



AmebaPro 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

COPYRIGHT

©2019 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

DISCLAIMER

Please Read Carefully:

Realtek Semiconductor Corp., (Realtek) reserves the right to make corrections, enhancements, improvements and other changes to its products and services. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

Reproduction of significant portions in Realtek data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Realtek is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions.

Buyers and others who are developing systems that incorporate Realtek products (collectively, “Customers”) understand and agree that Customers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Customers have full and exclusive responsibility to assure the safety of Customers' applications and compliance of their applications (and of all Realtek products used in or for Customers' applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Customer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Customer agrees that prior to using or distributing any applications that include Realtek products, Customer will thoroughly test such applications and the functionality of such Realtek products as used in such applications.

Realtek's provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation kits, (collectively, “Resources”) are intended to assist designers who are developing applications that incorporate Realtek products; by downloading, accessing or using Realtek's Resources in any way, Customer (individually or, if Customer is acting on behalf of a company, Customer's company) agrees to use any particular Realtek Resources solely for this purpose and subject to the terms of this Notice.

Realtek's provision of Realtek Resources does not expand or otherwise alter Realtek's applicable published warranties or warranty disclaimers for Realtek's products, and no additional obligations or liabilities arise from Realtek providing such Realtek Resources. Realtek reserves the right to make corrections, enhancements, improvements and other changes to its Realtek Resources. Realtek has not conducted any testing other than that specifically described in the published documentation for a particular Realtek Resource.

Customer is authorized to use, copy and modify any individual Realtek Resource only in connection with the development of applications that include the Realtek product(s) identified in such Realtek Resource. No other license, express or implied, by estoppel or otherwise to any other Realtek intellectual property right, and no license to any technology or intellectual property right of Realtek or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Realtek products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of Realtek Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from Realtek under the patents or other Realtek's intellectual property.

Realtek's Resources are provided “as is” and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding resources or use thereof, including but not limited to accuracy or completeness, title, any epidemic failure warranty and any implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third party intellectual property rights.

Realtek shall not be liable for and shall not defend or indemnify Customer against any claim, including but not limited to any infringement claim that related to or is based on any combination of products even if described in Realtek Resources or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek's Resources or use thereof, and regardless of whether Realtek has been advised of the possibility of such damages. Realtek is not responsible for any failure to meet such industry standard requirements.

Where Realtek specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Customers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any Realtek products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death. Such equipment includes, without limitation, all medical devices identified by the U.S.FDA as Class III devices and equivalent classifications outside the U.S.

Customers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Customers' own risk. Customers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Customer will fully indemnify Realtek and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

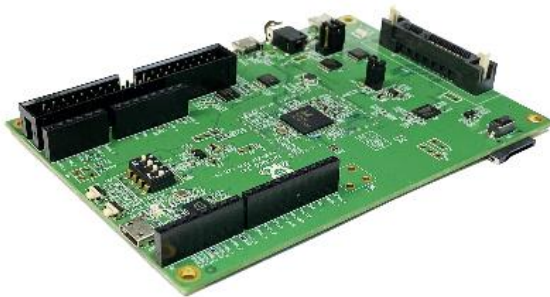
USING THIS DOCUMENT

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 AmebaPro RTL8715AD Board

1.1 AmebaPro Demo EVB

Ameba Demo board home page: <https://www.amebaiot.com/en/amebapro>



CPU

32-bit Arm v8M, up to 300MHz
32-bit Arm®Cortex®-M0, up to 4MHz



MEMORY

512KB RAM + 32MB LPDDR



KEY FEATURES

Integrated 802.11ac/n Wi-Fi SoC
Trustzone-M Security
Hardware SSL Engine
Root Trust Secure Boot
USB Host/Device
SD Host
LCDC
Codec
ISP
H.264



OTHER FEATURES

4 SPI interface
5 UART interface
2 I2S interface
4 I2C interface
11 ADC interface
16 PWM
2 PCM
Max 90 GPIO

1.2 PCB Layout Overview

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

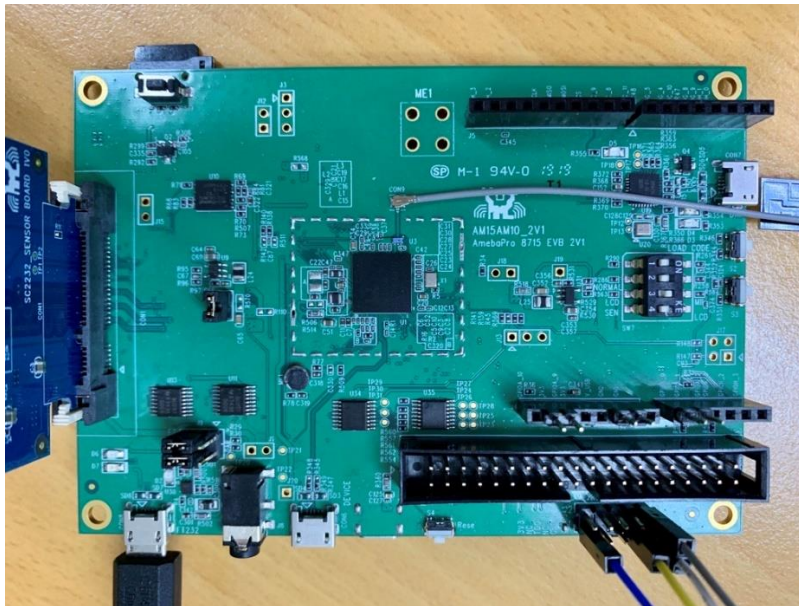


Fig 1-1 Demo board – PCB layout (2D)

1.3 Log UART

The log UART is shown in Fig 1-2.

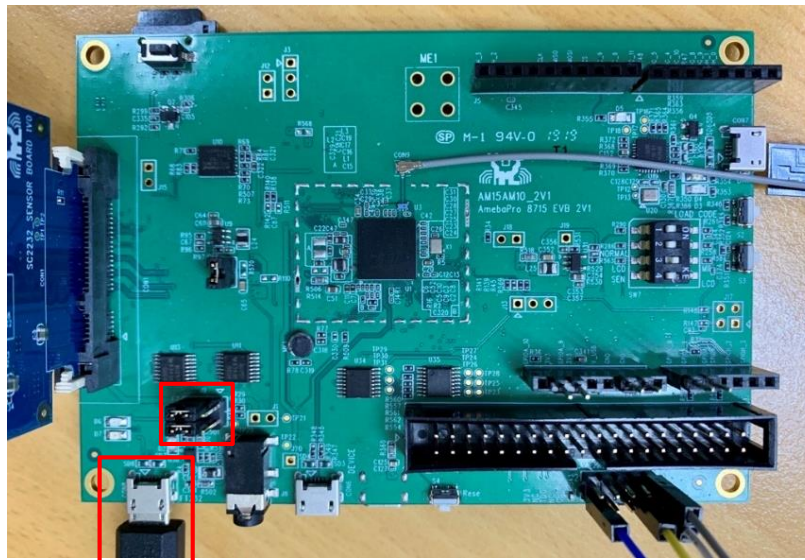


Fig 1-2 Demo board – log UART

1.4 JTAG/SWD

The SWD interface is shown in Fig 1-3.

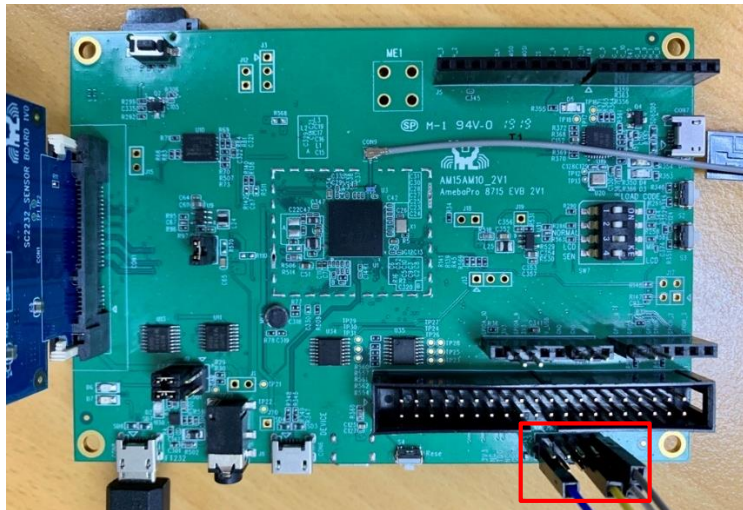


Fig 1-3 Demo board – JTAG/SWD

Note: If using 2V0 、2V1 version AmebaPro. Please check SW7 pin 3 switch to ON before connection.

1.5 Image Sensor

There is an image sensor socket as shown in Fig 1-4.

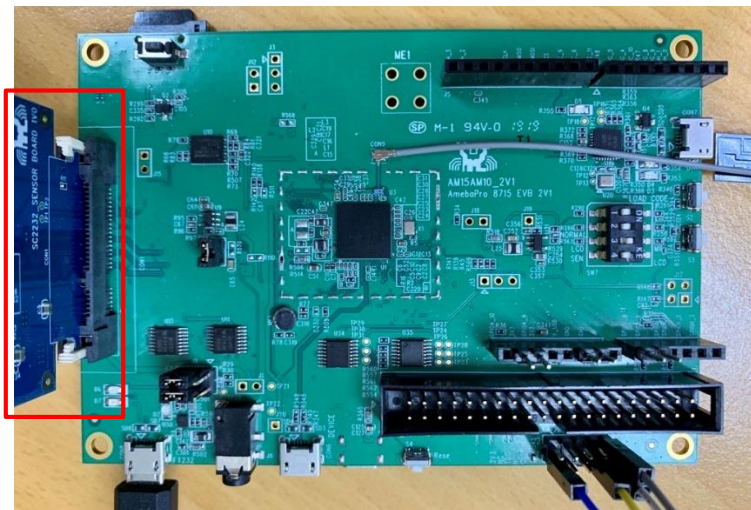


Fig 1-4 Demo board – image sensor

2 Configure AWS IoT Core

2.1 Set up your AWS account and Permissions

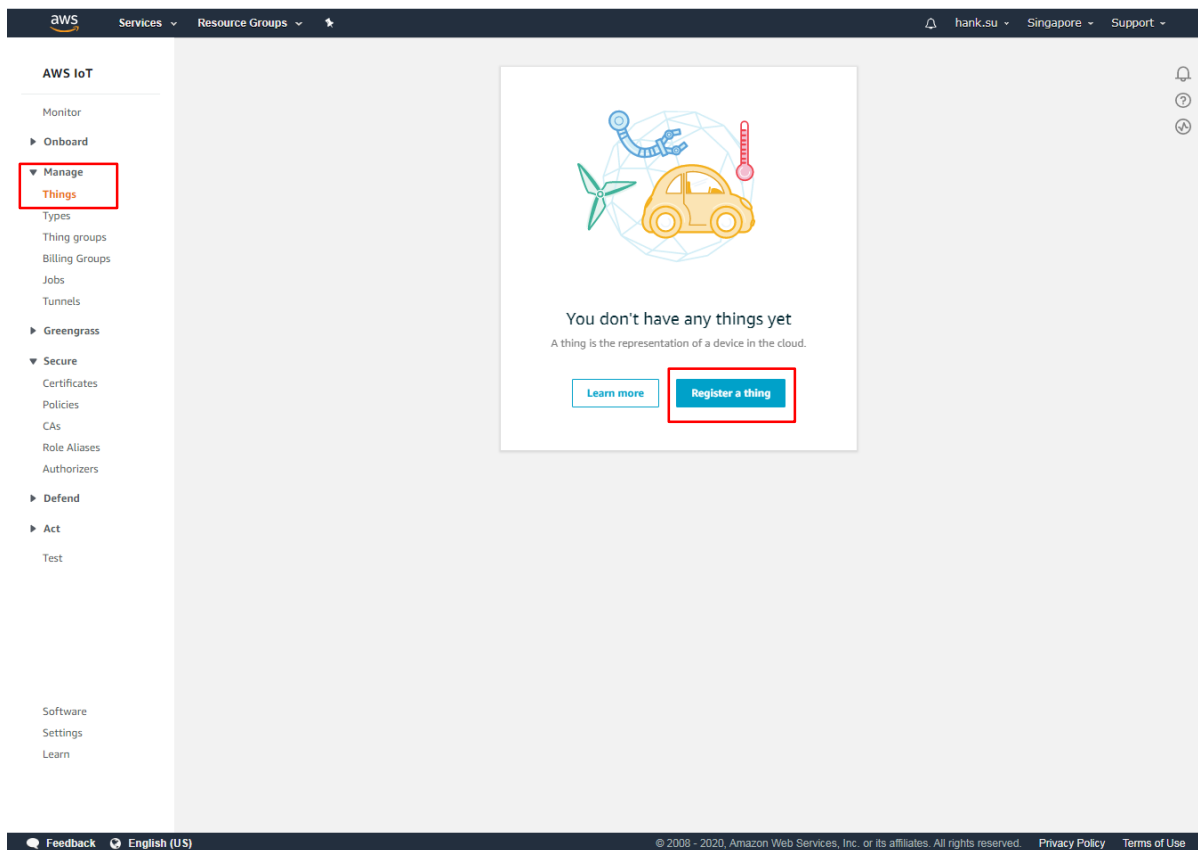
Refer to the instructions at Set up your AWS Account <https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html>. Follow the steps outlined in these sections to create your account and a user and get started:

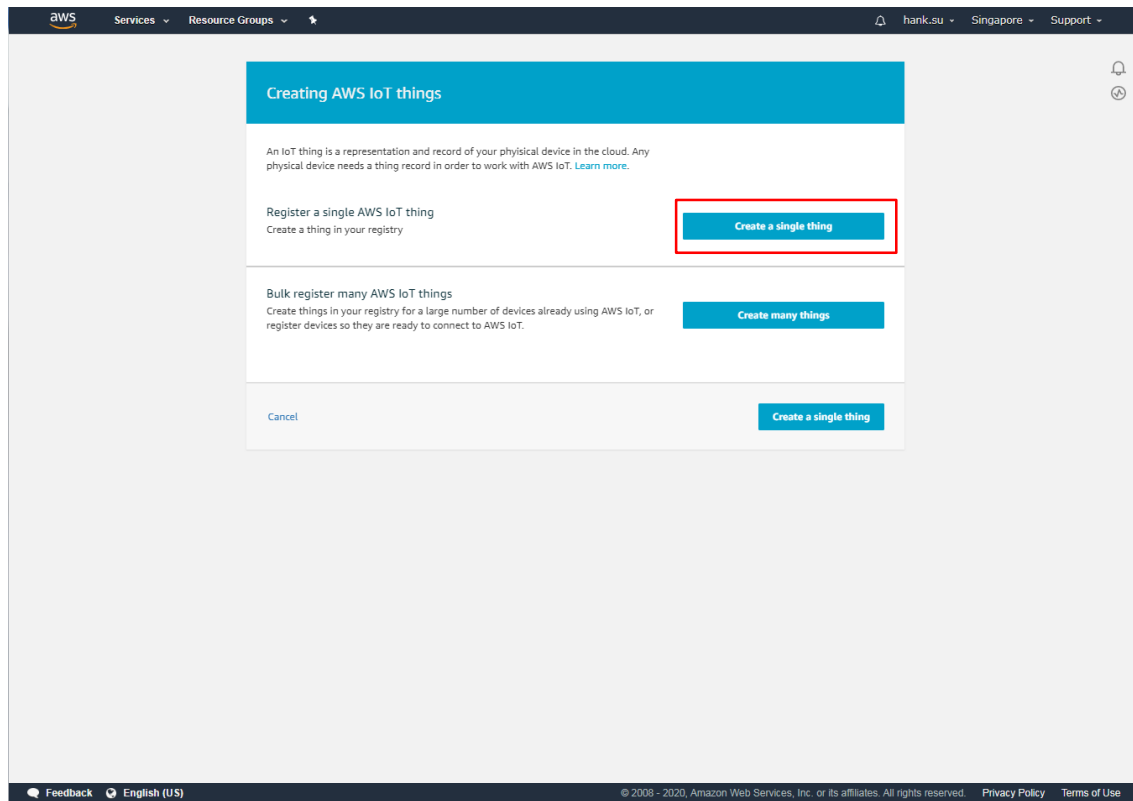
- Sign up for an AWS account
- Create a user and grant permissions
- Open the AWS IoT console

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 “Register a thing”.





Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more.](#)

Register a single AWS IoT thing
Create a thing in your registry

Create a single thing

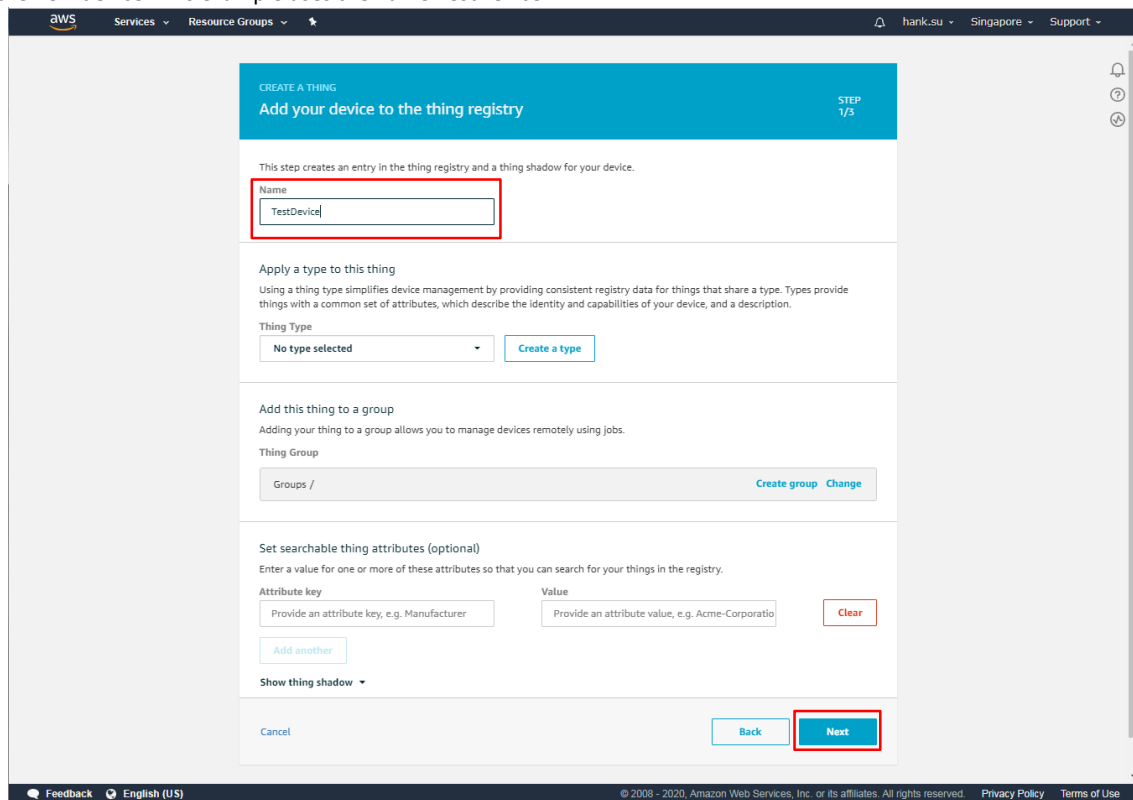
Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

Cancel **Create a single thing**

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

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



CREATE A THING **STEP 1/3**

Add your device to the thing registry

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

Name
TestDevice

Apply a type to this thing
Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type
No type selected **Create a type**

Add this thing to a group
Adding your thing to a group allows you to manage devices remotely using Jobs.

Thing Group
Groups / **Create group** **Change**

Set searchable thing attributes (optional)
Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key
Provide an attribute key, e.g. Manufacturer

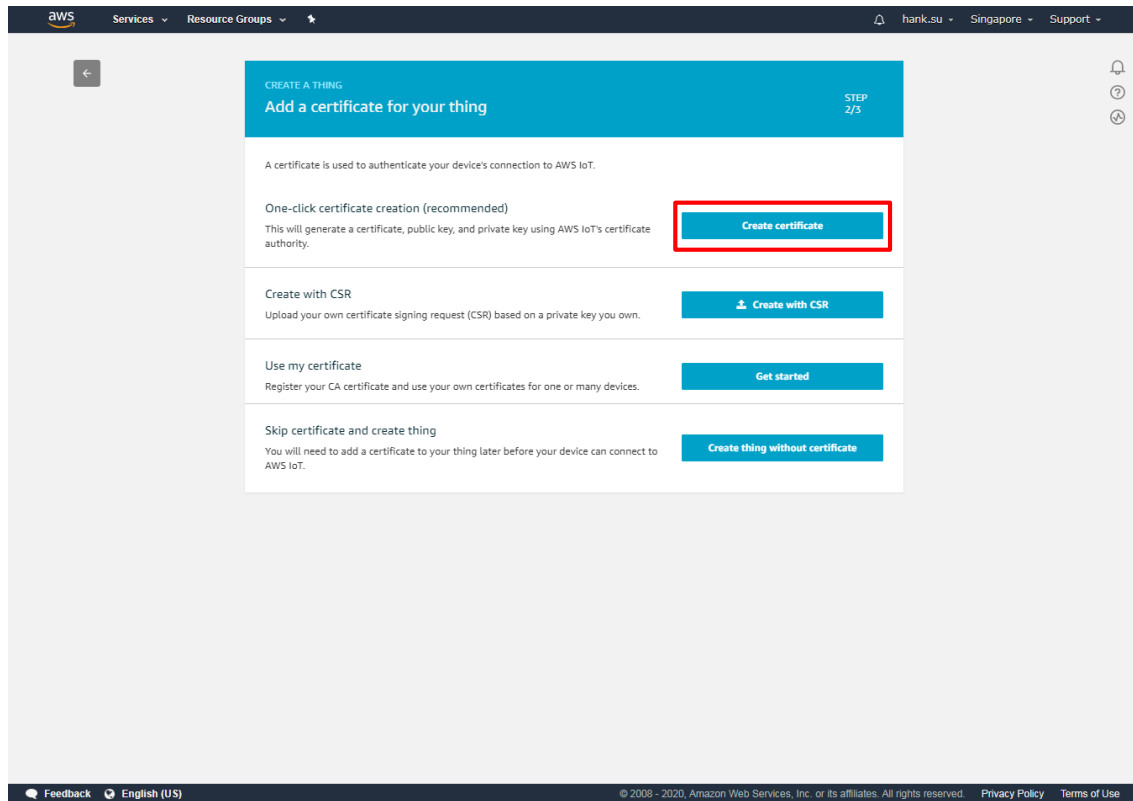
Value
Provide an attribute value, e.g. Acme-Corporatio **Clear**

Add another

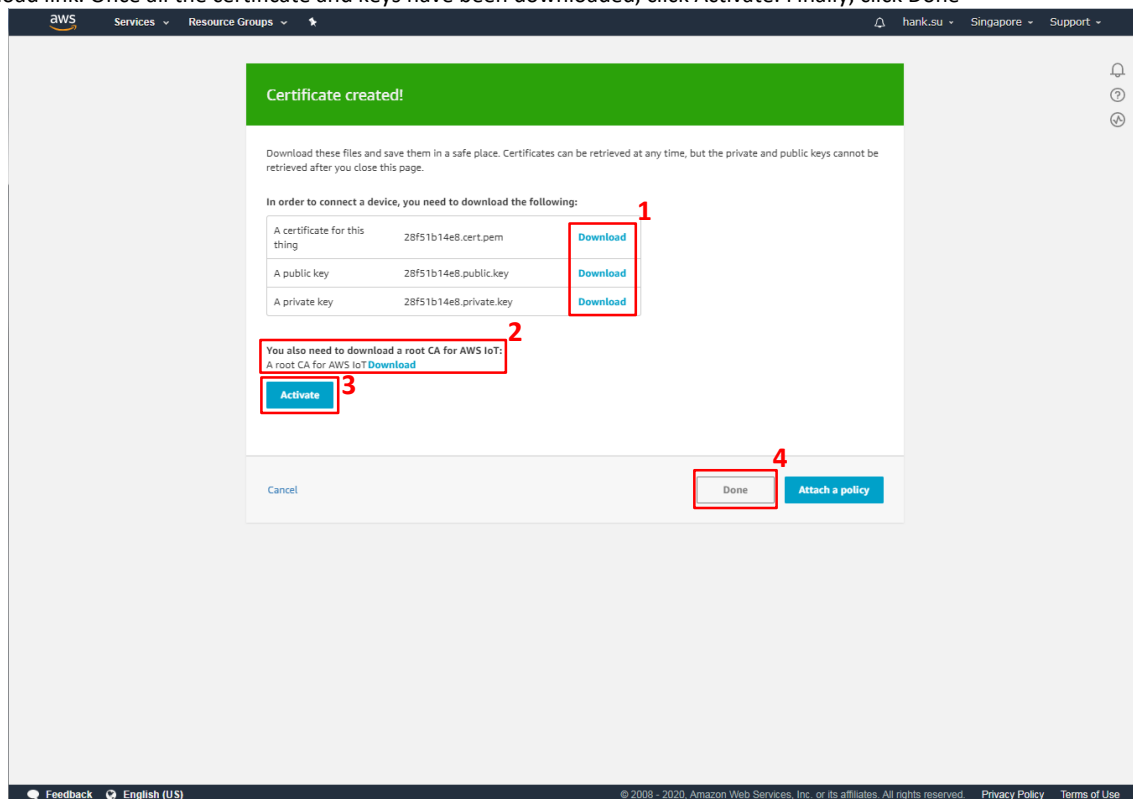
Show thing shadow

Cancel **Back** **Next**

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



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 Activate. Finally, click Done



CA certificates for server authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT Core server authentication certificates are signed by one of the following root CA certificates:

VeriSign Endpoints (legacy)

- RSA 2048 bit key: [VeriSign Class 3 Public Primary G5 root CA certificate](#)

Amazon Trust Services Endpoints (preferred)

Note

You might need to right click these links and select **Save link as...** to save these certificates as files.

- RSA 2048 bit key: [Amazon Root CA 1](#)
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: [Amazon Root CA 3](#)
- ECC 384 bit key: Amazon Root CA 4. Reserved for future use.

These certificates are all cross-signed by the [Starfield Root CA Certificate](#). All new AWS IoT Core regions, beginning with the May 9, 2018 launch of AWS IoT Core in the Asia Pacific (Mumbai) Region, serve only ATS certificates.

On this page

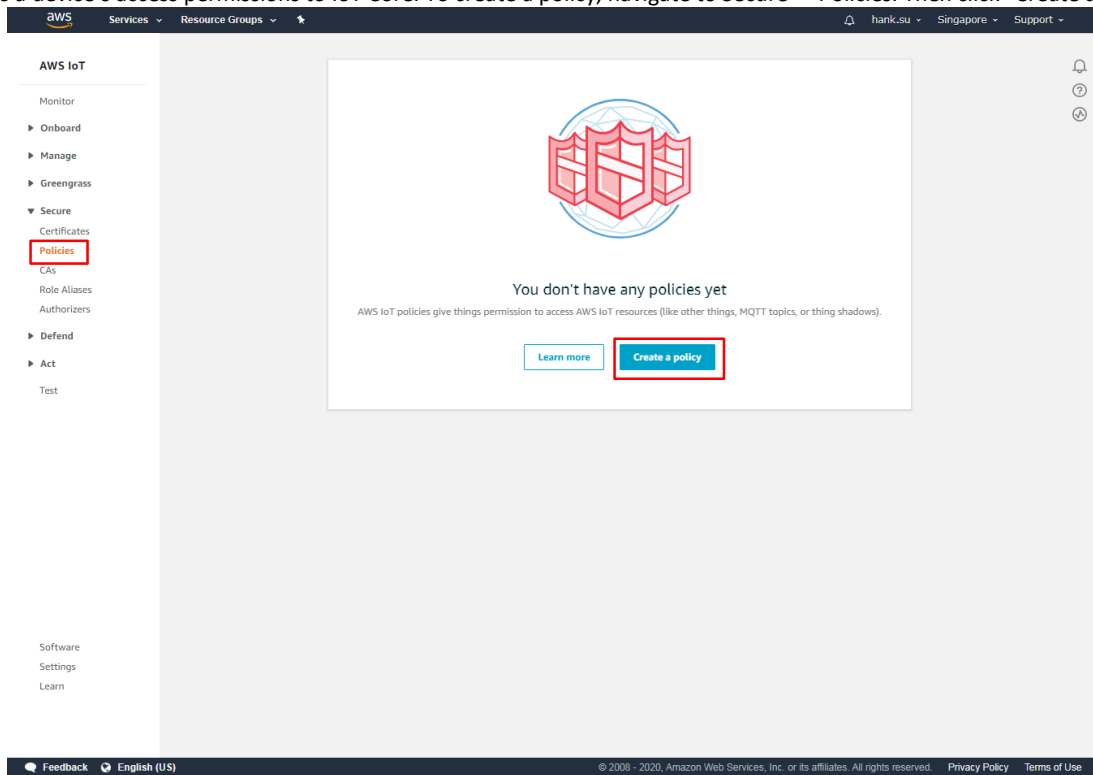
Endpoint types

CA certificates for server authentication

Server authentication guidelines

2.3 Create a policy

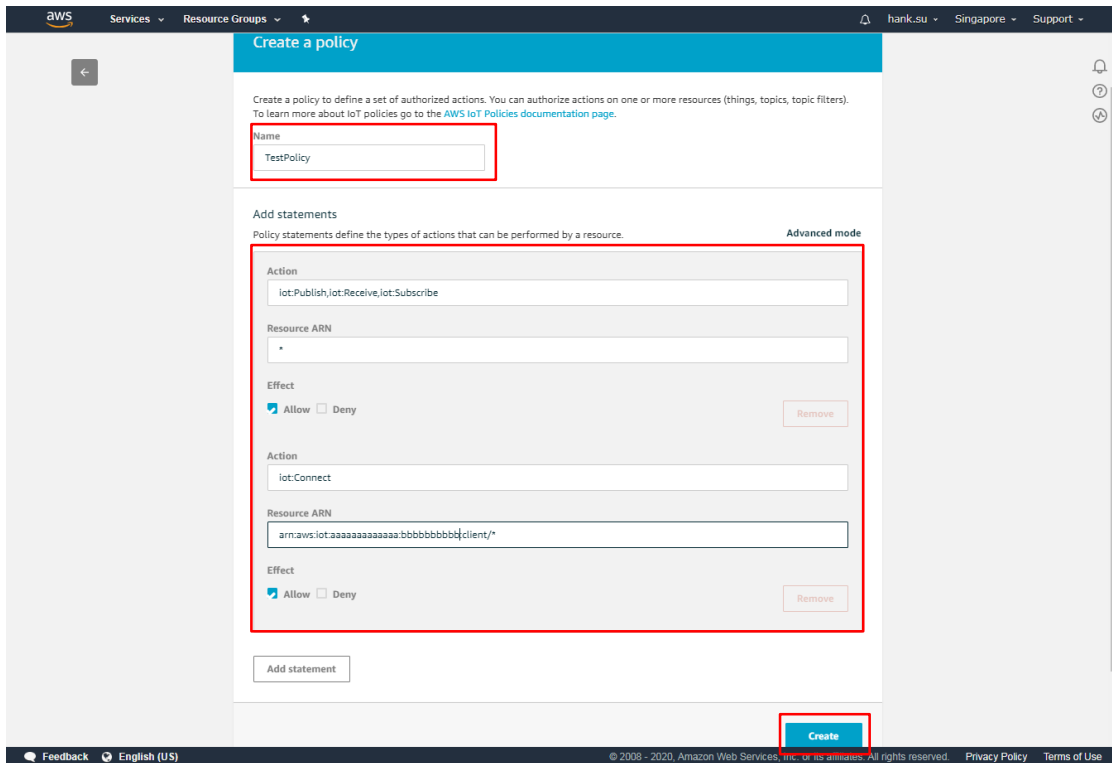
A policy defines a device's access permissions to IoT Core. To create a policy, navigate to Secure -> Policies. Then click "Create a policy"



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 a policy

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

Name
TestPolicy

Add statements
Policy statements define the types of actions that can be performed by a resource. **Advanced mode**

Action
iot:Publish,iot:Receive,iot:Subscribe

Resource ARN
*

Effect
☒ Allow ☐ Deny [Remove](#)

Action
iot:Connect

Resource ARN
arn:aws:iot:aaaaa:bbbbbb:client/*

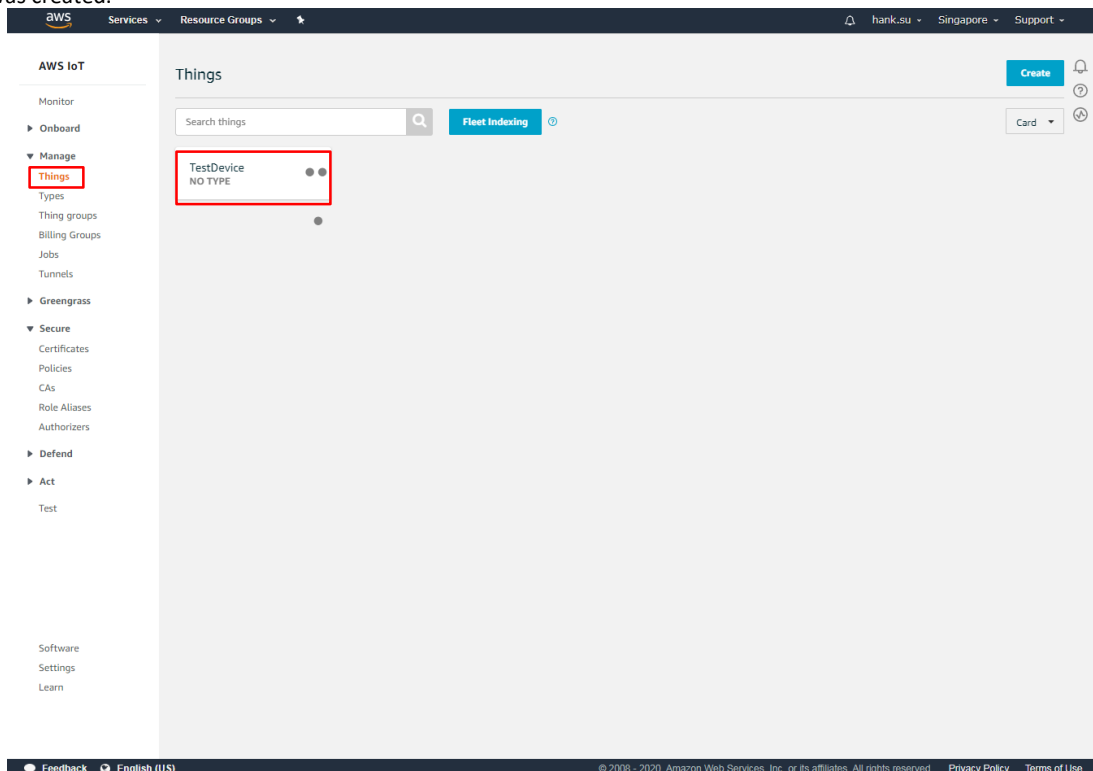
Effect
☒ Allow ☐ Deny [Remove](#)

[Add statement](#)

[Create](#)

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.



AWS IoT

Monitor

Onboard

Manage

Things

Types

Thing groups

Billing Groups

Jobs

Tunnels

Greengrass

Secure

Certificates

Policies

CAs

Role Aliases

Authorizers

Defend

Act

Test

Software

Settings

Learn

Things

Search things

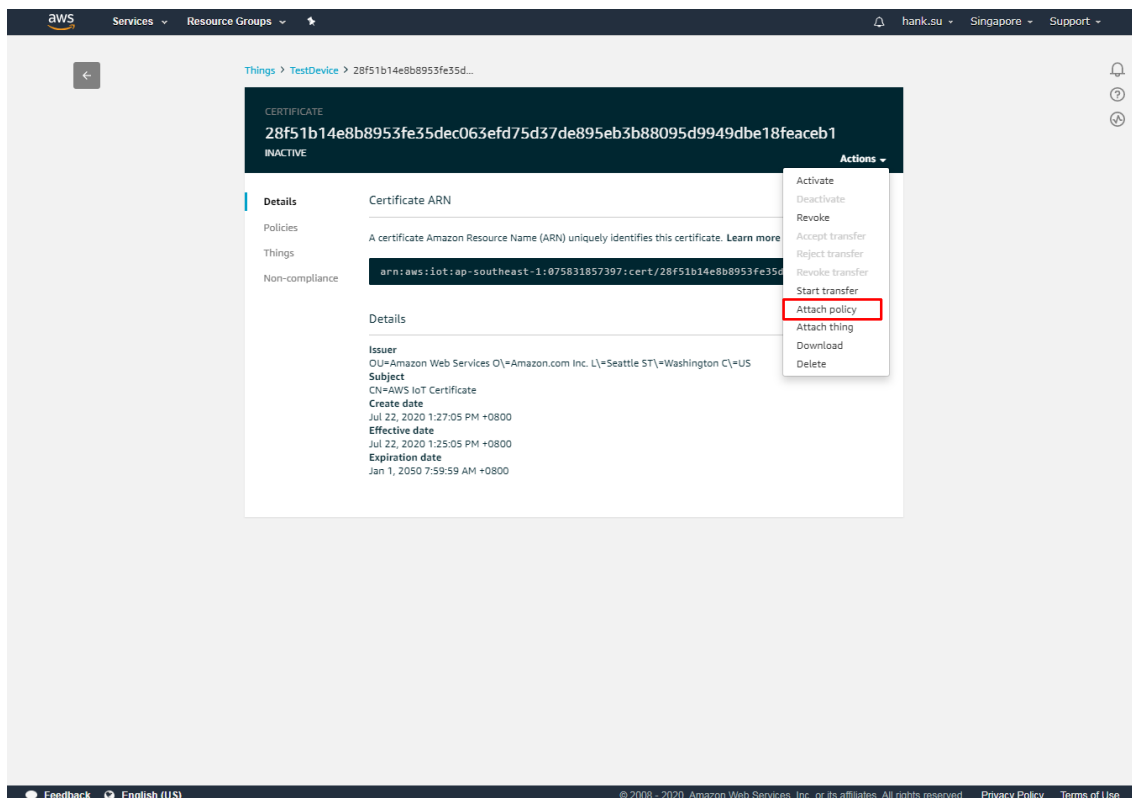
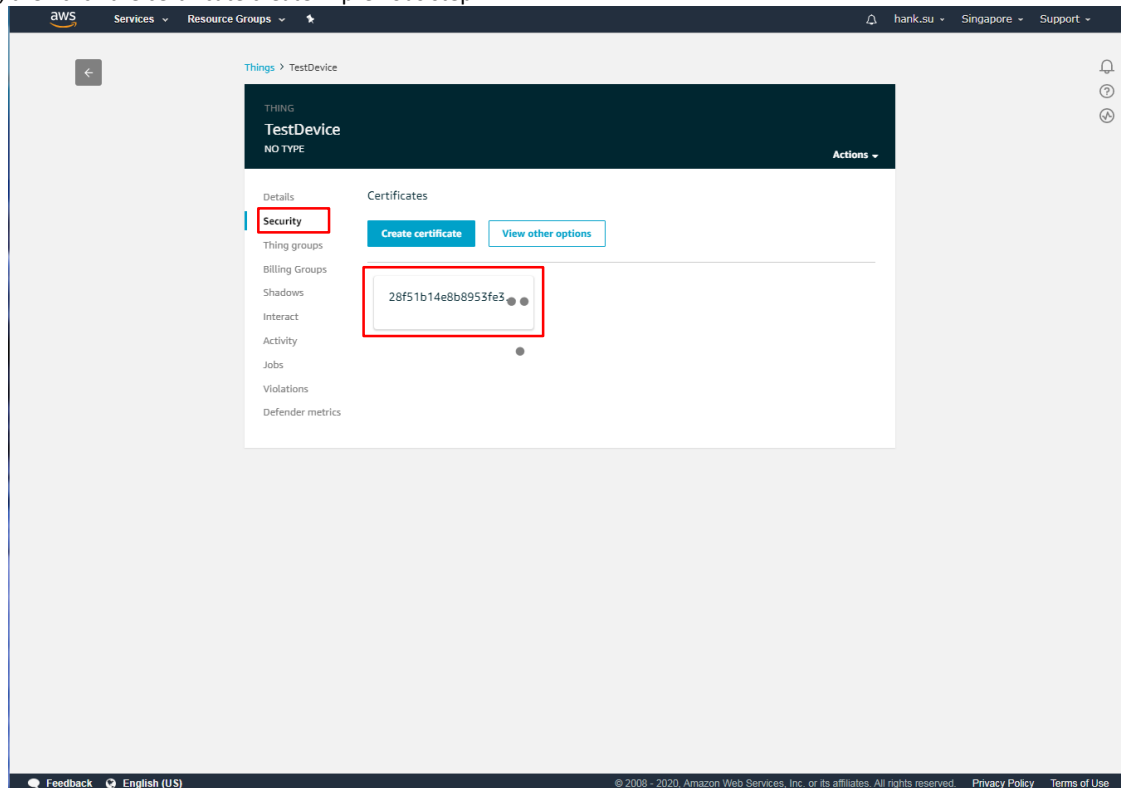
[Fleet indexing](#)

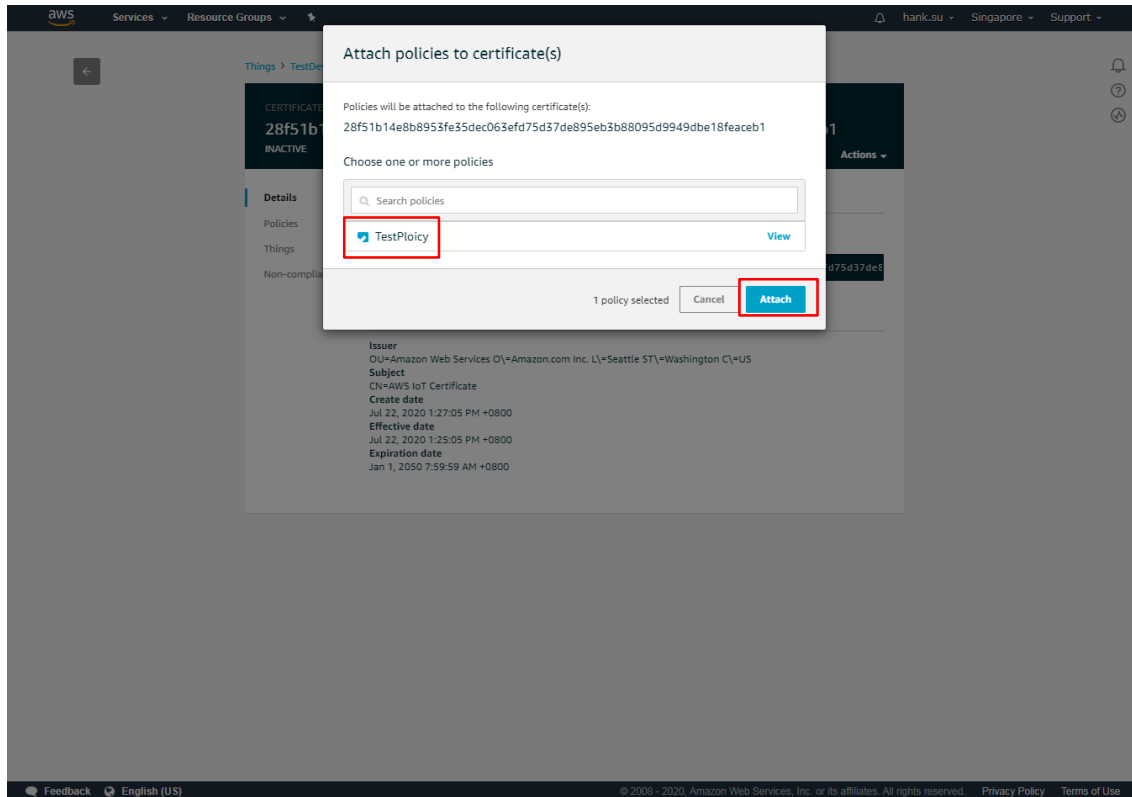
[Create](#)

[Card](#)

TestDevice
NO TYPE

Click Security, then click the certificate create in previous step.





3 Configure AmebaPro Amazon FreeRTOS

3.1 Download Source Code from Github

Open source link: https://github.com/ambiot/ambpro1_sdk and select branch: **main** to get newest source code.

The screenshot shows the GitHub interface for the repository `ambiot/ambpro1_sdk`. The `main` branch is selected and highlighted with a red box. The repository contains the following files and folders:

- `component`: [demo] modify the path of kvs demo source code (1 hour ago)
- `doc`: [amebapro] SDK first commit (3 days ago)
- `lib_amazon`: [demo] modify the path of kvs demo source code (1 hour ago)
- `project/realtek_amebapro_v0_example`: [demo] modify the path of kvs demo source code (1 hour ago)
- `tools`: [amebapro] SDK first commit (3 days ago)
- `.gitignore`: [amebapro] SDK first commit (3 days ago)
- `.gitmodules`: [demo] modify the path of kvs demo source code (1 hour ago)
- `README.md`: [amebapro] SDK first commit (3 days ago)
- `Realtek_Disclaimer-2019.pdf`: [amebapro] SDK first commit (3 days ago)
- `release_note.pdf`: [amebapro] SDK first commit (3 days ago)

The `README.md` file is open, showing the title **AmebaPro with KVS WebRTC and Producer**. The text below the title states: "This project is going to demonstrate how to use KVS Producer and WebRTC on AmebaPro" and "Amazon FreeRTOS-LTS libraries are supported on AmebaPro now, it can be used to communicate with AWS IoT core".

3.1.1 Download the Project

Run the command to download the whole project:

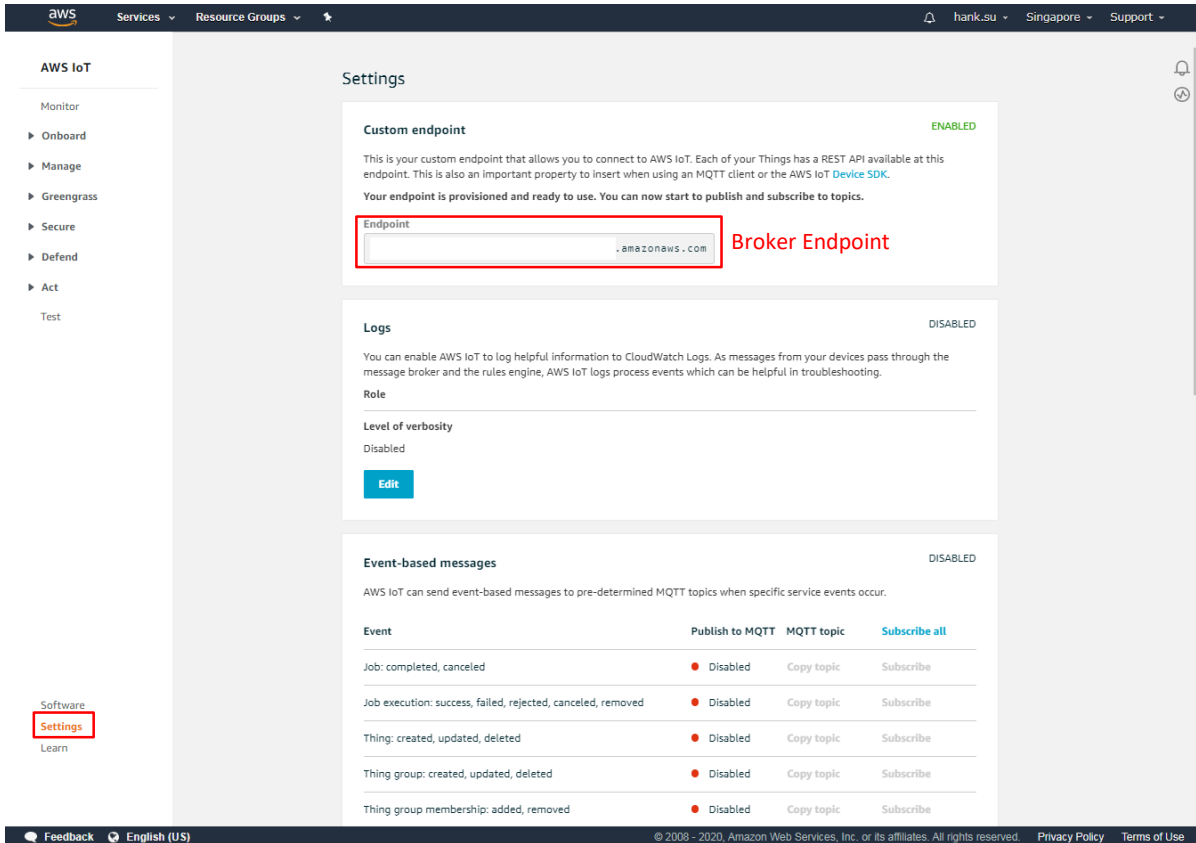
```
$ git clone --recurse-submodules https://github.com/ambiot/ambpro1_sdk.git
```

If you already have a checkout, run the following command to sync submodules recursively:

```
$ git submodule update --init --recursive
```

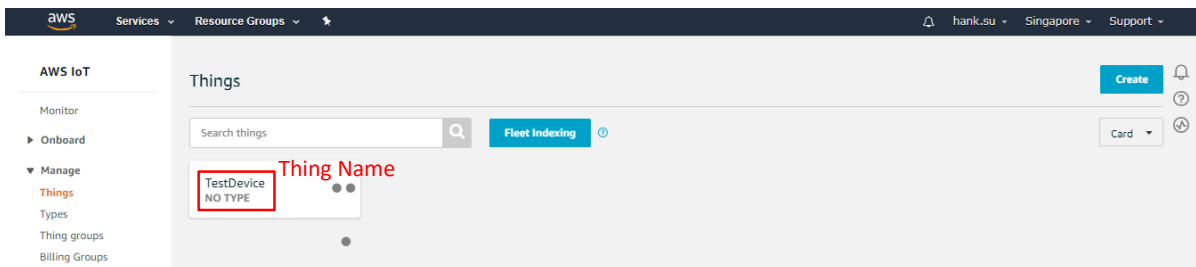
If there is GCC makefile error like: "No rule to make target ...", it may mean that some codes have not been downloaded correctly. Please run the above command again to download the missing codes.

3.2 Get Broker Endpoint by AWS IoT Core



The screenshot shows the AWS IoT Settings page. In the left sidebar, the 'Settings' link is highlighted with a red box. The main content area shows the 'Custom endpoint' section, which is also highlighted with a red box and labeled 'Broker Endpoint'. The 'Endpoint' field is visible, showing '.amazonaws.com'. The 'Logs' and 'Event-based messages' sections are also visible, both marked as 'DISABLED'.

3.3 Get Thing Name



The screenshot shows the AWS IoT Things page. A 'Thing' named 'TestDevice' is highlighted with a red box and labeled 'Thing Name'. The 'TestDevice' is listed under the 'Manage' section, with a 'NO TYPE' label. The 'Create' button is visible in the top right corner.

3.4 Setup IoT Core Information with AmebaPro Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in “[lib_amazon/amazon-freertos-202012.00/demos/include/aws_clientcredential.h](#)”

```

/*
 * @brief Host name.
 * @todo Set this to the unique name of your IoT Thing.
 */
#define clientcredentialMQTT_BROKER_ENDPOINT "xxxxxxxxxxxxx.amazonaws.com"

/*
 * @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 “[lib_amazon/amazon-freertos-202012.00/demos/include/aws_clientcredential_keys.h](#)” by xxxxxxxx-certifacte.pem and xxxxxxxx-private.pem.key.

Certificate created!

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

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

A certificate for this thing	28f51b14e8.cert.pem	Download
A public key	28f51b14e8.public.key	Download
A private key	28f51b14e8.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT [Download](#)

[Activate](#)

It can done by “[lib_amazon/amazon-freertos-202012.00/tools/certificate_configuration/CertificateConfigurator.html](#)”

Certificate Configuration Tool

FreeRTOS Developer Demos

Provide client certificate and private key PEM files downloaded from the AWS IoT Console.

Certificate PEM file:

選擇檔案 未選擇任何檔案

Private Key PEM file:

選擇檔案 未選擇任何檔案

⬇️ Generate and save `aws_clientcredential_keys.h`

 Save the generated header file to the *demos/common/include* folder of the demo project.

Copyright (C) 2017 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Final aws_clientcredential_keys.h overview.

[illegible]

3.4.2 Enable FreeRTOS demo on AmebaPro

Find `platform_opts.h` in “`project/realtek_amebapro_v0_example/inc`” and enable `CONFIG_EXAMPLE_AMAZON_FREERTOS`

```
/* For Amazon FreeRTOS LTS demo example */
#define CONFIG_EXAMPLE_AMAZON_FREERTOS 1
```

For example, if you would like to run MQTT mutual authentication demo, please find `aws_demo_config.h` in “`lib_amazon/amazon-freertos-202012.00/vendors/realtek/boards/amebaPro/aws_demos/config_files`” and enable `CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED`

```
/* To run a particular demo you need to define one of these.
 * Only one demo can be configured at a time
 *
 *
 * CONFIG_CORE_HTTP_MUTUAL_AUTH_DEMO_ENABLED
 * CONFIG_CORE_HTTP_S3_DOWNLOAD_DEMO_ENABLED
 * CONFIG_CORE_HTTP_S3_DOWNLOAD_MULTITHREADED_DEMO_ENABLED
 * CONFIG_CORE_HTTP_S3_UPLOAD_DEMO_ENABLED
 * CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
 * CONFIG_CORE_MQTT_CONNECTION_SHARING_DEMO_ENABLED
 * CONFIG_DEVICE_SHADOW_DEMO_ENABLED
 * CONFIG_DEVICE_DEFENDER_DEMO_ENABLED
 * CONFIG_JOBS_DEMO_ENABLED
 * CONFIG_MQTT_BLE_DEMO_ENABLED
 * CONFIG_GREENGRASS_DISCOVERY_DEMO_ENABLED
 * CONFIG_TCP_ECHO_CLIENT_DEMO_ENABLED
 * CONFIG_POSIX_DEMO_ENABLED
 * CONFIG_OTA_UPDATE_DEMO_ENABLED
 * CONFIG_BLE_GATT_SERVER_DEMO_ENABLED
 * CONFIG_BLE_NUMERIC_COMPARISON_DEMO_ENABLED
 *
 * These defines are used in iot_demo_runner.h for demo selection */

#define CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
// #define CONFIG_OTA_UPDATE_DEMO_ENABLED
// #define CONFIG_DEVICE_SHADOW_DEMO_ENABLED
// #define CONFIG_CORE_HTTP_S3_UPLOAD_DEMO_ENABLED
// #define CONFIG_CORE_HTTP_MUTUAL_AUTH_DEMO_ENABLED
```

Now you can start to compile AmebaPro Amazon FreeRTOS project !

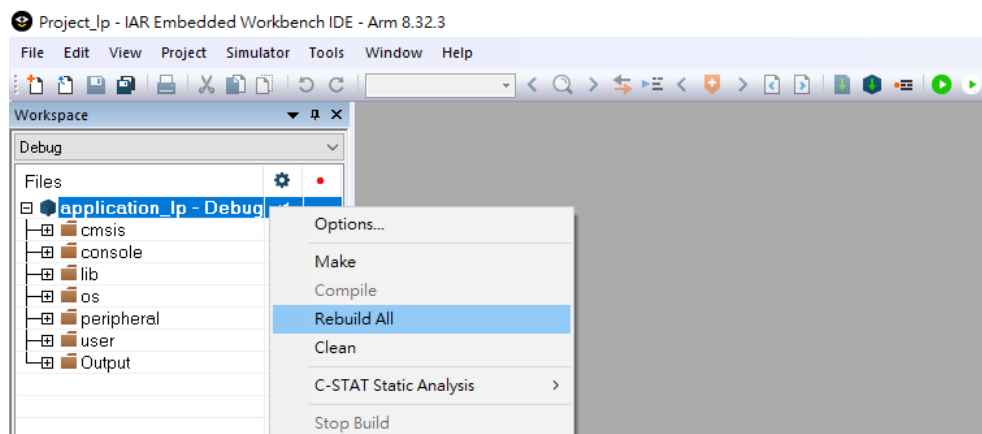
4 Compile AmebaPro Amazon FreeRTOS

4.1 IAR Embedded Workbench Build Environment Setup

AmebaPro use the newest Big-Little architecture. Since the big CPU will depend on the setting of small CPU, **it is necessary to compile the small CPU before the big CPU.**

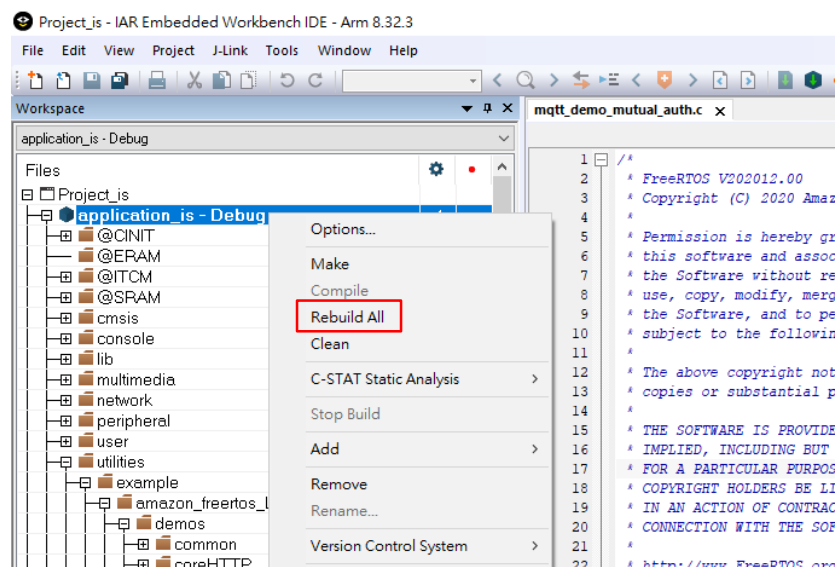
4.1.1 Compile Little CPU

- step 1. Open SDK/project/realtek_amebapro_v0_example/EWARMRELEASE/Project_lp.ewww.
- step 2. Confirm application_lp in WorkSpace, right click application_lp and choose "Rebuild All" to compile.
- step 3. Make sure there is no error after compile.



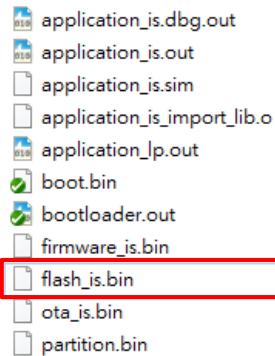
4.1.2 Compile Big CPU

- step 1. Open SDK/project/realtek_amebapro_v0_example/EWARMRELEASE/Project_is.ewww.
- step 2. Confirm application_is in WorkSpace, right click application_is and choose "Rebuild All" to compile.
- step 3. Make sure there is no error after compile.



4.1.3 Generating Image (Bin)

After compile, the images partition.bin, boot.bin, firmware_is.bin and flash_is.bin can be seen in the **EWARM-RELEASE\Debug\Exe**. flash_is.bin links partition.bin, boot.bin and firmware_is.bin. Users need to choose **flash_is.bin** when downloading the image to board by Image Tool.



4.2 Compile Program with GCC Toolchain

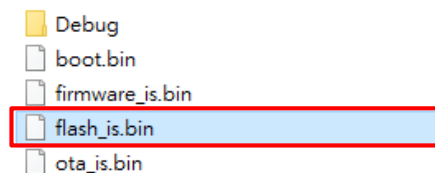
If using Linux environment or Cygwin on windows, follow the instructions below to build the project

```
$ cd project/realtek_amebapro_v0_example/GCC-RELEASE
```

Build the library and the example by running make in the directory

```
$ make all
```

If somehow it built failed, you can try to type **\$ make clean** and then redo the make procedure. After successfully build, there should be a directory named "application_is" created under GCC-RELEASE/ directory. The image file **flash_is.bin** is located in "application_is" directory.



Note:

If there is compile error with shell script in "component/soc/realtek/8195b/misc/gcc_utility/", you may need to run following command

```
$ dos2unix component/soc/realtek/8195b/misc/gcc_utility/*
```


5 Image Tool

The tool **ImageTool.exe** can be find in **project\tools\AmebaPro\Image_Tool\ImageTool.exe**

5.1 Introduction

As show in the following figure, Image Tool has two tab pages:

- Download: used as image download server to transmit images to AmebaPro through UART
- Generate: concat separate images and generate a final image

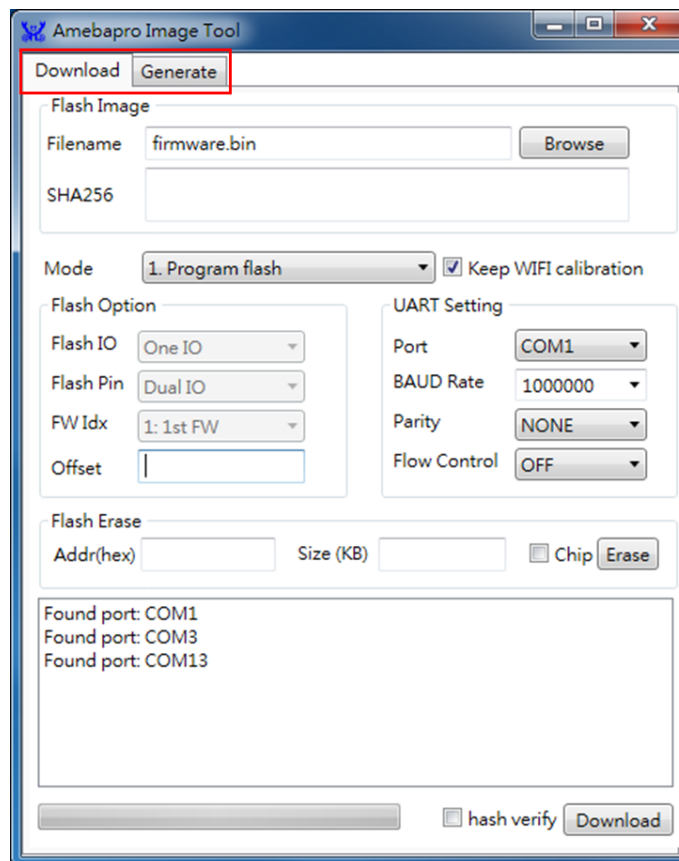


Fig 5-1 ImageTool UI

5.2 Environment Setup

5.2.1 Hardware Setup

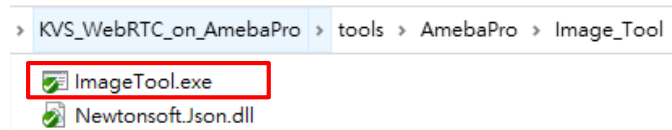
The hardware setup is shown in Fig 5-2.



Fig 5-2 Hardware setup

5.2.2 Software Setup

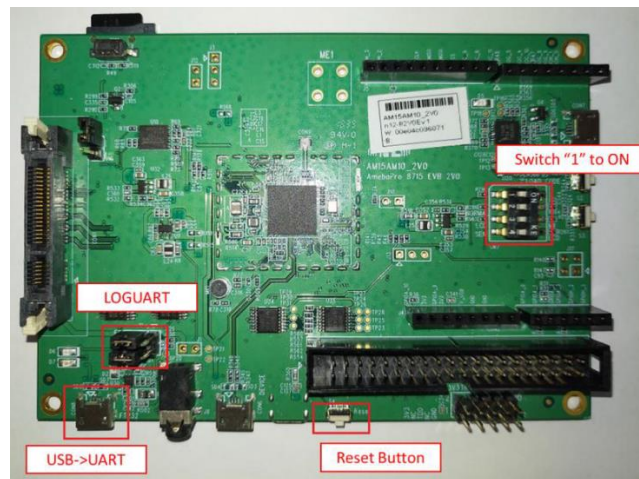
Execute **ImageTool.exe** from location **project\tools\AmebaPro\Image_Tool\ImageTool.exe**



5.3 Download

5.3.1 Enter the Download Mode to Ready

Image tool use UART to transmit image to AmebaPro board. Before performing image download function, AmebaPro need to enter UART_DOWNLOAD mode first. Please follow below steps to get AmebaPro into UART_DOWNLOAD mode:



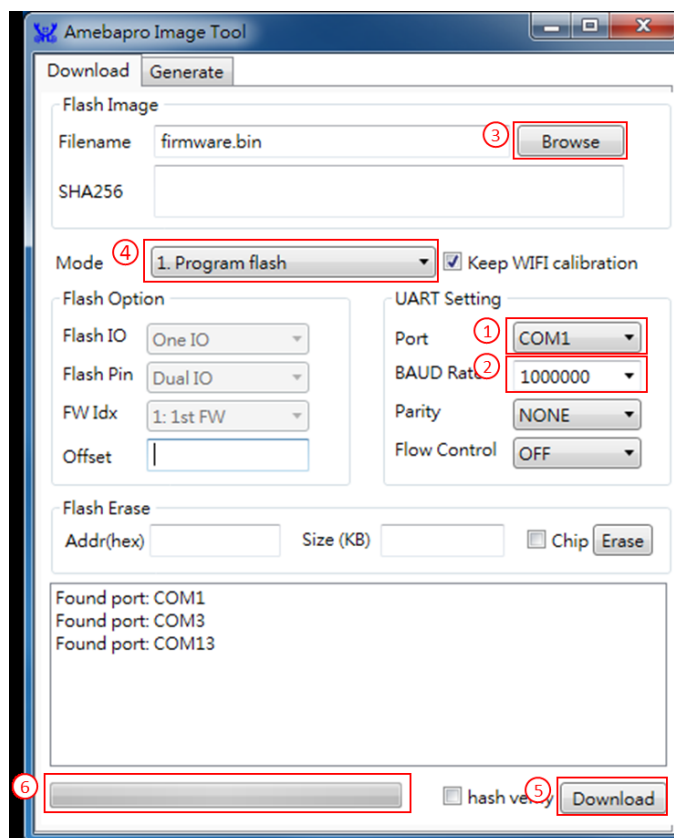
- step 1. Connect LOGUART with FT pin by jumper cap.
- step 2. Connect USB->UART to PC by using micro-USB wire.
- step 3. Switch "1" to ON from SW7(2V0 、 2V1) or Switch "2" to ON from SW7(1V0)
- step 4. Push reset button.

5.3.2 Download the Image to Flash

To download image through Image Tool, device need to enter UART_DOWNLOAD mode first.

Steps to download flash are as following:

- step 1. Application will scan available UART ports. Please choose correct UART port. Please close other UART connection for the target UART port.
- step 2. Choose desired baud rate between computer and AmebaPro.
- step 3. Choose target flash binary image file "flash_xx.bin"
- step 4. Check Mode is "1. Program flash"
- step 5. Click "Download"
- step 6. Progress will be shown on progress bar and result will be shown after download finish.
- step 7. Switch "1" to OFF from SW7(2V0、2V1) or Switch "2" to OFF from SW7(1V0)
- step 8. Push reset button to start the program.



6 MQTT Demo

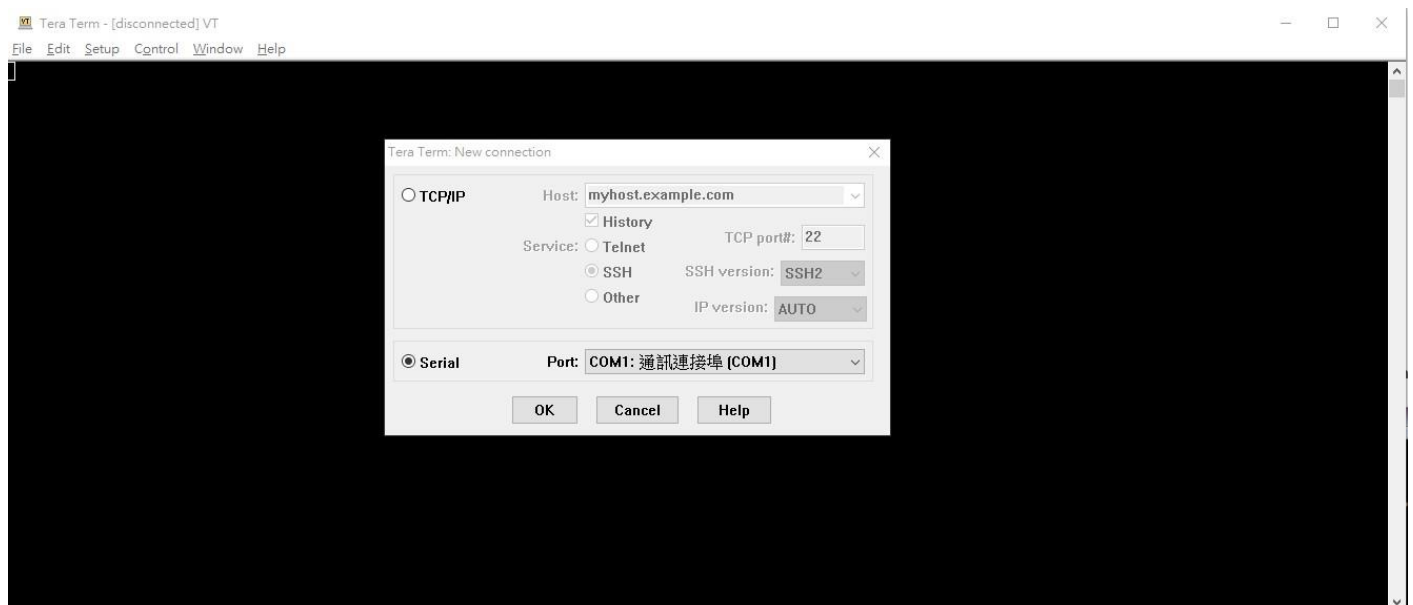
6.1 Get Device Log

Install **Tera Term** or other terminal emulator to get device log



Fig 6-1 Hardware setup

The serial port is same with ImageTool that get from 5.3.2 or use device manager to get the right serial port of device.



6.2 Run MQTT Demo

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

```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help

Initializing WIFI ...
RTL871X: [HALMAC]11692M
        HALMAC_MAJOR_VER = 1
        HALMAC_PROTOTYPE_VER = 4
        HALMAC_MINOR_VER = 14
        HALMAC_PATCH_VER = 0

RTL871X: efuse autoload en: 1

RTL871X: rfe_type=0x0, map=0x0

RTL871X: Download fw addr:9819ad80, size:115560
        - download_firmware_88xx HALMAC_RET_SUCCESS

RTL871X: fw: 24.4

RTL871X: RFE type = 0x0

Start LOG SERVICE MODE

# start_addr=(0x4000), end_addr=(0x8000), buffer_size=(0x4000), smp_number_max=(2048)

[Driver]: set ssid [lukai]
        0 175 [example_a] Write certificate...
1 264 [iot_threa] [INFO ][DEMO][264] -----STARTING DEMO-----
2 271 [iot_threa] [INFO ][INIT][271] SDK successfully initialized.

L
```

```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help

16 6811 [iot_threa] [INFO ][MQTT][core_mqtt_serializer.c:912] 17 6815 [iot_threa] Connection accepted.18 6820 [iot_threa]
19 6820 [iot_threa] [INFO ][MQTT][core_mqtt.c:1563] 20 6825 [iot_threa] Received MQTT CONNACK successfully from broker.21
6832 [iot_threa]
22 6833 [iot_threa] [INFO ][MQTT][core_mqtt.c:1829] 23 6838 [iot_threa] MQTT connection established with the broker.24 68
43 [iot_threa]
25 6846 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:820] 26 6852 [iot_threa] An MQTT connection is
established with a2zweh2b7yb784-ats.iot.ap-southeast-1.amazonaws.com.27 6862 [iot_threa]
28 6863 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:885] 29 6871 [iot_threa] Attempt to subscribe t
o the MQTT topic ameba-ota/example/topic.30 6877 [iot_threa]
31 6880 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:899] 32 6886 [iot_threa] SUBSCRIBE sent for to
pic ameba-ota/example/topic to broker.33 6895 [iot_threa]
34 7006 [iot_threa] [INFO ][MQTT][core_mqtt.c:886] 35 7010 [iot_threa] Packet received. ReceivedBytes=3.36 7014 [iot_threa]
a]
37 7017 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:1053] 38 7023 [iot_threa] Subscribed to the top
ic ameba-ota/example/topic with maximum QoS 1.39 7032 [iot_threa]
40 8032 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:533] 41 8037 [iot_threa] Publish to the MQTT to
pic ameba-ota/example/topic.42 8044 [iot_threa]
43 8047 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:543] 44 8054 [iot_threa] Attempt to receive pub
lish message from broker.45 8058 [iot_threa]
46 8284 [iot_threa] [INFO ][MQTT][core_mqtt.c:886] 47 8287 [iot_threa] Packet received. ReceivedBytes=2.48 8292 [iot_threa]
a]
49 8293 [iot_threa] [INFO ][MQTT][core_mqtt.c:1162] 50 8299 [iot_threa] Ack packet deserialized with result: MQTTSuccess.
51 8304 [iot_threa]
52 8307 [iot_threa] [INFO ][MQTT][core_mqtt.c:1175] 53 8311 [iot_threa] State record updated. New state=MQTTPublishDone.5
4 8317 [iot_threa]
55 8318 [iot_threa] [INFO ][MQTT_MutualAuth_Demo][mqtt_demo_mutual_auth.c:1031] 56 8326 [iot_threa] PUBACK received for p
acket Id 2.57 8330 [iot_threa]
58 8333 [iot_threa] [INFO ][MQTT][core_mqtt.c:886] 59 8336 [iot_threa] Packet received. ReceivedBytes=39.60 8342 [iot_threa]
ea]
```

```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help
Topic Name: ameba-ota/example/topic matches subscribed topic.Incoming Publish Message : Hello World!698 68052 [iot_threa]
699 68554 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:553] 700 68559 [iot_threa] Keeping Connection
Idle...701 68565 [iot_threa]
702 70566 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:561] 703 70571 [iot_threa] Unsubscribe from t
he MQTT topic ameba-ota/example/topic.704 70579 [iot_threa]
705 70707 [iot_threa] [INFO] [MQTT] [core_mqtt.c:886] 706 70710 [iot_threa] Packet received. ReceivedBytes=2.707 70716 [io
t_threa]
708 70717 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:1062] 709 70725 [iot_threa] Unsubscribed from
the topic ameba-ota/example/topic.710 70730 [iot_threa]
711 71733 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:583] 712 71738 [iot_threa] Disconnecting the
MQTT connection with a2zweh2b7yb784-ats.iot.ap-southeast-1.amazonaws.com.713 71749 [iot_threa]
714 71752 [iot_threa] [INFO] [MQTT] [core_mqtt.c:2149] 715 71756 [iot_threa] Disconnected from the broker.716 71760 [iot_t
hrea]
717 71763 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:612] 718 71770 [iot_threa] Demo completed an
iteration successfully.719 71775 [iot_threa]
720 71778 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:613] 721 71784 [iot_threa] Demo iteration 3 c
ompleted successfully.722 71790 [iot_threa]
723 71791 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:625] 724 71799 [iot_threa] Short delay before
starting the next iteration.... 725 71805 [iot_threa]
726 76808 [iot_threa] [INFO] [MQTT_MutualAuth_Demo] [mqtt_demo_mutual_auth.c:636] 727 76813 [iot_threa] Demo run is succes
sful with 3 successful loops out of total 3 loops.728 76822 [iot_threa]
729 77823 [iot_threa] [INFO] [DEMO][77823] Demo completed successfully.

LwIP DHCP: dhcp stop.
Deinitializing WIFI ...
[deinit_timer_wrapper] Need to delete 1 timer entry
WIFI deinitialized730 78085 [iot_threa] [INFO] [INIT][78085] SDK cleanup done.
731 78090 [iot_threa] [INFO] [DEMO][78090] -----DEMO FINISHED-----
```

Monitor connection summary.

6.3 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.

AWS IoT

Monitor

Activity

Onboard

Manage

Greengrass

Secure

Defend

Act

Test

Software

Settings

Learn

Documentation

New console experience

Tell us what you think

AWS IoT > MQTT test client

MQTT test client

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.

Subscribe to a topic

Publish to a topic

Topic filter

+ /example/topic

Additional configuration

Subscribe

Subscriptions

Topic

You have no topic subscriptions.

Subscribe to a topic to view incoming messages.

AWS IoT

Monitor

Activity

Onboard

Manage

Greengrass

Secure

Defend

Act

Test

Software

Settings

Learn

Documentation

New console experience

Tell us what you think

Subscriptions

+ /example/topic

Pause

Clear

Export

Edit

+ /example/topic

▼ 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 ec2sdsigner.key and ec2sdsigner.crt by openssl
you can create the key and certification by running:
\$ sudo openssl ecparam -name prime256v1 -genkey -out ec2sdsigner.key.pem
\$ sudo openssl req -new -x509 -days 3650 -key ec2sdsigner.key.pem -out ec2sdsigner.crt.pem
- ...
- Step 6. Add certificate pem(ec2sdsigner.crt.pem) into lib_amazon/amazon-freertos-202012.00/demos/include/aws_ota_codesigner_certificate.h

```
#ifndef __AWS_CODESIGN_KEYS_H__
#define __AWS_CODESIGN_KEYS_H__

/*
 * PEM-encoded code signer certificate
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"
 * "...base64 data...\n"
 * "-----END CERTIFICATE-----\n";
 */
static const char signingcredentialSIGNING_CERTIFICATE_PEM[] =
"-----BEGIN CERTIFICATE-----\n"
[REDACTED]
"-----END CERTIFICATE-----";

#endif
```

7.2 Set the Firmware Version and App Version to Image File

The serial number in "amebapro_firmware_is.json" decide the firmware version of AmebaPro:

```
{
  "ISP": {
    "source": "binary",
    "header": {
      "type": "ISP",
      "serial": 100
    },
    "FST": {
      "validpat": "0001020304050607",
      "hash_en": true
    },
    "list": ["sram"],
    "sram": {
      "file": "../../../component/soc/realtek/8195b/misc/bsp/image/isp.bin"
    }
  }
}
```

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

```
#ifndef _AWS_APPLICATION_VERSION_H_
#define _AWS_APPLICATION_VERSION_H_

#include "iot_appversion32.h"
extern const AppVersion32_t xAppFirmwareVersion;

#define APP_VERSION_MAJOR    100
#define APP_VERSION_MINOR    9
#define APP_VERSION_BUILD    2

#endif
```

Usually, the APP_VERSION_MAJOR in “aws_application_version.h” and FW version (serial number) in “amebapro_firmware_is.json” must same.

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 AmebaPro, and wait the OTA job coming.
- Another image with newer version will be uploaded to S3 bucket. Then, create a new job for OTA.

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 How Custom Signed Image File is Created

We use custom signing feature provided by amazon to manually sign the OTA binary and attach the signatures along with the **firmware_is.bin**:

1. The firmware_is.bin is manually signed using the ECDSA P-256 key provided by user.
2. The ECDSA signatures are then appended to the end of the firmware_is_sig.bin with signature sizes.
3. The signatures are received as a separate packet and formatted accordingly to verify the OTA image which was updated.

The custom signing process is executed by a python script – **python_custom_ecdsa_Pro.py**, that provided in the folder (component\common\application\amazon\amazon_ota_tools\python_custom_ecdsa_Pro.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.

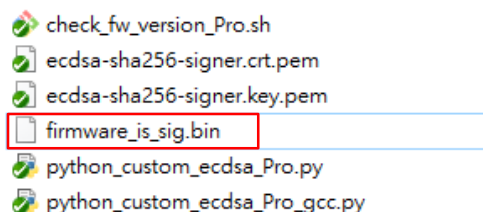
(**Note:** The key pair in SDK are just for example, please generated new key by openssl !)

Run the python script in folder: component\common\application\amazon\amazon_ota_tools

- cmd command after IAR build : **\$ python python_custom_ecdsa_Pro.py**
- shell command after GCC build : **\$ python3 python_custom_ecdsa_Pro_gcc.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 custom signed binary named **firmware_is_sig.bin** inside the component\common\application\amazon\amazon_ota_tools folder.



After getting the custom signed **firmware_is_sig.bin**, you can upload it to the 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?p=icr&cp=bn&ad=c>. Then, follow the following steps to create an AWS OTA task for AmebaPro:

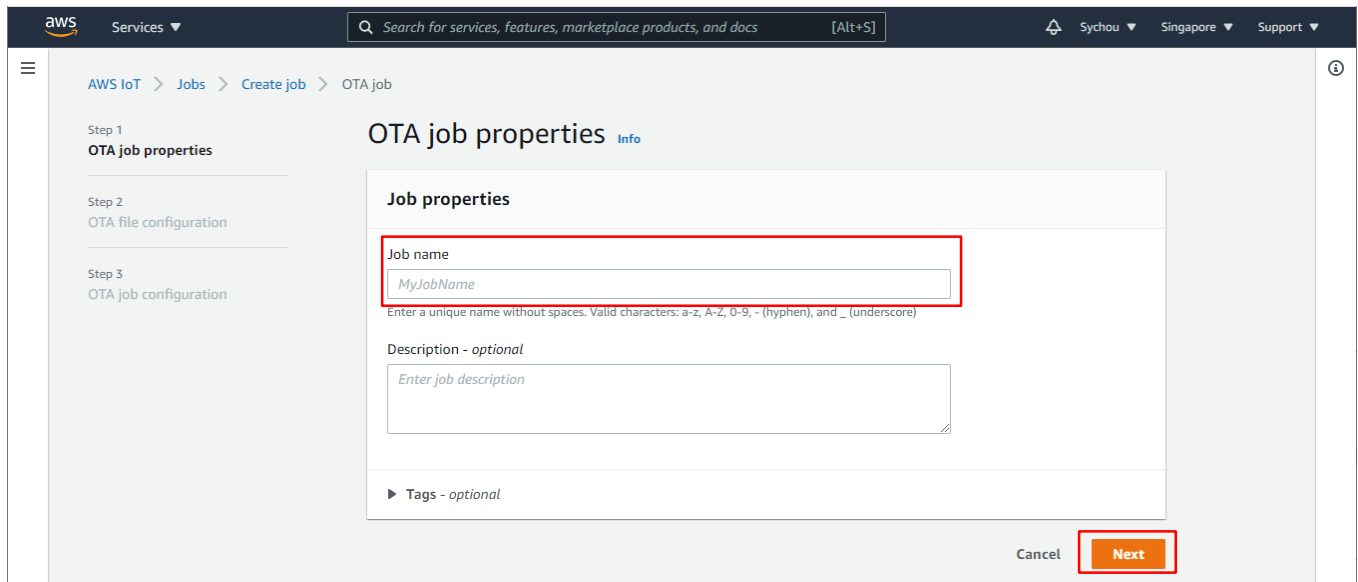
Step 1. Click on 'Create OTA update job', select your job type and then click next.

The first screenshot shows the AWS IoT console home page. In the left-hand navigation menu, the 'Jobs' option is highlighted with a red rectangle.

The second screenshot shows the 'Jobs' page. A table lists existing jobs, with one job named 'AFR_OTA-Pro_OTA_test_0615_1' shown as 'Completed'. A red rectangle highlights the 'Create job' button in the top right corner.

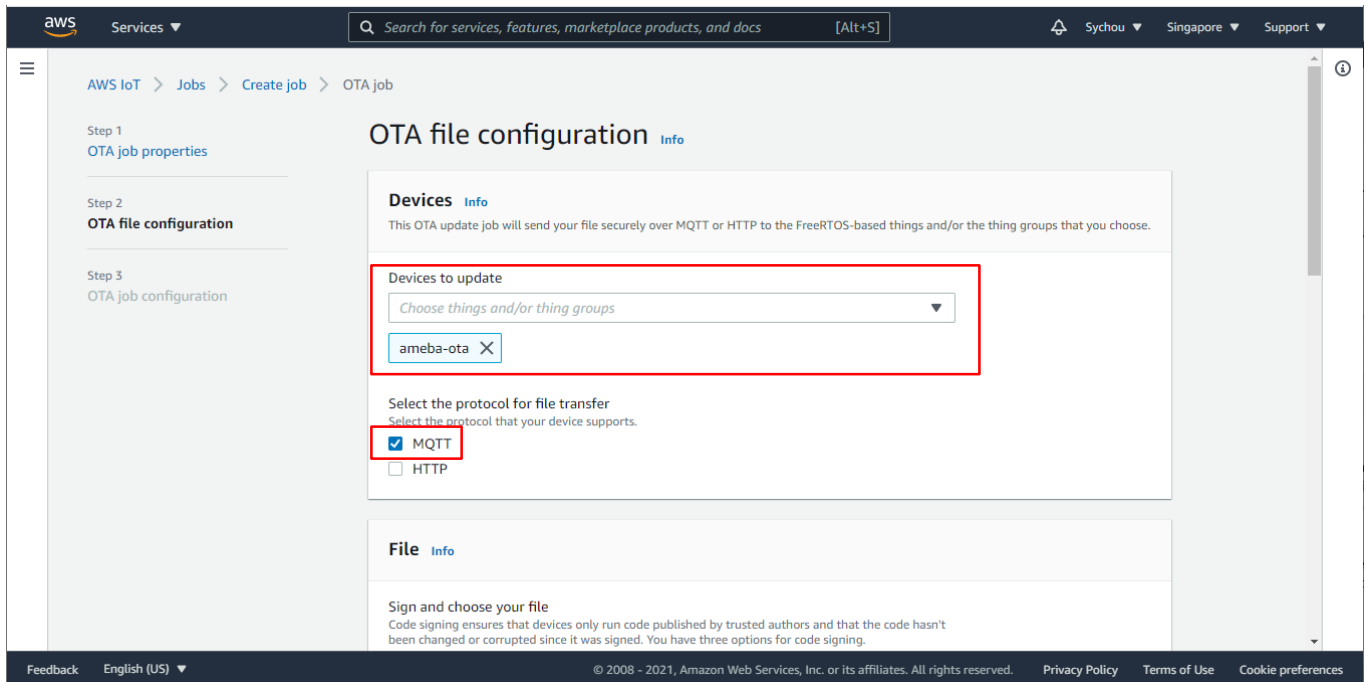
The third screenshot shows the 'Create job' wizard. Under the 'Job type' section, the 'Create FreeRTOS OTA update job' option is selected and highlighted with a red rectangle. At the bottom right, the 'Next' button is also highlighted with a red rectangle.

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



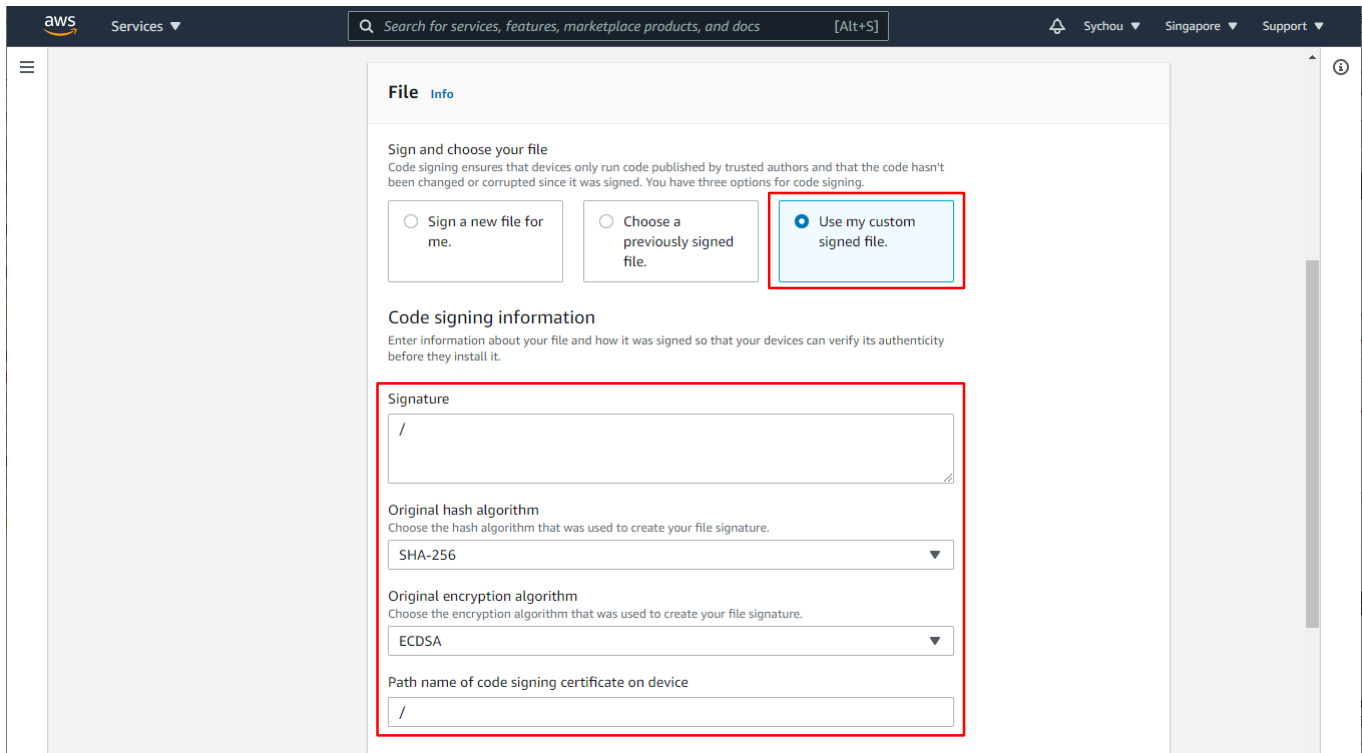
The screenshot shows the 'OTA job properties' page in the AWS IoT console. The left sidebar indicates the current step is 'Step 1: OTA job properties'. The main content area has a heading 'OTA job properties' and a sub-heading 'Job properties'. Below this, there is a 'Job name' field with the text 'MyJobName' entered. A red box highlights this field. Below the 'Job name' field is a 'Description - optional' text area. At the bottom right, there are 'Cancel' and 'Next' buttons, with the 'Next' button highlighted by a red box.

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



The screenshot shows the 'OTA file configuration' page in the AWS IoT console. The left sidebar indicates the current step is 'Step 2: OTA file configuration'. The main content area has a heading 'OTA file configuration' and a sub-heading 'Devices'. Below this, there is a 'Devices to update' dropdown menu with the text 'Choose things and/or thing groups' and 'ameba-ota' selected. A red box highlights this dropdown. Below the dropdown is a section 'Select the protocol for file transfer' with two options: 'MQTT' (checked) and 'HTTP'. The 'MQTT' checkbox is highlighted with a red box. At the bottom, there is a 'File' section with a heading 'Sign and choose your file'.

- Step 4. In the following page, choose the option 'Use my custom signed firmware image'.
In the signature field just enter '/'. Choose hash algorithm as 'SHA-256'. Choose encryption algorithm as 'ECDSA'.
In "pathname of code signing certificate", enter '/'



File Info

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.

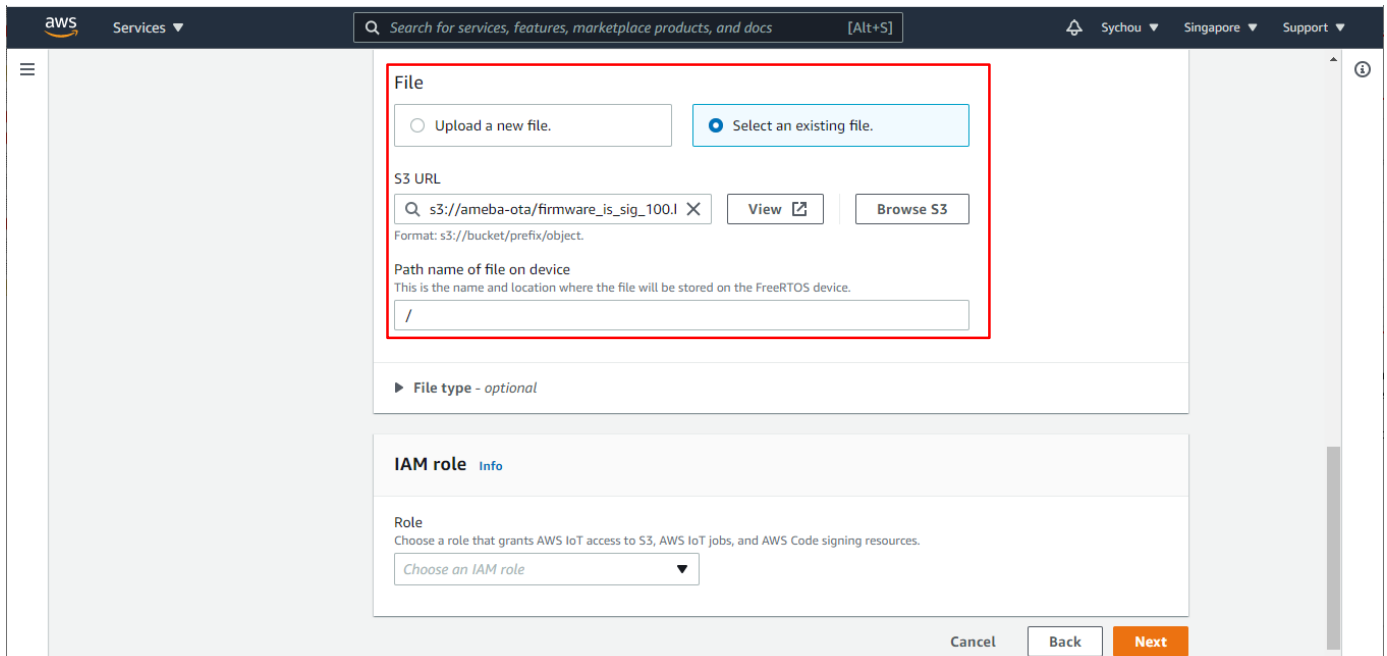
Signature
/

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-ota/firmware_is_sig_100.I View Browse S3
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.
/

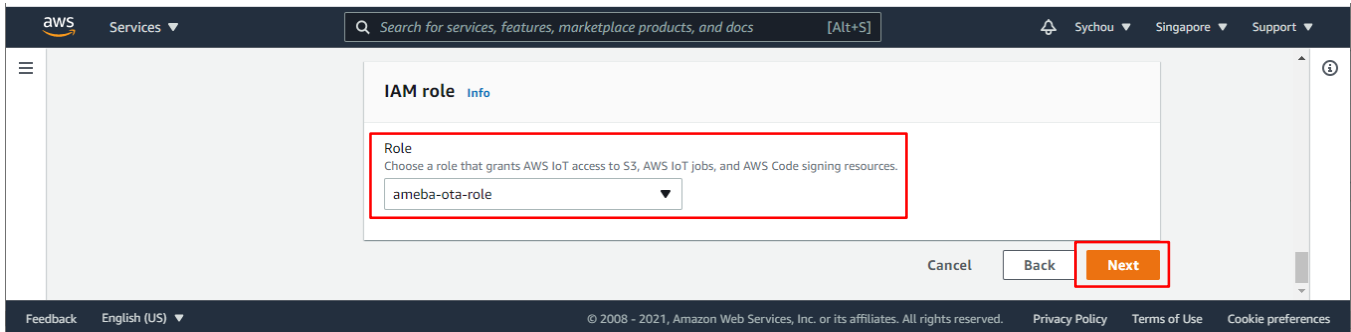
File type - optional

IAM role Info

Role
Choose a role that grants AWS IoT access to S3, AWS IoT jobs, and AWS Code signing resources.
Choose an IAM role

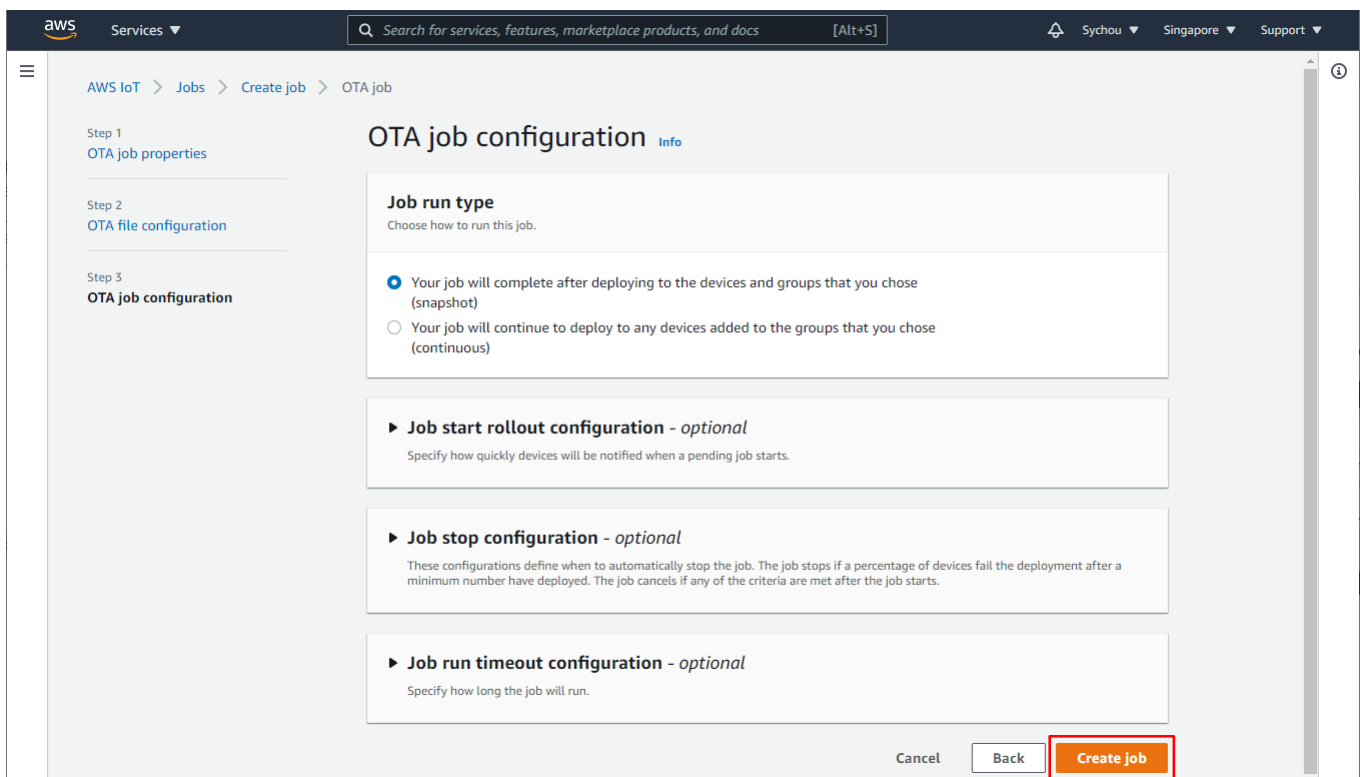
Cancel Back Next

Step 6. Choose the IAM role for OTA update job. (This is the same IAM role as any OTA update job)

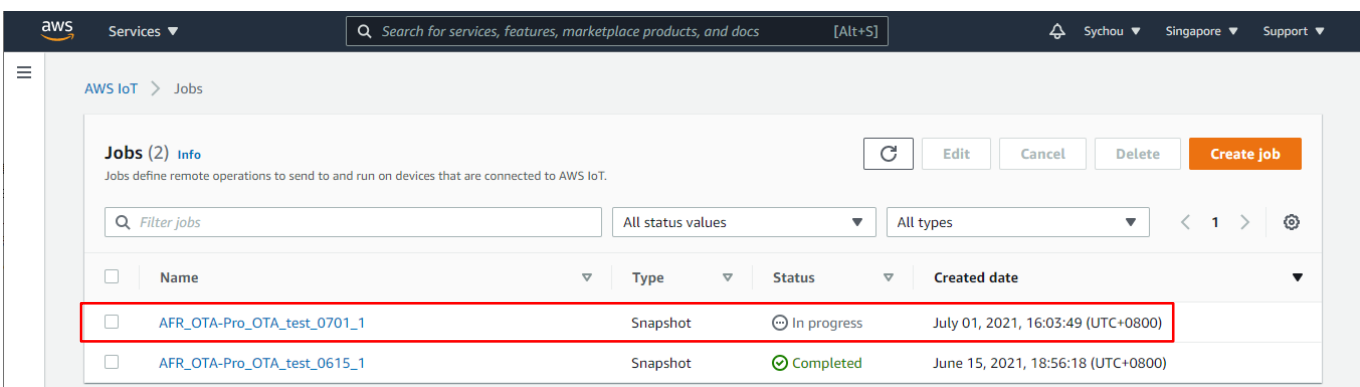


The screenshot shows the AWS IAM console 'IAM role' configuration page. The 'Role' dropdown menu is set to 'ameba-ota-role'. The 'Next' button is highlighted with a red box.

Step 7. Click next, and create your OTA job.



The screenshot shows the AWS IoT console 'OTA job configuration' page. The 'Job run type' is set to 'Your job will complete after deploying to the devices and groups that you chose (snapshot)'. The 'Create job' button is highlighted with a red box.



The screenshot shows the AWS IoT console 'Jobs (2)' list. The table lists two jobs: 'AFR_OTA-Pro_OTA_test_0701_1' (In progress) and 'AFR_OTA-Pro_OTA_test_0615_1' (Completed). The first job is highlighted with a red box.

Name	Type	Status	Created date
AFR_OTA-Pro_OTA_test_0701_1	Snapshot	In progress	July 01, 2021, 16:03:49 (UTC+0800)
AFR_OTA-Pro_OTA_test_0615_1	Snapshot	Completed	June 15, 2021, 18:56:18 (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 AmebaPro to request the update.

Next, download the image file with older version number to AmebaPro 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:

```
wlan_wrtie_reconnect_data_to_flash():not the same ssid/passphrase/channel/offer_ip, write new profile to flash
Interface 0 IP address : 192.168.0.143 5568 [iot_threa] [INFO ][DEMO][5568] Successfully initialized the demo. Network type for the demo: 1
4 5578 [iot_threa] [INFO ][MQTT][5577] MQTT library successfully initialized.
5 5583 [iot_threa] [INFO ][DEMO][5583] OTA demo version 99.9.2
6 5590 [iot_threa] [INFO ][DEMO][5590] Connecting to broker...
7 5595 [iot_threa] [INFO ][DEMO][5595] MQTT demo client identifier is ameba-ota (length 9).
8 6947 [iot_threa] [INFO ][MQTT][6947] Establishing new MQTT connection.
9 6955 [iot_threa] [INFO ][MQTT][6955] (MQTT connection 7026ef40, CONNECT operation 7026f0a0) Waiting for operation completion.
10 7090 [NetRecv] [INFO ][MQTT] [core_mqtt_serializer.c:970] 11 7094 [NetRecv] CONNACK session present bit not set.12 7099 [NetRecv]
13 7101 [NetRecv] [INFO ][MQTT] [core_mqtt_serializer.c:912] 14 7106 [NetRecv] Connection accepted.15 7109 [NetRecv]
16 7111 [iot_threa] [INFO ][MQTT][7111] (MQTT connection 7026ef40, CONNECT operation 7026f0a0) Wait complete with result SUCCESS.
17 7124 [iot_threa] [INFO ][MQTT][7124] New MQTT connection 20110f50 established.
18 7131 [iot_threa] [OTA AgentInit_internal] OTA Task is Ready.
19 7135 [OTA Agent] [prvPAL_GetPlatformImageState_amebaPro] Image current state (0x02)
20 7143 [OTA Agent] [prvExecuteHandler] Called handler. Current State [Ready] Event [Start] New state [RequestingJob]
21 7154 [OTA Agent] [INFO ][MQTT][7154] (MQTT connection 7026ef40) SUBSCRIBE operation scheduled.
22 7162 [OTA Agent] [INFO ][MQTT][7162] (MQTT connection 7026ef40, SUBSCRIBE operation 7026f1e0) Waiting for operation completion.
23 7282 [OTA Agent] [INFO ][MQTT][7282] (MQTT connection 7026ef40, SUBSCRIBE operation 7026f1e0) Wait complete with result SUCCESS.
24 7293 [OTA Agent] [prvSubscribeToJobNotificationTopics] OK: $aws/things/ameba-ota/jobs/$next/get/accepted
25 7303 [OTA Agent] [INFO ][MQTT][7303] (MQTT connection 7026ef40) SUBSCRIBE operation scheduled.
26 7311 [OTA Agent] [INFO ][MQTT][7311] (MQTT connection 7026ef40, SUBSCRIBE operation 7026f1e0) Waiting for operation completion.
27 7430 [OTA Agent] [INFO ][MQTT][7430] (MQTT connection 7026ef40, SUBSCRIBE operation 7026f1e0) Wait complete with result SUCCESS.
28 7440 [OTA Agent] [prvSubscribeToJobNotificationTopics] OK: $aws/things/ameba-ota/jobs/notify-next
29 7449 [OTA Agent] [prvRequestJob_Mqtt] Request #0
30 7455 [OTA Agent] [INFO ][MQTT][7455] (MQTT connection 7026ef40) MQTT PUBLISH operation queued.
31 7462 [OTA Agent] [INFO ][MQTT][7462] (MQTT connection 7026ef40, PUBLISH operation 7026f1e0) Waiting for operation completion.
32 7702 [OTA Agent] [INFO ][MQTT][7702] (MQTT connection 7026ef40, PUBLISH operation 7026f1e0) Wait complete with result SUCCESS.
33 7712 [OTA Agent] [prvExecuteHandler] Called handler. Current State [RequestingJob] Event [RequestJobDocument] New state [WaitingForJob]
34 7725 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ clientToken: 0:ameba-ota ]
35 7733 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ timestamp: 1625133128 ]
36 7740 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ jobId: AFR_OTA-Pro_OTA_test_0701_4 ]
37 7749 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ protocols: ["MQTT"] ]
38 7757 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ streamname: AFR_OTA-d7c5e0c5-6d12-45c3-8fe4-7167753724dc ]
39 7767 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ filepath: / ]
40 7774 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ filesize: 1712200 ]
41 7782 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ fileId: 0 ]
42 7788 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ certfile: / ]
43 7795 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: /"}]]}}... ]
44 7804 [OTA Agent] [prvParseJobDoc] Job was accepted. Attempting to start transfer.
```

We can see that the OTA process start!

```
60 8174 [iot_threa] [INFO ][DEMO][8174] State: RequestingJob Received: 1 Queued: 0 Processed: 0 Dropped: 0
61 8427 [OTA Agent] [prvIngestDataBlock] Received file block 5, size 1024
[update_ota_prepare_addr] Get loaded_fw_idx 1
[update_ota_prepare_addr] NewFWAddr 00240000
[update_ota_erase_upg_region] NewFWLen 1712128
[update_ota_erase_upg_region] NewFWBlkSize 418 0x1a2[OTA][prvPAL_WriteBlock_amebaPro] NewImg2BlkSize 418
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0x1400: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x241400 (0x241400)
62 23029 [OTA Agent] [prvIngestDataBlock] Remaining: 1672
63 23033 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
64 23048 [OTA Agent] [INFO ][MQTT][23048] (MQTT connection 7026ef40) MQTT PUBLISH operation queued.
65 23055 [OTA Agent] [prvRequestFileBlock_Mqtt] OK: $aws/things/ameba-ota/streams/AFR_OTA-d7c5e0c5-6d12-45c3-8fe4-7167753724dc/get/cbor
66 23067 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [RequestTimer] New state [WaitingForFileBlock]
67 23081 [OTA Agent] [prvIngestDataBlock] Received file block 4, size 1024
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0x1000: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x241000 (0x241000)
68 23101 [OTA Agent] [prvIngestDataBlock] Remaining: 1671
69 23105 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
70 23120 [OTA Agent] [prvIngestDataBlock] Received file block 3, size 1024
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0xc00: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x240c00 (0x240c00)
```

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.

```
[OTA][prvPAL_WriteBlock_amebaPro] iFileSize 1712200, iOffset: 0x1a2000: iBlockSize: 0x48
[OTA] Final block with signature arrived
[OTA][prvPAL_WriteBlock_amebaPro] OTA1 Sig Size is 71
[OTA] image download is already done, dropped, aws_ota_imgsz=0x1A2000, ImgLen=0x1A2000
10226 140083 [OTA Agent] [prvIngestDataBlock] Received final expected block of file.
10227 140091 [OTA Agent] [prvStopRequestTimer] Stopping request timer.
[OTA] Authenticating and closing file.
10228 140100 [OTA Agent] [prvPAL_CloseFile] Authenticating and closing file.
10229 140107 [OTA Agent] [prvPAL_CheckFileSignature_amebaPro] Started sig-sha256-ecdsa signature verification, file: /
10230 140118 [OTA Agent] Assume Cert - No such file: /. Using header file
10231 140682 [OTA Agent] [prvIngestDataBlock] File receive complete and signature is valid.
10232 140689 [OTA Agent] [prvStopRequestTimer] Stopping request timer.
10233 140695 [OTA Agent] [prvPublishStatusMessage] Msg: {"status":"IN_PROGRESS","statusDetails":{"self_test":"ready","updatedBy":"0x63090002"}}
10234 140709 [OTA Agent] [INFO ][MQTT][140709] (MQTT connection 7026ef40) MQTT PUBLISH operation queued.
10235 140717 [OTA Agent] [INFO ][MQTT][140717] (MQTT connection 7026ef40, PUBLISH operation 7026f3c0) Waiting for operation completion.
10236 140735 [iot_threa] [INFO ][DEMO][140735] State: WaitingForFileBlock Received: 3343 Queued: 0 Processed: 0 Dropped: 0

10237 140955 [OTA Agent] [INFO ][MQTT][140955] (MQTT connection 7026ef40, PUBLISH operation 7026f3c0) Wait complete with result SUCCESS.
10238 140965 [OTA Agent] [prvPublishStatusMessage] 'IN_PROGRESS' to $aws/things/ameba-ota/jobs/AFR_OTa-Pro_OTa_test_0701_4/update
10239 140977 [OTA Agent] [INFO ][DEMO][140977] Received eOTA_JobEvent_Activate callback from OTA Agent.

10240 140986 [OTA Agent] [INFO ][MQTT][140986] (MQTT connection 7026ef40) Disconnecting connection.
10241 140995 [OTA Agent] [INFO ][MQTT] [core_mqtt.c:2149] 10242 141000 [OTA Agent] Disconnected from the broker.10243 141004 [OTA Agent]
10244 141007 [OTA Agent] [INFO ][MQTT][141007] (MQTT connection 7026ef40) Network connection closed.
10245 141015 [OTA Agent] [INFO ][DEMO][141015] Mqtt disconnected due to invoking diconnect function.

10246 141025 [OTA Agent] [INFO ][MQTT][141025] (MQTT connection 7026ef40) Network connection destroyed.

[update_ota_prepare_addr] Get loaded_fw_idx 1

[update_ota_prepare_addr] NewFWAddr 00240000

[update_ota_signature] Append OTA signature
[update_ota_signature] signature:
E6 8C 79 0B 33 80 72 F7 C3 D2 63 D1 A5 BE 05 CA
66 EE 58 4B BD F9 69 A2 53 9E C9 F9 0B DA 46 CC[OTA] [prvPAL_ActivateNewImage_amebaPro] Update OTA success!
[OTA] 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 100.9.2, which is bigger than old one 99.9.2.

```
46 7625 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: /" ]}}}}}}... ]
47 7633 [OTA Agent] [prvParseJobDoc] In self test mode.
48 7638 [OTA Agent] [prvValidateUpdateVersion] The update version is newer than the version on device.
49 7647 [OTA Agent] [prvParseJobDoc] Setting image state to Testing for file ID 0
50 7654 [OTA Agent] [prvSetImageState_amebaPro] Testing image.
51 7660 [OTA Agent] [prvPublishStatusMessage] Msg: {"status":"IN_PROGRESS","statusDetails":{"self_test":"active","updatedBy":"0x64090002"}}
52 7673 [OTA Agent] [INFO ][MQTT][7673] (MQTT connection 7026af20) MQTT PUBLISH operation queued.
53 7681 [OTA Agent] [INFO ][MQTT][7681] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Waiting for operation completion.
54 7916 [OTA Agent] [INFO ][MQTT][7916] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Wait complete with result SUCCESS.
55 7926 [OTA Agent] [prvPublishStatusMessage] 'IN_PROGRESS' to $aws/things/ameba-ota/jobs/AFR_OTa-Pro_OTa_test_0701_4/update
56 7937 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForJob] Event [ReceivedJobDocument] New state [CreatingFile]
57 7949 [OTA Agent] [prvInSelfTestHandler] prvInSelfTestHandler, platform is in self-test.
58 7957 [OTA Agent] [prvPAL_GetPlatformImageState_amebaPro] Image current state (0x01)
59 7964 [OTA Agent] [INFO ][DEMO][7964] Received eOTA_JobEvent_StartTest callback from OTA Agent.

60 7974 [OTA Agent] [prvSetImageState_amebaPro] Accepted and committed final image.

[update_ota_prepare_addr] Get loaded_fw_idx 2

[update_ota_prepare_addr] NewFWAddr 00040000
61 7989 [OTA Agent] [prvStopSelfTestTimer] Stopping the self test timer.
62 7995 [OTA Agent] [prvPublishStatusMessage] Msg: {"status":"SUCCEEDED","statusDetails":{"reason":"accepted v100.9.2"}}
63 8007 [OTA Agent] [INFO ][MQTT][8007] (MQTT connection 7026af20) MQTT PUBLISH operation queued.
64 8015 [OTA Agent] [INFO ][MQTT][8014] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Waiting for operation completion.
65 8026 [iot_threa] [INFO ][DEMO][8026] State: RequestingJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

66 8243 [OTA Agent] [INFO ][MQTT][8243] (MQTT connection 7026af20, PUBLISH operation 7026b3a0) Wait complete with result SUCCESS.
67 8253 [OTA Agent] [prvPublishStatusMessage] 'SUCCEEDED' to $aws/things/ameba-ota/jobs/AFR_OTa-Pro_OTa_test_0701_4/update
68 8264 [OTA Agent] [prvExecuteHandler] Called handler. Current State [CreatingFile] Event [StartSelfTest] New state [WaitingForJob]
69 9036 [iot_threa] [INFO ][DEMO][9036] State: CreatingFile Received: 1 Queued: 0 Processed: 0 Dropped: 0
```

In the final, the log imply that the OTA status is changed to “SUCCEEDED” !

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 "lib_amazon/amazon-freertos-202012.00/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 "lib_amazon/amazon-freertos-202012.00/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 "lib_amazon/amazon-freertos-202012.00/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-----
```