the top of the console, there are two green status messages: 'You successfully created thing IoTTest1.' and 'You successfully created certificate fd7d2df6b71cedcee6d0ee27d6e8005c77004333057f9d534c5551e4f3e9db8.' Both messages are highlighted with red boxes. The main content area shows the 'Things (4)' list with columns for Name and Thing type. The listed items are IoTTest1, awsiottestdevice1, MacIoT, and MacVirtualDevice.

32. Once you click on the MQTT test client, in the new page, under Subscribe to a topic, enter the name of the topic you are subscribing to. For now, write **topic** and then click on subscribe.



33. Once you subscribe, you will start receiving data from the device or Linux machine or macOS.



34. Now next step is to create a rule for this device, which will invoke a Lambda function. That Lambda function will write this IoT device data into AWS Timestream.