AmebaPro2 Amazon FreeRTOS-LTS - Getting Started Guide

Realtek Semiconductor Corp.

No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan

Tel.: +886-3-578-0211. Fax: +886-3-577-6047

www.realtek.com

USING THIS DOCUMENT

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

1 AmebaPro2 RTL8735B Board

1.1 AmebaPro2 Demo EVB

Ameba Demo board home page: https://www.amebaiot.com/en/amebapro2/

MCU

Part Number: RTL8735B

32-bit Arm v8M, up to 500MHz

MEMORY

768KB ROM

512KB RAM

Supports MCM embedded DDR2/DDR3L memory up to 128MB

External Flash up to 64MB

KEY FEATURES

Integrated 802.11 a/b/g/n Wi-Fi, 2.4GHz/5GHz

Bluetooth Low Energy (BLE) 4.2

Integrated Intelligent Engine @ 0.4 TOPS

Ethernet Interface

USB Host/Device

SD Host

ISP

Audio Codec

H.264/H.265

Secure Boot

Crypto Engine

OTHER FEATURES

2 SPI interfaces

1 I2C interface

8 PWM interfaces

3 UART interfaces

3 ADC interfaces

2 GDMA interfaces

Max 23 GPIO

1.2 PCB Layout Overview

The PCB layout of AmebaPro2 is shown in Fig 1-1.

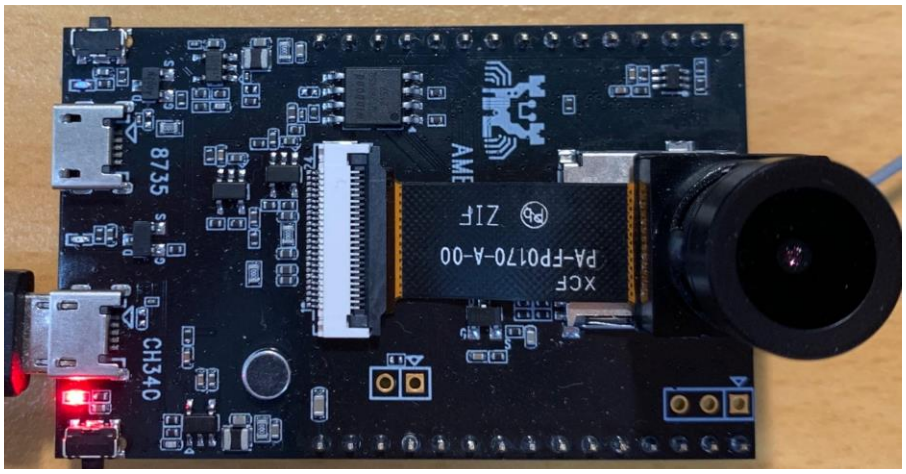


Fig 1-1 Demo board – PCB layout (2D)

1.3 Log UART

The USB Type-C log UART is shown in Fig 1-2.

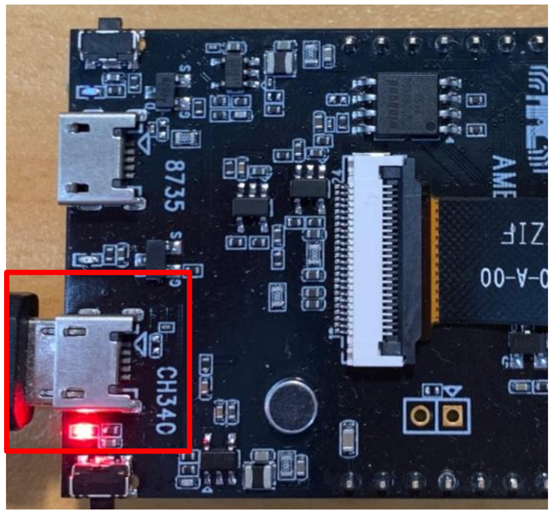


Fig 1-2 Demo board – log UART

2 Configure AWS IoT Core

2.1 Set up your AWS account and Permissions

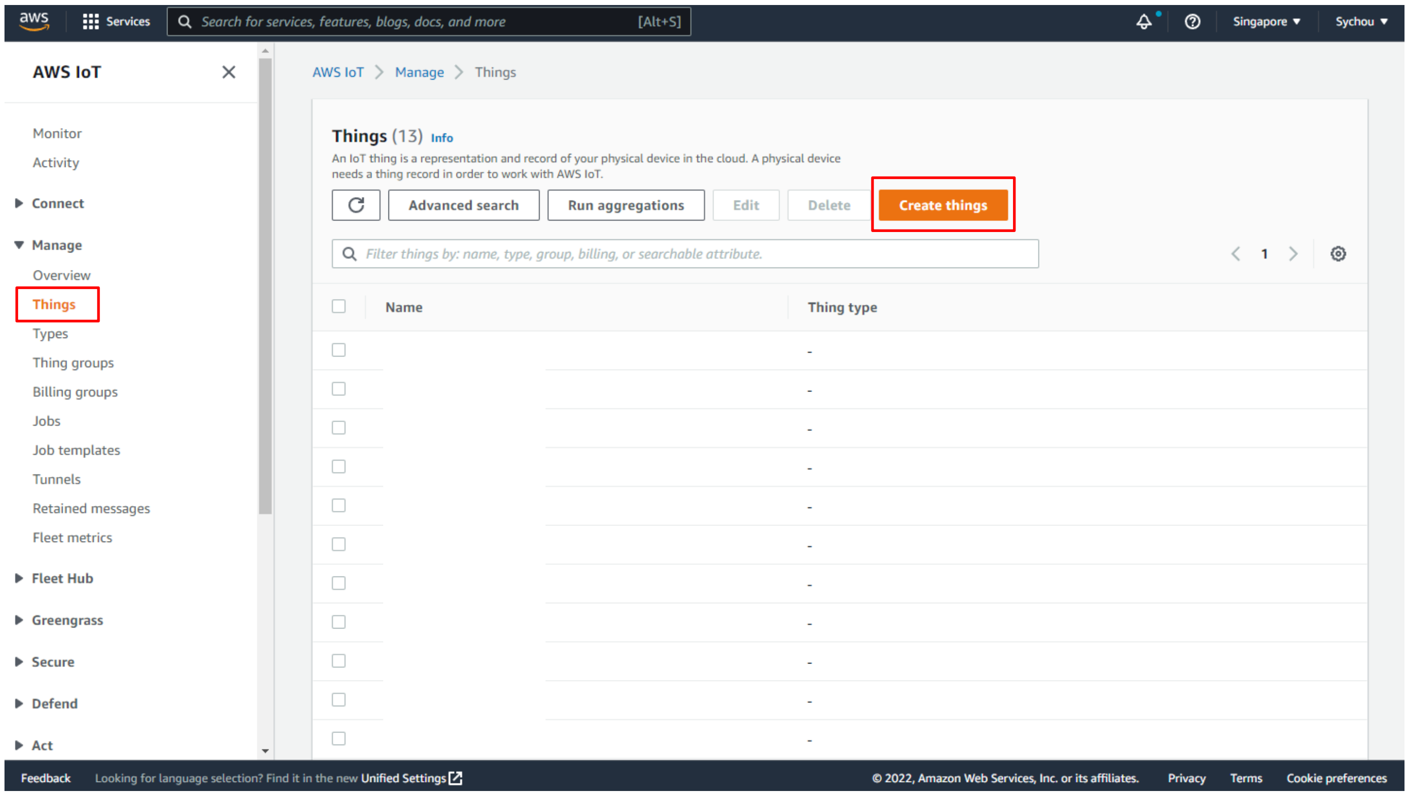
Refer to the instructions at Set up your AWS Account https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html. Follow the steps outlined in these sections to create your account and a user and get started:

- Sign up for an AWS account
- Create a user and grant permissions
- Open the AWS IoT console

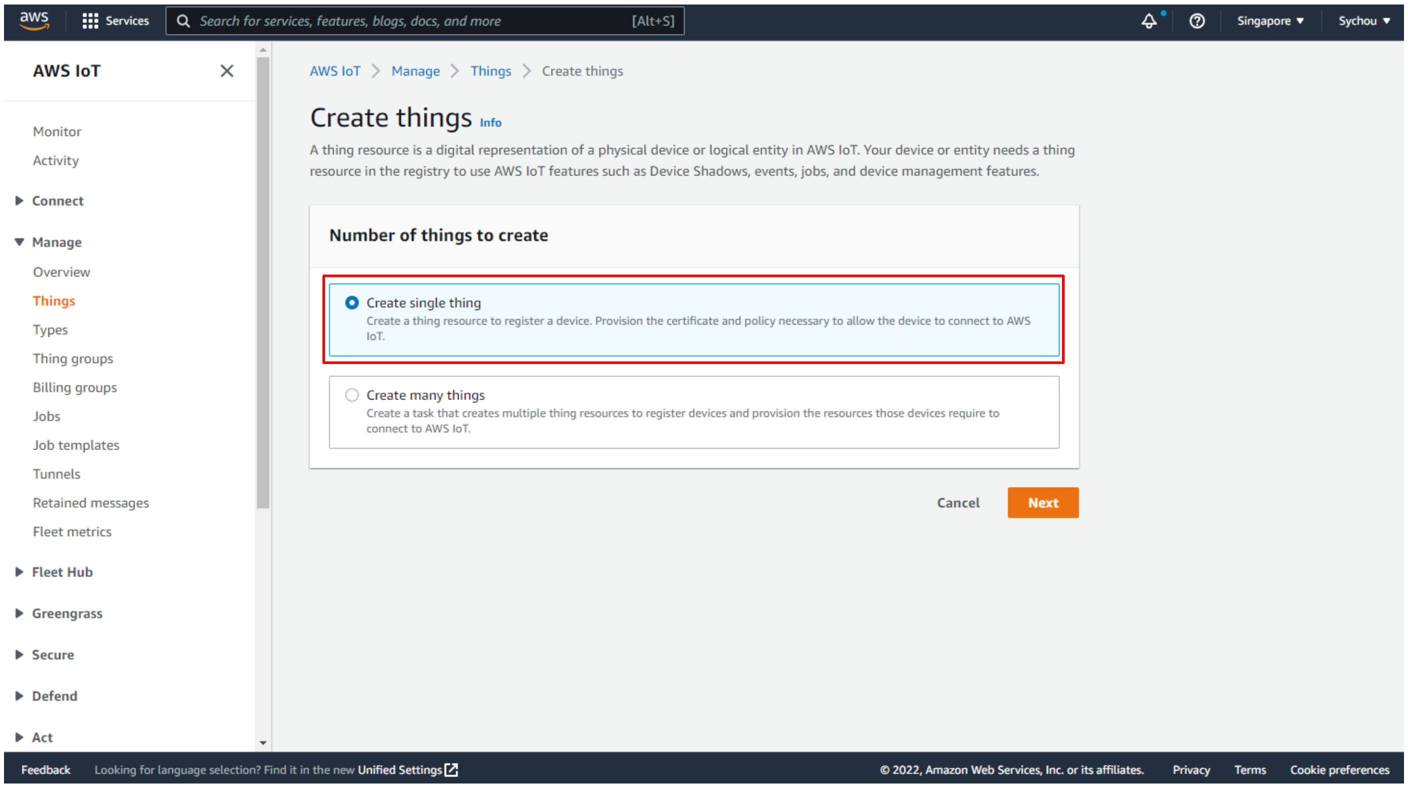
Please pay special attention to the Notes in AWS webpage.

2.2 Create a New Device

To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click “Create things”.



An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.



Create things

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.

Number of things to create

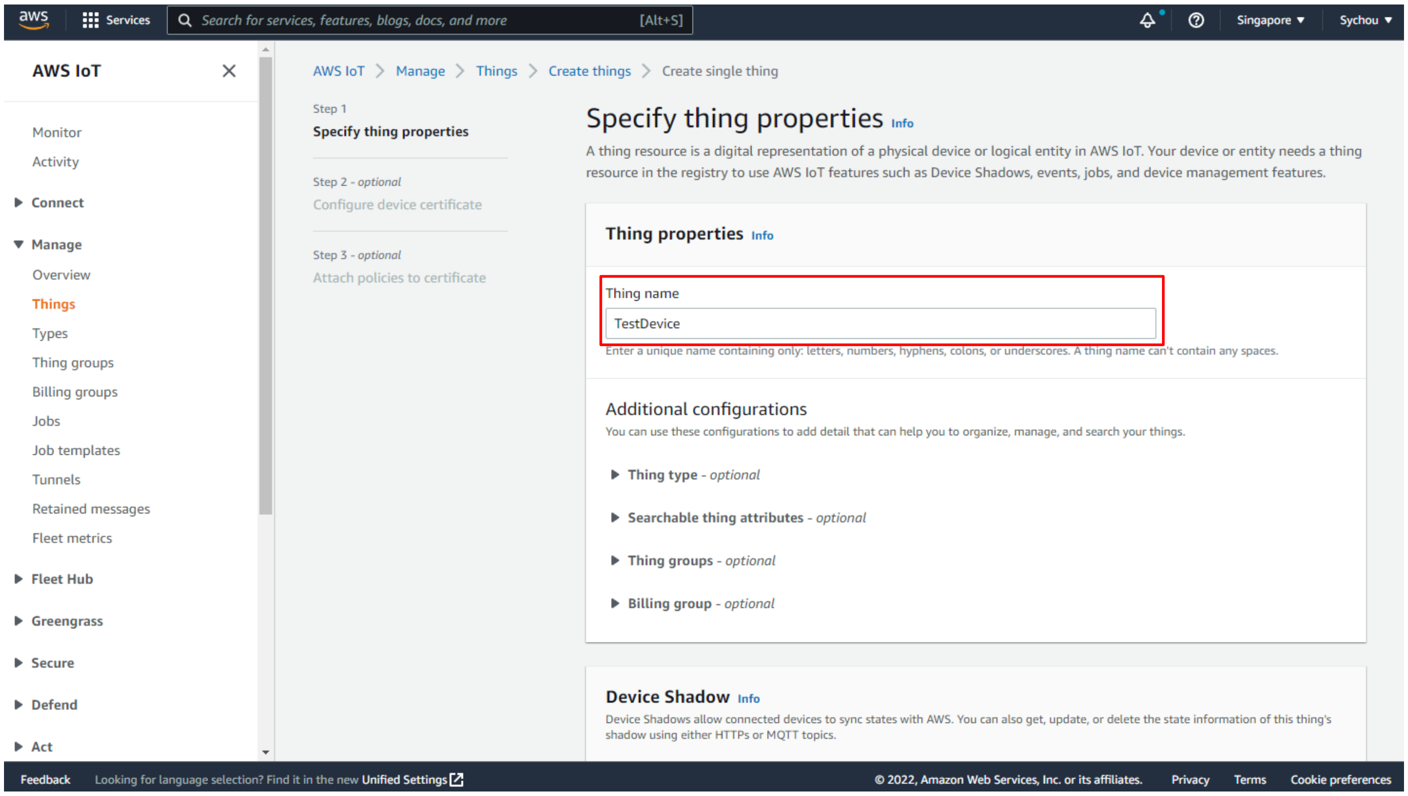
[✓ Create single thing]
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

Create many things
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.

Then, name the new device. This example uses the name TestDevice.



Specify thing properties

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

Thing name: TestDevice

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

Additional configurations

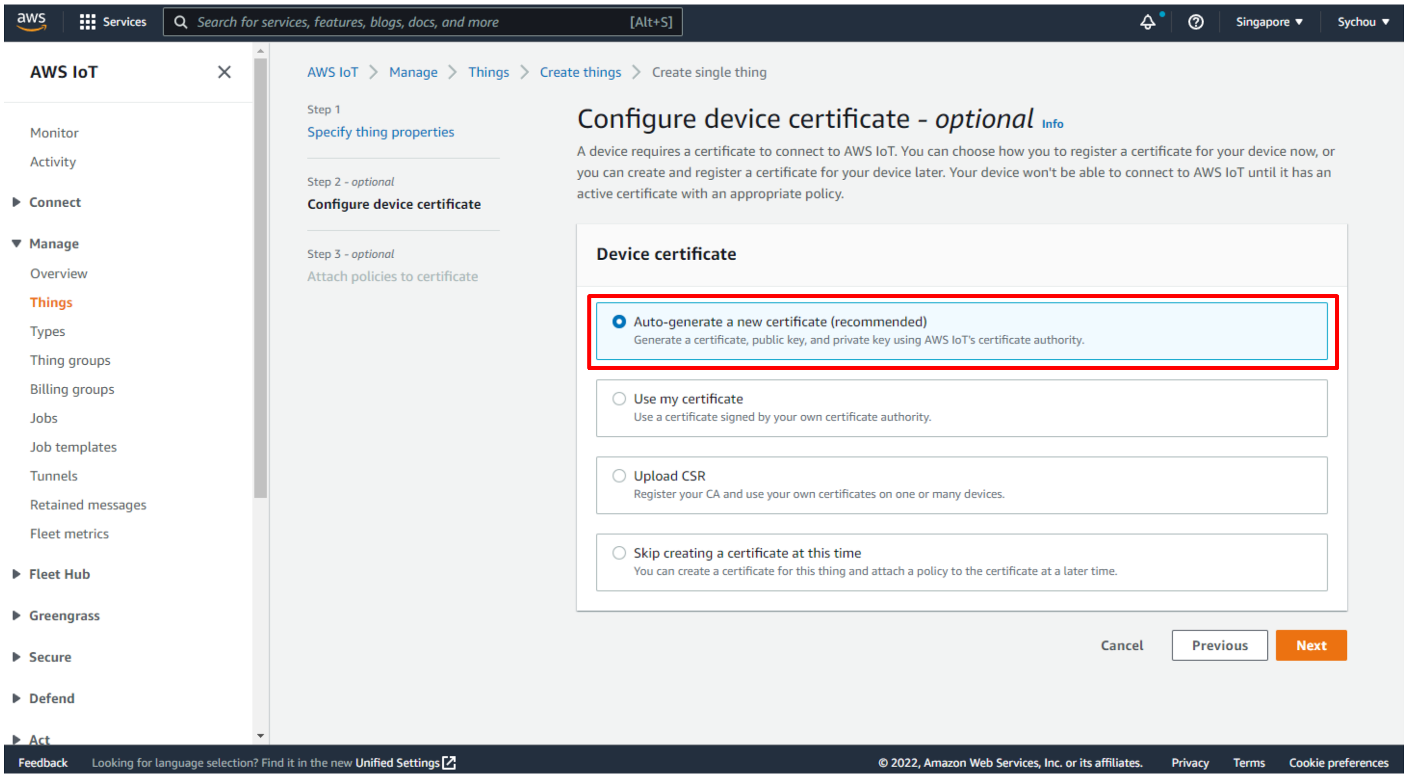
You can use these configurations to add detail that can help you to organize, manage, and search your things.

Device Shadow

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.

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.



Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how you 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.

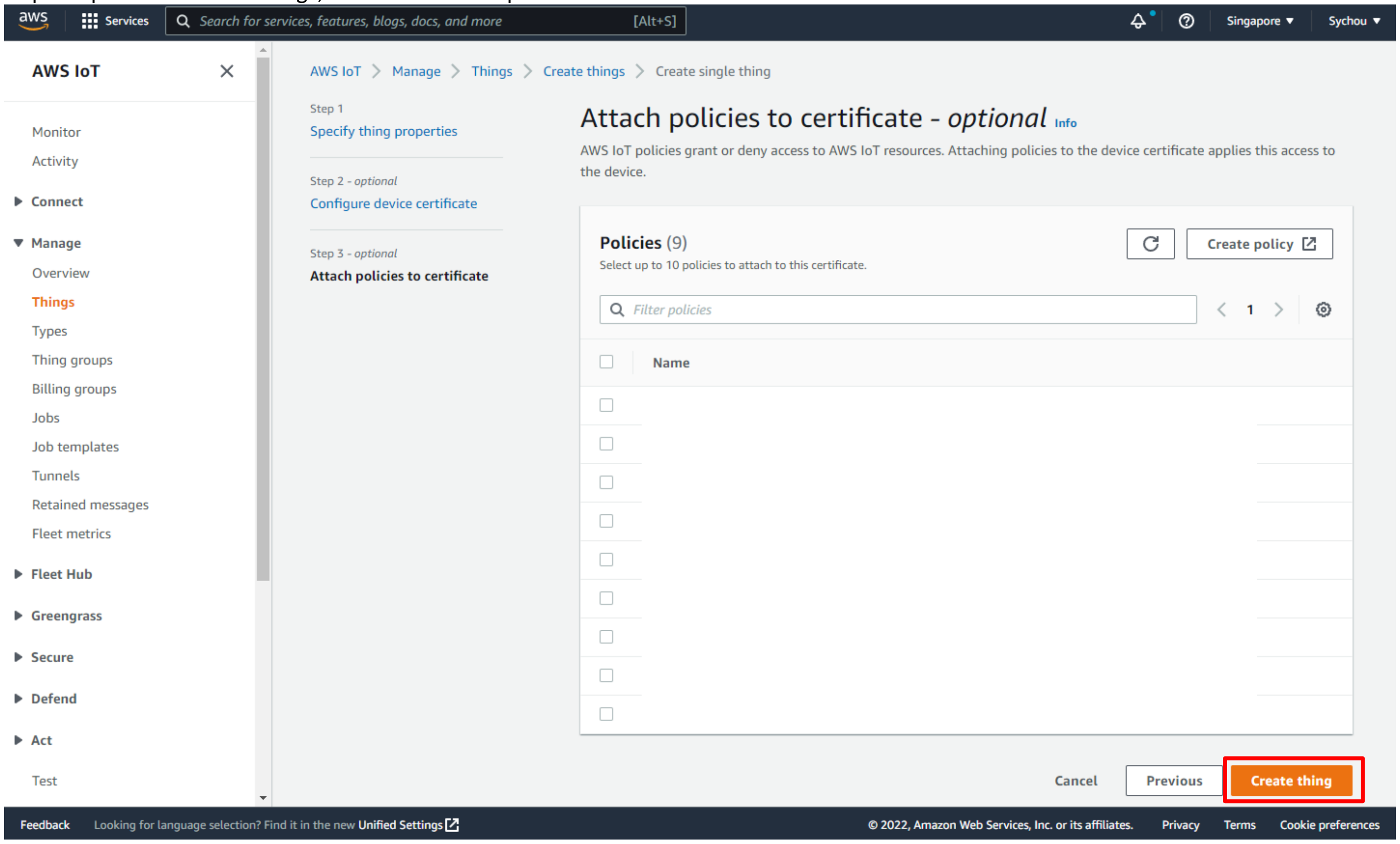
Device certificate

- [✓ 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.

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.

Skip this part and “Create thing”, we will attach the policies to certificate later.



Attach policies to certificate - *optional* Info

AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

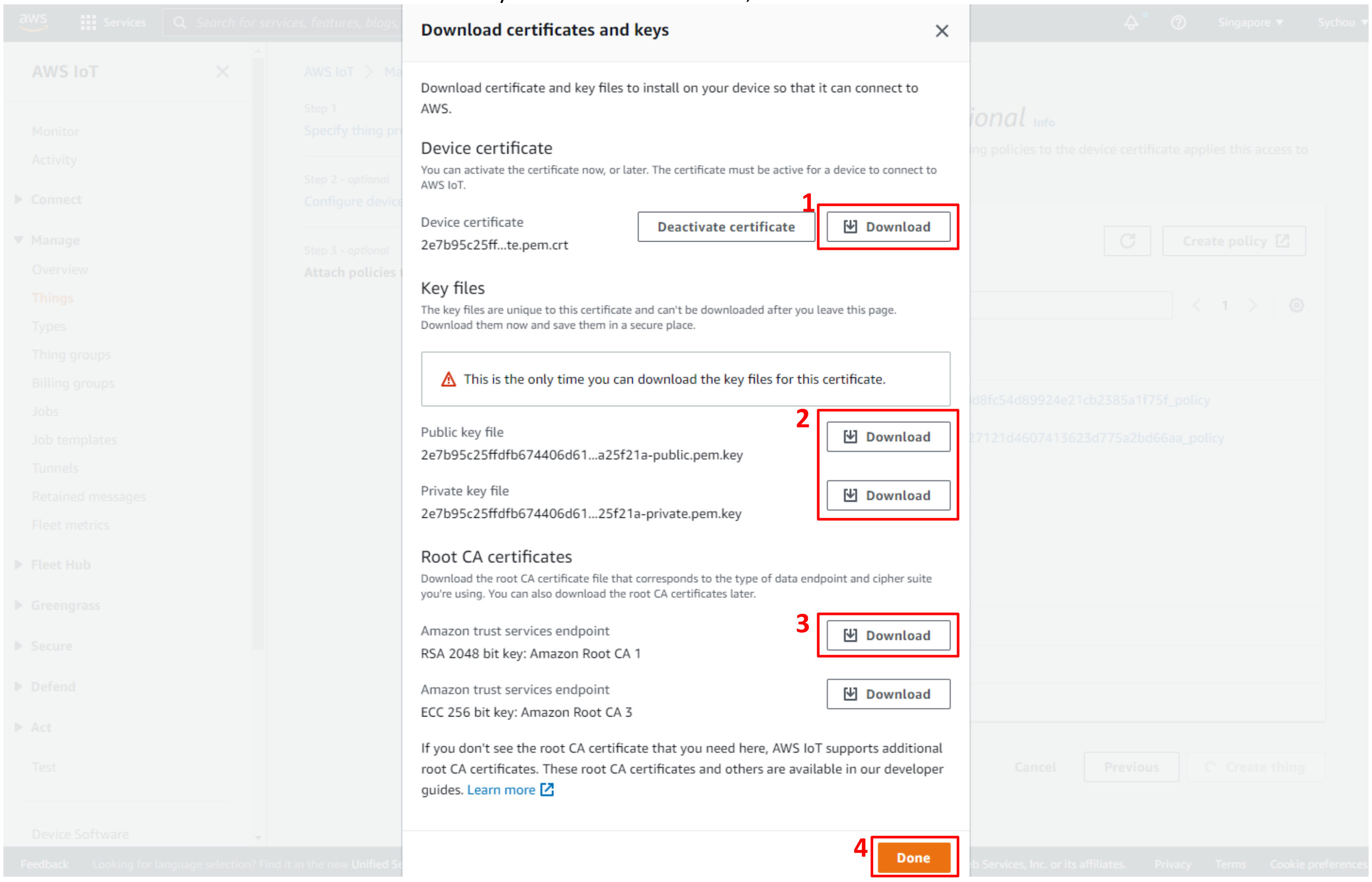
Policies (9)

Select up to 10 policies to attach to this certificate.

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates.

Download the certificate, public key, and private key for the device by clicking Download. Next, download the root CA for AWS IoT by clicking to the Download link. Once all the certificate and keys have been downloaded, click Done



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
2e7b95c25ff...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file
2e7b95c25ffdfb674406d61...a25f21a-public.pem.key

Private key file
2e7b95c25ffdfb674406d61...25f21a-private.pem.key

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

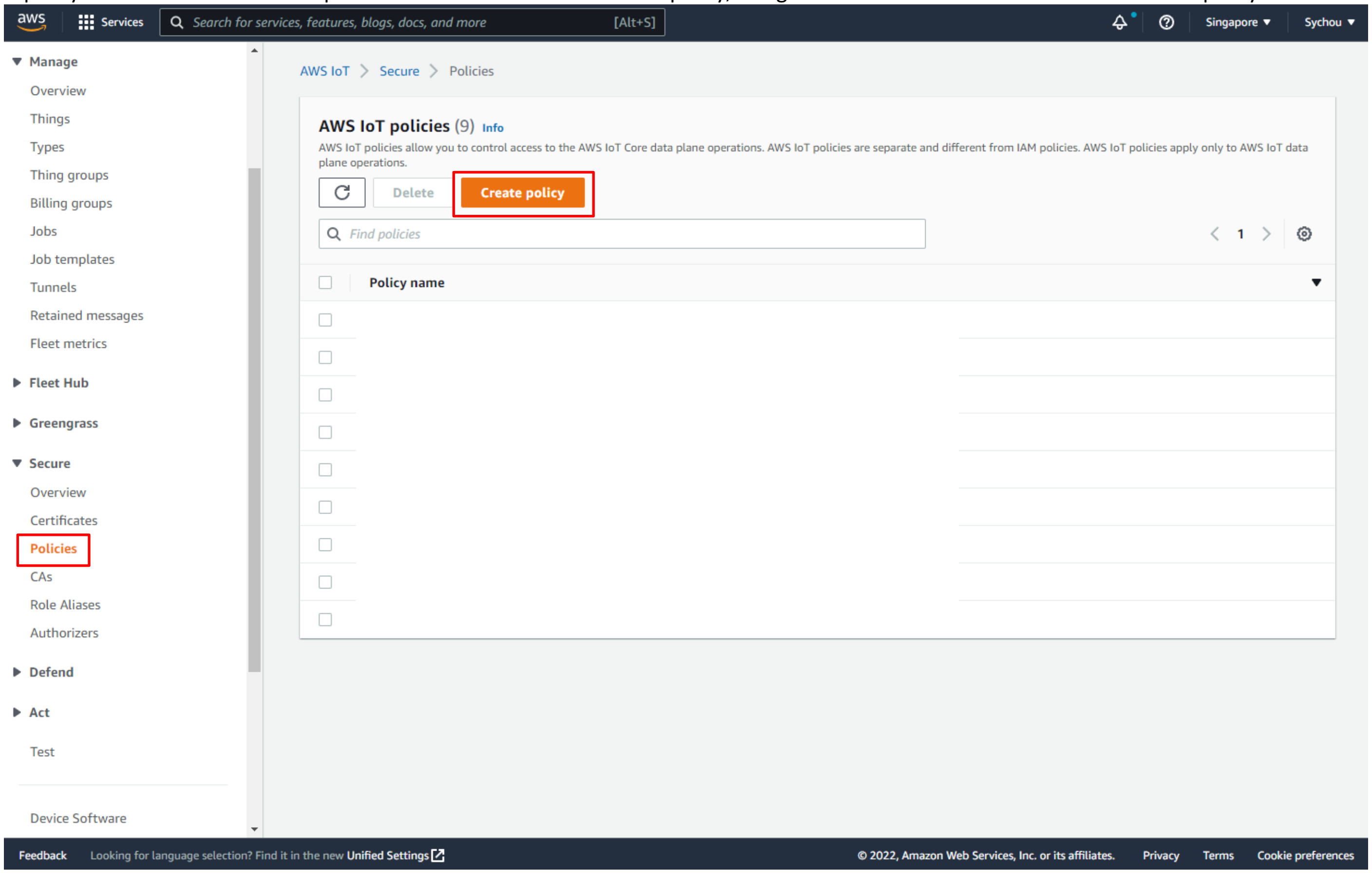
Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more

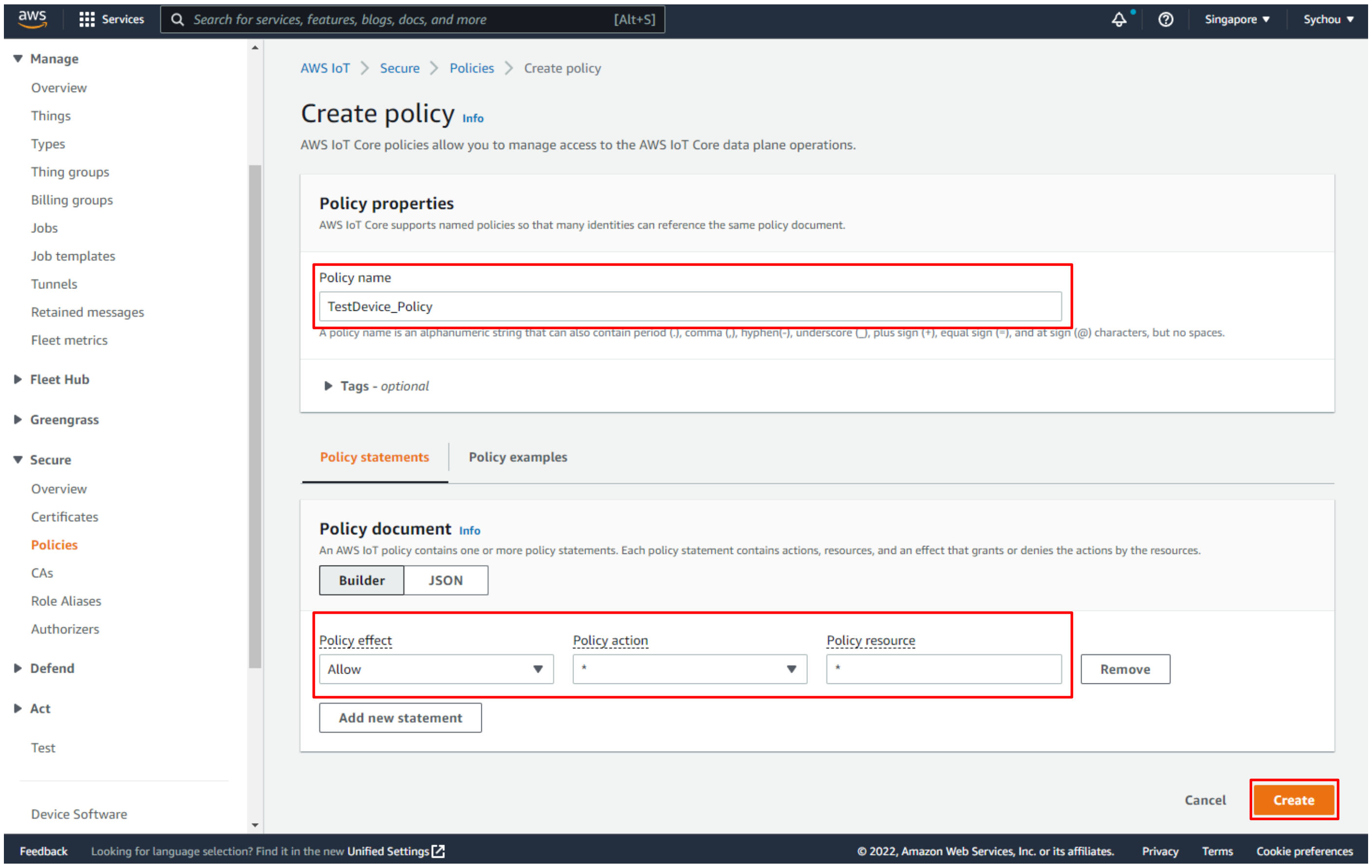
2.3 Create a policy

A policy defines a device's access permissions to IoT Core. To create a policy, navigate to Secure -> Policies. Then click "Create policy"



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.

Note: this policy grants unrestricted access for all iot operations, and is to be used only in a development environment. For non-dev environments, all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions such as publishing messages or subscribing to topics with specific scope and context. The specific permission policies can vary for your use cases. Identify the permission policies that best meet your business and security requirements.
For sample policies, refer to https://docs.aws.amazon.com/iot/latest/developerguide/example-iot-policies.html.
Also refer to https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html



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: TestDevice_Policy

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.

Policy document Info

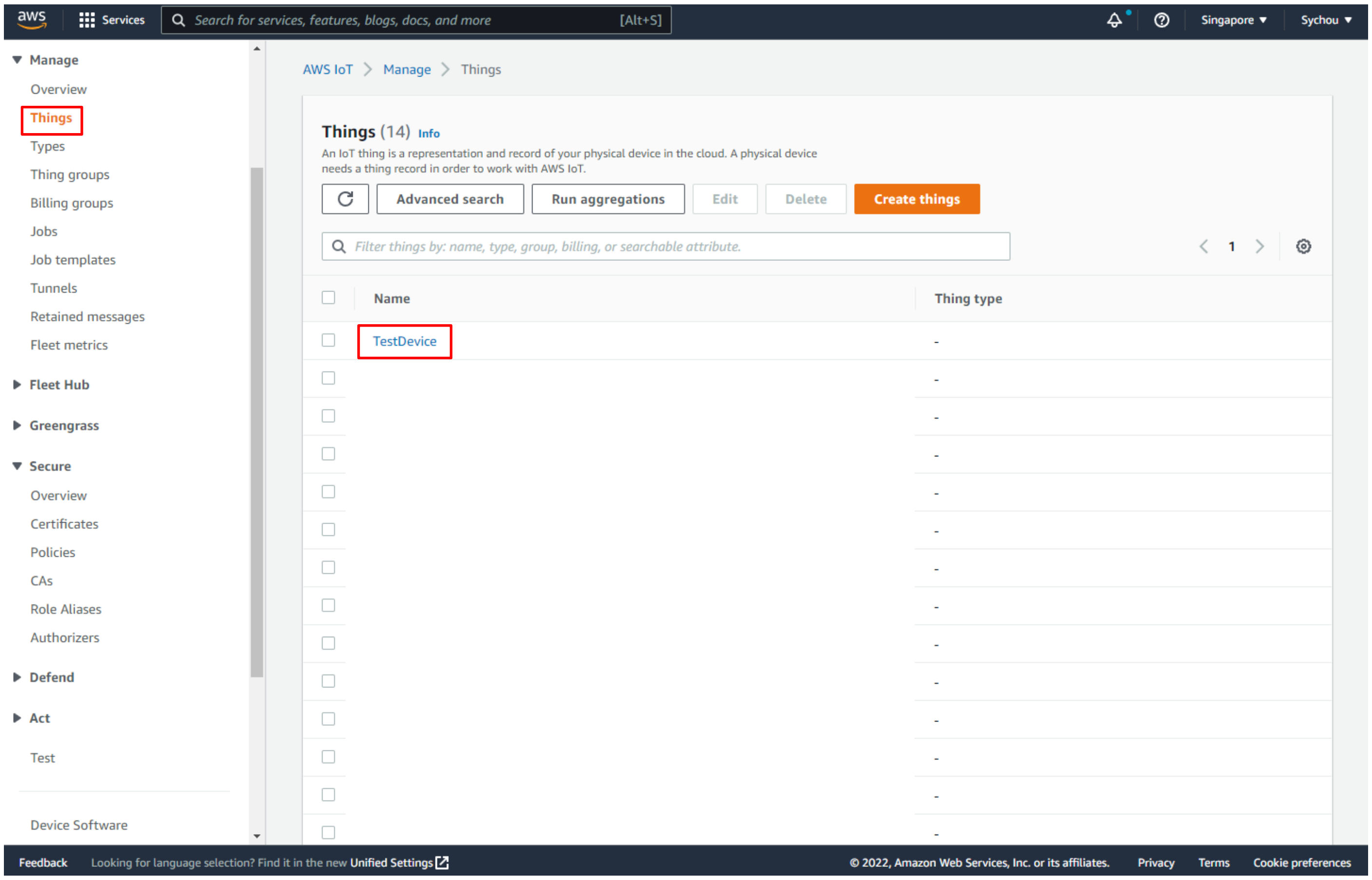
An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect: Allow
Policy action: *
Policy resource: *

Remove
Add new statement
Cancel
Create
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

2.4 Attach Policy

The last step to configuring the device is attaching a policy. To attach a policy to new device, navigate to Manage -> Things. Then click on the device which was created.

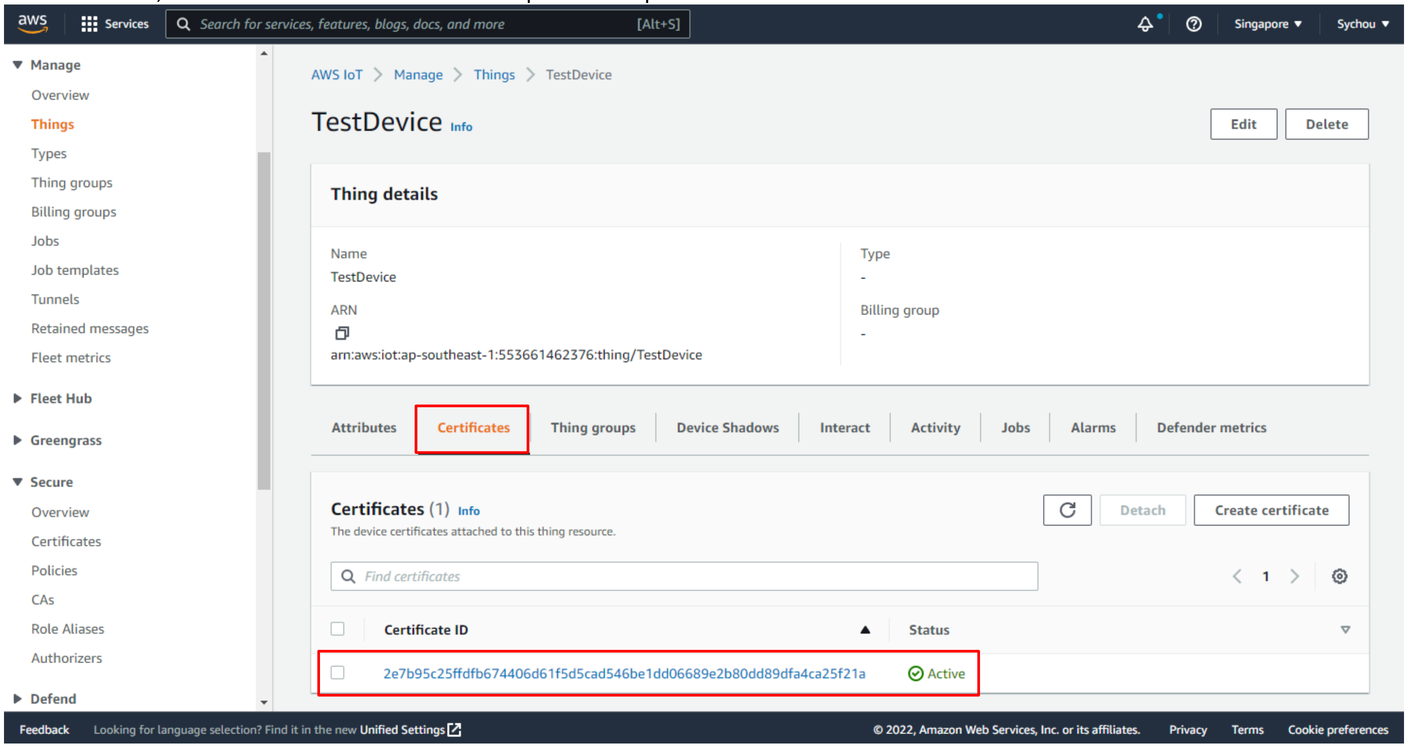


Things (14) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Name	Thing type
TestDevice	-
	-
	-
	-
	-
	-
	-
	-
	-
	-
	-
	-
	-
	-

Click Certificate, then choose the certificate create in previous step.



TestDevice Info

Thing details

Name
TestDevice

ARN
arn:aws:iot:ap-southeast-1:553661462376:thing/TestDevice

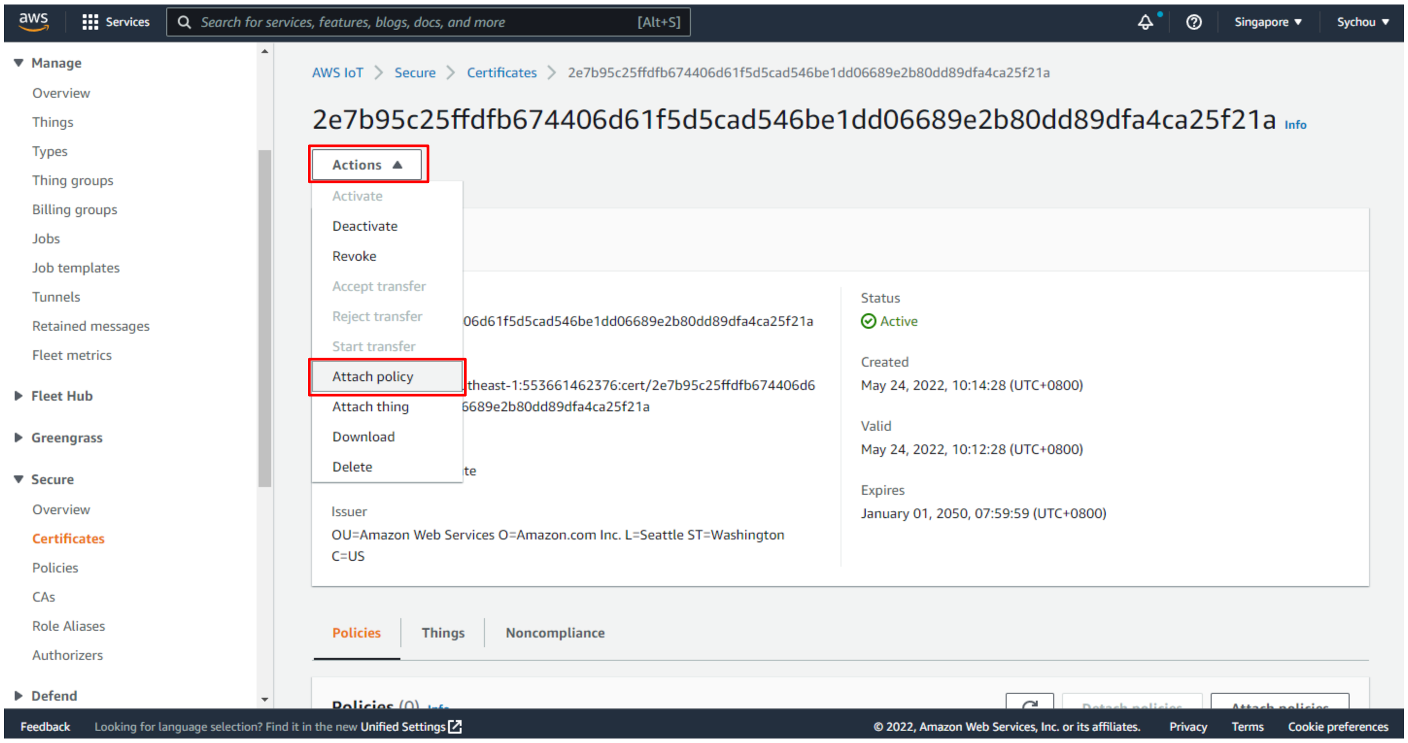
Type
-

Billing group
-

Certificates (1) Info

The device certificates attached to this thing resource.

Certificate ID	Status
2e7b95c25ffdfb674406d61f5d5cad546be1dd06689e2b80dd89dfa4ca25f21a	Active


AWS IoT > Secure > Certificates > 2e7b95c25ffdfb674406d61f5d5cad546be1dd06689e2b80dd89dfa4ca25f21a

2e7b95c25ffdfb674406d61f5d5cad546be1dd06689e2b80dd89dfa4ca25f21a

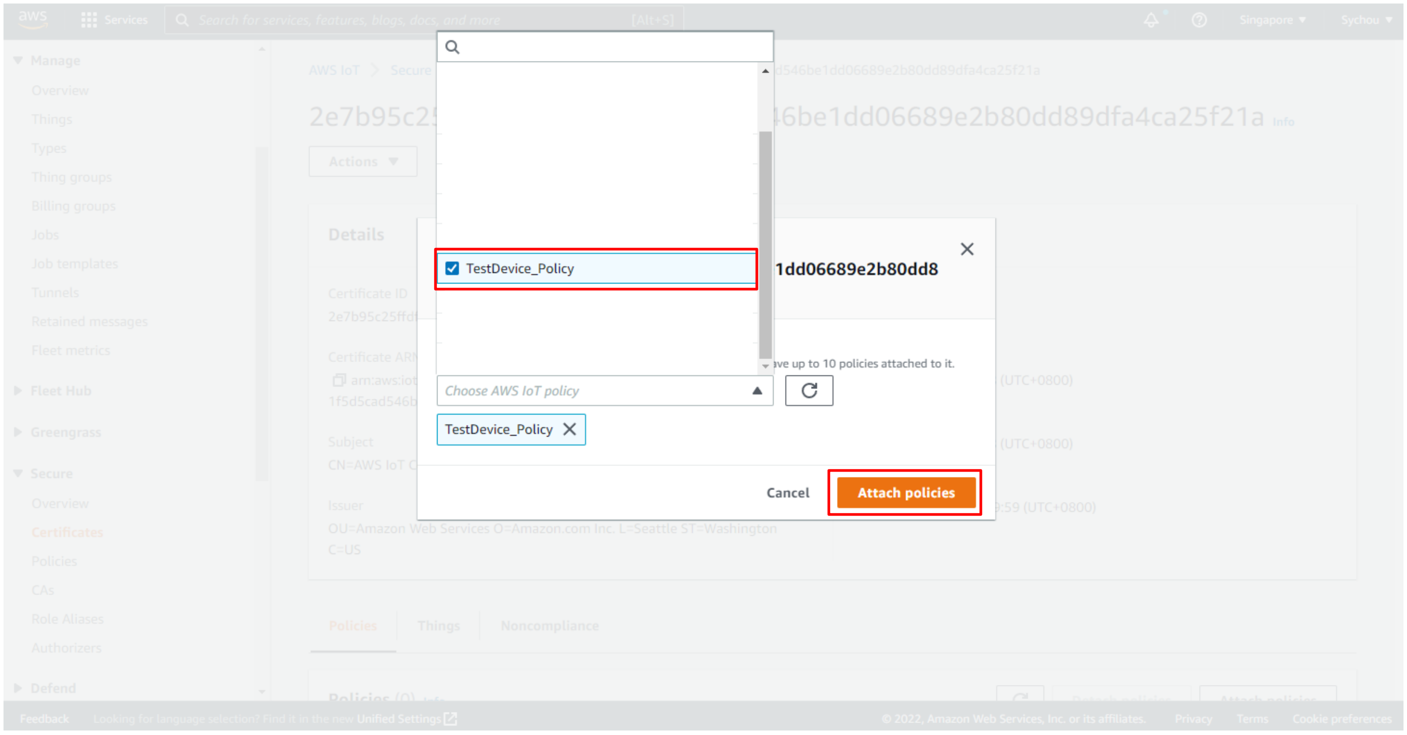
Status: Active

Created: May 24, 2022, 10:14:28 (UTC+0800)

Valid: May 24, 2022, 10:12:28 (UTC+0800)

Expires: January 01, 2050, 07:59:59 (UTC+0800)

Issuer: OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US



3 Configure AmebaPro2 Amazon FreeRTOS

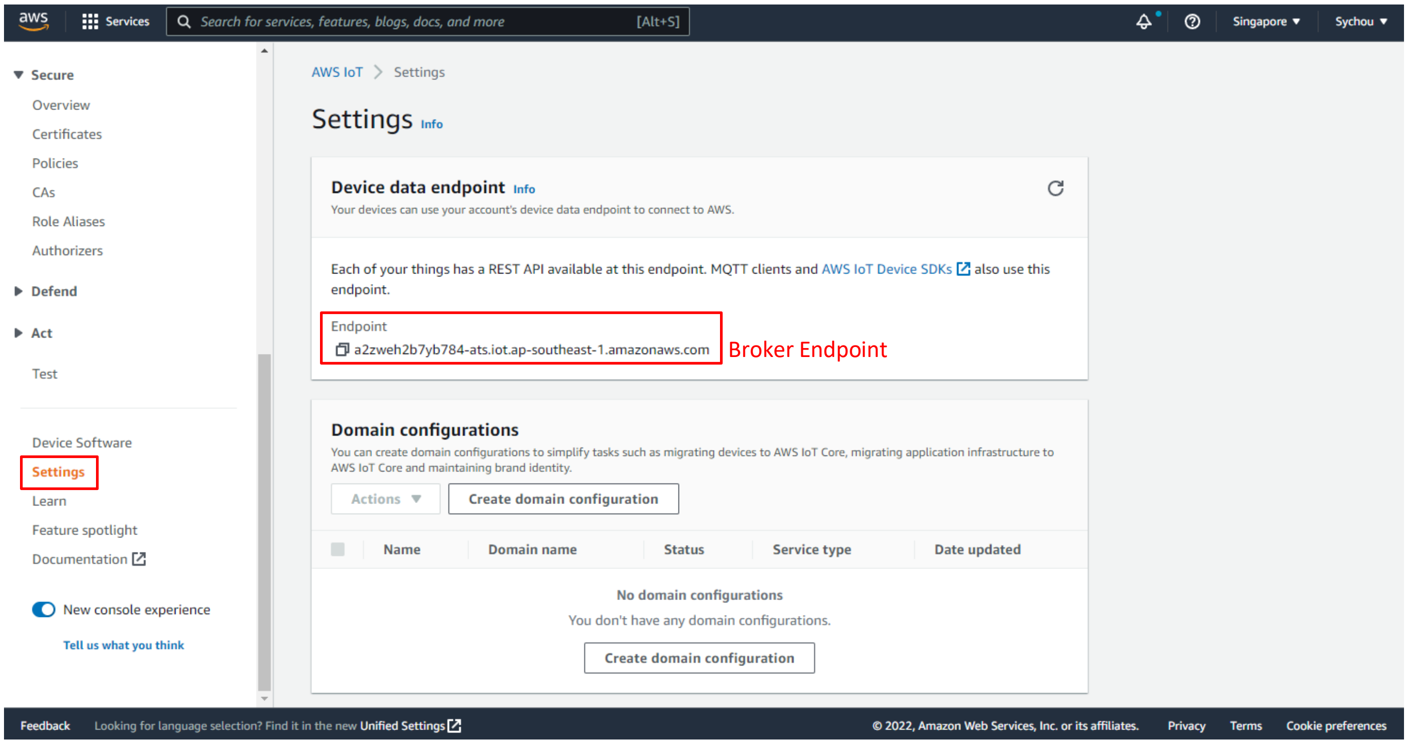
3.1 Download FreeRTOS-LTS Library Source Code from Github

Open source link: https://github.com/ambiot/amazon-freertos/tree/amebaPro2-9.x-202107.00-LTS
branch: **amebaPro2-9.x-202107.00-LTS**

Go to "AmebaPro2_SDK/project/realtek_amebapro2_v0_example/src":

```
$ cd project/realtek_amebapro2_v0_example/src
$ git clone --recurse-submodules -b amebaPro2-9.x-202107.00-LTS https://github.com/ambiot/amazon-freertos.git aws_iot_freertos_lts
```

3.2 Get Broker Endpoint by AWS IoT Core



Device data endpoint Info

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint
a2zweh2b7yb784-ats.iot.ap-southeast-1.amazonaws.com

Broker Endpoint

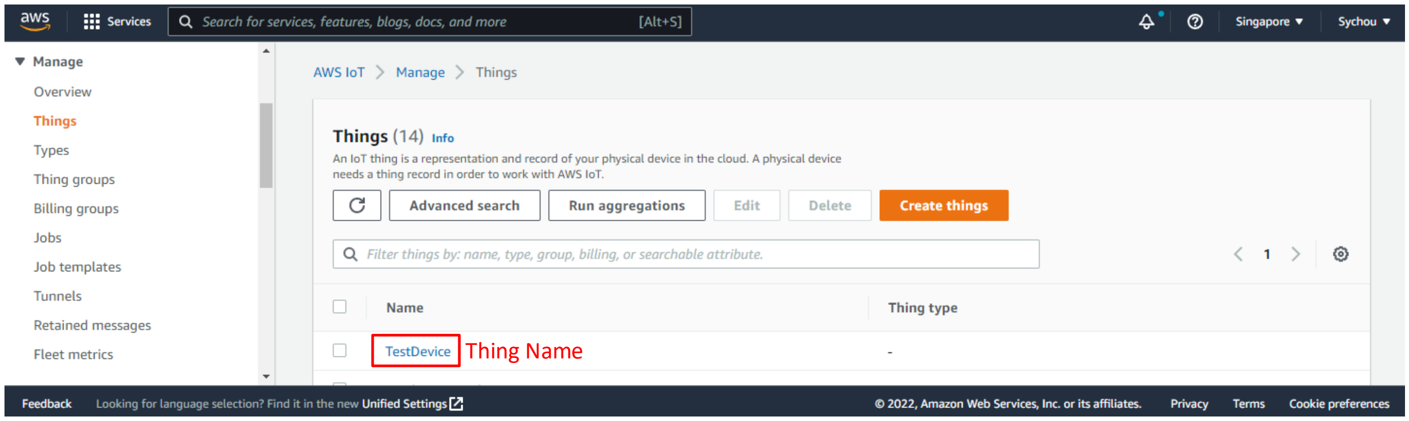
Domain configurations

You can create domain configurations to simplify tasks such as migrating devices to AWS IoT Core, migrating application infrastructure to AWS IoT Core and maintaining brand identity.

Name	Domain name	Status	Service type	Date updated

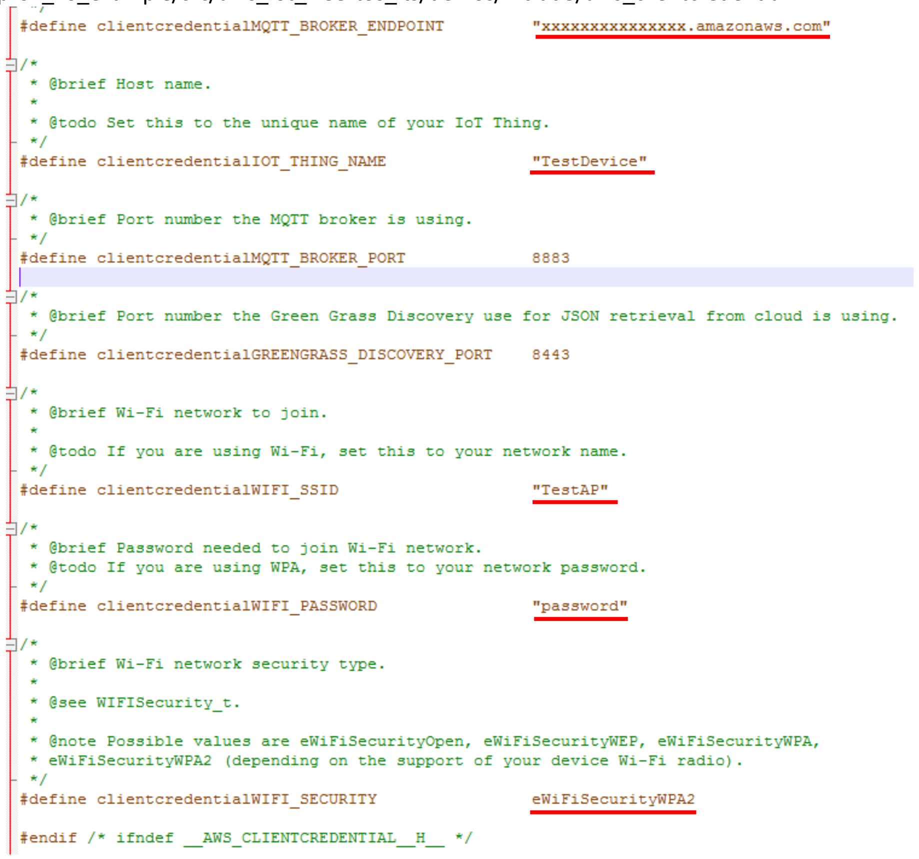
No domain configurations
You don't have any domain configurations.

3.3 Get Thing Name


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Singapore
Sychou
Manage
Overview
Things
Types
Thing groups
Billing groups
Jobs
Job templates
Tunnels
Retained messages
Fleet metrics
AWS IoT > Manage > Things
Things (14) Info
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.
Advanced search
Run aggregations
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.
Name
Thing type
TestDevice
Thing Name
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

3.4 Setup IoT Core Information with AmebaPro2 Amazon FreeRTOS

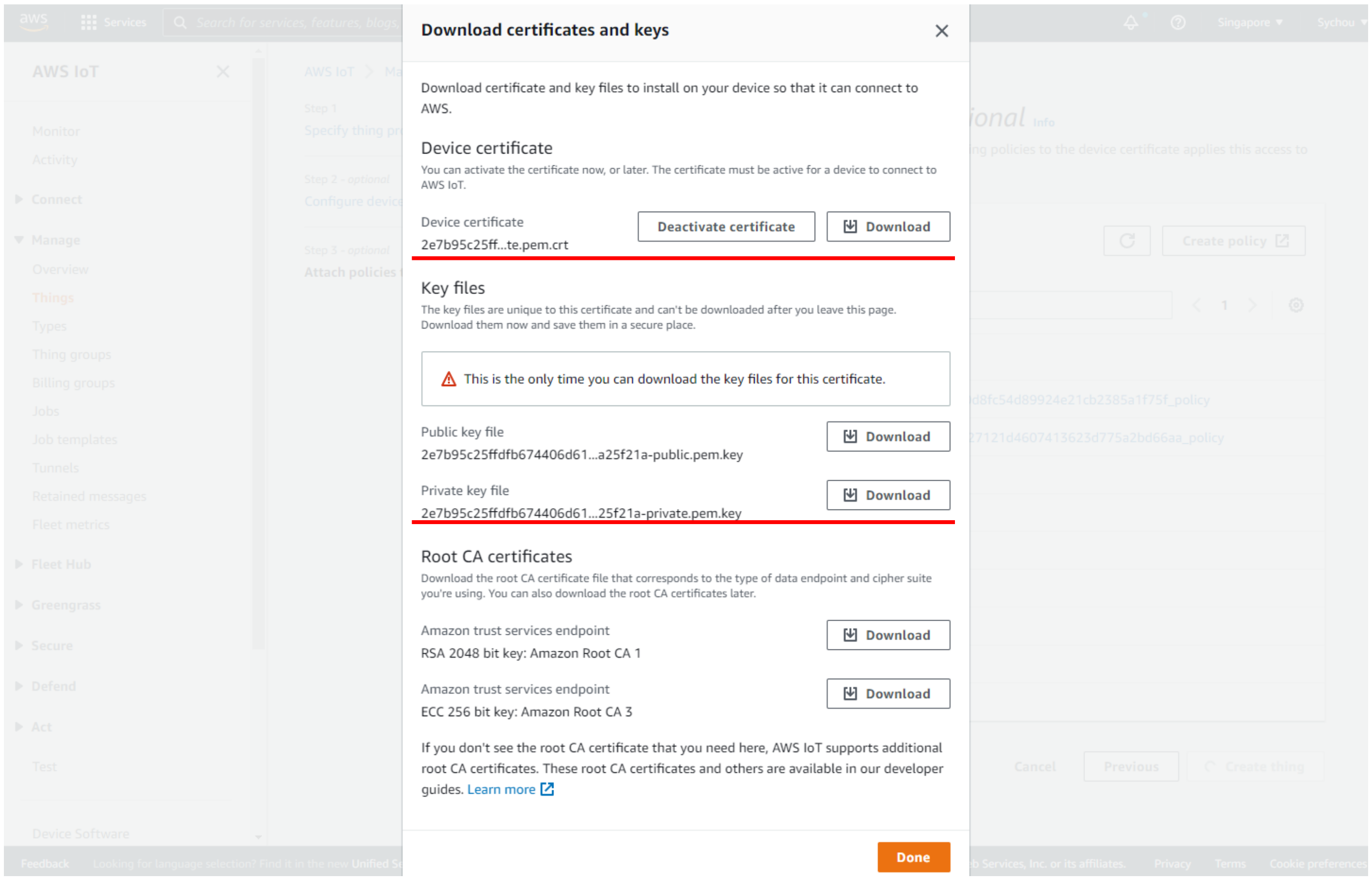
Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential.h"



```
#define clientcredentialMQTT_BROKER_ENDPOINT        "xxxxxxxxxxxxxxx.amazonaws.com"

/*
 * @brief Host name.
 *
 * @todo Set this to the unique name of your IoT Thing.
 */
#define clientcredentialIOT_THING_NAME              "TestDevice"

/*
 * @brief Port number the MQTT broker is using.
 */
#define clientcredentialMQTT_BROKER_PORT            8883

/*
 * @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
 */
#define clientcredentialGREENGRASS_DISCOVERY_PORT   8443

/*
 * @brief Wi-Fi network to join.
 *
 * @todo If you are using Wi-Fi, set this to your network name.
 */
#define clientcredentialWIFI_SSID                   "TestAP"

/*
 * @brief Password needed to join Wi-Fi network.
 * @todo If you are using WPA, set this to your network password.
 */
#define clientcredentialWIFI_PASSWORD               "password"

/*
 * @brief Wi-Fi network security type.
 *
 * @see WIFISecurity_t.
 *
 * @note Possible values are eWiFiSecurityOpen, eWiFiSecurityWEP, eWiFiSecurityWPA,
 * eWiFiSecurityWPA2 (depending on the support of your device Wi-Fi radio).
 */
#define clientcredentialWIFI_SECURITY               eWiFiSecurityWPA2

#endif /* ifndef __AWS_CLIENTCREDENTIAL__H__ */
```

3.4.1 Setup Thing's Private Key and Certificate

Fill keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-certifiacte.pem and xxxxxxxx-private.pem.key.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
2e7b95c25ff...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file
2e7b95c25ffdfb674406d61...a25f21a-public.pem.key

Private key file
2e7b95c25ffdfb674406d61...25f21a-private.pem.key

Root CA certificates

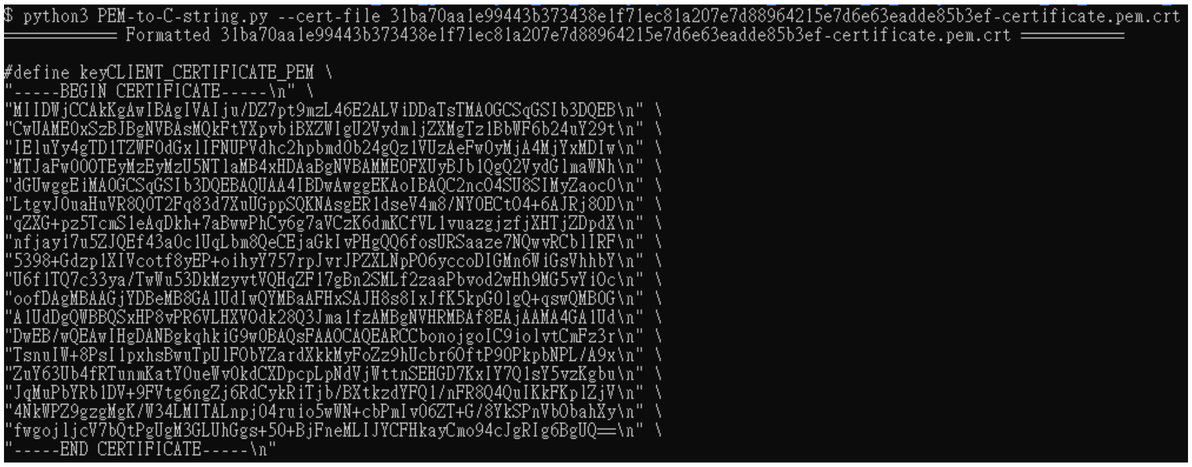
Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3

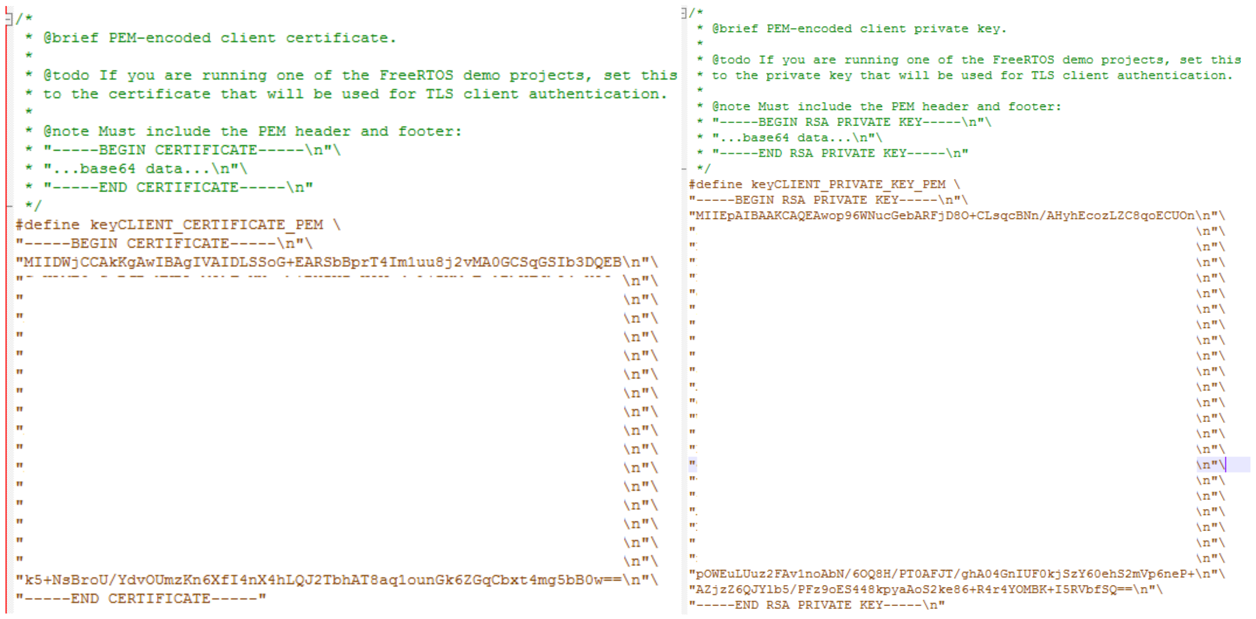
If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more

It can be done by the script PEM-to-C-string.py provided by AWS. It can be downloaded from https://github.com/aws/amazon-freertos/tree/main/tools/certificate_configuration.



```
$ python3 PEM-to-C-string.py --cert-file 31ba70aa1e99443b373438e1f71ec81a207e7d88964215e7d6e63eadde85b3ef-certificate.pem.crt
============ Formatted 31ba70aa1e99443b373438e1f71ec81a207e7d88964215e7d6e63eadde85b3ef-certificate.pem.crt ============

#define keyCLIENT_CERTIFICATE_PEM \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDWjCCAkKgAwIBAgIVAIju/DZ7pt9mzL46E2ALViDDaTsTMA0GCSqGSIb3DQEB\n" \
"CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b24uY29t\n" \
"IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0yMjA4MjYxMDIw\n" \
"MTJaFw0OOTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJb1QgQ2VydGlmaWNh\n" \
"dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2nc04SU8SIMyZaoc0\n" \
"LtgvJ0uaHuVR8Q0T2Fq83d7XuUGppSQKNAsgER1dseV4m8/NY0ECt04+6AJRj8OD\n" \
"qZXG+pz5TcmS1eAqDkh+7aBwwPhCy6g7aVCzK6dmKCfVL1vuazgjzfjXHTjZDpdX\n" \
"nfjayi7u5ZJQEf43a0c1UqLbm8QeCEjaGkIvPHgQQ6fosURSaaze7NQwvRCblIRF\n" \
"5398+Gdzp1XIVcotf8yEP+oihyY757rpJvrJPZXLNpPO6yccoDIGMn6WiGsVhhbY\n" \
"U6f1TQ7c33ya/TwWu53DkMzyvtVQHqZF17gBn2SMLf2zaaPbvod2wHh9MG5vYi0c\n" \
"oofDAgMBAAGjYDBeMB8GA1UdIwQYMBaAFHxSAJH8s8IxJfK5kpG0lgQ+qswQMB0G\n" \
"A1UdDgQWBBQSxHP8vPR6VLHXV0dk28Q3Jma1fzAMBgNVHRMBAf8EAjAAMA4GA1Ud\n" \
"DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEARCCbonojgoIC9iolvtCmFz3r\n" \
"TsnuIW+8PsI1pxhsBwuTpUlFObYZardXkkMyFoZz9hUcbr60ftP90PkpbNPL/A9x\n" \
"ZuY63Ub4fRTunmKatY0ueWv0kdCXDpcpLpNdVjWttnSEHGD7KxIY7Q1sY5vzKgbu\n" \
"JqMuPbYRb1DV+9FVtg6ngZj6RdCykRiTjb/BXtkzdYFQ1/nFR8Q4QuIKkFKplZjV\n" \
"4NkWPZ9gzgMgK/W34LMITALnpj04ruio5wWN+cbPmIv06ZT+G/8YkSPnVbObahXy\n" \
"fwgojljcV7bQtPgUgM3GLUhGgs+50+BjFneMLIJYCFHkayCmo94cJgRIg6BgUQ==\n" \
"-----END CERTIFICATE-----\n"
```

Final aws_clientcredential_keys.h overview.

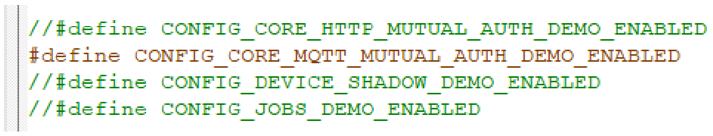


```
/*
 * @brief PEM-encoded client certificate.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the certificate that will be used for TLS client authentication.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"\
 * "...base64 data...\n"\
 * "-----END CERTIFICATE-----\n"
 */
#define keyCLIENT_CERTIFICATE_PEM \
"-----BEGIN CERTIFICATE-----\n"\
"MIIDWjCCAkKgAwIBAgIVAIDLSSoG+EARSbBprT4Im1uu8j2vMA0GCSqGSIb3DQEB\n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"k5+NsBroU/YdvOUmzKn6XfI4nX4hLQJ2TbhAT8aq1ounGk6ZGqCbxt4mg5bB0w==\n"\
"-----END CERTIFICATE-----"
```

```
/*
 * @brief PEM-encoded client private key.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the private key that will be used for TLS client authentication.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN RSA PRIVATE KEY-----\n"\
 * "...base64 data...\n"\
 * "-----END RSA PRIVATE KEY-----\n"
 */
#define keyCLIENT_PRIVATE_KEY_PEM \
"-----BEGIN RSA PRIVATE KEY-----\n"\
"MIIEpAIBAAKCAQEAwop96WNucGebARFjD8O+CLsqcBNn/AHyhEcozLZC8qoECUOn\n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"pOWEuLUuz2FAv1noAbN/6OQ8H/PT0AFJT/ghA04GnIUF0kjSzY60ehS2mVp6neP+\n"\
"AZjzZ6QJYlb5/PFz9oES448kpyaAoS2ke86+R4r4YOMBK+I5RVbfSQ==\n"\
"-----END RSA PRIVATE KEY-----\n"
```

3.4.2 Enable FreeRTOS demo on AmebaPro2

For example, if you would like to run MQTT mutual authentication demo, please find aws_demo_config.h in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/vendors/realtek/boards/amebaPro2/aws_demos/config_files/" and enable **CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED**



```
//#define CONFIG_CORE_HTTP_MUTUAL_AUTH_DEMO_ENABLED
#define CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
//#define CONFIG_DEVICE_SHADOW_DEMO_ENABLED
//#define CONFIG_JOBS_DEMO_ENABLED
```

Now you can start to compile AmebaPro2 Amazon FreeRTOS project !

4 Compile AmebaPro2 Amazon FreeRTOS

4.1 Compile Program with GCC Toolchain

Run following commands to build the image with option `-DEXAMPLE=amazon_freertos`

```
$ cd project/realtek_amebapro2_v0_example/GCC-RELEASE
$ mkdir build
$ cd build
$ cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DEXAMPLE=amazon_freertos
$ cmake --build . --target flash -j4
```

After successfully build, there should be an image file **flash_ntz.bin** located in “build/” directory.

5 Image Download Tool

Use image tool to download the image to AmebaPro2. The tool can be find in **tools/Pro2_PG_tool_linux_v1.3.x.**

5.1 Environment Setup

The hardware setup is shown in Fig 5-1.

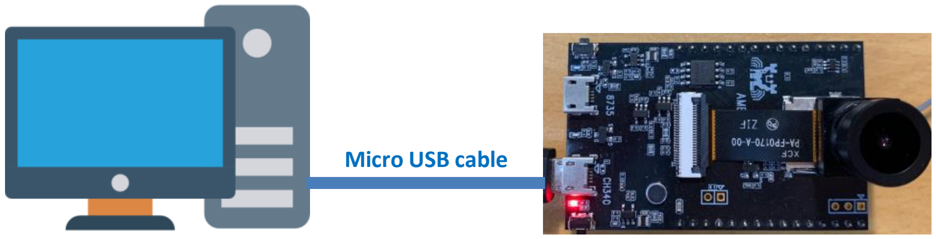


Fig 5-1 Hardware setup

5.2 Enter the Download Mode

Press these two buttons in following figure simultaneously to enter download mode.

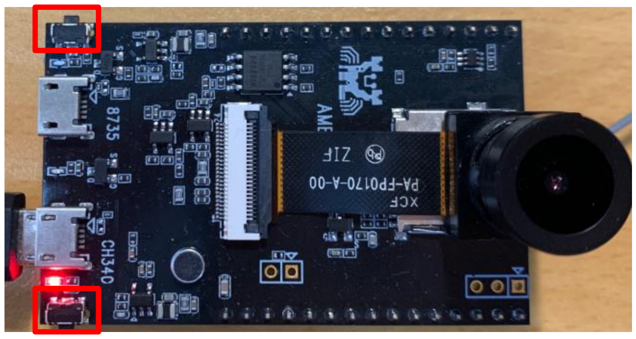


Fig 5-2 Enter download mode

5.3 Download Firmware Image

Copy the image **flash_ntz.nn.bin** to image tool folder **tools/Pro2_PG_tool_linux_v1.3.x**.
Then, Use Pro2_PG_tool_linux_v1.3.x command line tool to download image.
Nor flash

```
$ ./uartfwburn.linux -p /dev/ttyUSB? -f flash_ntz.nn.bin -b 2000000 -U
```

Nand flash

```
$ ./uartfwburn.linux -p /dev/ttyUSB? -f flash_ntz.nn.bin -b 2000000 -n pro2
```

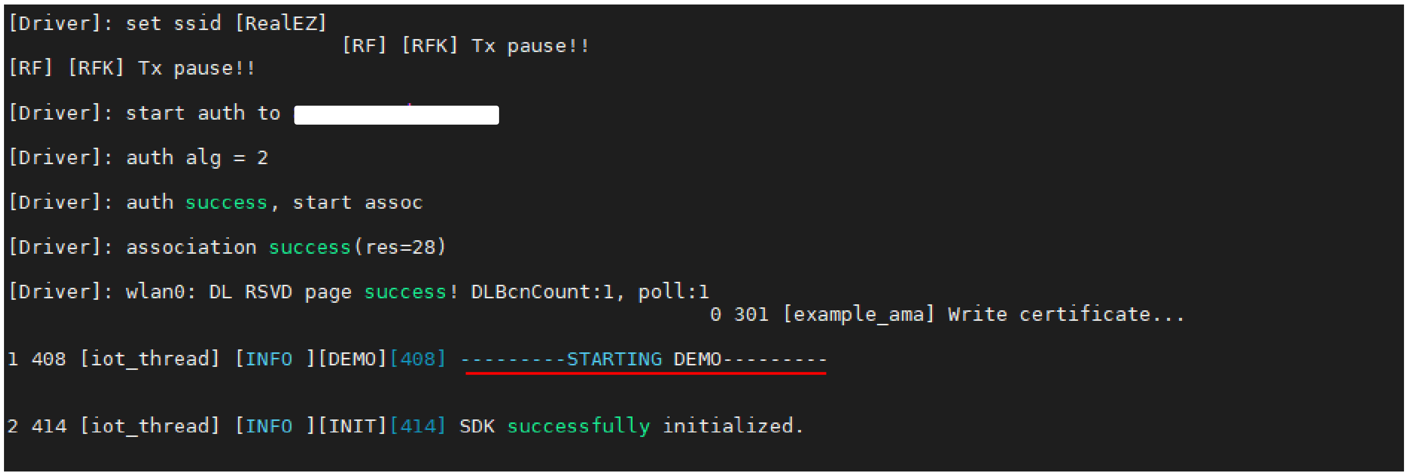
After firmware image downloaded, press the reset button (beside the LED) to reboot the device and open terminal console to check the log.

Note: If using windows, replace *uartfwburn.linux* with *uartfwburn.exe* and replace */dev/ttyUSB?* with *COM?*

6 MQTT Demo

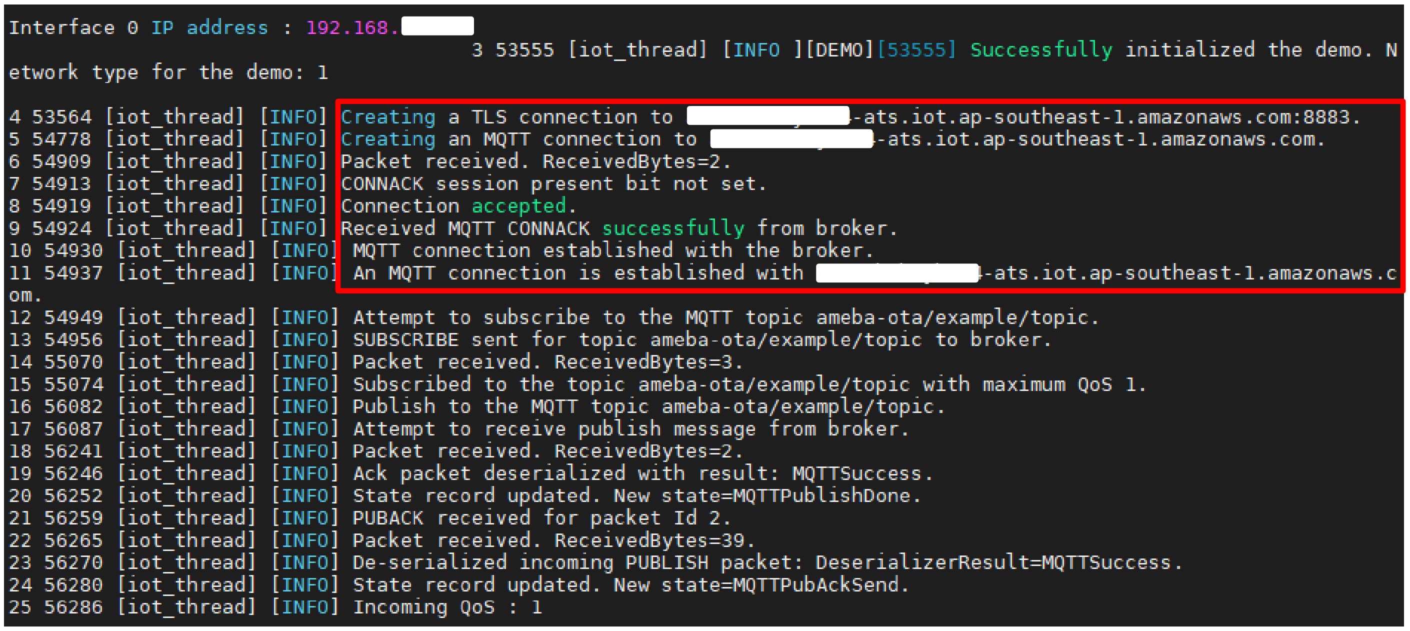
6.1 Run MQTT Demo

Default setting of SDK are enable MQTT demo. Once the AmebaPro2 EVB has rebooted, the application will automatically start run MQTT demo and communicate to IoT Core.



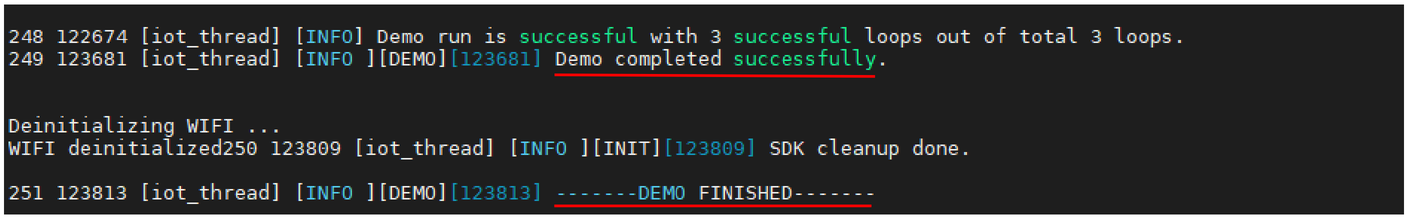
```
[Driver]: set ssid [RealEZ]
                            [RF] [RFK] Tx pause!!
[RF] [RFK] Tx pause!!

[Driver]: start auth to [redacted]

[Driver]: auth alg = 2

[Driver]: auth success, start assoc

[Driver]: association success(res=28)

[Driver]: wlan0: DL RSVD page success! DLBcnCount:1, poll:1
                                                           0 301 [example_ama] Write certificate...

1 408 [iot_thread] [INFO ][DEMO][408] ---------STARTING DEMO---------


2 414 [iot_thread] [INFO ][INIT][414] SDK successfully initialized.
```

…



```
Interface 0 IP address : 192.168.[redacted]
                                               3 53555 [iot_thread] [INFO ][DEMO][53555] Successfully initialized the demo. N
etwork type for the demo: 1

4 53564 [iot_thread] [INFO] Creating a TLS connection to [redacted]-ats.iot.ap-southeast-1.amazonaws.com:8883.
5 54778 [iot_thread] [INFO] Creating an MQTT connection to [redacted]-ats.iot.ap-southeast-1.amazonaws.com.
6 54909 [iot_thread] [INFO] Packet received. ReceivedBytes=2.
7 54913 [iot_thread] [INFO] CONNACK session present bit not set.
8 54919 [iot_thread] [INFO] Connection accepted.
9 54924 [iot_thread] [INFO] Received MQTT CONNACK successfully from broker.
10 54930 [iot_thread] [INFO] MQTT connection established with the broker.
11 54937 [iot_thread] [INFO] An MQTT connection is established with [redacted]-ats.iot.ap-southeast-1.amazonaws.c
om.
12 54949 [iot_thread] [INFO] Attempt to subscribe to the MQTT topic ameba-ota/example/topic.
13 54956 [iot_thread] [INFO] SUBSCRIBE sent for topic ameba-ota/example/topic to broker.
14 55070 [iot_thread] [INFO] Packet received. ReceivedBytes=3.
15 55074 [iot_thread] [INFO] Subscribed to the topic ameba-ota/example/topic with maximum QoS 1.
16 56082 [iot_thread] [INFO] Publish to the MQTT topic ameba-ota/example/topic.
17 56087 [iot_thread] [INFO] Attempt to receive publish message from broker.
18 56241 [iot_thread] [INFO] Packet received. ReceivedBytes=2.
19 56246 [iot_thread] [INFO] Ack packet deserialized with result: MQTTSuccess.
20 56252 [iot_thread] [INFO] State record updated. New state=MQTTPublishDone.
21 56259 [iot_thread] [INFO] PUBACK received for packet Id 2.
22 56265 [iot_thread] [INFO] Packet received. ReceivedBytes=39.
23 56270 [iot_thread] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
24 56280 [iot_thread] [INFO] State record updated. New state=MQTTPubAckSend.
25 56286 [iot_thread] [INFO] Incoming QoS : 1
```

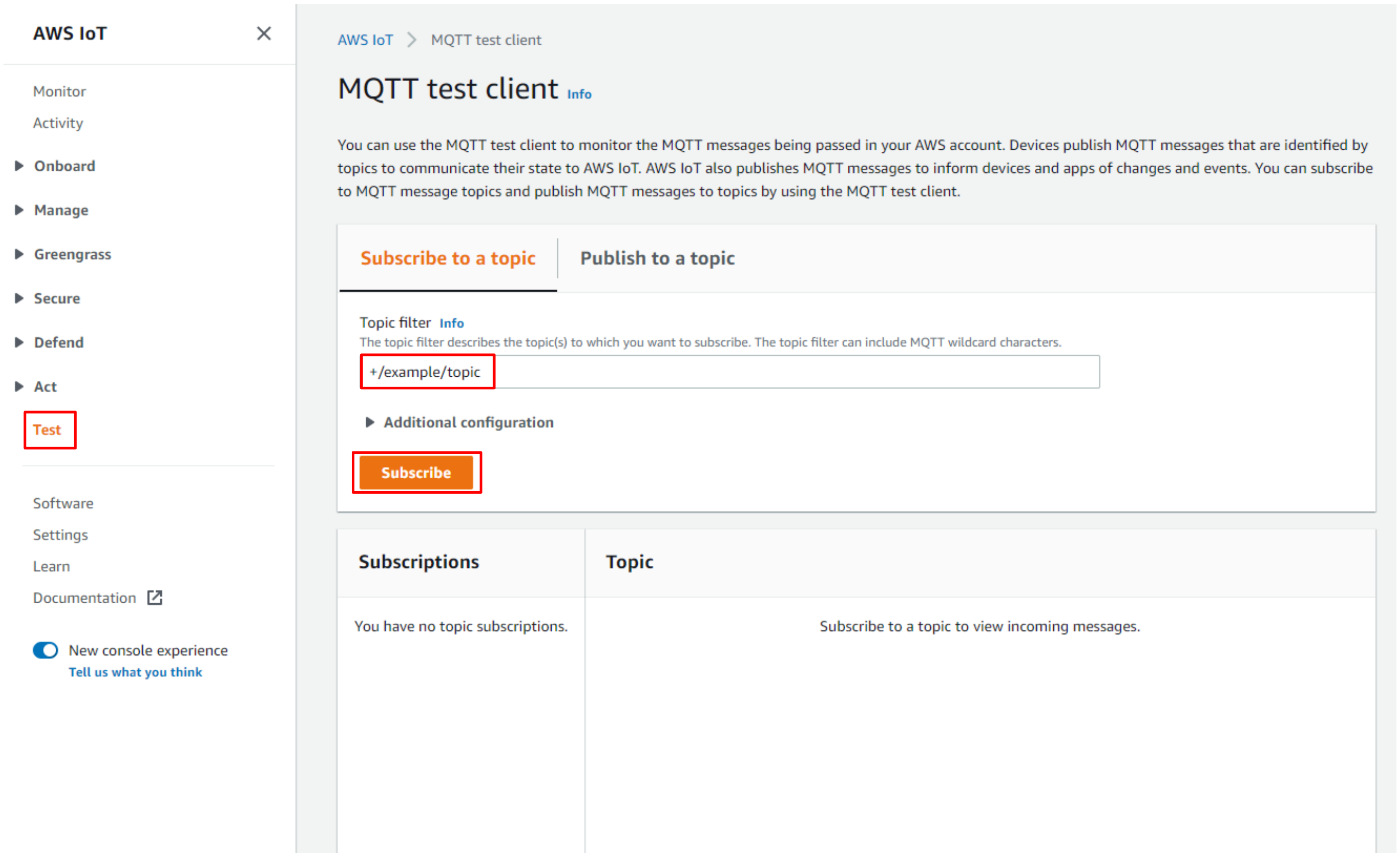
…



```
248 122674 [iot_thread] [INFO] Demo run is successful with 3 successful loops out of total 3 loops.
249 123681 [iot_thread] [INFO ][DEMO][123681] Demo completed successfully.


Deinitializing WIFI ...
WIFI deinitialized250 123809 [iot_thread] [INFO ][INIT][123809] SDK cleanup done.

251 123813 [iot_thread] [INFO ][DEMO][123813] -------DEMO FINISHED-------
```

6.2 Monitoring MQTT Messages on the Cloud

To subscribe to the MQTT topic with the AWS IoT MQTT client
1. Sign in to the AWS IoT console.
2. In the navigation pane, choose Test to open the MQTT client.
3. In Subscription topic, enter “+/example/topic”, and then choose Subscribe to topic.

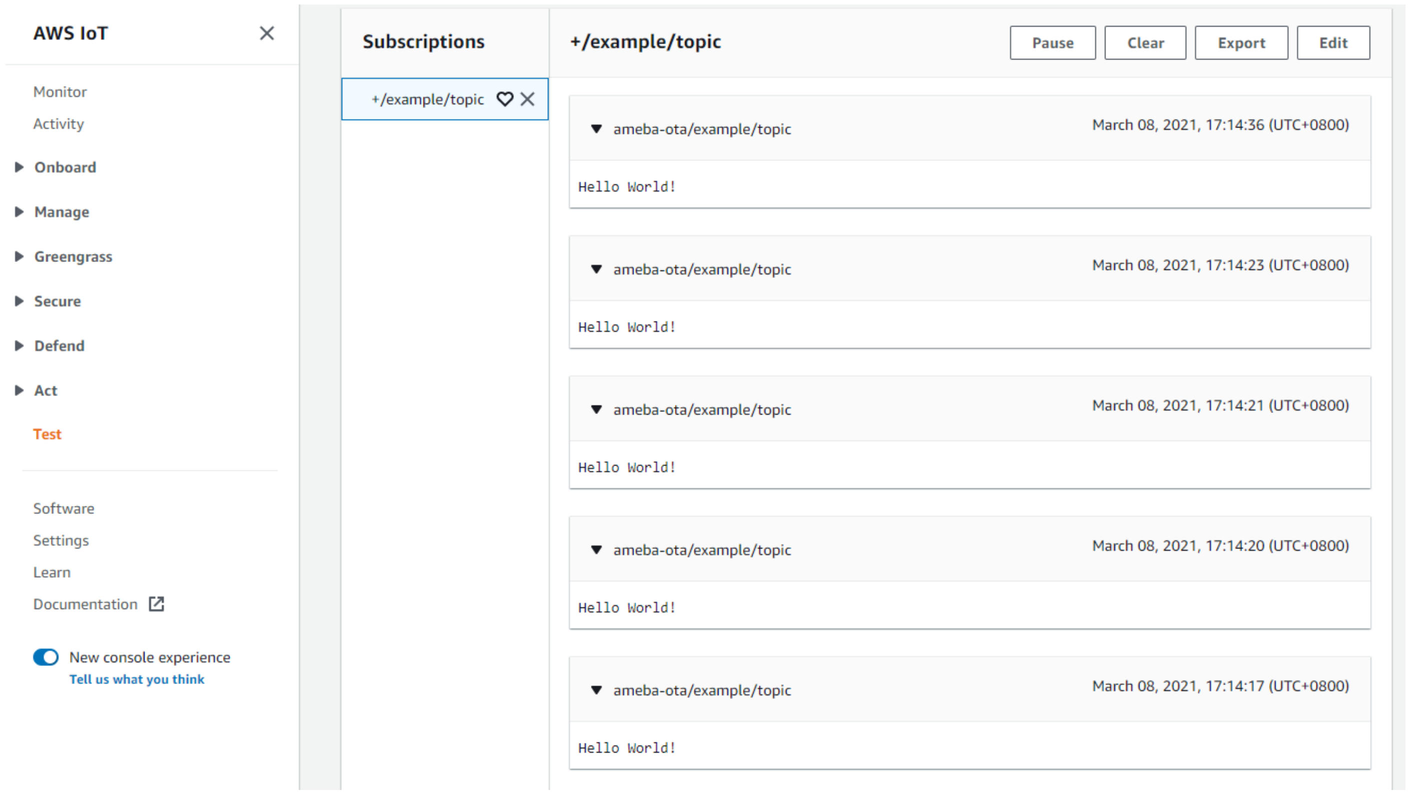


MQTT test client Info

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

Topic filter Info
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.
+/example/topic

You have no topic subscriptions.
Subscribe to a topic to view incoming messages.



ameba-ota/example/topic March 08, 2021, 17:14:36 (UTC+0800)

```
Hello World!
```

ameba-ota/example/topic March 08, 2021, 17:14:23 (UTC+0800)

```
Hello World!
```

ameba-ota/example/topic March 08, 2021, 17:14:21 (UTC+0800)

```
Hello World!
```

ameba-ota/example/topic March 08, 2021, 17:14:20 (UTC+0800)

```
Hello World!
```

ameba-ota/example/topic March 08, 2021, 17:14:17 (UTC+0800)

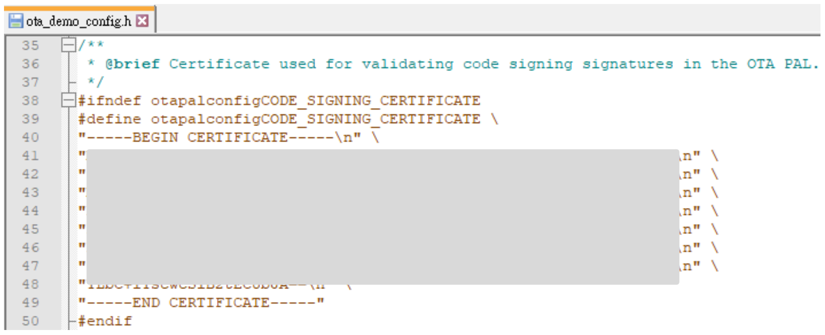
```
Hello World!
```

7 OTA Demo

7.1 OTA Update Prerequisites

Please refer to the AWS official guide (https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html) and finish the following steps:

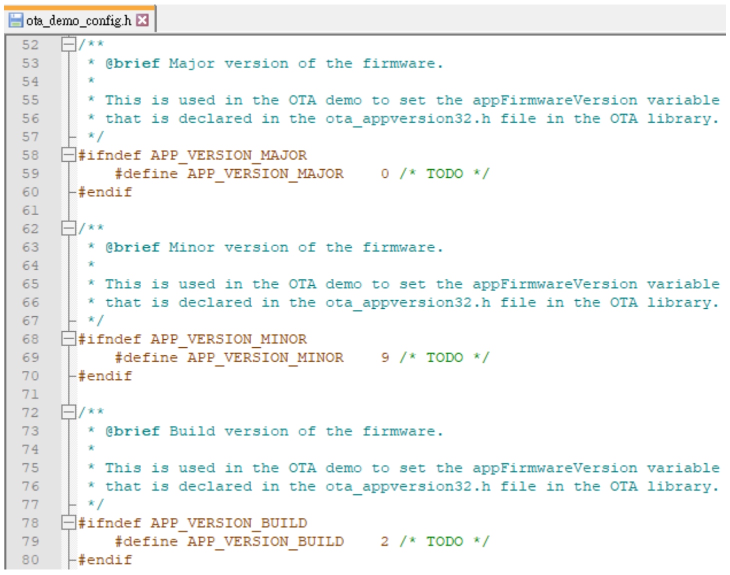
Step 1. Prerequisites for OTA updates using MQTT
Step 2. Create an Amazon S3 bucket to store your update
Step 3. Create an OTA Update service role
Step 4. Create an OTA user policy
Step 5. Create esdsasigner.key and ecdsasigner.crt by openSSL
you can create the key and certification by running:
$ sudo openssl ecparam -name prime256v1 -genkey -out ecdsa-sha256-signer.key.pem
$ sudo openssl req -new -x509 -days 3650 -key ecdsa-sha256-signer.key.pem -out ecdsa-sha256-signer.crt.pem
Step 6. Add certificate pem(ecdsa-sha256-signer.crt.pem) into *project\realtek_amebapro2_v0_example\src\aws_iot_freertos_lts\vendors\realtek\boards\amebaPro2\aws_demos\config_files\ota_demo_config.h*


ota_demo_config.h

```
/**
 * @brief Certificate used for validating code signing signatures in the OTA PAL.
 */
#ifndef otapalconfigCODE_SIGNING_CERTIFICATE
#define otapalconfigCODE_SIGNING_CERTIFICATE \
"-----BEGIN CERTIFICATE-----\n" \
[redacted]
"-----END CERTIFICATE-----"
#endif
```

7.2 Set the App Version to Image File

The app number in “ota_demo_config.h” decide the version of application code:


ota_demo_config.h

```
/**
 * @brief Major version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable
 * that is declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MAJOR
    #define APP_VERSION_MAJOR    0 /* TODO */
#endif

/**
 * @brief Minor version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable
 * that is declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MINOR
    #define APP_VERSION_MINOR    9 /* TODO */
#endif

/**
 * @brief Build version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable
 * that is declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    2 /* TODO */
#endif
```

Please note that the newer image file must have the bigger version number. So now, you need two image file to perform this demo.

- One image with older version should be downloaded to your AmebaPro2, and wait the OTA job coming.
- Another image with newer version will be uploaded to S3 bucket. Then, create a new job for OTA.

Your newer image for OTA - **ota.bin** will also be located in "build/" directory after compilation.

```
ota.bin                 2022/9/12
```

Note: newer version image file should be signed by a private key before uploading. Next section will introduce how to sign the image.

7.3 Custom-Signed Image and Signature

We use custom signing feature provided by amazon to manually sign the OTA binary to get the signatures used to verify the integrity of the firmware after download. The ota.bin is manually signed using the ECDSA P-256 key provided by user.

The custom signing process is executed by a python script – **python_custom_ecdsa_Pro2.py**, that provided in the folder "project\realtek_amebapro2_v0_example\src\aws_iot_freertos_lts\tools\amazon_ota_tools\python_custom_ecdsa_Pro2.py"

The python script requires the following pre-requisites to work

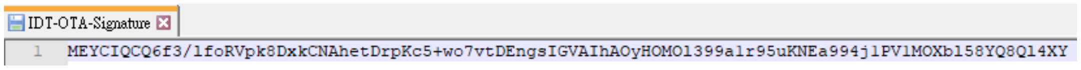
1. Python must be installed in the windows system with version 3.7.x or later
2. Pyopenssl library must be installed using 'pip install pyopenssl'
3. The ECDSA signing key and the Certificate pair must be present in the same folder as the python script and must be named 'ecdsa-sha256-signer.key.pem' and 'ecdsa-sha256-signer.crt.pem' respectively.
   (**Note:** *The key pair in SDK are just for example, please generated new key by openssl !*)

Run the python script in folder: "project\realtek_amebapro2_v0_example\src\aws_iot_freertos_lts\tools\amazon_ota_tools\"

- command after GCC build : **$ python3 python_custom_ecdsa_Pro2.py**

There might be some error if there are packages lack in your environment (like openssl...). Please install the package and run the script again.

Once all these are present and the python script is run, it will generate a signature file – **IDT-OTA-Signature** in "project\realtek_amebapro2_v0_example\src\aws_iot_freertos_lts\tools\amazon_ota_tools". This will be used in next step – crate OTA job in AWS-IOT Core.



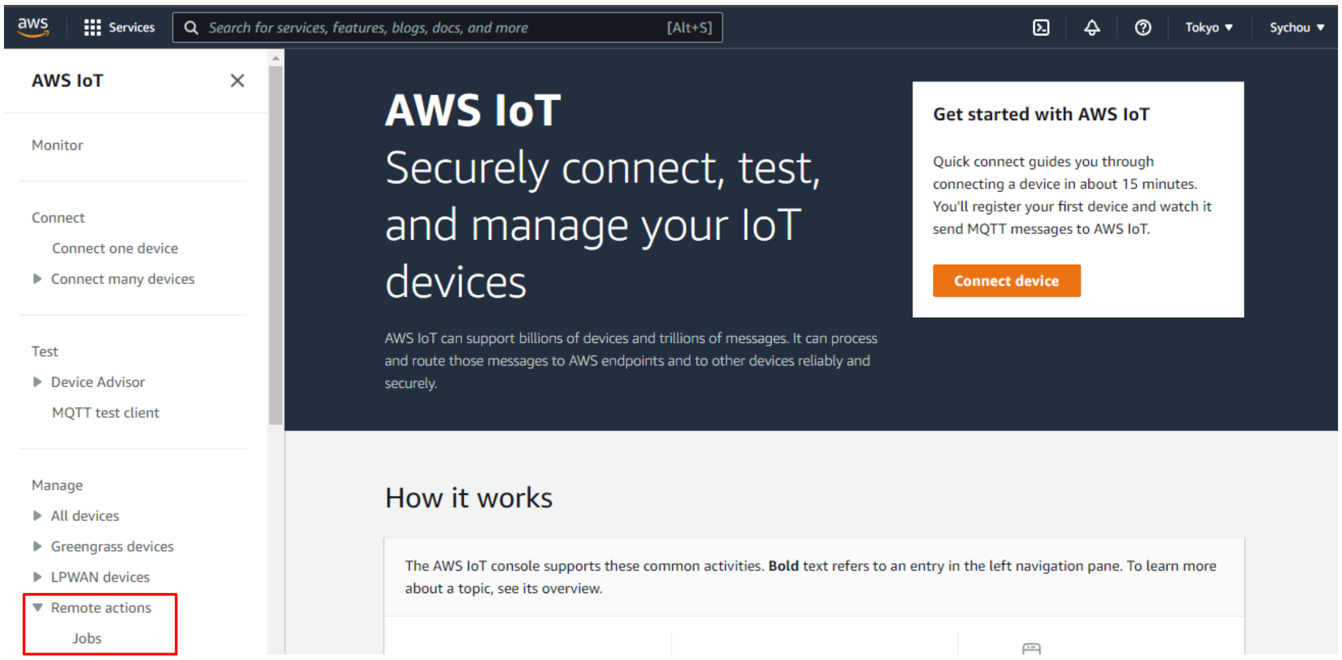
```
IDT-OTA-Signature
MEYCIQCQ6f3/1foRVpk8DxkCNAhetDrpKc5+wo7vtDEngsIGVAIhAOyHOMO1399a1r95uKNEa994j1PV1MOXb158YQ8Q14XY
```

After getting the custom signed signature, you can upload **ota.bin** to your S3 bucket.

7.4 How to Trigger a Custom Signed OTA Job in Amazon AWS IOT Core

Go to AWS IoT Core https://console.aws.amazon.com/iot/home. Then, follow the following steps to create an AWS OTA task for AmebaPro2:

Step 1. Click on 'Create OTA update job', select your job type and then click next.



AWS IoT

Securely connect, test, and manage your IoT devices

AWS IoT can support billions of devices and trillions of messages. It can process and route those messages to AWS endpoints and to other devices reliably and securely.

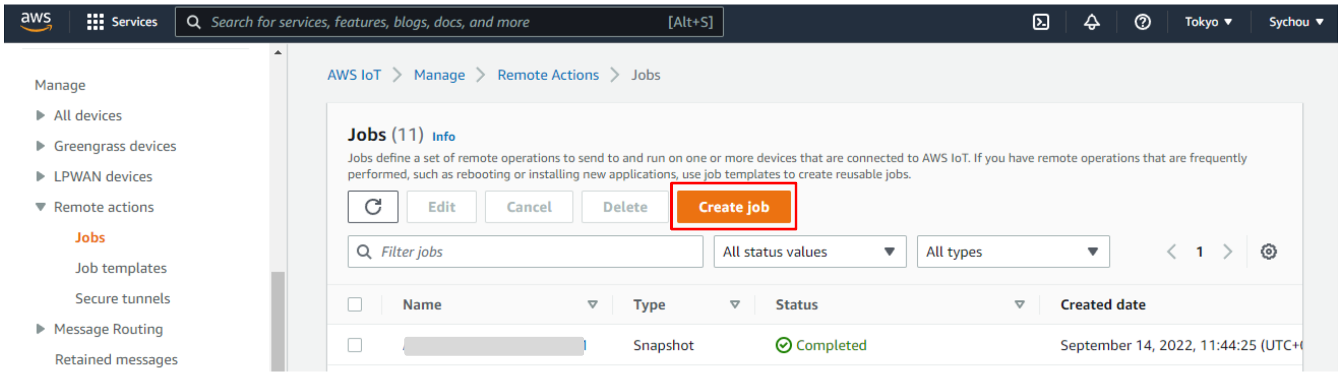
Get started with AWS IoT

Quick connect guides you through connecting a device in about 15 minutes. You'll register your first device and watch it send MQTT messages to AWS IoT.

How it works

The AWS IoT console supports these common activities. **Bold** text refers to an entry in the left navigation pane. To learn more about a topic, see its overview.

…



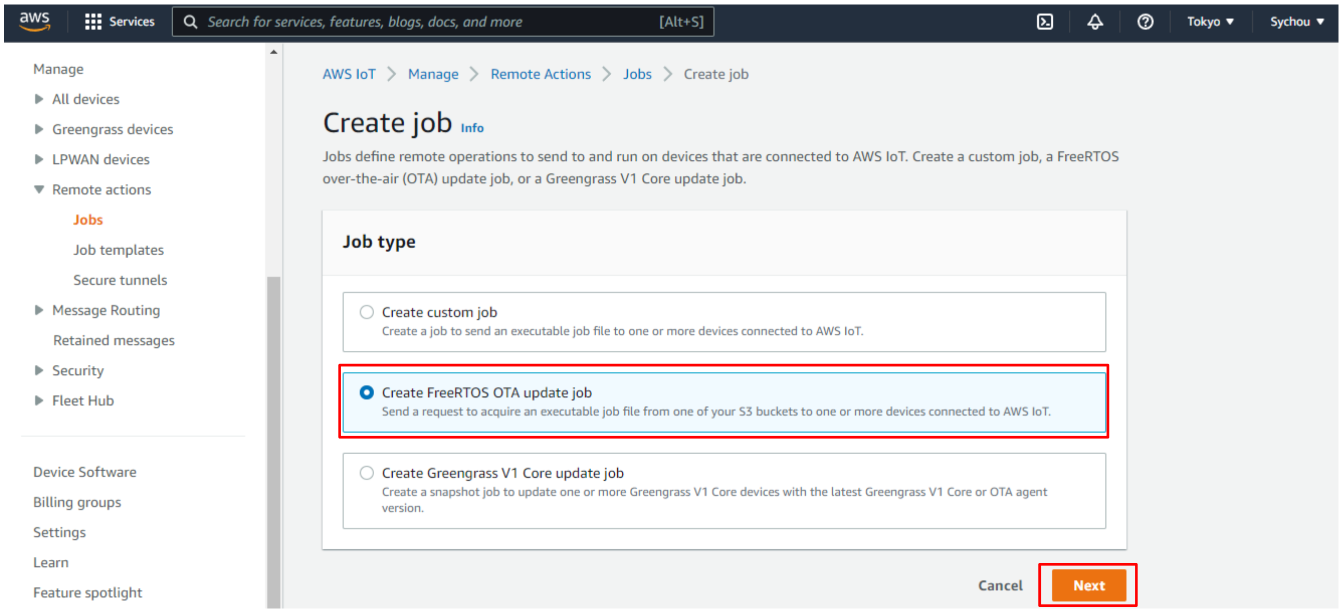
AWS IoT > Manage > Remote Actions > Jobs

Jobs (11) Info

Jobs define a set of remote operations to send to and run on one or more devices that are connected to AWS IoT. If you have remote operations that are frequently performed, such as rebooting or installing new applications, use job templates to create reusable jobs.

Name	Type	Status	Created date
[redacted]	Snapshot	Completed	September 14, 2022, 11:44:25 (UTC+

…



AWS IoT > Manage > Remote Actions > Jobs > Create job

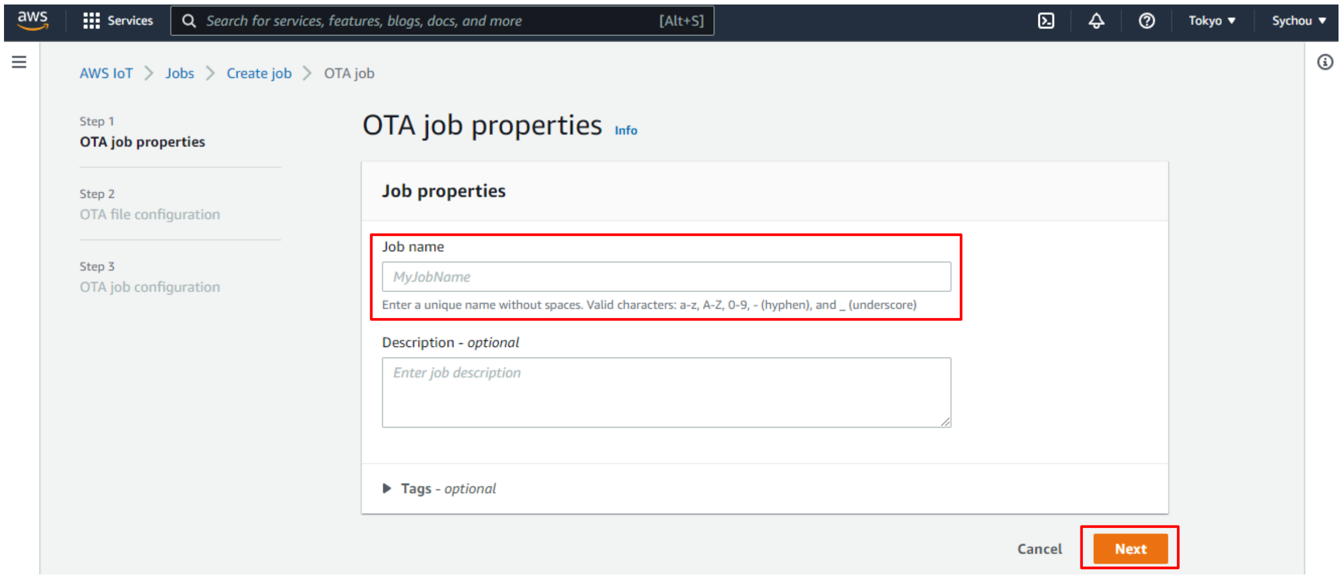
Create job Info

Jobs define remote operations to send to and run on devices that are connected to AWS IoT. Create a custom job, a FreeRTOS over-the-air (OTA) update job, or a Greengrass V1 Core update job.

Job type

- Create custom job
  Create a job to send an executable job file to one or more devices connected to AWS IoT.
- [✓ Create FreeRTOS OTA update job]
  Send a request to acquire an executable job file from one of your S3 buckets to one or more devices connected to AWS IoT.
- Create Greengrass V1 Core update job
  Create a snapshot job to update one or more Greengrass V1 Core devices with the latest Greengrass V1 Core or OTA agent version.

Step 2. For Job properties, give a unique name to your OTA job, then click next.



OTA job properties Info

Job properties

Job name

MyJobName

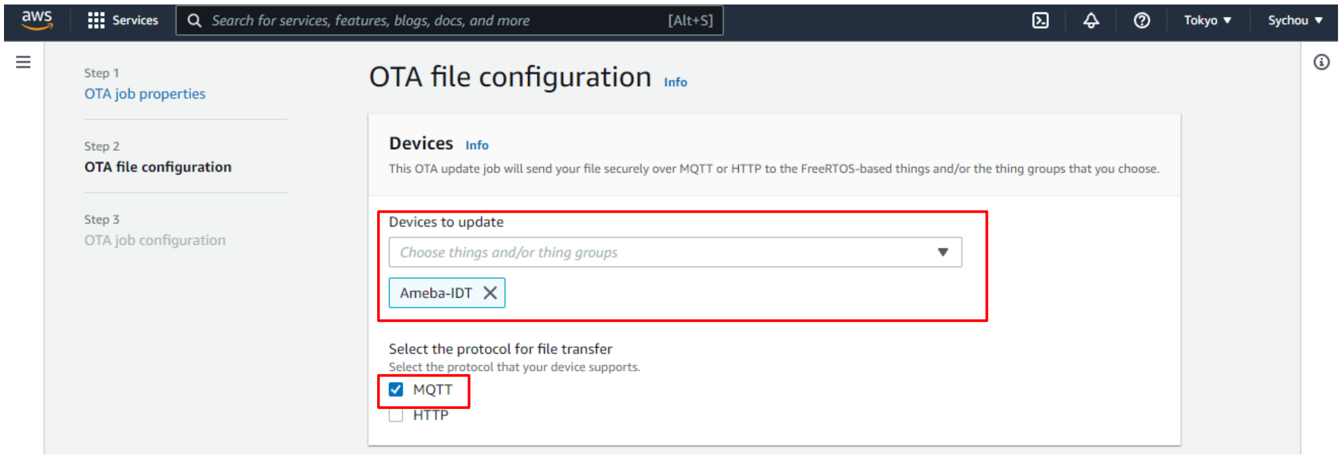
Enter a unique name without spaces. Valid characters: a-z, A-Z, 0-9, - (hyphen), and _ (underscore)

Description - *optional*

Enter job description

Tags - *optional*

Step 3. In the following page, choose your device to update and select the protocol for file transfer



OTA file configuration Info

Devices Info

This OTA update job will send your file securely over MQTT or HTTP to the FreeRTOS-based things and/or the thing groups that you choose.

Devices to update

Choose things and/or thing groups

Ameba-IDT

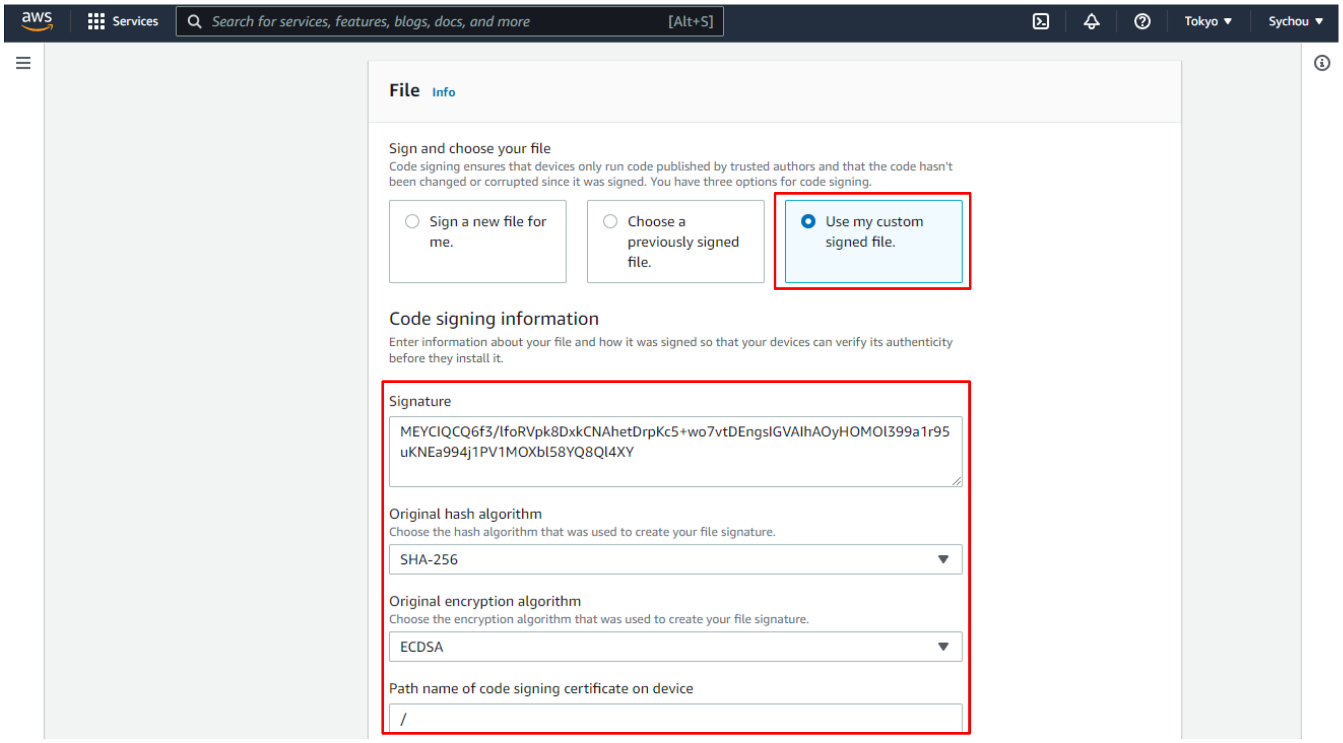
Select the protocol for file transfer

Select the protocol that your device supports.

☑ MQTT

☐ HTTP

Step 4. In the following page, choose the option 'Use my custom signed firmware image'.
In the signature field, copy and paste the content of **IDT-OTA-Signature** (generated in previous section).
Choose hash algorithm as 'SHA-256'.
Choose encryption algorithm as 'ECDSA'.
In "pathname of code signing certificate", enter '/'



Sign and choose your file
Code signing ensures that devices only run code published by trusted authors and that the code hasn't been changed or corrupted since it was signed. You have three options for code signing.

Sign a new file for me. Choose a previously signed file. [✓ Use my custom signed file.]

Code signing information
Enter information about your file and how it was signed so that your devices can verify its authenticity before they install it.

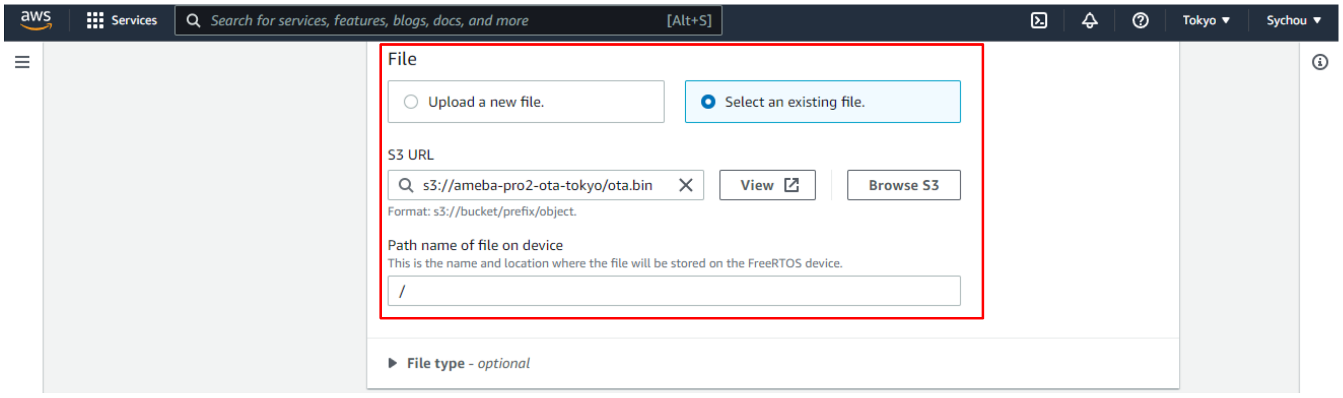
Signature
MEYCIQCQ6f3/lfoRVpk8DxkCNAhetDrpKc5+wo7vtDEngsIGVAIhAOyHOMOl399a1r95uKNEa994j1PV1MOXbl58YQ8Ql4XY

Original hash algorithm
Choose the hash algorithm that was used to create your file signature.
SHA-256

Original encryption algorithm
Choose the encryption algorithm that was used to create your file signature.
ECDSA

Path name of code signing certificate on device
/

Step 5. Choose your custom signed firmware binary that was generated by the python script from S3 bucket.
In "Pathname of file on device", enter '/'



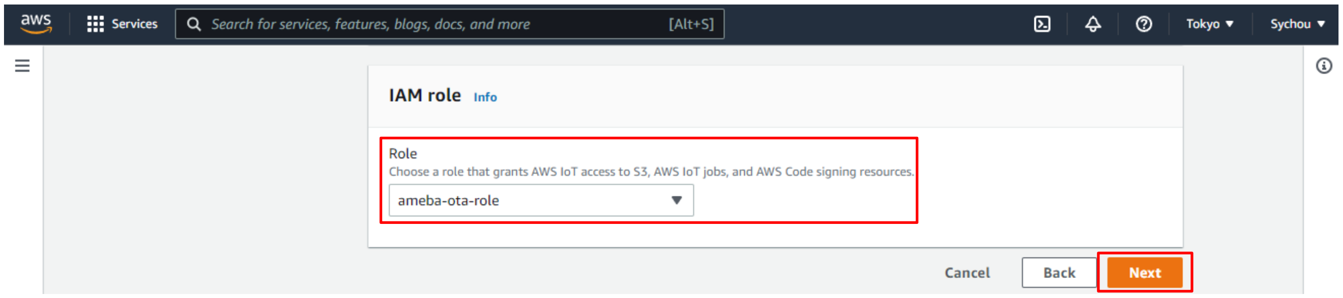
File
Upload a new file. [✓ Select an existing file.]

S3 URL
s3://ameba-pro2-ota-tokyo/ota.bin
Format: s3://bucket/prefix/object.

Path name of file on device
This is the name and location where the file will be stored on the FreeRTOS device.
/

File type - *optional*

Step 7. Choose the IAM role for OTA update job. (This is the same IAM role as any OTA update job)

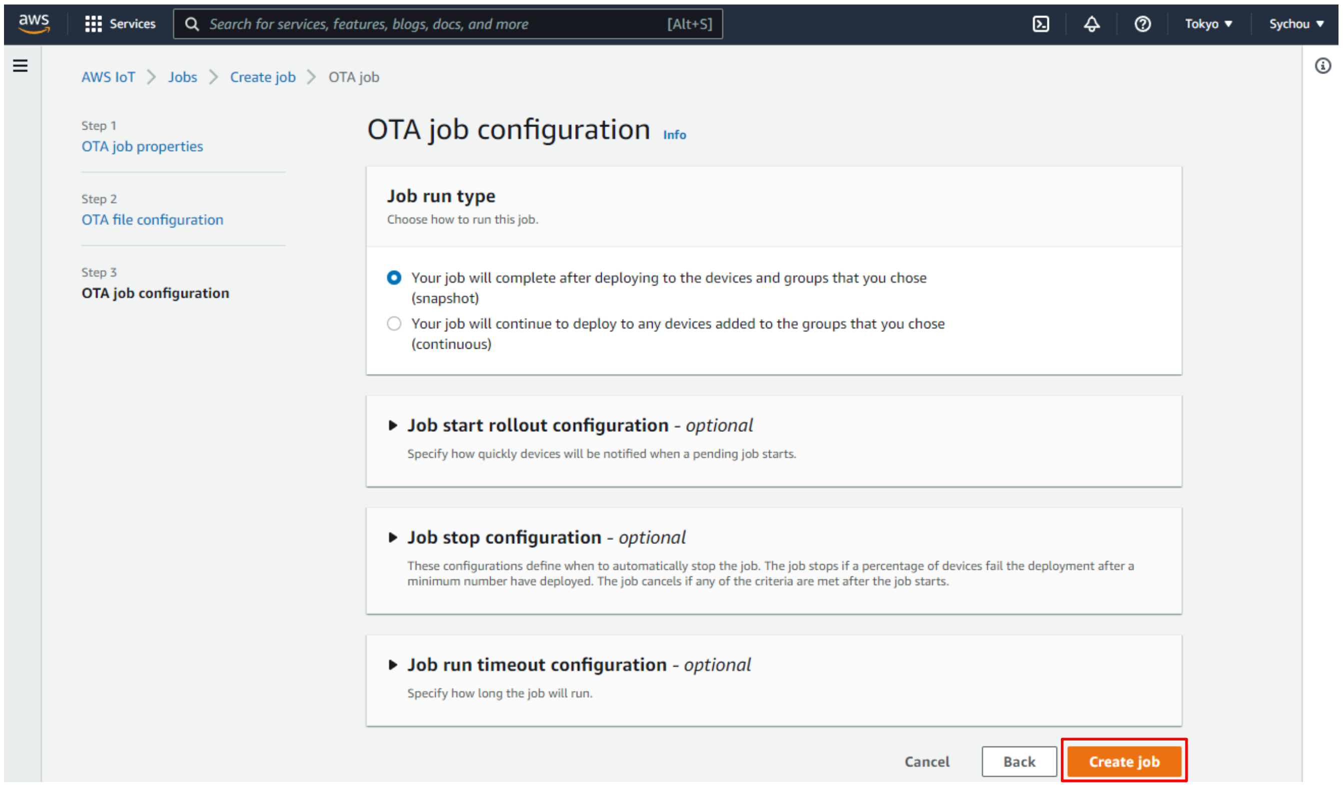


IAM role Info

Role
Choose a role that grants AWS IoT access to S3, AWS IoT jobs, and AWS Code signing resources.
ameba-ota-role

Services
Search for services, features, blogs, docs, and more [Alt+S]
Tokyo
Sychou
Cancel
Back
Next

Step 8. Click next, and create your OTA job.



AWS IoT > Jobs > Create job > OTA job

Step 1
OTA job properties
Step 2
OTA file configuration
Step 3
OTA job configuration

OTA job configuration Info

Job run type
Choose how to run this job.

- [✓] Your job will complete after deploying to the devices and groups that you chose (snapshot)
- Your job will continue to deploy to any devices added to the groups that you chose (continuous)

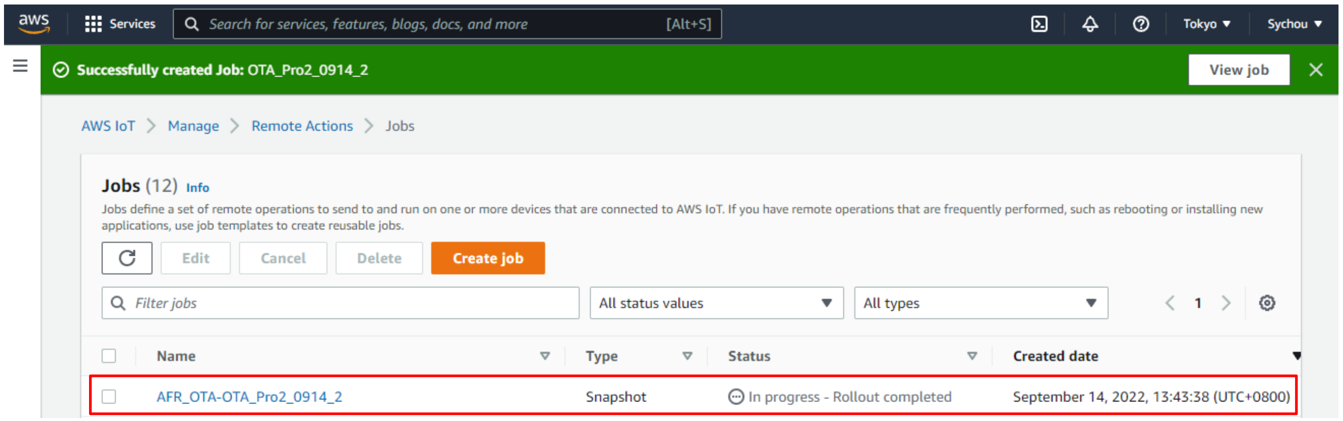
Job start rollout configuration - *optional*
Specify how quickly devices will be notified when a pending job starts.

Job stop configuration - *optional*
These configurations define when to automatically stop the job. The job stops if a percentage of devices fail the deployment after a minimum number have deployed. The job cancels if any of the criteria are met after the job starts.

Job run timeout configuration - *optional*
Specify how long the job will run.

Services
Search for services, features, blogs, docs, and more [Alt+S]
Tokyo
Sychou
Cancel
Back
Create job

...



Successfully created Job: OTA_Pro2_0914_2

AWS IoT > Manage > Remote Actions > Jobs

Jobs (12) Info

Jobs define a set of remote operations to send to and run on one or more devices that are connected to AWS IoT. If you have remote operations that are frequently performed, such as rebooting or installing new applications, use job templates to create reusable jobs.

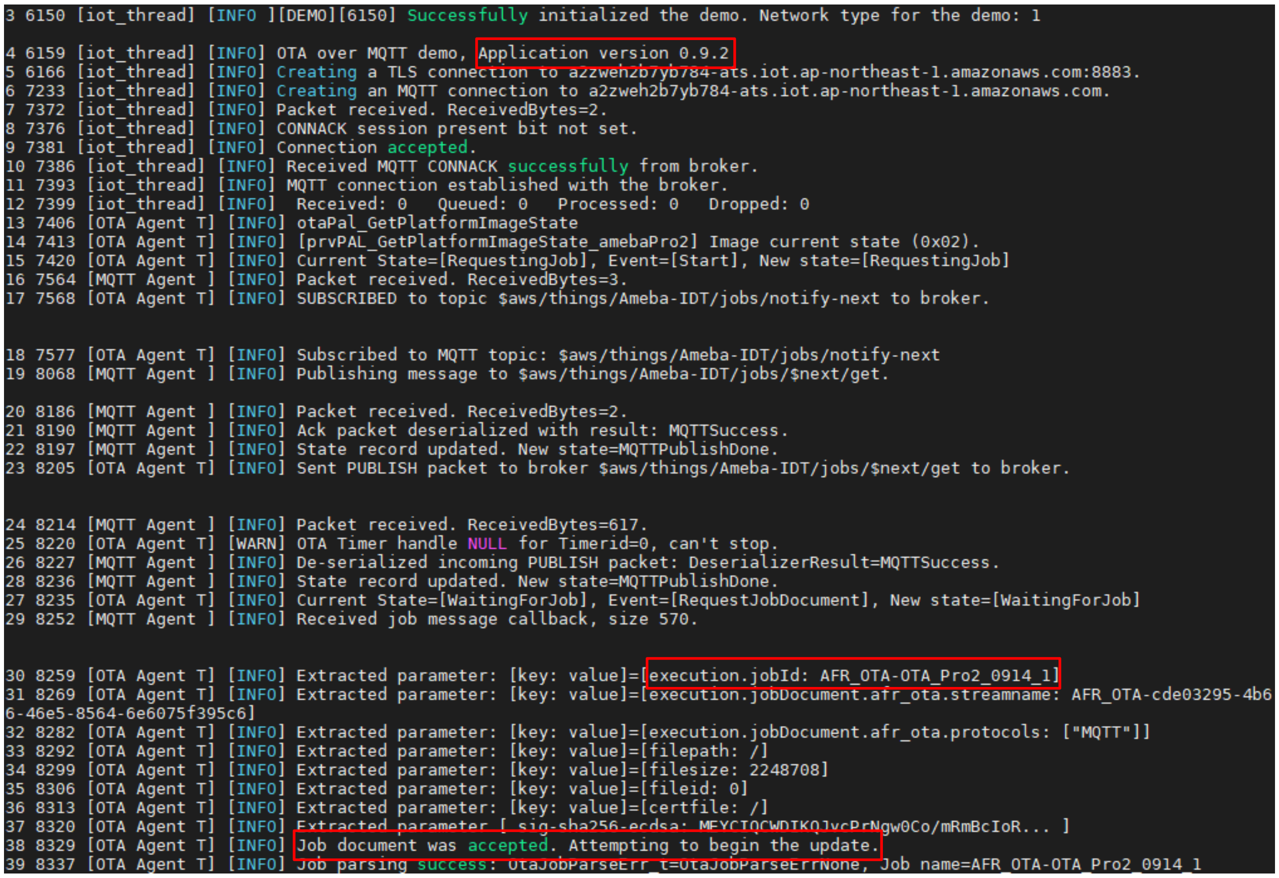
Name	Type	Status	Created date
AFR_OTA-OTA_Pro2_0914_2	Snapshot	In progress - Rollout completed	September 14, 2022, 13:43:38 (UTC+0800)

7.5 Run OTA Demo

Now we can see that the status of OTA job on AWS IoT Core is “in progress”. It means that it is waiting AmebaPro2 to request the update.

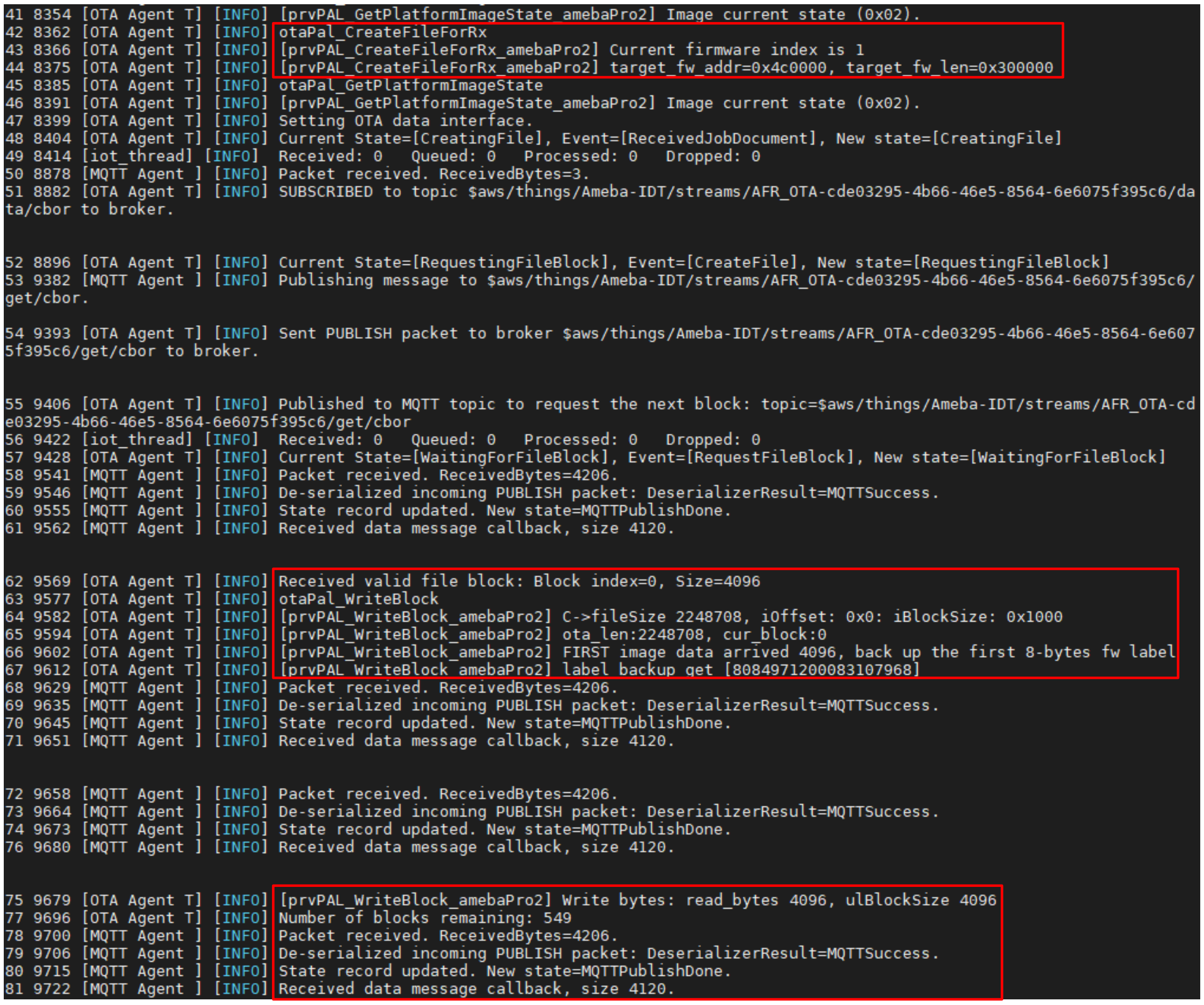
Next, download the image file with older version number 0.9.2 to AmebaPro2 and then reboot the device, the application will automatically start run OTA demo.

In the beginning, we can check the app version of this running firmware, and the OTA process by the job ID:



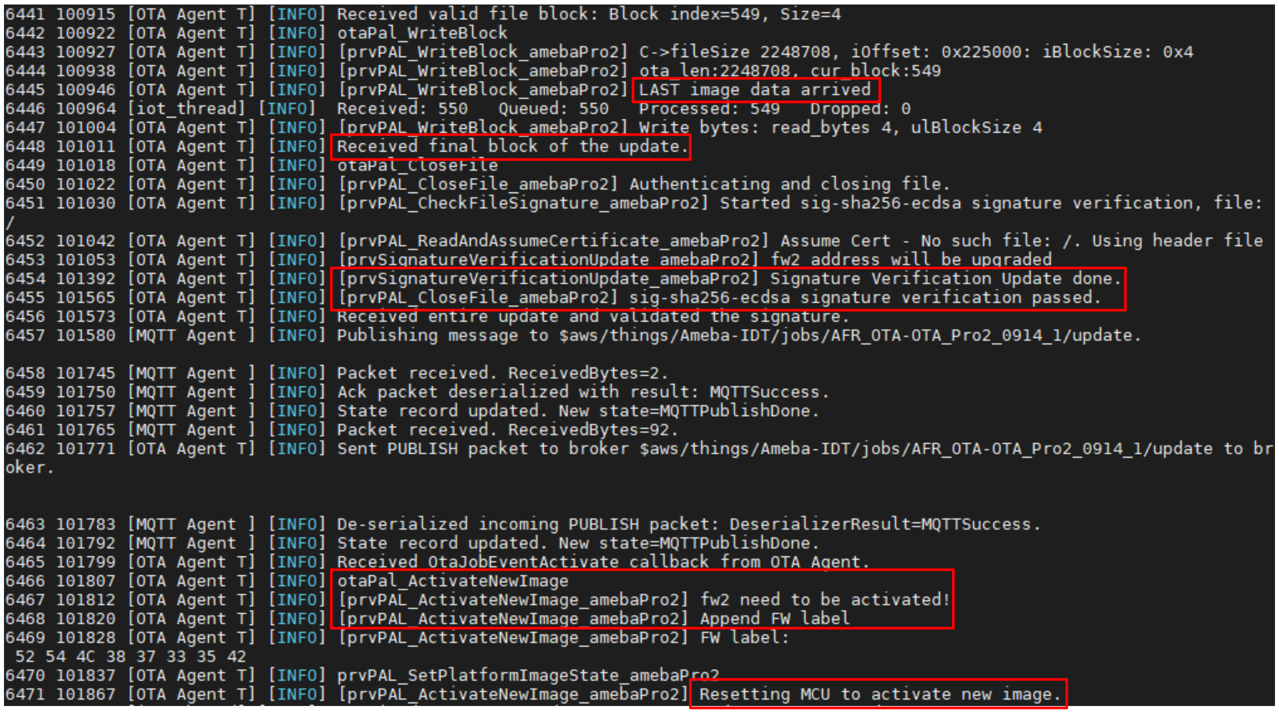
```
3 6150 [iot_thread] [INFO ][DEMO][6150] Successfully initialized the demo. Network type for the demo: 1

4 6159 [iot_thread] [INFO] OTA over MQTT demo, Application version 0.9.2
5 6166 [iot_thread] [INFO] Creating a TLS connection to a2zweh2b7yb784-ats.iot.ap-northeast-1.amazonaws.com:8883.
6 7233 [iot_thread] [INFO] Creating an MQTT connection to a2zweh2b7yb784-ats.iot.ap-northeast-1.amazonaws.com.
7 7372 [iot_thread] [INFO] Packet received. ReceivedBytes=2.
8 7376 [iot_thread] [INFO] CONNACK session present bit not set.
9 7381 [iot_thread] [INFO] Connection accepted.
10 7386 [iot_thread] [INFO] Received MQTT CONNACK successfully from broker.
11 7393 [iot_thread] [INFO] MQTT connection established with the broker.
12 7399 [iot_thread] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
13 7406 [OTA Agent T] [INFO] otaPal_GetPlatformImageState
14 7413 [OTA Agent T] [INFO] [prvPAL_GetPlatformImageState_amebaPro2] Image current state (0x02).
15 7420 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
16 7564 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=3.
17 7568 [OTA Agent T] [INFO] SUBSCRIBED to topic $aws/things/Ameba-IDT/jobs/notify-next to broker.


18 7577 [OTA Agent T] [INFO] Subscribed to MQTT topic: $aws/things/Ameba-IDT/jobs/notify-next
19 8068 [MQTT Agent ] [INFO] Publishing message to $aws/things/Ameba-IDT/jobs/$next/get.

20 8186 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=2.
21 8190 [MQTT Agent ] [INFO] Ack packet deserialized with result: MQTTSuccess.
22 8197 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
23 8205 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/Ameba-IDT/jobs/$next/get to broker.


24 8214 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=617.
25 8220 [OTA Agent T] [WARN] OTA Timer handle NULL for Timerid=0, can't stop.
26 8227 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
28 8236 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
27 8235 [OTA Agent T] [INFO] Current State=[WaitingForJob], Event=[RequestJobDocument], New state=[WaitingForJob]
29 8252 [MQTT Agent ] [INFO] Received job message callback, size 570.


30 8259 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[execution.jobId: AFR_OTA-OTA_Pro2_0914_1]
31 8269 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[execution.jobDocument.afr_ota.streamname: AFR_OTA-cde03295-4b6
6-46e5-8564-6e6075f395c6]
32 8282 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[execution.jobDocument.afr_ota.protocols: ["MQTT"]]
33 8292 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[filepath: /]
34 8299 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[filesize: 2248708]
35 8306 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[fileid: 0]
36 8313 [OTA Agent T] [INFO] Extracted parameter: [key: value]=[certfile: /]
37 8320 [OTA Agent T] [INFO] Extracted parameter [ sig-sha256-ecdsa: MEYCIQCWDIKQJvcPrNgw0Co/mRmBcIoR... ]
38 8329 [OTA Agent T] [INFO] Job document was accepted. Attempting to begin the update.
39 8337 [OTA Agent T] [INFO] Job parsing success: OtaJobParseErr_t=OtaJobParseErrNone, Job name=AFR_OTA-OTA_Pro2_0914_1
```

We can see that the OTA process start!



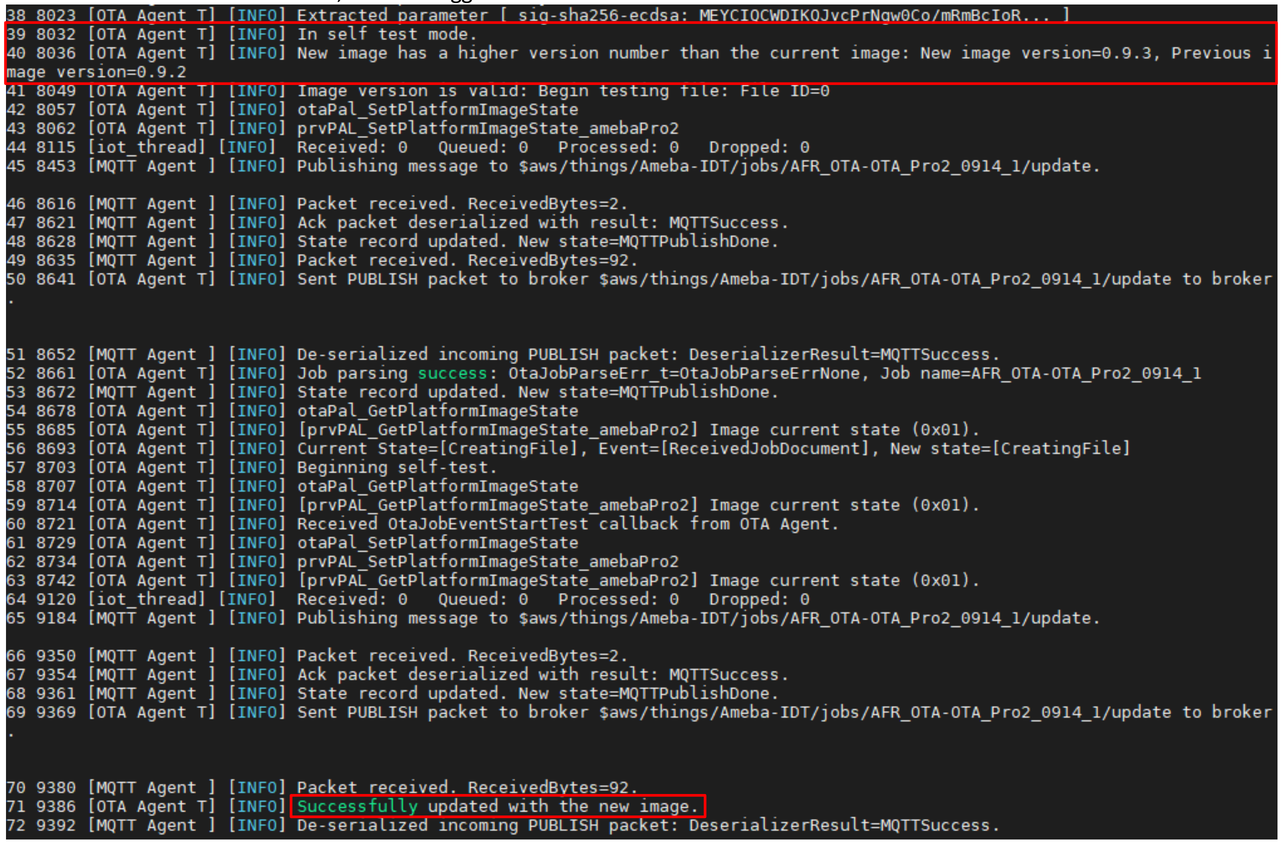
```
41 8354 [OTA Agent T] [INFO] [prvPAL_GetPlatformImageState_amebaPro2] Image current state (0x02).
42 8362 [OTA Agent T] [INFO] otaPal_CreateFileForRx
43 8366 [OTA Agent T] [INFO] [prvPAL_CreateFileForRx_amebaPro2] Current firmware index is 1
44 8375 [OTA Agent T] [INFO] [prvPAL_CreateFileForRx_amebaPro2] target_fw_addr=0x4c0000, target_fw_len=0x300000
45 8385 [OTA Agent T] [INFO] otaPal_GetPlatformImageState
46 8391 [OTA Agent T] [INFO] [prvPAL_GetPlatformImageState_amebaPro2] Image current state (0x02).
47 8399 [OTA Agent T] [INFO] Setting OTA data interface.
48 8404 [OTA Agent T] [INFO] Current State=[CreatingFile], Event=[ReceivedJobDocument], New state=[CreatingFile]
49 8414 [iot_thread] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
50 8878 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=3.
51 8882 [OTA Agent T] [INFO] SUBSCRIBED to topic $aws/things/Ameba-IDT/streams/AFR_OTA-cde03295-4b66-46e5-8564-6e6075f395c6/da
ta/cbor to broker.


52 8896 [OTA Agent T] [INFO] Current State=[RequestingFileBlock], Event=[CreateFile], New state=[RequestingFileBlock]
53 9382 [MQTT Agent ] [INFO] Publishing message to $aws/things/Ameba-IDT/streams/AFR_OTA-cde03295-4b66-46e5-8564-6e6075f395c6/
get/cbor.

54 9393 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/Ameba-IDT/streams/AFR_OTA-cde03295-4b66-46e5-8564-6e607
5f395c6/get/cbor to broker.


55 9406 [OTA Agent T] [INFO] Published to MQTT topic to request the next block: topic=$aws/things/Ameba-IDT/streams/AFR_OTA-cd
e03295-4b66-46e5-8564-6e6075f395c6/get/cbor
56 9422 [iot_thread] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
57 9428 [OTA Agent T] [INFO] Current State=[WaitingForFileBlock], Event=[RequestFileBlock], New state=[WaitingForFileBlock]
58 9541 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=4206.
59 9546 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
60 9555 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
61 9562 [MQTT Agent ] [INFO] Received data message callback, size 4120.


62 9569 [OTA Agent T] [INFO] Received valid file block: Block index=0, Size=4096
63 9577 [OTA Agent T] [INFO] otaPal_WriteBlock
64 9582 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] C->fileSize 2248708, iOffset: 0x0: iBlockSize: 0x1000
65 9594 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] ota_len:2248708, cur_block:0
66 9602 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] FIRST image data arrived 4096, back up the first 8-bytes fw label
67 9612 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] label backup get [8084971200083107968]
68 9629 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=4206.
69 9635 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
70 9645 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
71 9651 [MQTT Agent ] [INFO] Received data message callback, size 4120.


72 9658 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=4206.
73 9664 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
74 9673 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
76 9680 [MQTT Agent ] [INFO] Received data message callback, size 4120.


75 9679 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] Write bytes: read_bytes 4096, ulBlockSize 4096
77 9696 [OTA Agent T] [INFO] Number of blocks remaining: 549
78 9700 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=4206.
79 9706 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
80 9715 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
81 9722 [MQTT Agent ] [INFO] Received data message callback, size 4120.
```

After receiving the final block, the signature will be checked if valid or not. If signature is valid, the OTA process is successful!
Then, the device will reboot with new firmware automatically.



```
6441 100915 [OTA Agent T] [INFO] Received valid file block: Block index=549, Size=4
6442 100922 [OTA Agent T] [INFO] otaPal_WriteBlock
6443 100927 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] C->fileSize 2248708, iOffset: 0x225000: iBlockSize: 0x4
6444 100938 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] ota_len:2248708, cur_block:549
6445 100946 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] LAST image data arrived
6446 100964 [iot_thread] [INFO]  Received: 550   Queued: 550   Processed: 549   Dropped: 0
6447 101004 [OTA Agent T] [INFO] [prvPAL_WriteBlock_amebaPro2] Write bytes: read_bytes 4, ulBlockSize 4
6448 101011 [OTA Agent T] [INFO] Received final block of the update.
6449 101018 [OTA Agent T] [INFO] otaPal_CloseFile
6450 101022 [OTA Agent T] [INFO] [prvPAL_CloseFile_amebaPro2] Authenticating and closing file.
6451 101030 [OTA Agent T] [INFO] [prvPAL_CheckFileSignature_amebaPro2] Started sig-sha256-ecdsa signature verification, file:
/
6452 101042 [OTA Agent T] [INFO] [prvPAL_ReadAndAssumeCertificate_amebaPro2] Assume Cert - No such file: /. Using header file
6453 101053 [OTA Agent T] [INFO] [prvSignatureVerificationUpdate_amebaPro2] fw2 address will be upgraded
6454 101392 [OTA Agent T] [INFO] [prvSignatureVerificationUpdate_amebaPro2] Signature Verification Update done.
6455 101565 [OTA Agent T] [INFO] [prvPAL_CloseFile_amebaPro2] sig-sha256-ecdsa signature verification passed.
6456 101573 [OTA Agent T] [INFO] Received entire update and validated the signature.
6457 101580 [MQTT Agent ] [INFO] Publishing message to $aws/things/Ameba-IDT/jobs/AFR_OTA-OTA_Pro2_0914_1/update.

6458 101745 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=2.
6459 101750 [MQTT Agent ] [INFO] Ack packet deserialized with result: MQTTSuccess.
6460 101757 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
6461 101765 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=92.
6462 101771 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/Ameba-IDT/jobs/AFR_OTA-OTA_Pro2_0914_1/update to br
oker.


6463 101783 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
6464 101792 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
6465 101799 [OTA Agent T] [INFO] Received OtaJobEventActivate callback from OTA Agent.
6466 101807 [OTA Agent T] [INFO] otaPal_ActivateNewImage
6467 101812 [OTA Agent T] [INFO] [prvPAL_ActivateNewImage_amebaPro2] fw2 need to be activated!
6468 101820 [OTA Agent T] [INFO] [prvPAL_ActivateNewImage_amebaPro2] Append FW label
6469 101828 [OTA Agent T] [INFO] [prvPAL_ActivateNewImage_amebaPro2] FW label:
 52 54 4C 38 37 33 35 42
6470 101837 [OTA Agent T] [INFO] prvPAL_SetPlatformImageState_amebaPro2
6471 101867 [OTA Agent T] [INFO] [prvPAL_ActivateNewImage_amebaPro2] Resetting MCU to activate new image.
```

After booting with newer image, the device will start a self-test mode to check the app version is newer than before.
We can see that the version now is 0.9.2, which is bigger than old one 0.9.3.



```
38 8023 [OTA Agent T] [INFO] Extracted parameter [ sig-sha256-ecdsa: MEYCIQCWDIKQJvcPrNgw0Co/mRmBcIoR... ]
39 8032 [OTA Agent T] [INFO] In self test mode.
40 8036 [OTA Agent T] [INFO] New image has a higher version number than the current image: New image version=0.9.3, Previous i
mage version=0.9.2
41 8049 [OTA Agent T] [INFO] Image version is valid: Begin testing file: File ID=0
42 8057 [OTA Agent T] [INFO] otaPal_SetPlatformImageState
43 8062 [OTA Agent T] [INFO] prvPAL_SetPlatformImageState_amebaPro2
44 8115 [iot_thread] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
45 8453 [MQTT Agent ] [INFO] Publishing message to $aws/things/Ameba-IDT/jobs/AFR_OTA-OTA_Pro2_0914_1/update.

46 8616 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=2.
47 8621 [MQTT Agent ] [INFO] Ack packet deserialized with result: MQTTSuccess.
48 8628 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
49 8635 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=92.
50 8641 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/Ameba-IDT/jobs/AFR_OTA-OTA_Pro2_0914_1/update to broker
.


51 8652 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
52 8661 [OTA Agent T] [INFO] Job parsing success: OtaJobParseErr_t=OtaJobParseErrNone, Job name=AFR_OTA-OTA_Pro2_0914_1
53 8672 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
54 8678 [OTA Agent T] [INFO] otaPal_GetPlatformImageState
55 8685 [OTA Agent T] [INFO] [prvPAL_GetPlatformImageState_amebaPro2] Image current state (0x01).
56 8693 [OTA Agent T] [INFO] Current State=[CreatingFile], Event=[ReceivedJobDocument], New state=[CreatingFile]
57 8703 [OTA Agent T] [INFO] Beginning self-test.
58 8707 [OTA Agent T] [INFO] otaPal_GetPlatformImageState
59 8714 [OTA Agent T] [INFO] [prvPAL_GetPlatformImageState_amebaPro2] Image current state (0x01).
60 8721 [OTA Agent T] [INFO] Received OtaJobEventStartTest callback from OTA Agent.
61 8729 [OTA Agent T] [INFO] otaPal_SetPlatformImageState
62 8734 [OTA Agent T] [INFO] prvPAL_SetPlatformImageState_amebaPro2
63 8742 [OTA Agent T] [INFO] [prvPAL_GetPlatformImageState_amebaPro2] Image current state (0x01).
64 9120 [iot_thread] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
65 9184 [MQTT Agent ] [INFO] Publishing message to $aws/things/Ameba-IDT/jobs/AFR_OTA-OTA_Pro2_0914_1/update.

66 9350 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=2.
67 9354 [MQTT Agent ] [INFO] Ack packet deserialized with result: MQTTSuccess.
68 9361 [MQTT Agent ] [INFO] State record updated. New state=MQTTPublishDone.
69 9369 [OTA Agent T] [INFO] Sent PUBLISH packet to broker $aws/things/Ameba-IDT/jobs/AFR_OTA-OTA_Pro2_0914_1/update to broker
.


70 9380 [MQTT Agent ] [INFO] Packet received. ReceivedBytes=92.
71 9386 [OTA Agent T] [INFO] Successfully updated with the new image.
72 9392 [MQTT Agent ] [INFO] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
```

In the final, the log imply that OTA success!

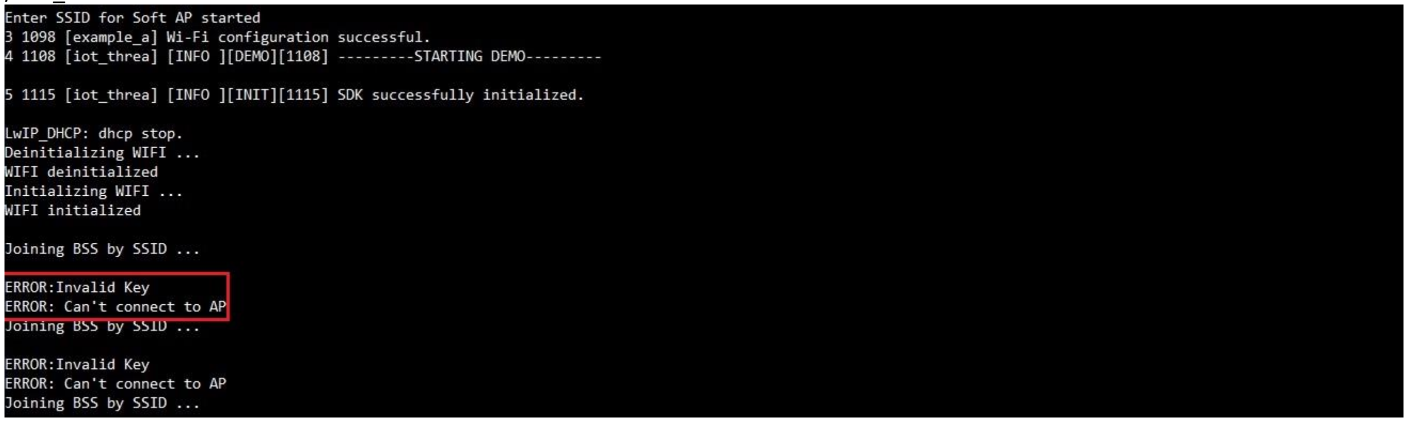
8 Troubleshooting

If these steps don't work, look at the device log in the serial terminal. You should see some text that indicates the source of the problem.

For general troubleshooting information about Getting Started with FreeRTOS, see Troubleshooting getting started.

8.1 ERROR: Invalid Key

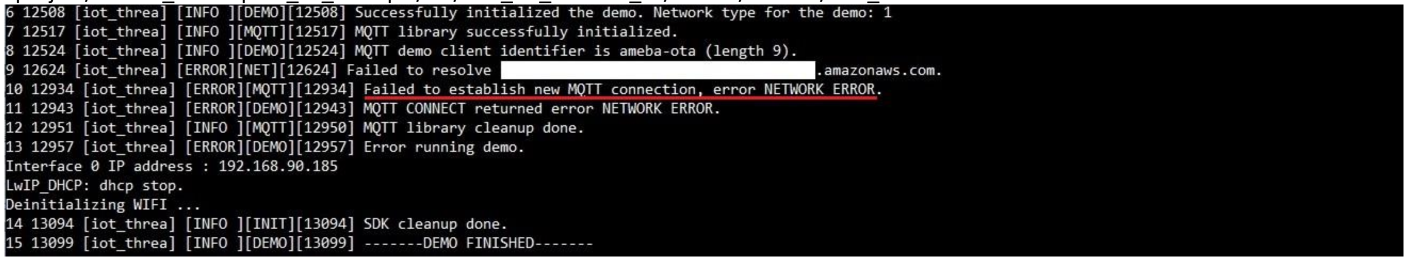
Please check **WIFI_SSID** and **WIFI_PASSWORD** in in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include /aws_clientcredential.h"



```
Enter SSID for Soft AP started
3 1098 [example_a] Wi-Fi configuration successful.
4 1108 [iot_threa] [INFO ][DEMO][1108] ---------STARTING DEMO---------

5 1115 [iot_threa] [INFO ][INIT][1115] SDK successfully initialized.

LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
WIFI deinitialized
Initializing WIFI ...
WIFI initialized

Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...
```

8.2 Failed to establish new MQTT connection

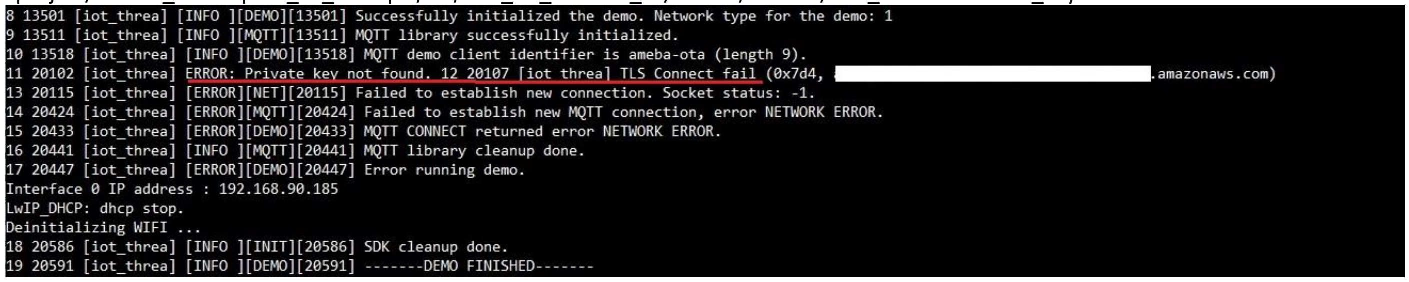
Please check **clientcredentialMQTT_BROKER_ENDPOINT** in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential.h"



```
6 12508 [iot_threa] [INFO ][DEMO][12508] Successfully initialized the demo. Network type for the demo: 1
7 12517 [iot_threa] [INFO ][MQTT][12517] MQTT library successfully initialized.
8 12524 [iot_threa] [INFO ][DEMO][12524] MQTT demo client identifier is ameba-ota (length 9).
9 12624 [iot_threa] [ERROR][NET][12624] Failed to resolve [redacted].amazonaws.com.
10 12934 [iot_threa] [ERROR][MQTT][12934] Failed to establish new MQTT connection, error NETWORK ERROR.
11 12943 [iot_threa] [ERROR][DEMO][12943] MQTT CONNECT returned error NETWORK ERROR.
12 12951 [iot_threa] [INFO ][MQTT][12950] MQTT library cleanup done.
13 12957 [iot_threa] [ERROR][DEMO][12957] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
14 13094 [iot_threa] [INFO ][INIT][13094] SDK cleanup done.
15 13099 [iot_threa] [INFO ][DEMO][13099] -------DEMO FINISHED-------
```

8.3 TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in "project/realtek_amebapro2_v0_example/src/aws_iot_freertos_lts/demos/include/aws_clientcredential_keys.h"



```
8 13501 [iot_threa] [INFO ][DEMO][13501] Successfully initialized the demo. Network type for the demo: 1
9 13511 [iot_threa] [INFO ][MQTT][13511] MQTT library successfully initialized.
10 13518 [iot_threa] [INFO ][DEMO][13518] MQTT demo client identifier is ameba-ota (length 9).
11 20102 [iot_threa] ERROR: Private key not found. 12 20107 [iot_threa] TLS_Connect fail (0x7d4, [redacted].amazonaws.com)
13 20115 [iot_threa] [ERROR][NET][20115] Failed to establish new connection. Socket status: -1.
14 20424 [iot_threa] [ERROR][MQTT][20424] Failed to establish new MQTT connection, error NETWORK ERROR.
15 20433 [iot_threa] [ERROR][DEMO][20433] MQTT CONNECT returned error NETWORK ERROR.
16 20441 [iot_threa] [INFO ][MQTT][20441] MQTT library cleanup done.
17 20447 [iot_threa] [ERROR][DEMO][20447] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
18 20586 [iot_threa] [INFO ][INIT][20586] SDK cleanup done.
19 20591 [iot_threa] [INFO ][DEMO][20591] -------DEMO FINISHED-------
```