AmebaD Amazon FreeRTOS 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

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 AmebaD RTL8722DM Board

1.1 AmebaD Demo EVB

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

Ameba RTL8722DM Board (AMB 21)

Manual / Schematic / Layout

Buy it

CPU
- 32-bit Arm®Cortex®-M4, up to 200MHz
- 32-bit Arm®Cortex®-M0, up to 20MHz

Memory
- 512KB SRAM + 4MB PSRAM

Key Features
- Integrated 802.11a/n Wi-Fi SoC
- Trustzone-M Security
- Hardware SSL Engine
- Root Trust Secure Boot
- USB Host/Device
- SD Host
- BLE5.0
- Codec
- LCDC
- Key Matrix

Other Features
- 1 PCM interface
- 4 UART interface
- 1 I2S Interface
- 2 I2C interface
- 7 ADC
- 17 PWM
- Max 54 GPIO

1.2 PCB Layout Overview

The PCB layout of 2D and 3D are shown in Fig 1-1 and Fig 1-2.

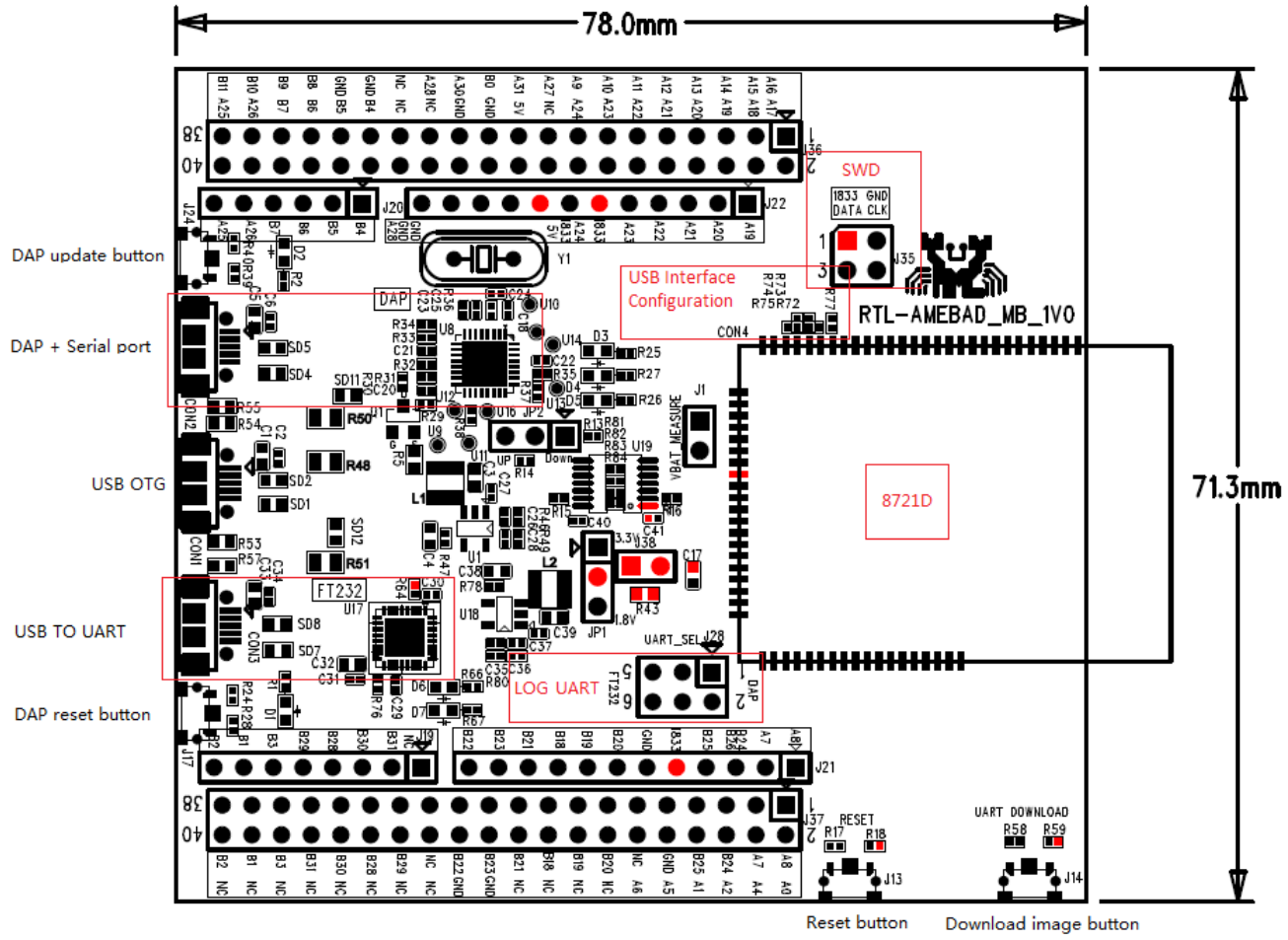


Fig 1-1 Demo board – PCB layout (2D)

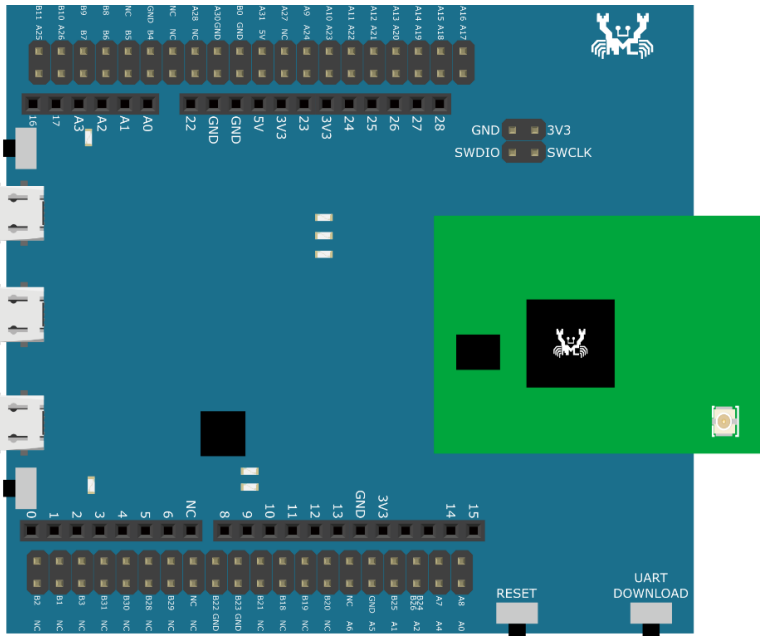


Fig 1-2 Demo board – PCB layout (3D)

1.3 Pin Out

The pin out board is shown in Fig 1-3.

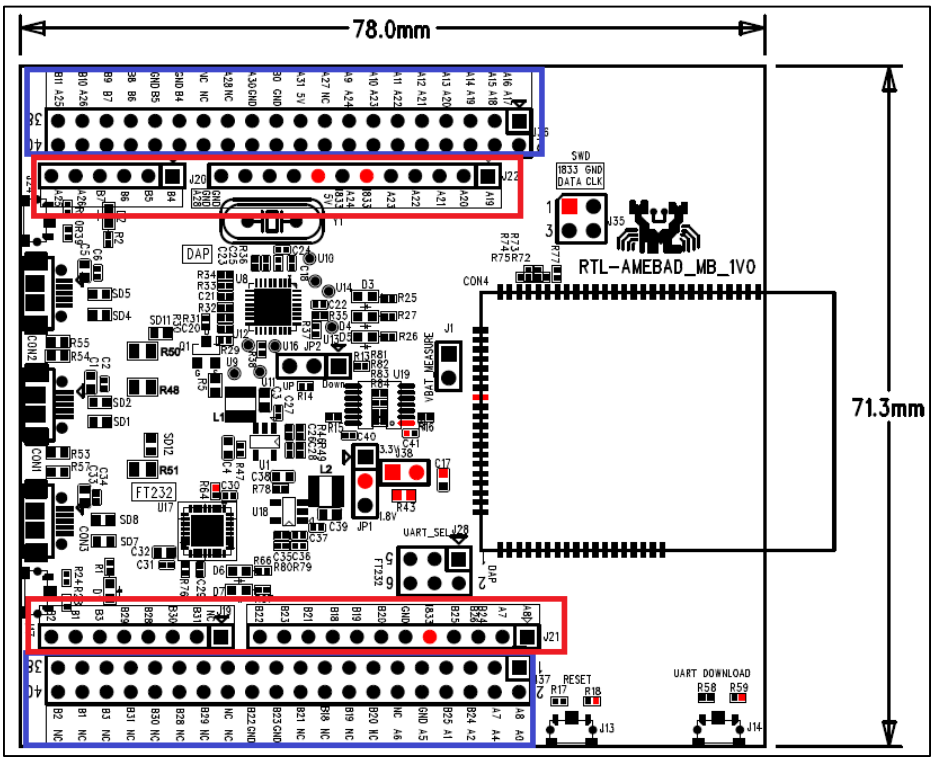


Fig 1-3 Demo board – pin out

There are four rows of pins on the board.

- The pins in the red box are used for Arduino REF.
- The pins in the blue box are all the GPIO pins.

1.4 DC Power Supply

The 3.3V/1.8V power supply board is shown in Fig 1-4.

- Jump JP1 is used to select 3.3V or 1.8V power supply
- Jump J38 is for current test. You can test the current power after taking off the R43.

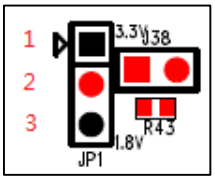


Fig 1-4 Demo board – 3.3V/1.8V power supply

When you select power supply, refer to Table 1-1.

Table 1-1 3.3V/1.8V power supply selection

Power Supply Select	JP1
3.3V	1-2 connected
1.8V	2-3 connected

1.5 USB Interface Configuration

The USB interface configuration board is shown in Fig 1-5 and Fig 1-6.

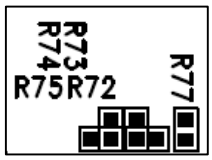


Fig 1-5 Mother board – USB interface configuration

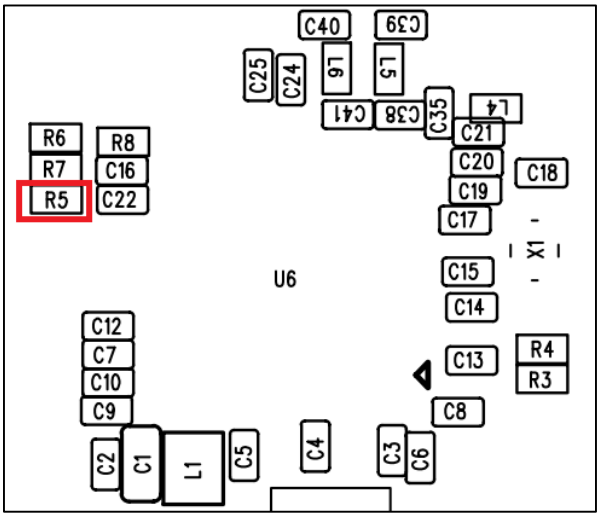


Fig 1-6 Module board – USB interface configuration

For normal GPIO usage by default, R72/R75/R77 on mother board will part on with 0 Ohm resistors, R5 on module board needs to take off. For USB usage, you need to take off R77, part on R73&R74 with 0 Ohm resistors on mother board and part on R5 on module board with a 12K Ohm 1% precision resistor.

1.6 LOGUART

The LOGUART board is shown in Fig 1-7. When you select LOGUART, please refer to Table 1-2.

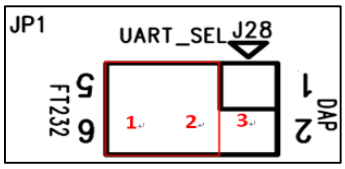


Fig 1-7 Demo board – LOGUART

Table 1-2 LOGUART selection

LOGUART Select	JP1
FT232	1-2 connected
DAP	2-3 connected

1.7 SWD

The SWD board is shown in Fig 1-8.

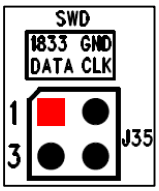


Fig 1-8 Demo board – SWD

Note: For 1V0 board, there is an issue, you should use CLK as DATA, and use DATA as CLK.

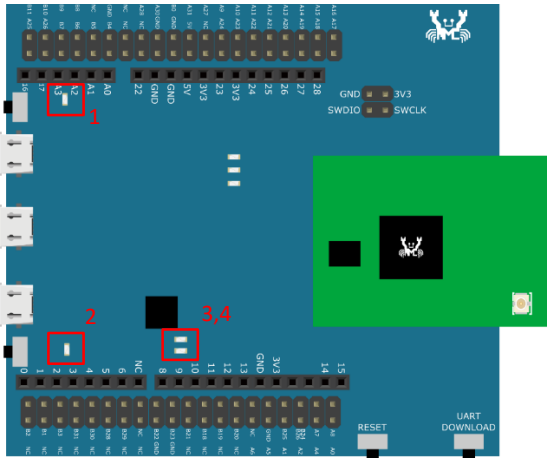
1.8 VBAT ADC

The VBAT ADC board is shown in Fig 1-9. J1 is used to test VBAT ADC.

Fig 1-9 Demo board – VBAT ADC

1.9 LED State

There are four LED on the AmebaD EVB. LED1 and LED2 lights steady green when device have power. LED3 and LED4 go with log uart, they flash red and green when uart communicating.



2 Configure AWS IoT Core

2.1 Set up your AWS account and Permissions

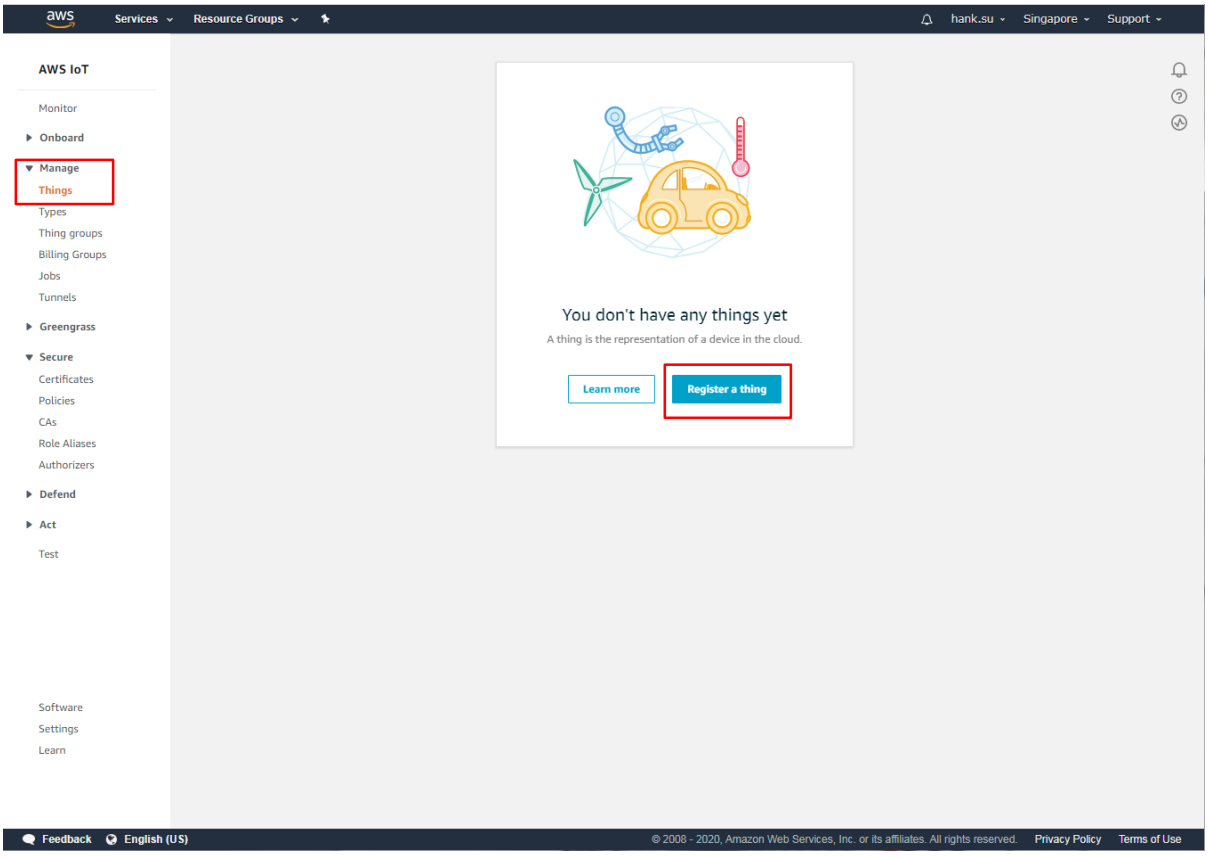
Refer to the instructions at Set up your AWS Account https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html. Follow the steps outlined in these sections to create your account and a user and get started:

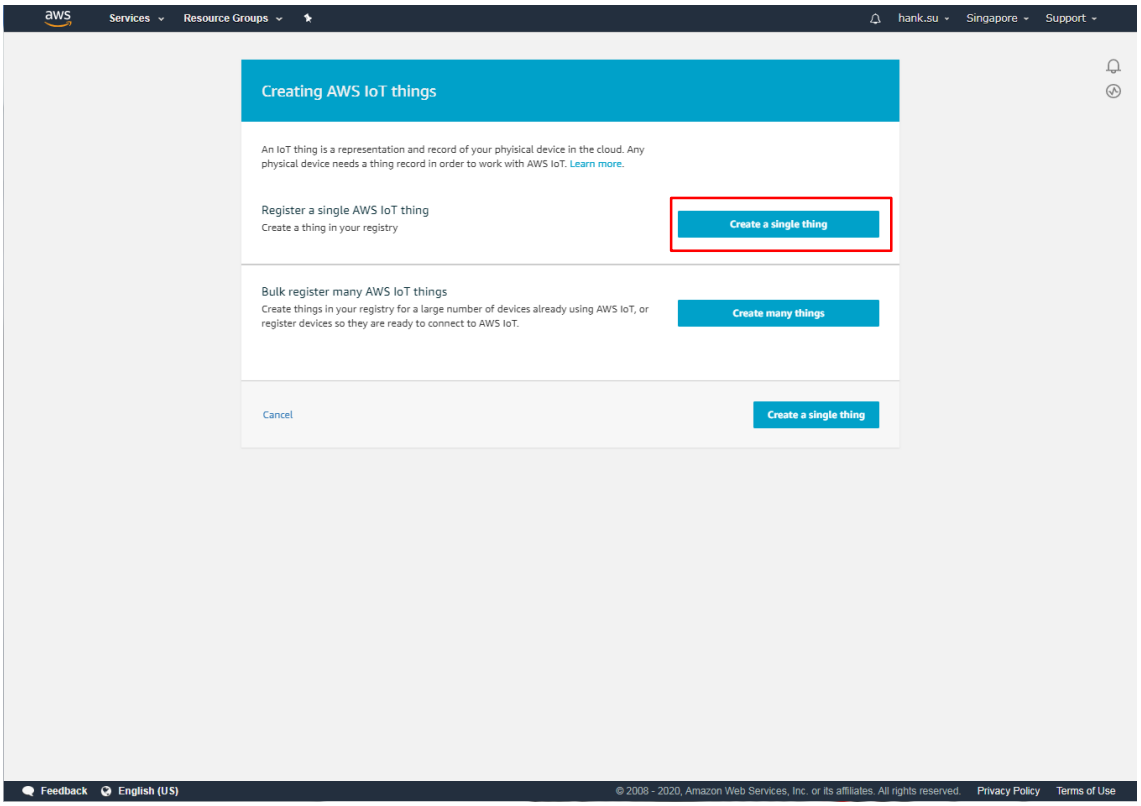
- Sign up for an AWS account
- Create a user and grant permissions
- Open the AWS IoT console

Please pay special attention to the Notes in AWS webpage.

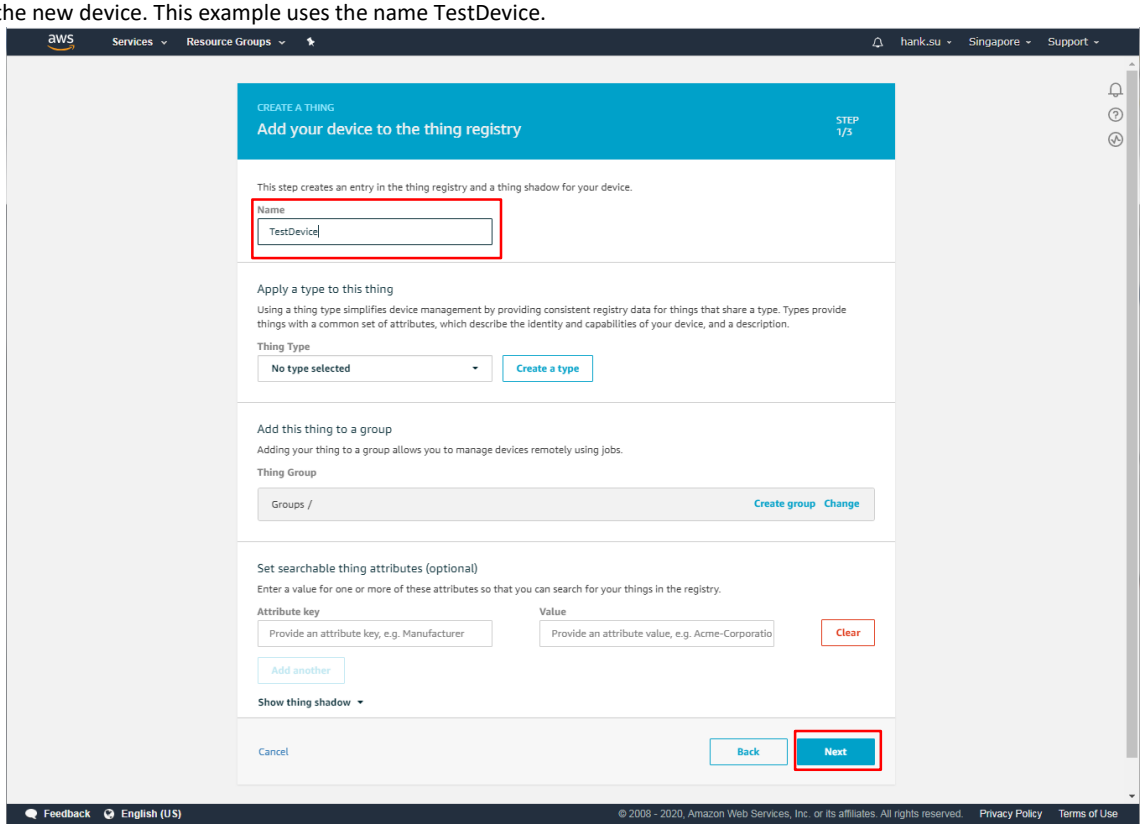
2.2 Create a New Device

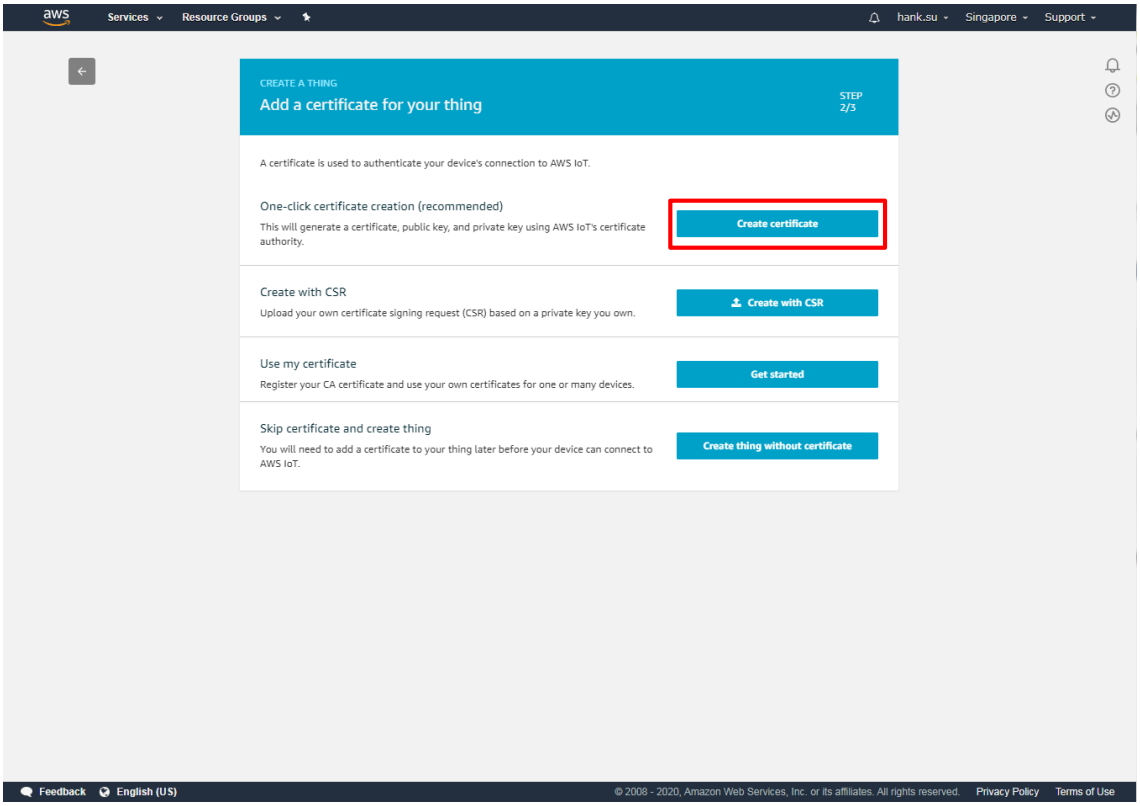
To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click “Register a thing”.



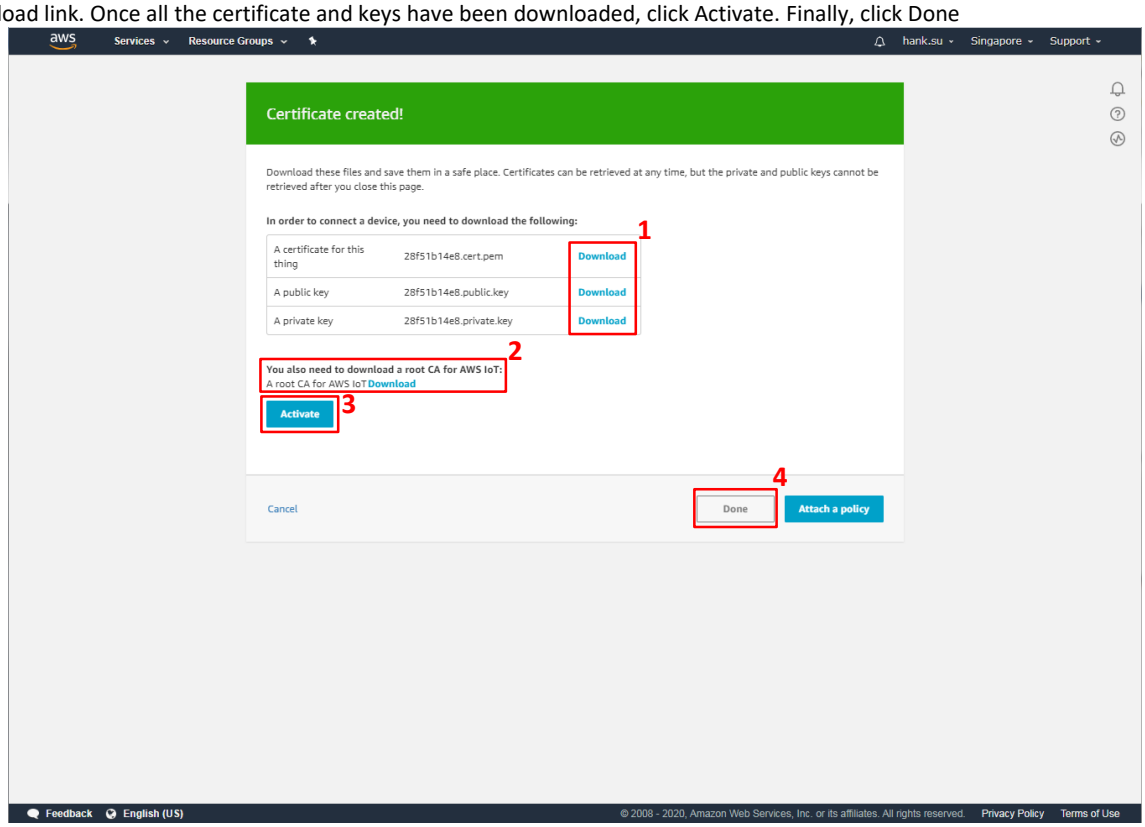


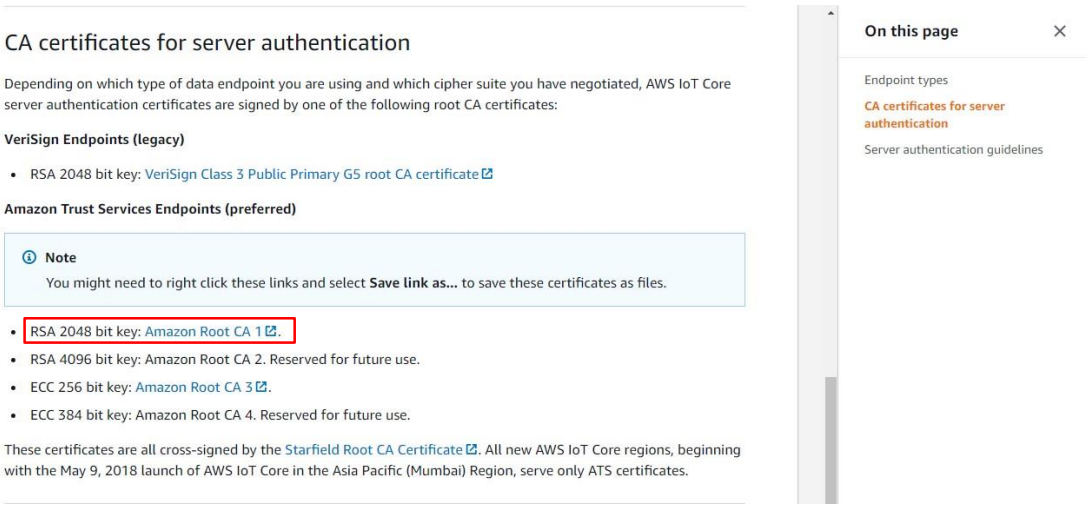
Then, name the new device. This example uses the name TestDevice.





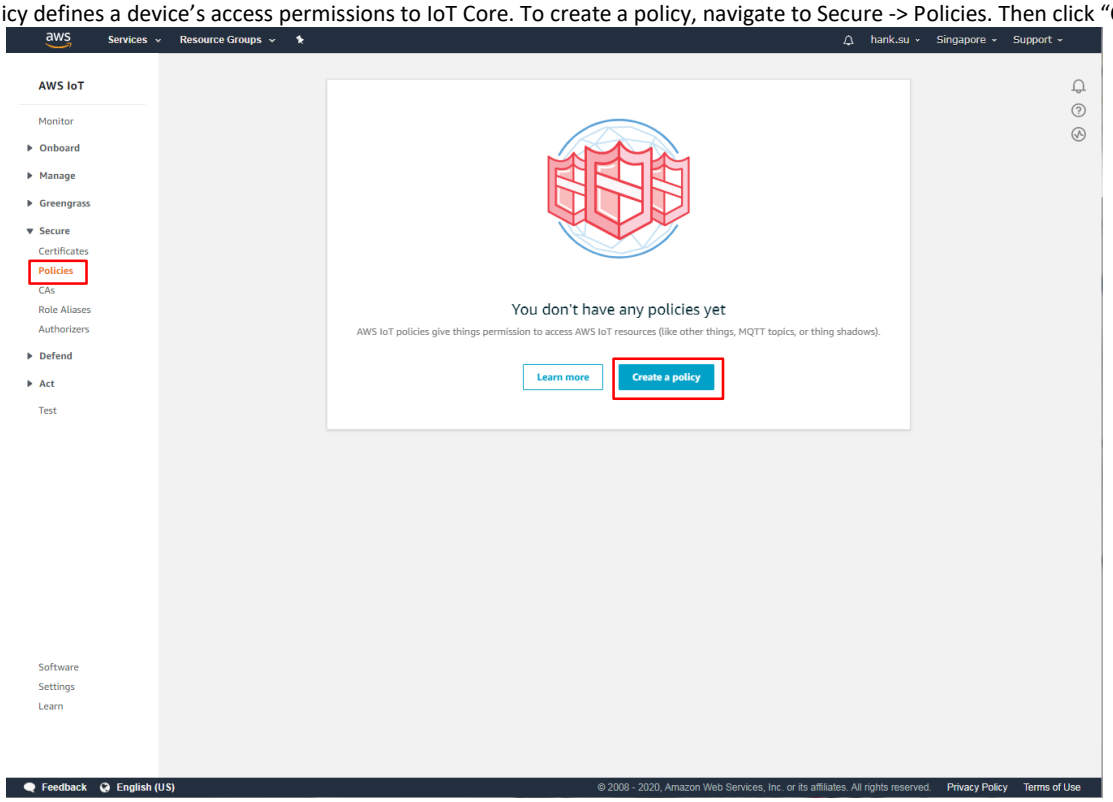
Download the certificate, public key, and private key for the device by clicking Download. Next, download the root CA for AWS IoT by clicking to the Download link. Once all the certificate and keys have been downloaded, click Activate. Finally, click Done



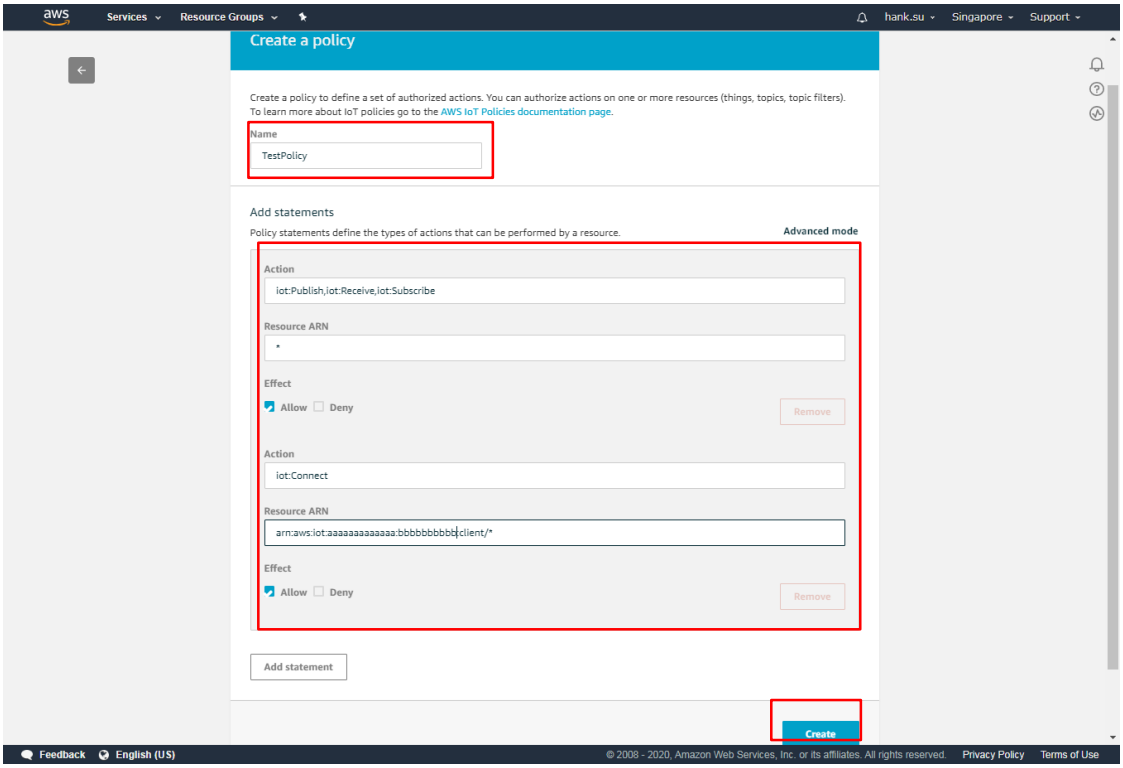


2.3 Create a policy

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

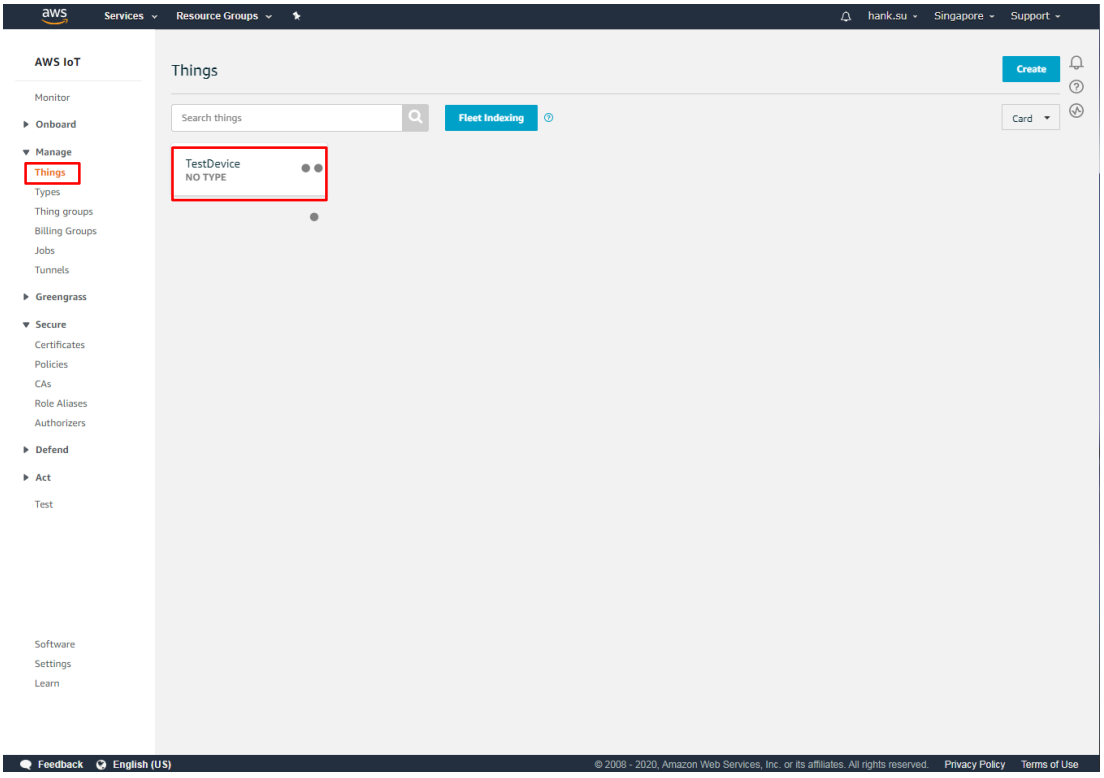


NOTE – this policy grants unrestricted access for all iot operations, and is to be used only in a development environment. For non-dev environments, all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions 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

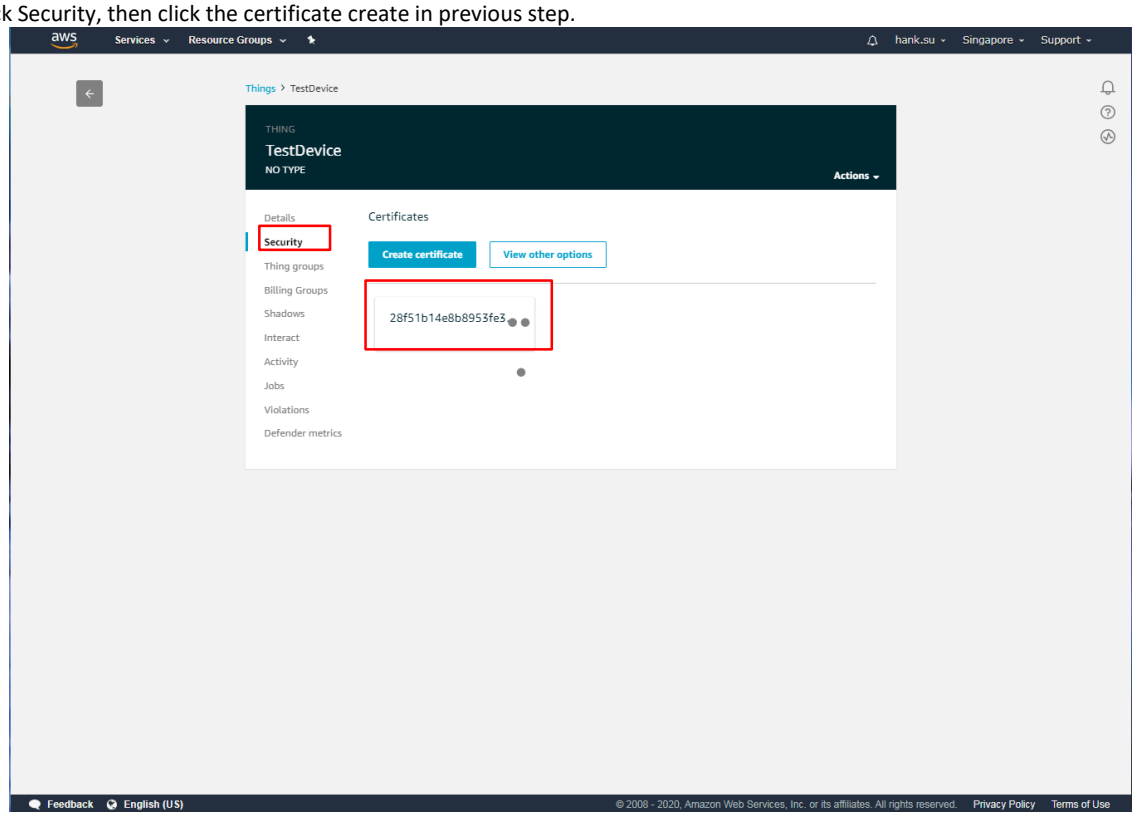


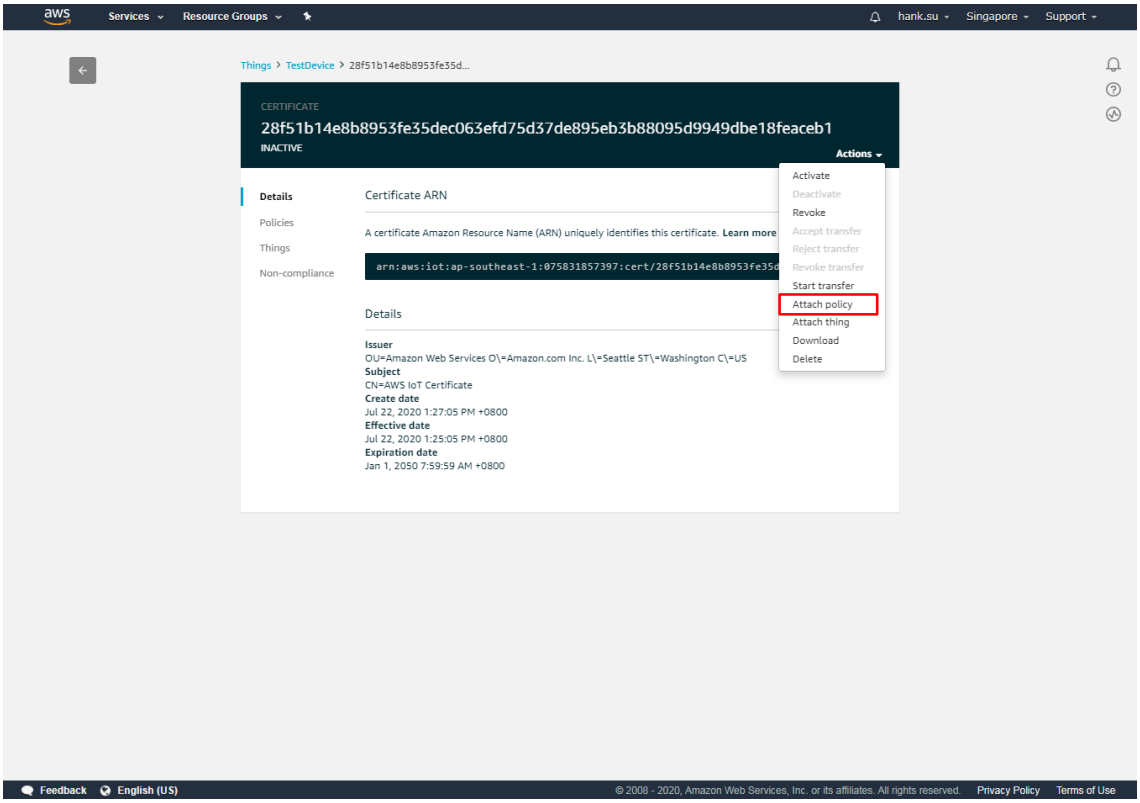
2.4 Attach Policy

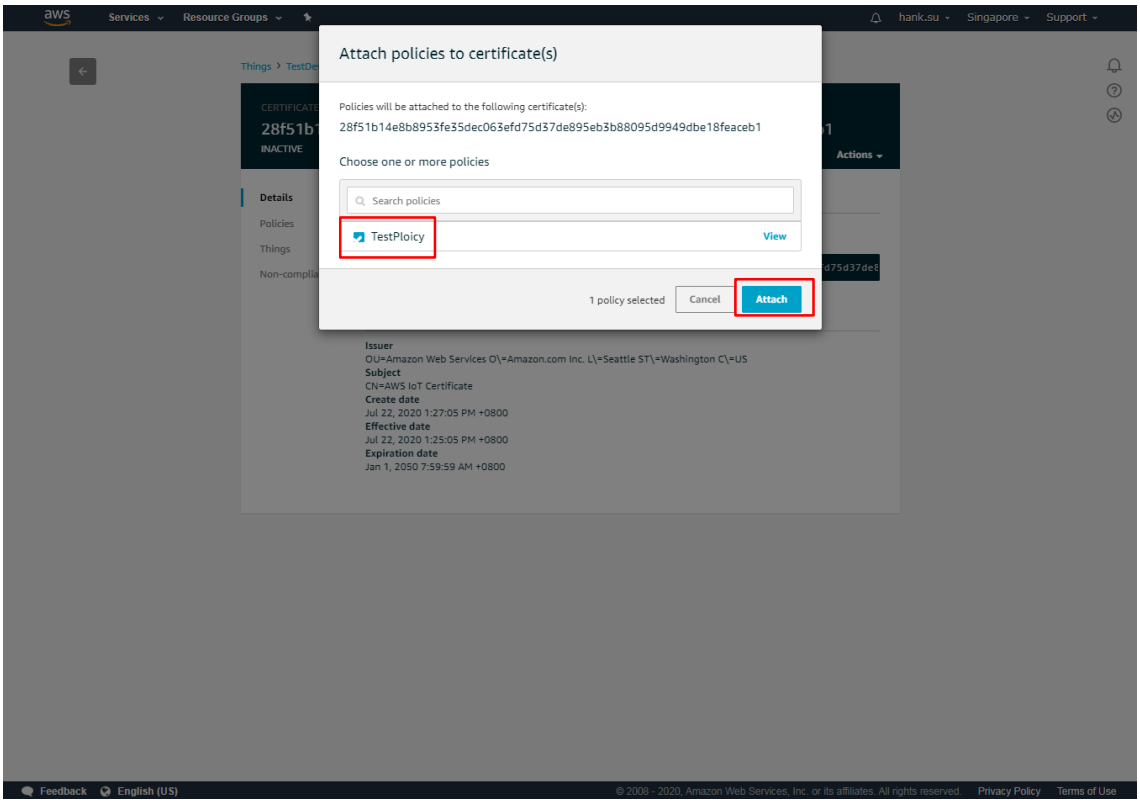
The last step to configuring the device is attaching a policy. To attach a policy to new device, navigate to Manage -> Things. Then click on the device which was created.



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



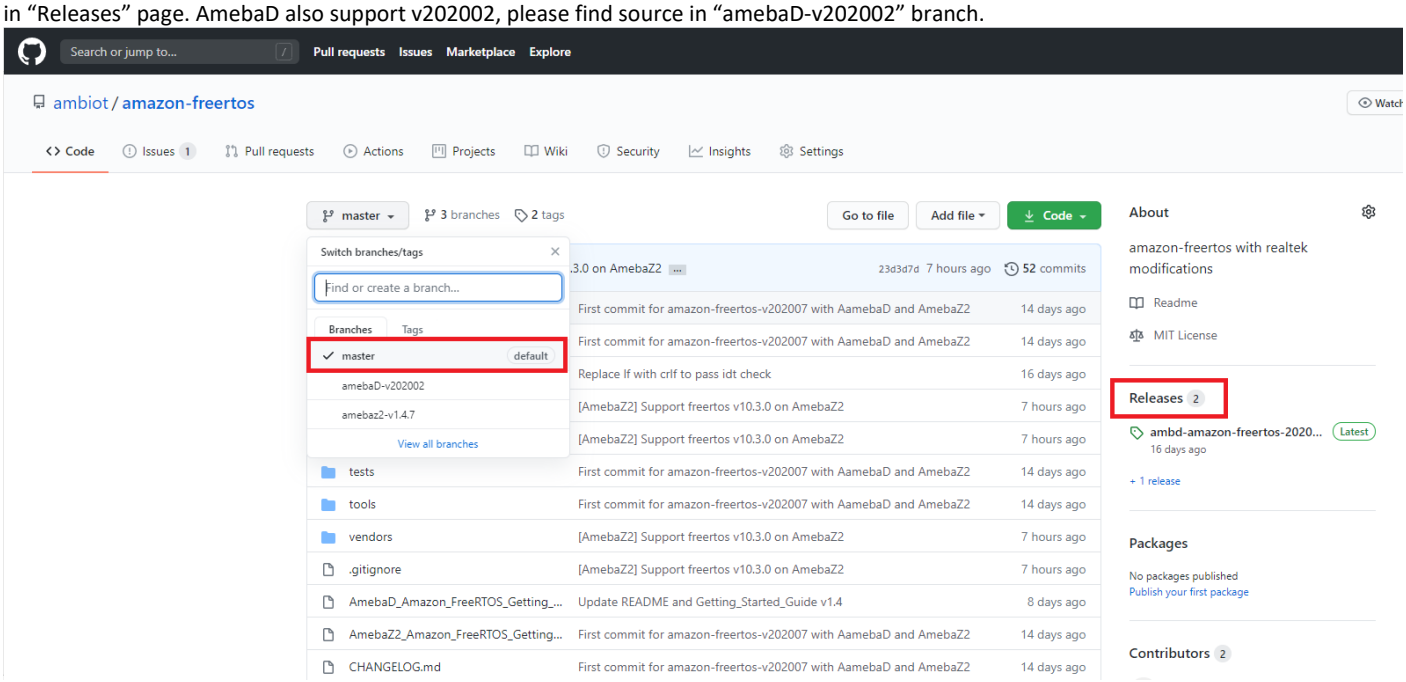




3 Configure AmebaD Amazon FreeRTOS

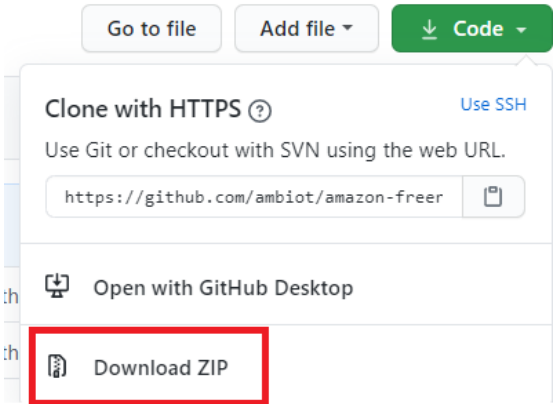
3.1 Download Source Code from github

Open source link: https://github.com/ambiot/amazon-freertos and select master to get newest source code. The stable version could be found in “Releases” page. AmebaD also support v202002, please find source in “amebaD-v202002” branch.



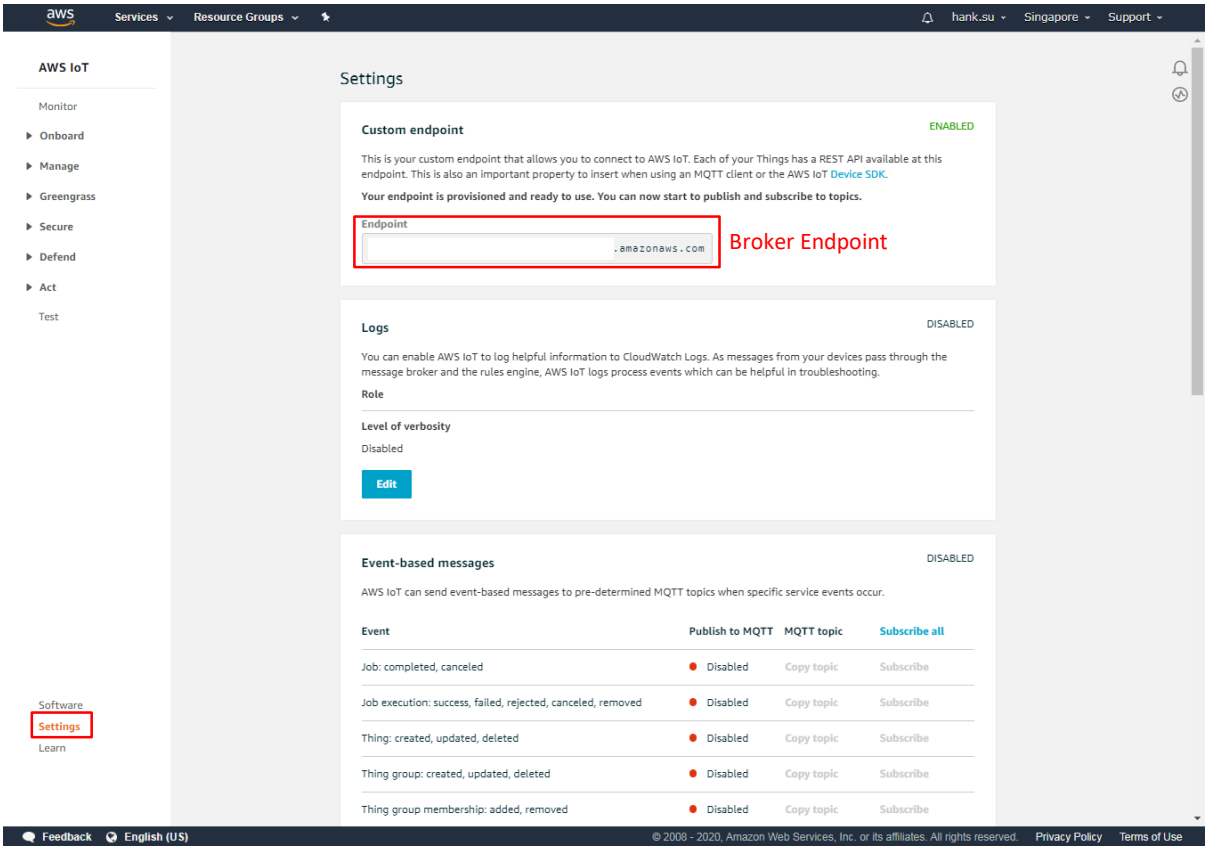
3.1.1 Cloning a repository by Download ZIP

1. On GitHub, navigate to the main page of the repository.
2. Above the list of files, click **Code**.
3. Click **Donwload ZIP** to get source code.

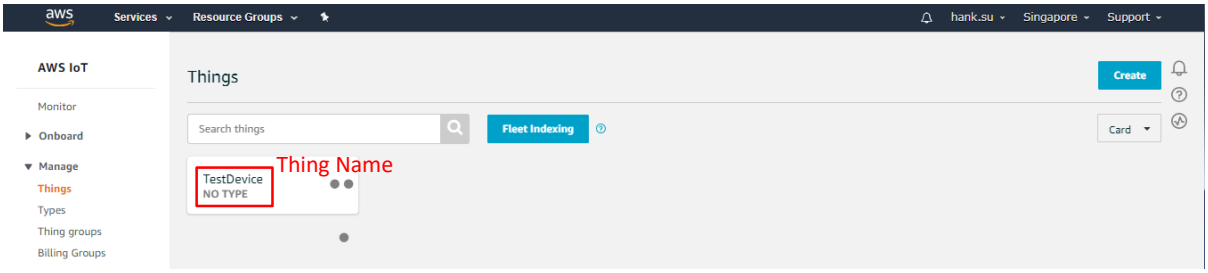


For more information, please refer "Cloning a repository from GitHub to GitHub Desktop."

3.2 Get Broker Endpoint by AWS IoT Core



3.3 Get Thing Name



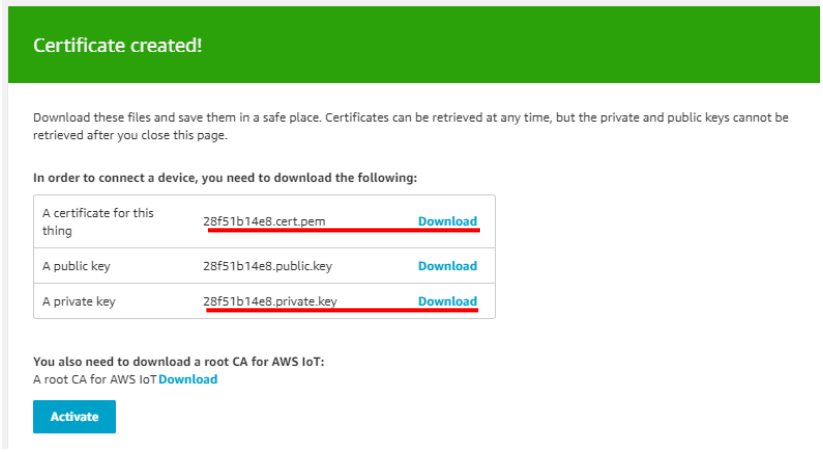
3.4 Setup IoT Core Information with AmebaD Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in “amazon-freertos/demos/include/aws_clientcredential.h”

```
#define clientcredentialMQTT_BROKER_ENDPOINT        "xxxxxxxxxxxxxx.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 “amazon-freertos/demos /include/aws_clientcredential_keys.h” by xxxxxxxx-certifiacte.pem and xxxxxxxx-private.pem.key.



It can done by amazon-freertos/tools/certificate_configuration/CertificateConfigurator.html

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"\
"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"\
"pOWEuLUuz2FAv1noAbN/6OQ8H/PT0AFJT/ghA04GnIUF0kjSzY60ehS2mVp6neP+\n"\
"AZjzZ6QJYlb5/PFz9oES448kpyaAoS2ke86+R4r4YOMBK+I5RVbfSQ==\n"\
"-----END RSA PRIVATE KEY-----\n"
```

3.4.2 Enable FreeRTOS demo on AmebaD

Find platform_opts.h in amazon-freertos\vendors\realtek\boards\amebaD\aws_demos\config_files and enable **CONFIG_EXAMPLE_AMAZON_FREERTOS**

```
/* For Amazon FreeRTOS SDK example */
#define CONFIG_EXAMPLE_AMAZON_FREERTOS   1
```

Find aws_demo_config.h in amazon-freertos\vendors\realtek\boards\amebaD\aws_demos\config_files and add **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
```

Now you can start to compile AmebaD Amazon FreeRTOS

4 Compile AmebaD Amazon FreeRTOS

4.1 IAR Build Environment Setup

Currently the amazon-freertos of AmebaD supported by the IAR Embedded workbench ver.8.30.1. **For windows operating system only**. This chapter illustrates how to setup IAR development environment for Realtek Ameba-D SDK, including building projects and downloading images.

4.2 Pre-Requisite

- Required source code. (https://github.com/ambiot/amazon-freertos)
- AmebaD Demo board
- Realtek Image Tool
- IAR Embedded Workbench ver.8.30.1

IAR provides an IDE environment for code building, downloading, and debugging. Check “IAR Embedded Workbench” on http://www.iar.com/, and a trail version is available for 30 days.

Note: To support ARMv8-M with Security Extension (Ameba-D HS CPU, also called KM4), IAR version must be 8.30 or higher.

4.3 How to Use IAR SDK

4.3.1 IAR Project Introduction

Because Ameba-D is a dual-core CPU platform, two workspaces provided to build for each core in **amazon-freertos/projects/realtek/amebaD/IAR/aws_demos**

- Project_lp_release.eww (KM0 workspace) contains the following projects:
  - km0_bootloader
  - km0_application
- Project_hp_release.eww (KM4 workspace) contains the following projects:
  - km4_bootloader
  - km4_application

4.3.2 IAR Build

When building SDK for the first time, you should build both KM0 project and KM4 project. Other times, you only need to rebuild the modified project.

4.3.2.1 Building KM0 Project

The following steps show how to build KM0 project:

(1) Open **amazon-freertos\projects\realtek\amebaD\IAR\aws_demos\Project_lp_release.eww**.

(2) Make sure km0_bootloader and km0_application are in Workspace. Click **Project** > **Options**, **General Options** > **Target** > **Processor Variant** > **Core**, verify the CPU configurations according to Fig 4-1

(3) Right click the project and choose “Rebuild All”, as Fig 4-2 shows. The km0_bootloader and km0_application should compile in order.

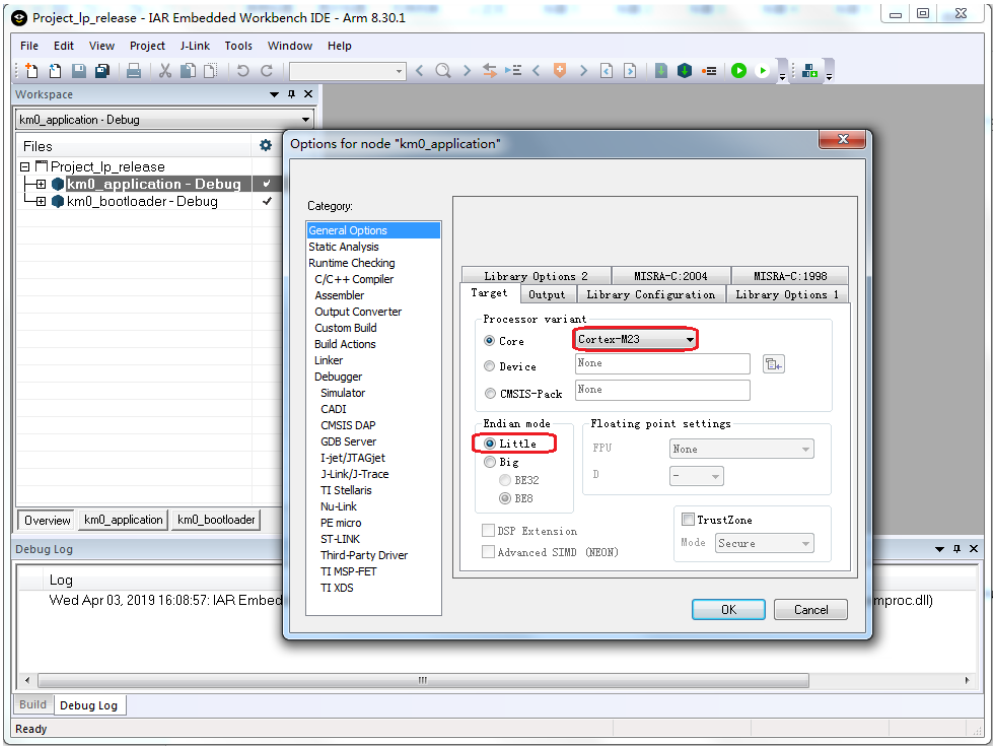


Fig 4-1 KM0 processor options

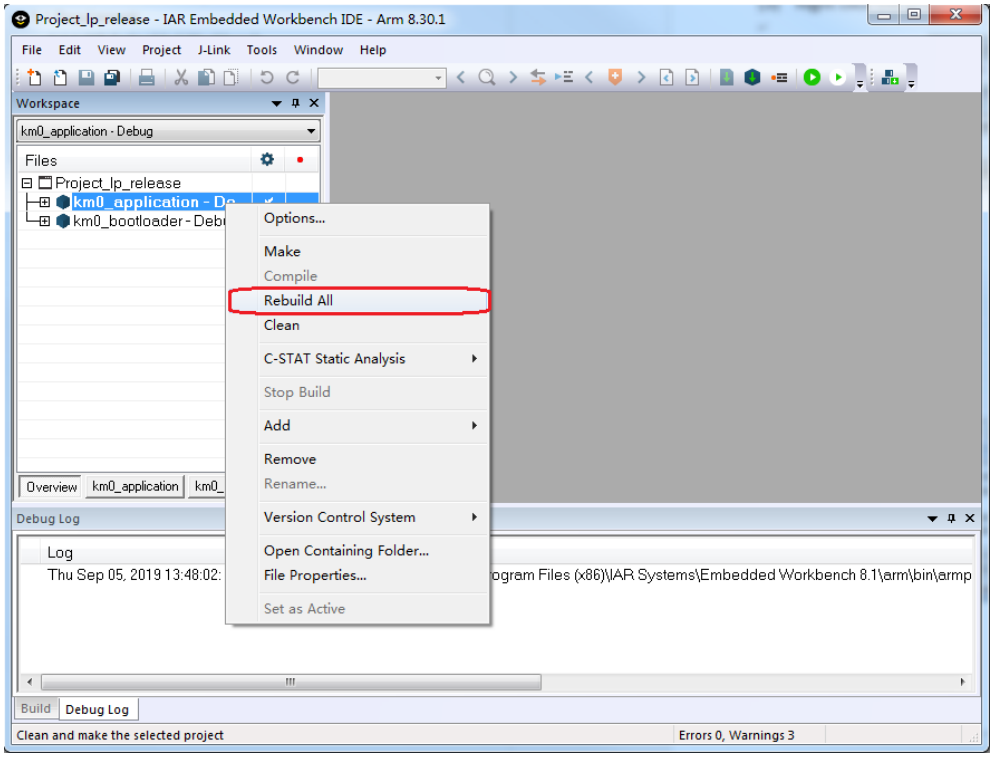
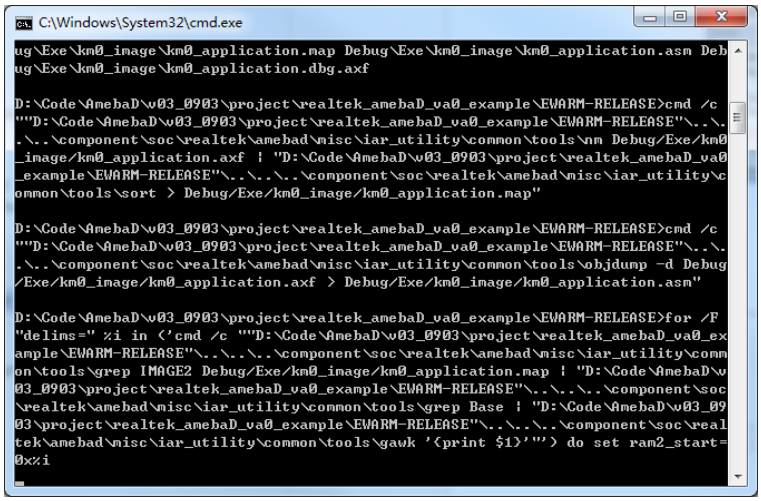


Fig 4-2 Building KM0 project

Note: After building each project, IAR will pop up a command prompt window to execute post-build action to generate images from executable files. This may takes several seconds. Do not stop it while it is in progress. After post-build action is completed, the window would disappear automatically.



(4) After compile, the images km0_boot_all.bin and km0_image2_all.bin can be find in **amazon-freertos\projects\realtek\amebaD\IAR\aws_demos\Debug\Exe\km0_image**.

4.3.2.2 Building KM4 Project

The following steps show how to build KM4 project:

(1) Open **amazon-freertos\projects\realtek\amebaD\IAR\aws_demos\Project_hp_release.eww**.

(2) Refer to 4.3.1 and choose the build configurations for each project according to your application.

(3) Click **Project** > **Options**, **General Options** > **Target** > **Processor Variant** > **Core**, verify the CPU configurations according to Fig 4-3.

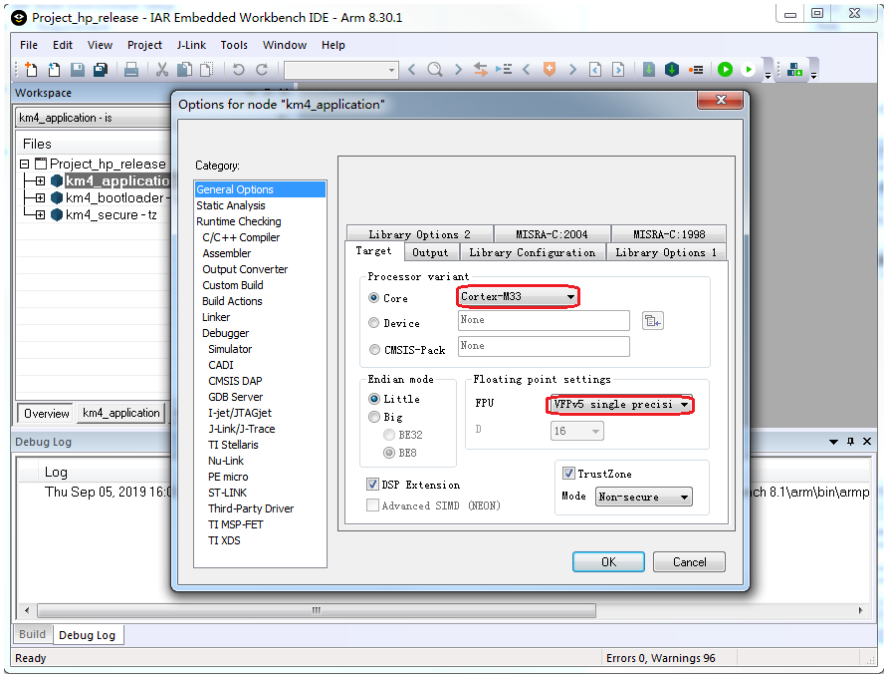


Fig 4-3 KM4 processor options

(4) Right click the project and choose “Rebuild All”, as Fig 4-4 shows. The km4_bootloader, km4_application should compile in order.

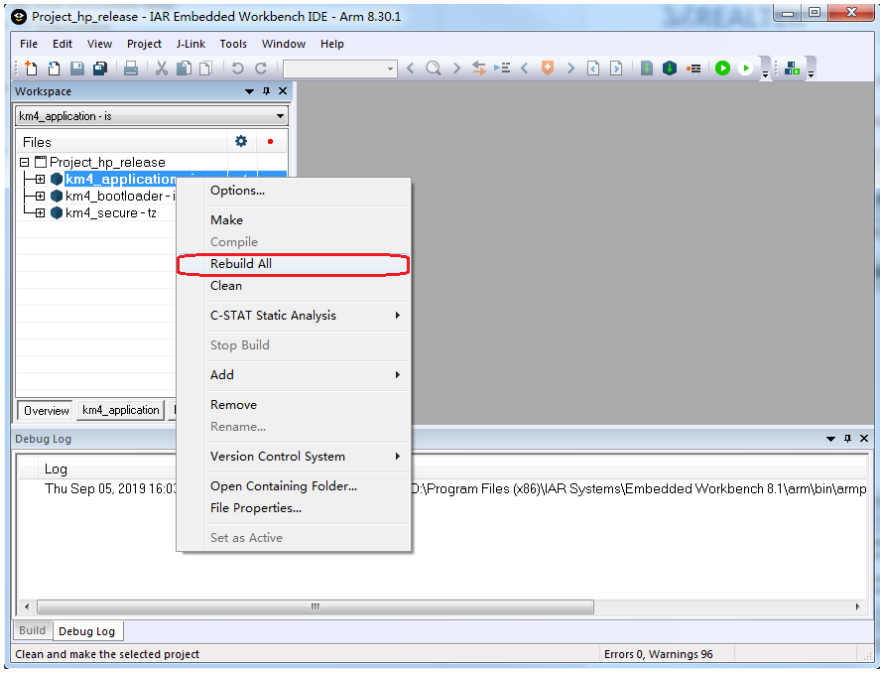
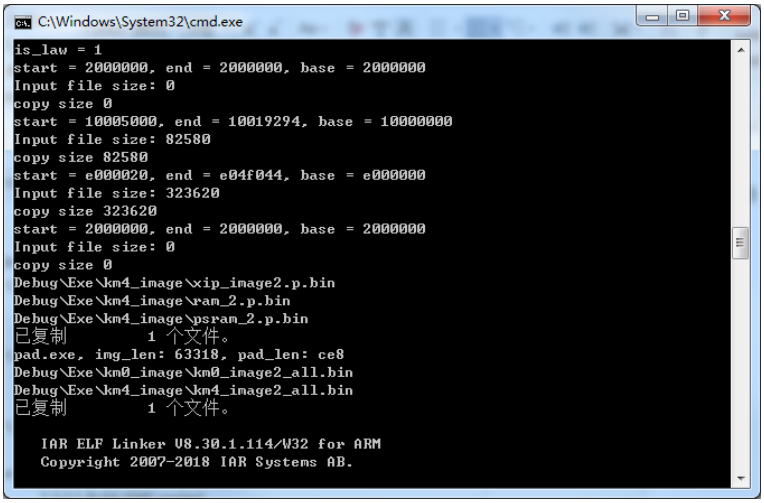


Fig 4-4 Building KM4 project

Note:

- After building each project, IAR will pop up a command prompt window shown in bellow to execute post-build action to generate images from executable files. This may takes several seconds. Do not stop it while it is in progress. After post-build action is completed, the window would disappear automatically.



(5) After compile, the images km4_boot_all.bin and km0_km4_image2.bin can be find in **amazon-freertos\projects\realtek\amebaD\IAR\aws_demos\Debug\Exe\km4_image**.

(6) The generated images can be downloaded to flash by ImageTool:

5 ImageTool

The tool can be find in amazon-freertos\vendors\realtek\tools\ameba-image-Tool-v2.4.1\

5.1 Introduction

This chapter introduces how to use ImageTool to encrypt, generate and download images. As show in Fig 5-1, ImageTool has four tabpages.

- Download: used as image download server to transmit images to Ameba through UART.

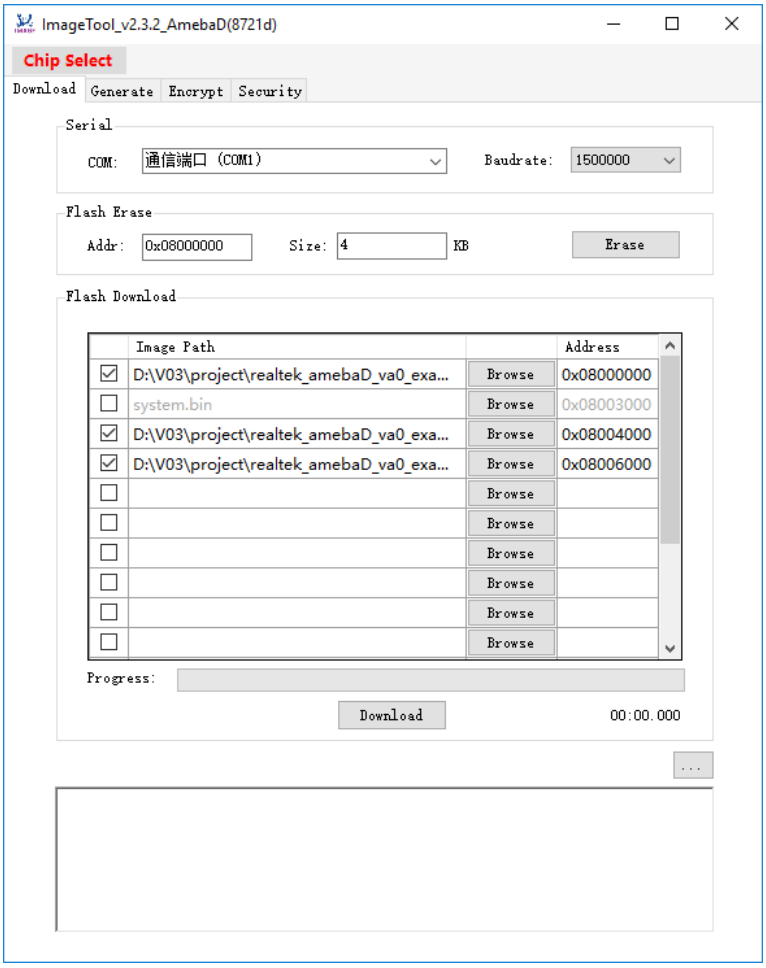


Fig 5-1 ImageTool UI

5.2 Environment Setup

5.2.1 Hardware Setup

The hardware setup is shown in Fig 5-2.

Note: If using external UART to download images, FT232 USB to UART dongle must be used.

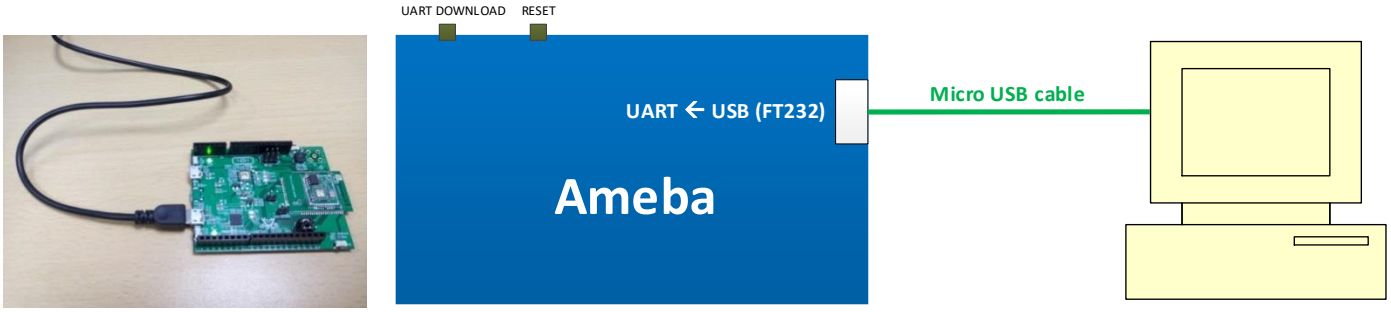


Fig 5-2 Hardware setup

5.2.2 Software Setup

- Environment Requirements: EX. WinXP, Win 7 Above, Microsoft .NET Framework 3.5
- ImageTool.exe Location: **vendors\realtek\tools\ameba-image-Tool-v2.4.1\ImageTool.exe**

Name	Date modified	Type	Size
ChangeLog.txt	7/29/2019 11:52 AM	Text Document	4 KB
Download.ini	11/4/2019 5:44 PM	Configuration sett...	2 KB
Encrypt.ini	11/4/2019 5:44 PM	Configuration sett...	1 KB
ImageTool.exe	7/29/2019 11:52 AM	Application	282 KB
ImageTool.pdb	7/29/2019 11:52 AM	VisualStudio.pdb....	178 KB
ImageTool.vshost.exe	8/20/2018 1:41 PM	Application	14 KB
ImageTool.vshost.exe.manifest	8/20/2018 1:41 PM	MANIFEST File	1 KB
imgtool_flashloader_amebad.bin	6/6/2019 3:15 PM	BIN File	5 KB
imgtool_flashloader_amebaz.bin	6/6/2019 3:15 PM	BIN File	6 KB
SB.exe	8/20/2018 1:41 PM	Application	189 KB
system.bin	8/6/2019 9:53 AM	BIN File	4 KB
TestListView.dll	8/20/2018 1:41 PM	Application extens...	5 KB
TestListView.pdb	8/20/2018 1:41 PM	VisualStudio.pdb....	14 KB

5.3 Download

5.3.1 Image Download

Assuming that the ImageTool on PC is a server, it sends images files to Ameba (client) through UART. There are two ways to download images to board.

5.3.1.1 Based on Hardware Reset

The way based on hardware reset is a manual method to download images, and it is the primary and recommended method.

(1) Enter into UART_DOWNLOAD mode.
   a) Push the **UART DOWNLOAD** button and keep it pressed.
   b) Re-power on the board or press the **Reset** button.
   c) Release the **UART DOWNLOAD** button.
   Now, Ameba board gets into UART_DOWNLOAD mode and is ready to receive data.

(2) Click **Chip Select** (in red) on UI and select chip (AmebaD).

(3) Select the corresponding serial port and transmission baud rate. The default baud rate is 1.5Mbps (recommended).

(4) Click the **Browse** button to select the images (**km0_boot_all.bin/km4_boot_all.bin/km0_km4_image2.bin**) to be programmed and input addresses.
   - The image path is located in **{*path*}\projects\realtek\amebaD\IAR\aws_demos\Debug\Exe\km0_image** and **{*path*}\projects\realtek\amebaD\IAR\aws_demos\Debug\Exe\km4_image**, where **{*path*}** is the location of the project on your own computer.
   - The default target address is the SDK default image address, you can use it directly.

(5) Click **Download** button to start. The progress bar will show the transmit progress of each image. You can also get the message of operation successfully or errors from the log window.

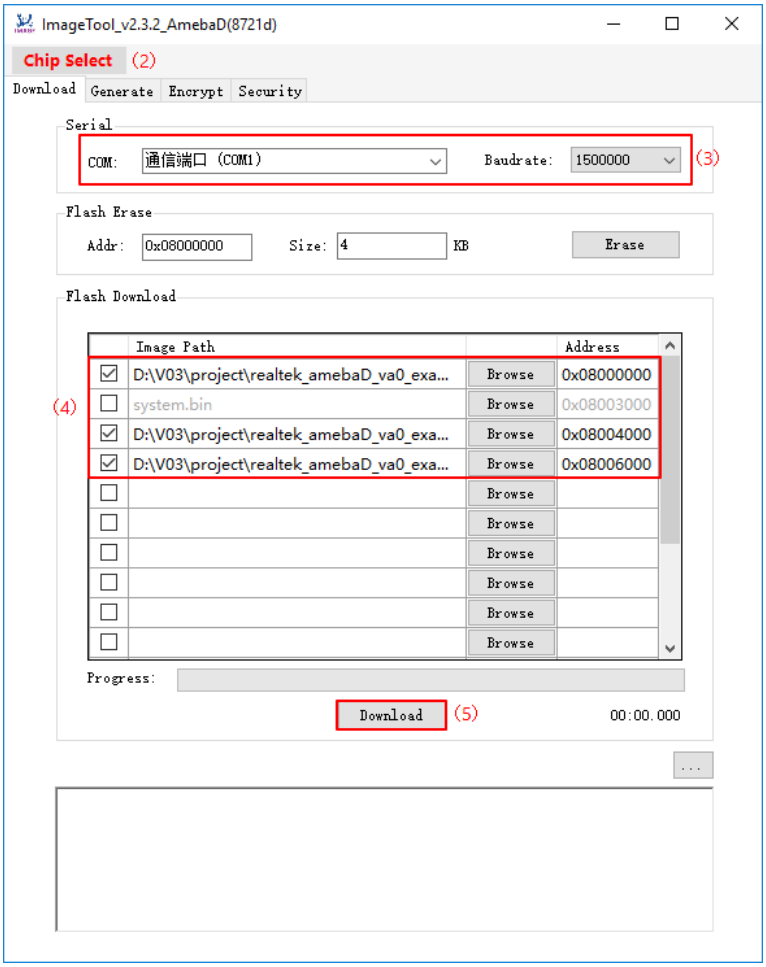


Fig 5-3 ImageTool 'Download' tabpage setting

6 MQTT Demo

6.1 Get Device Log

Install Tera Term to get device log

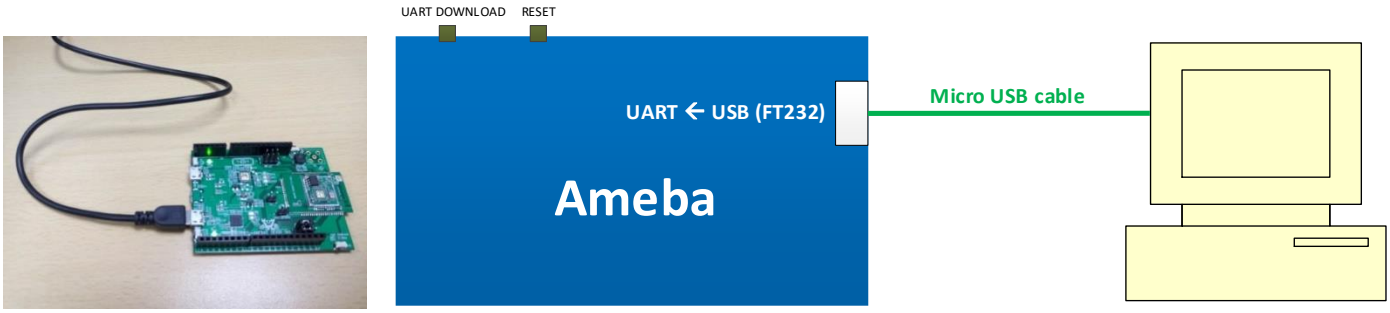
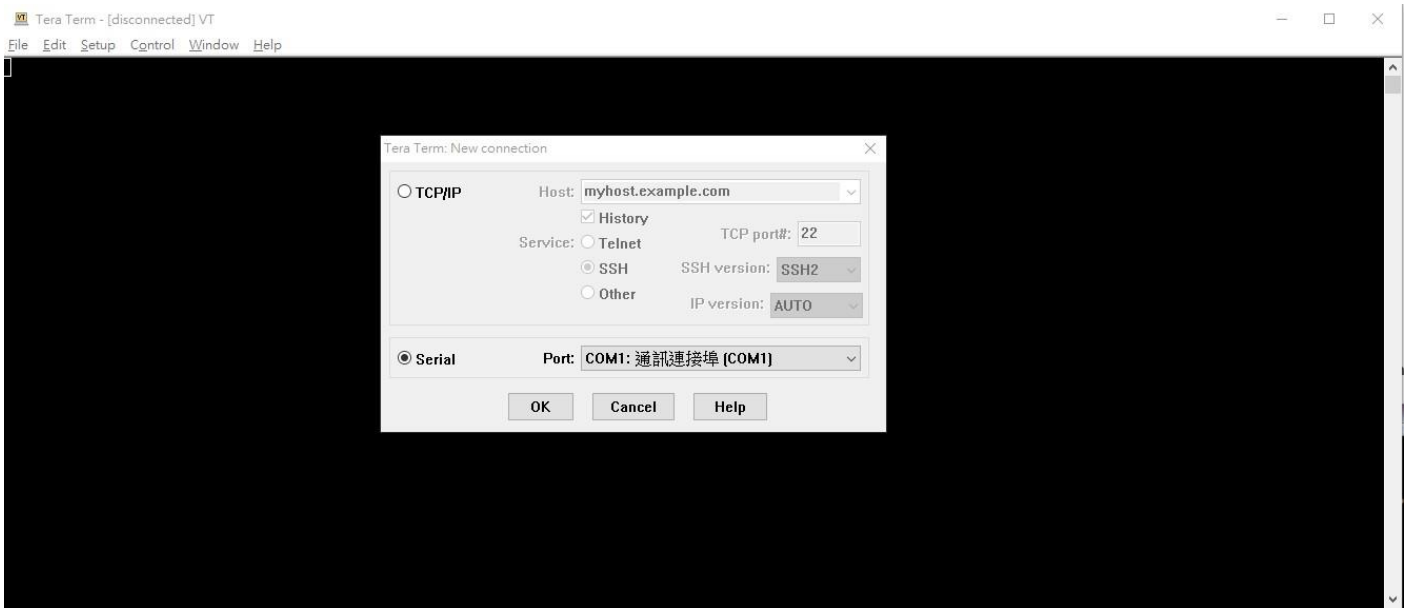


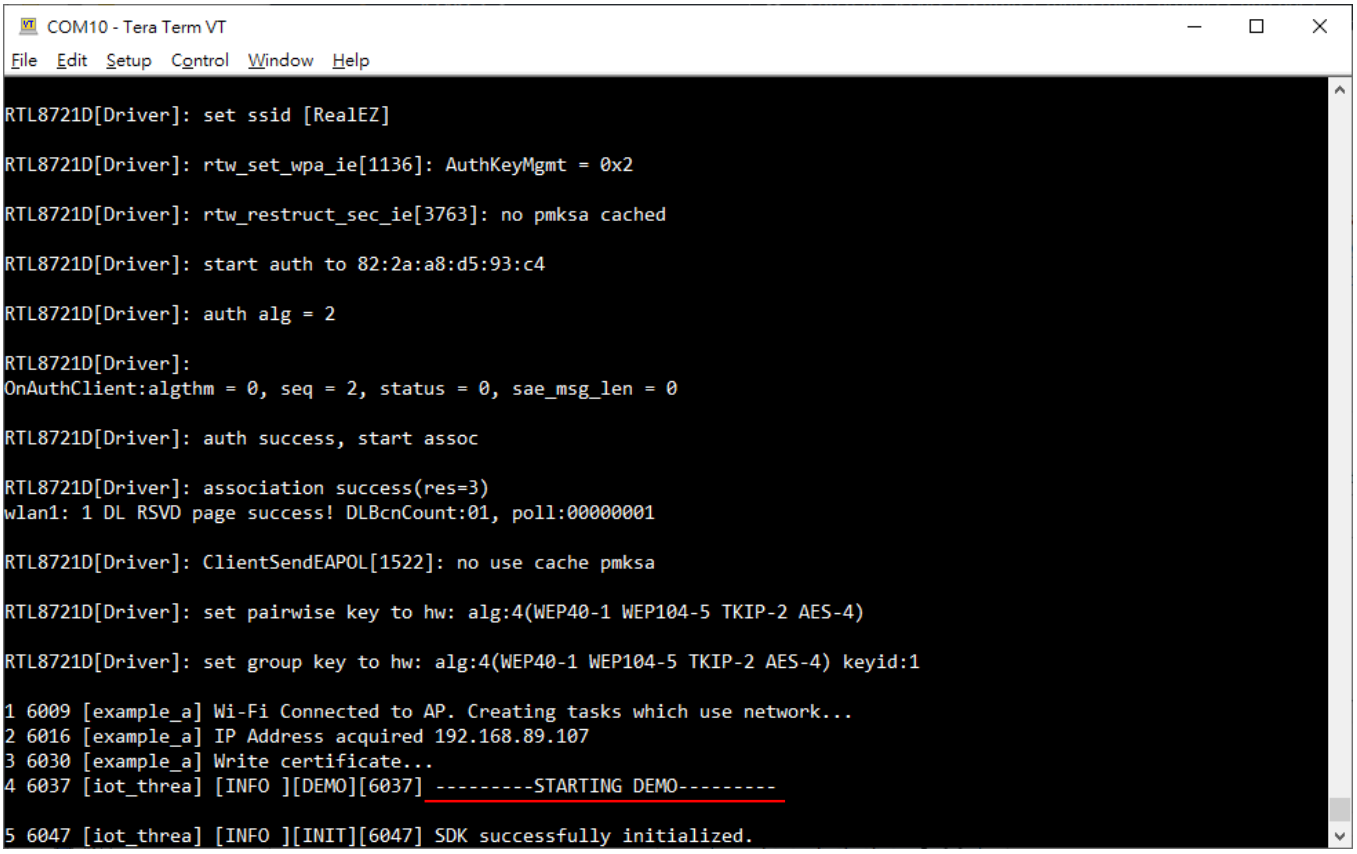
Fig 6-1 Hardware setup

The serial port is same with ImageTool that get from 5.3.1.1 step (3).

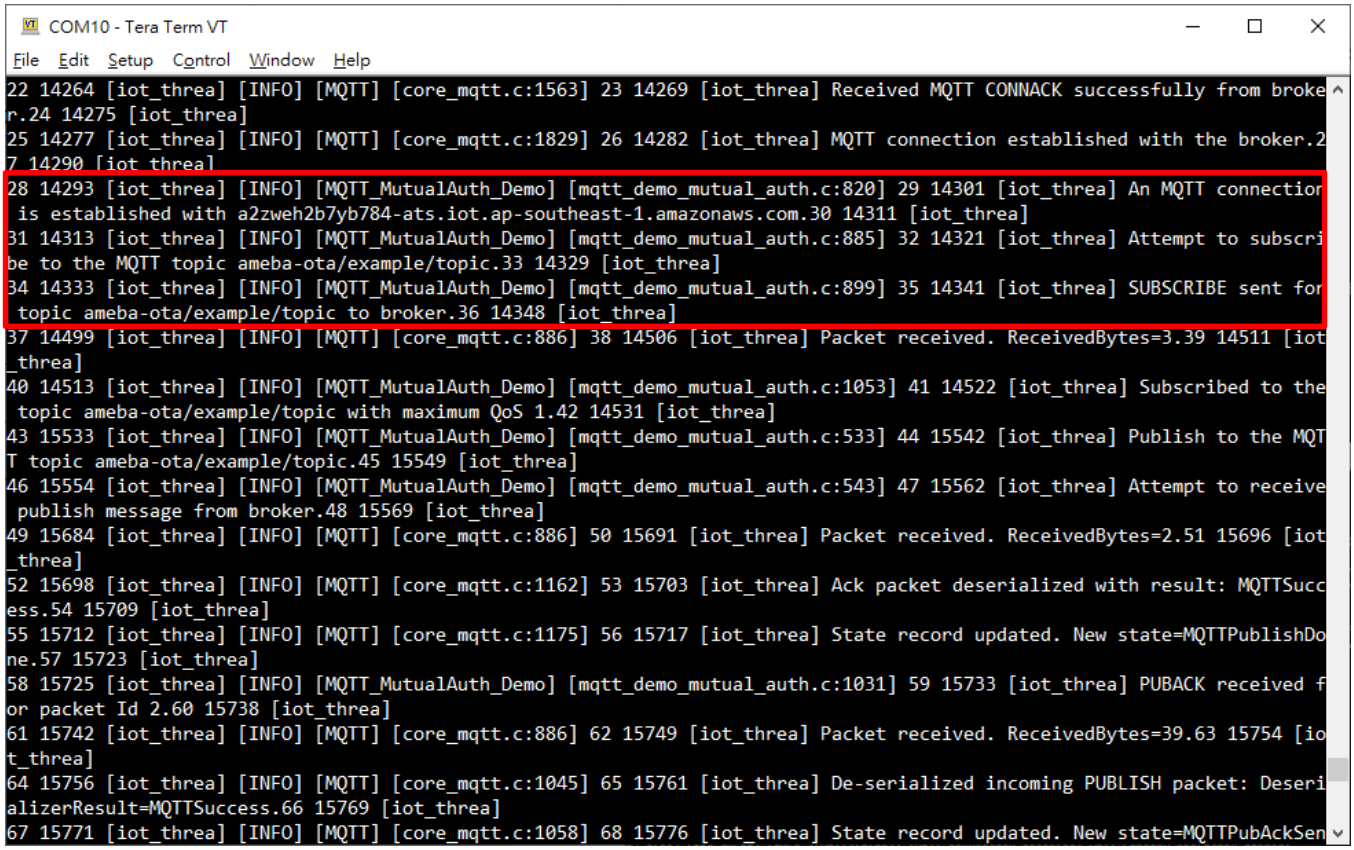


6.2 Run MQTT Demo

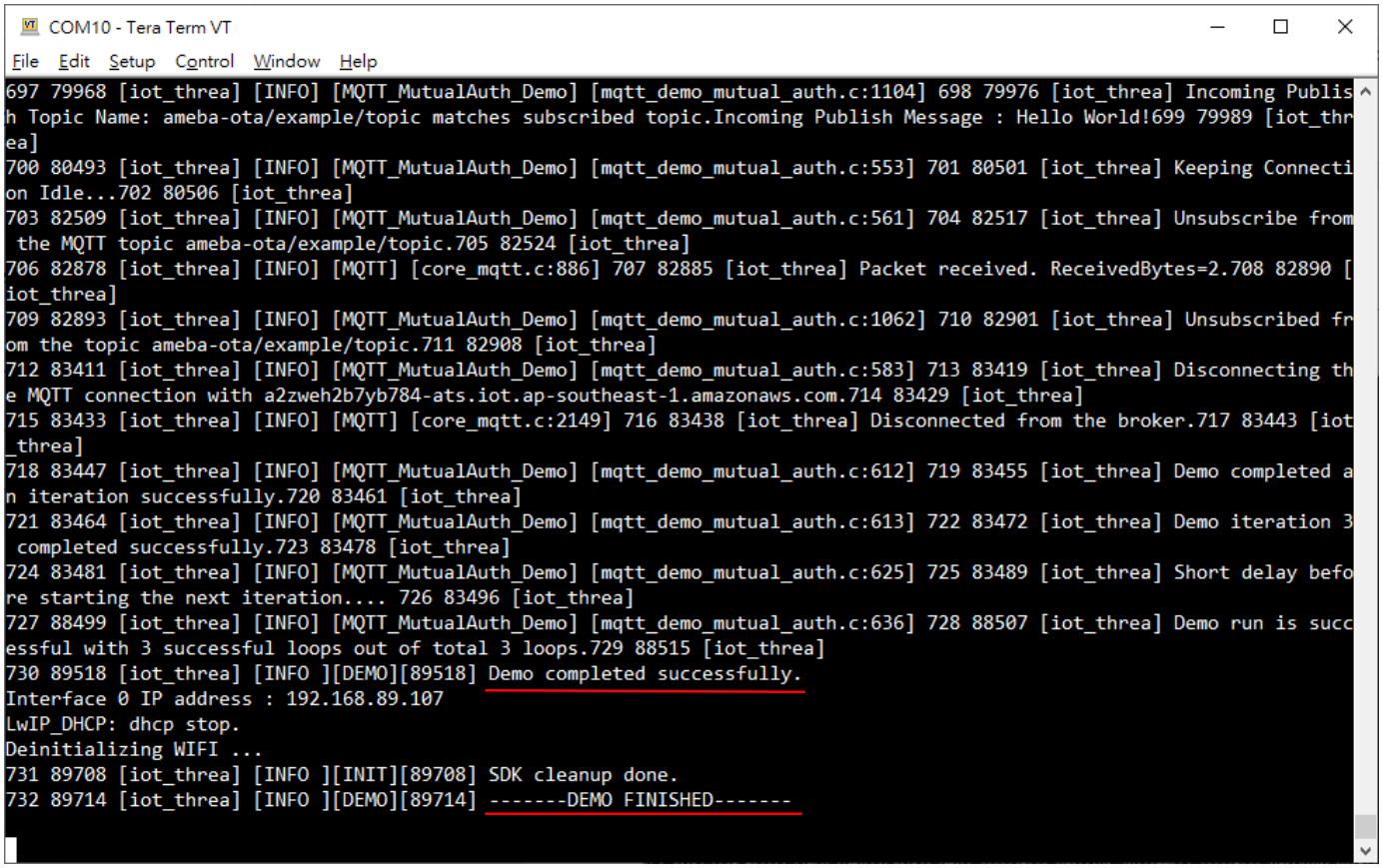
Default setting of SDK are enable MQTT demo. Once the AmebaD EVB has rebooted, the application will automatically start run MQTT demo and communicate to IoT Core.



...



...

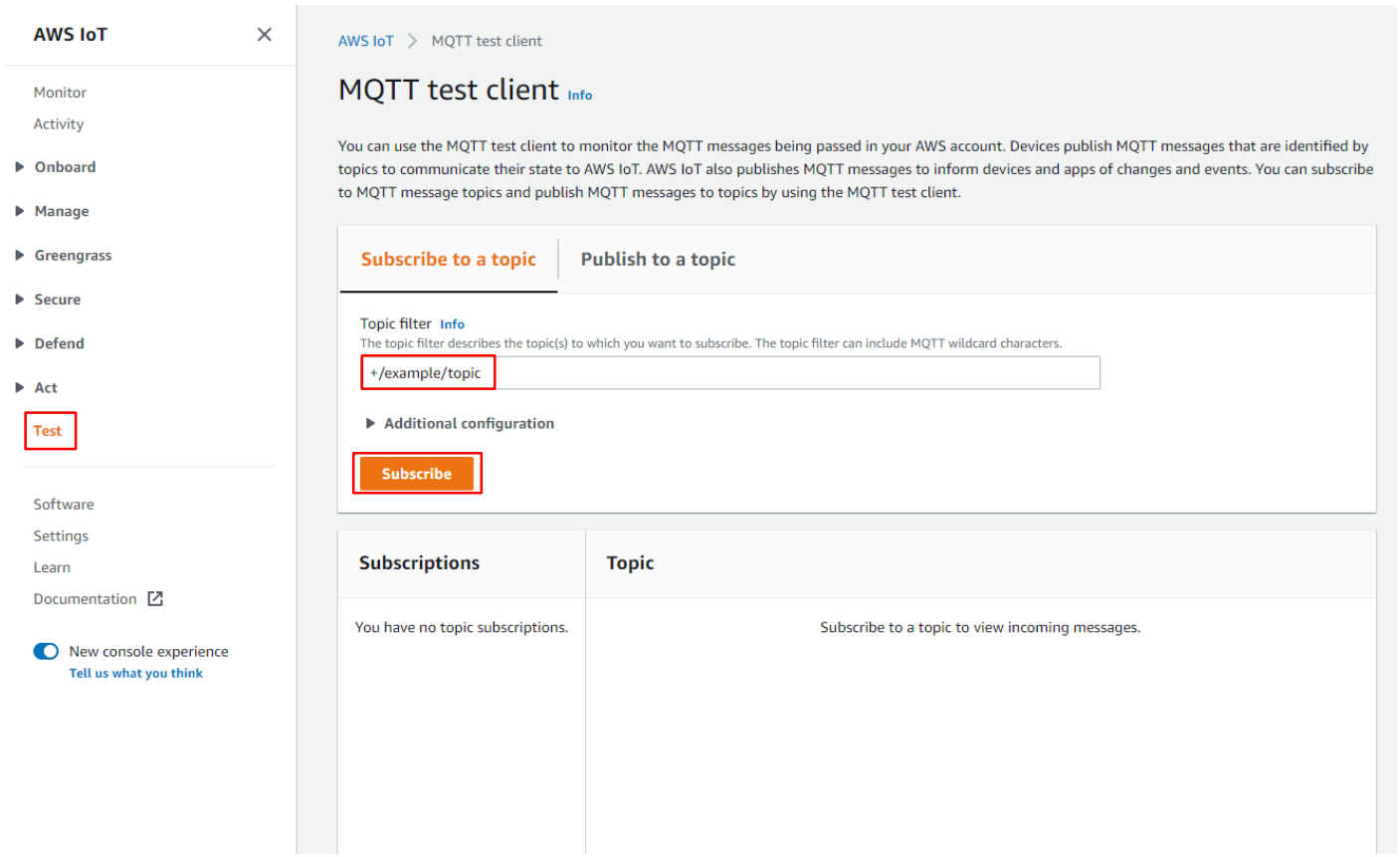


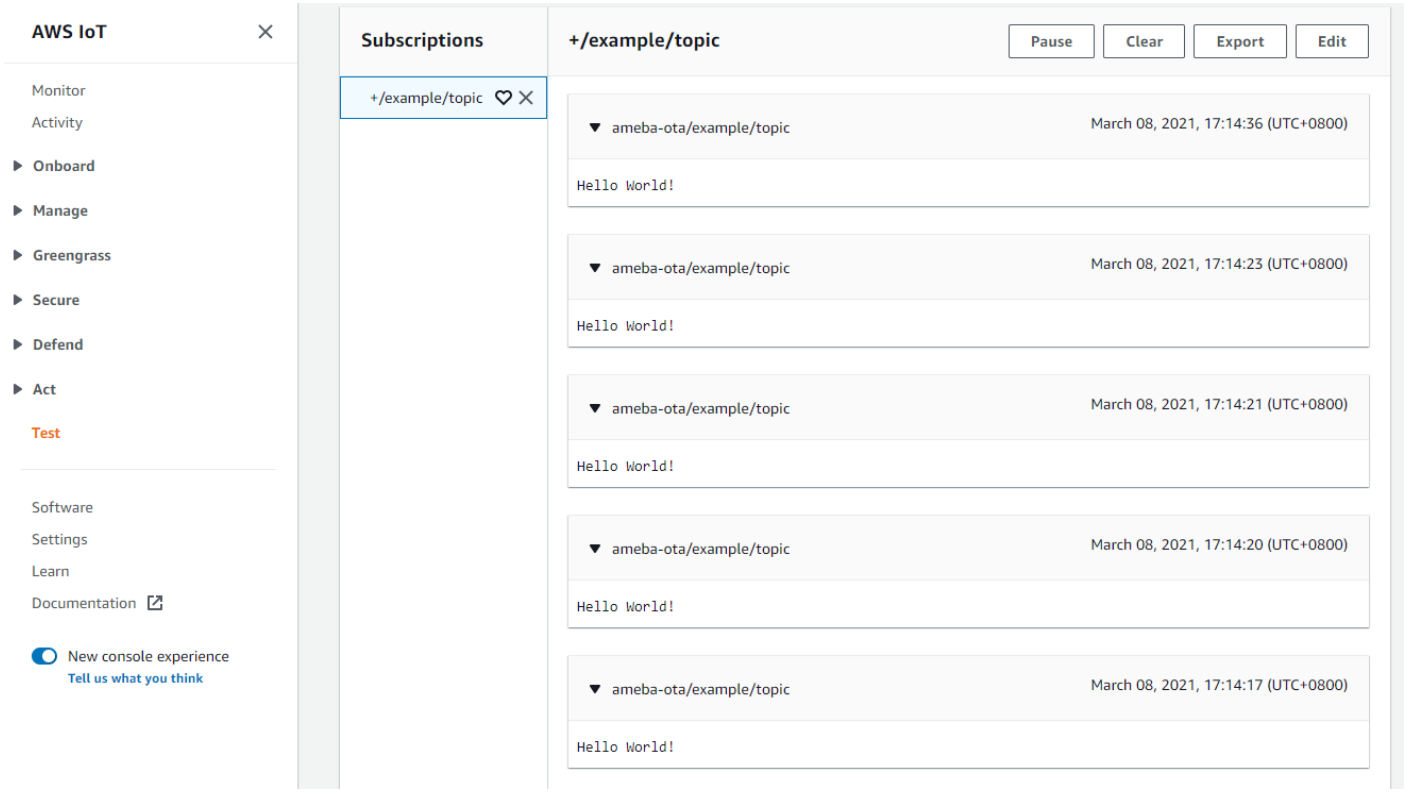
Monitor connection summary.

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.





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 component\common\application\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"
"-----END CERTIFICATE-----";

#endif
```

7.2 Set the App Version to Image File

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    0
#define APP_VERSION_MINOR    9
#define APP_VERSION_BUILD    2

#endif
```

Now, configure the **aws_demo_config.h** for OTA demo. Then, build the project and get the image file (.bin).

```
//#define CONFIG_CORE_MQTT_MUTUAL_AUTH_DEMO_ENABLED
#define CONFIG_OTA_UPDATE_DEMO_ENABLED
//#define CONFIG_DEVICE_SHADOW_DEMO_ENABLED
```

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

- One image with older version should be downloaded to your AmebaD, 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 **OTA-ALL.bin**:

1. The **km0_km4_image2.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 **OTA_ALL.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_D.py**, that provided in the folder (component\common \application\amazon\amazon_ota_tools\python_custom_ecdsa_D.py)

The python script requires the following pre-requisites to work

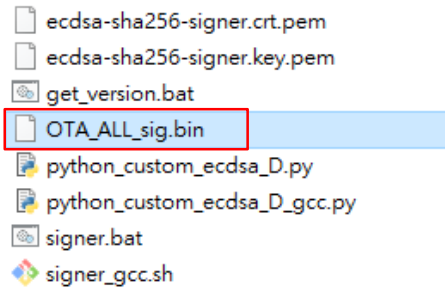
1. Python must be installed in the windows system with version 3.7.x or later
2. Pyopenssl library must be installed using 'pip install pyopenssl'
3. The ECDSA signing key and the Certificate pair must be present in the same folder as the python script and must be named 'ecdsa-sha256-signer.key.pem' 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_D.py**
- shell command after GCC build : **$ python3 python_custom_ecdsa_D_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 **OTA_ALL_sig.bin** inside the component\common\application\amazon\amazon_ota_tools folder.

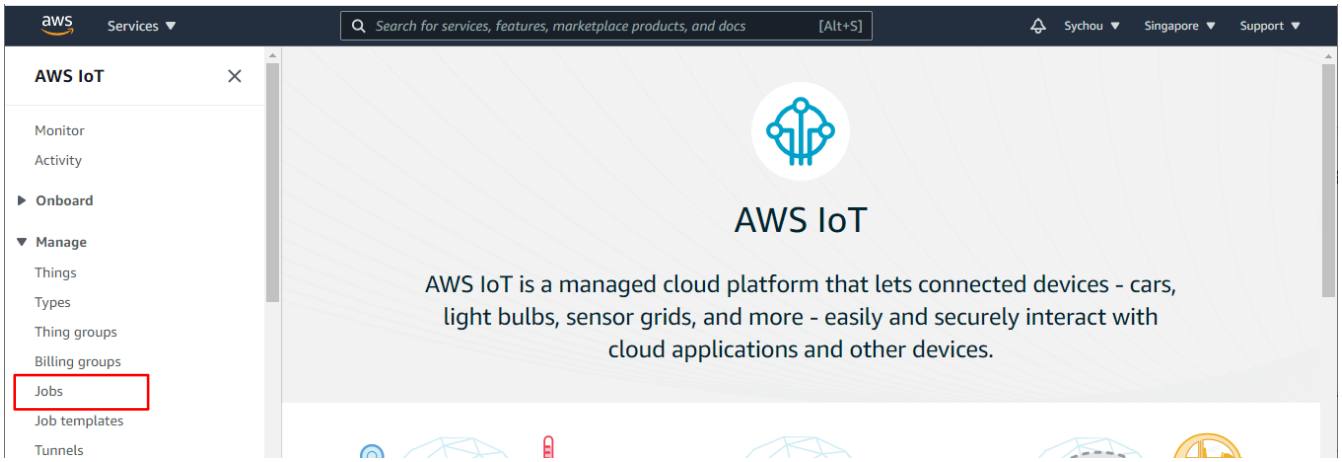


After getting the custom signed **OTA_ALL_sig.bin**, you can upload it to the S3 bucket.

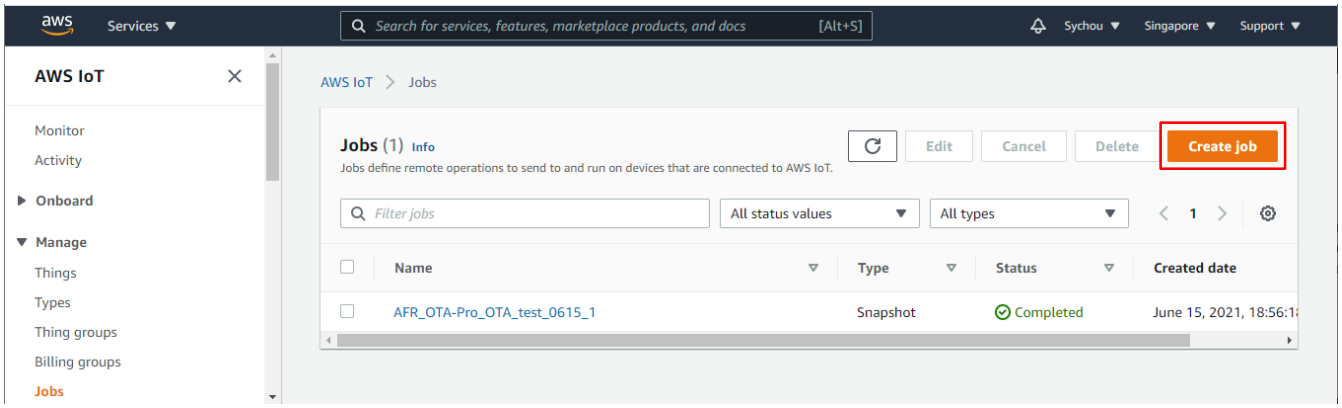
7.4 How to Trigger a Custom Signed OTA Job in Amazon AWS IOT Core

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 AmebaD:

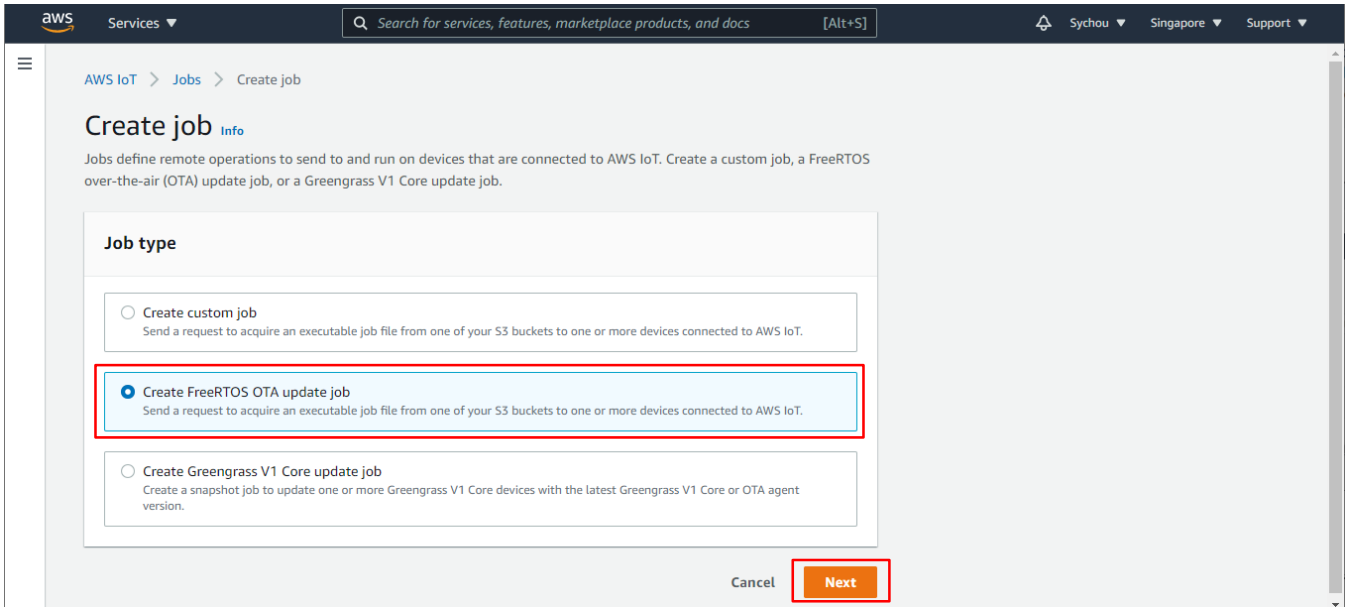
Step 1. Click on 'Create OTA update job', select your job type and then click next.



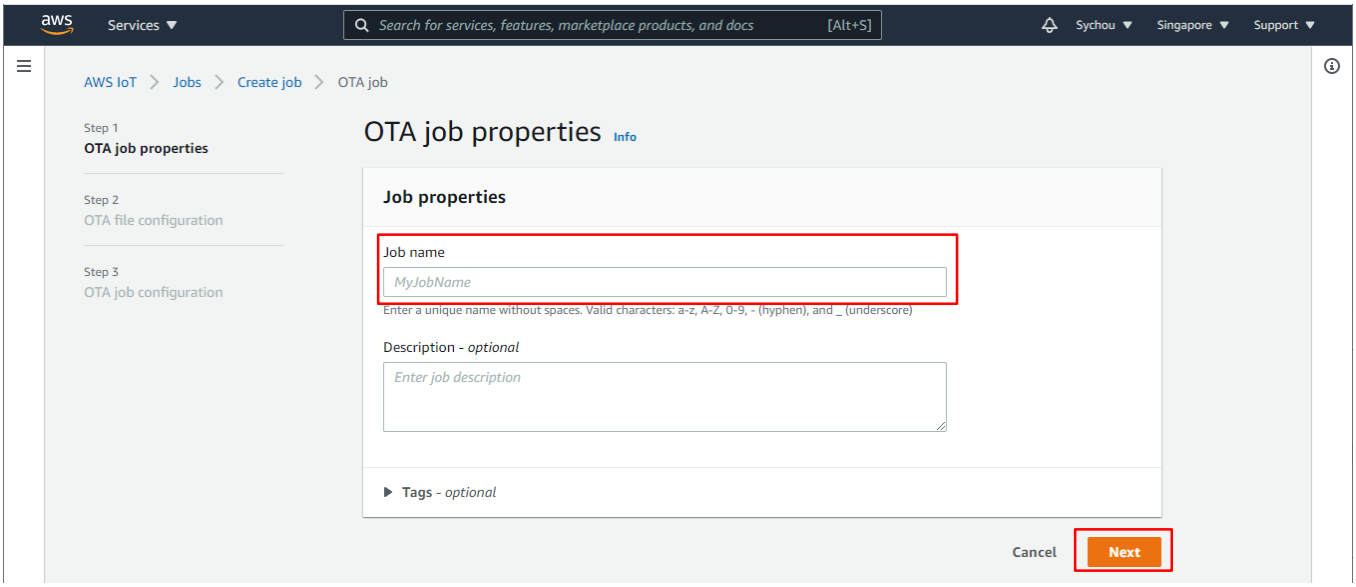
...



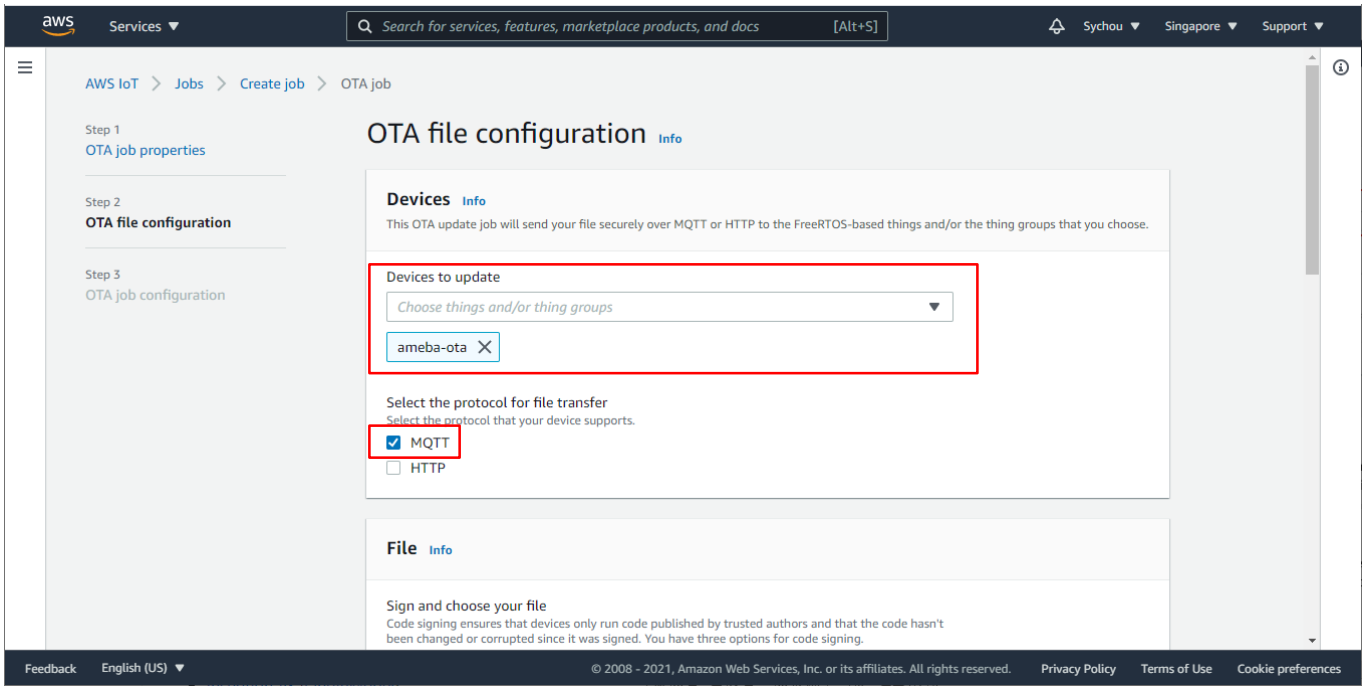
...



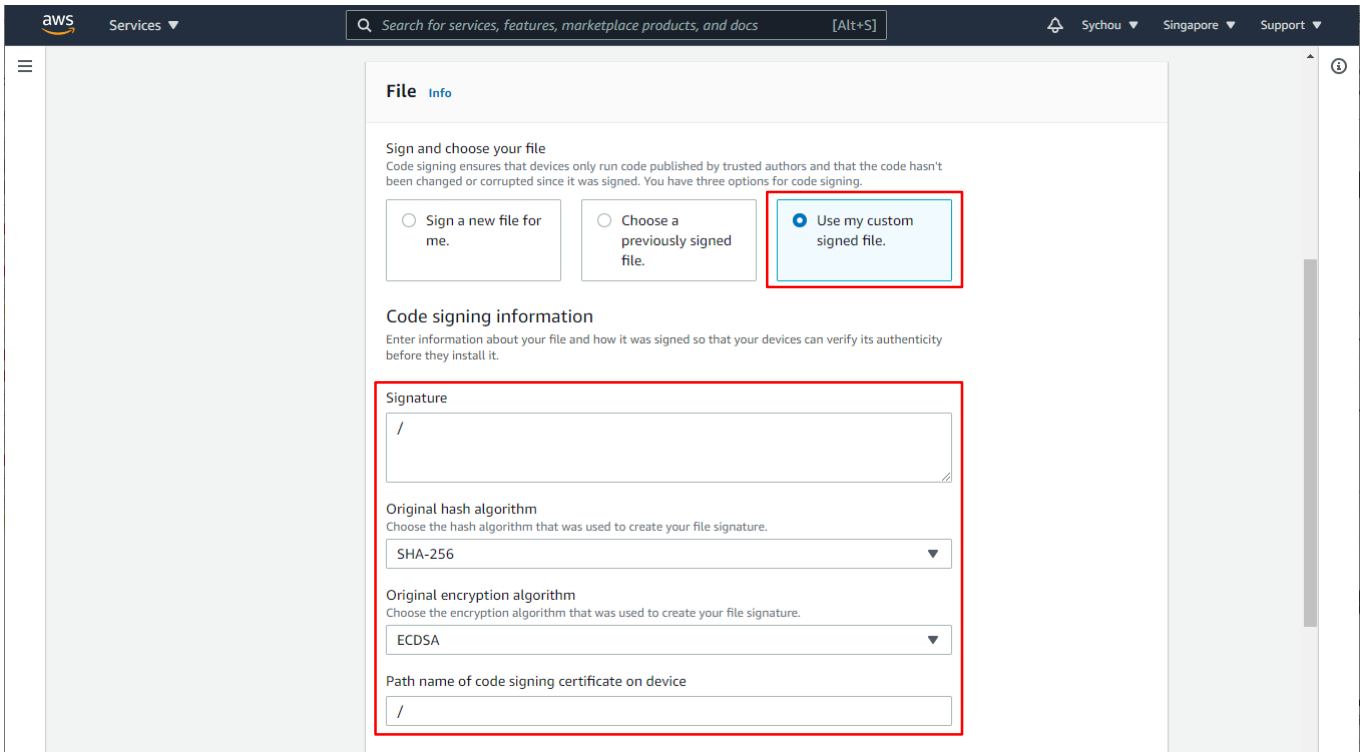
Step 2. For Job properties, give a unique name to your OTA job, then click next.



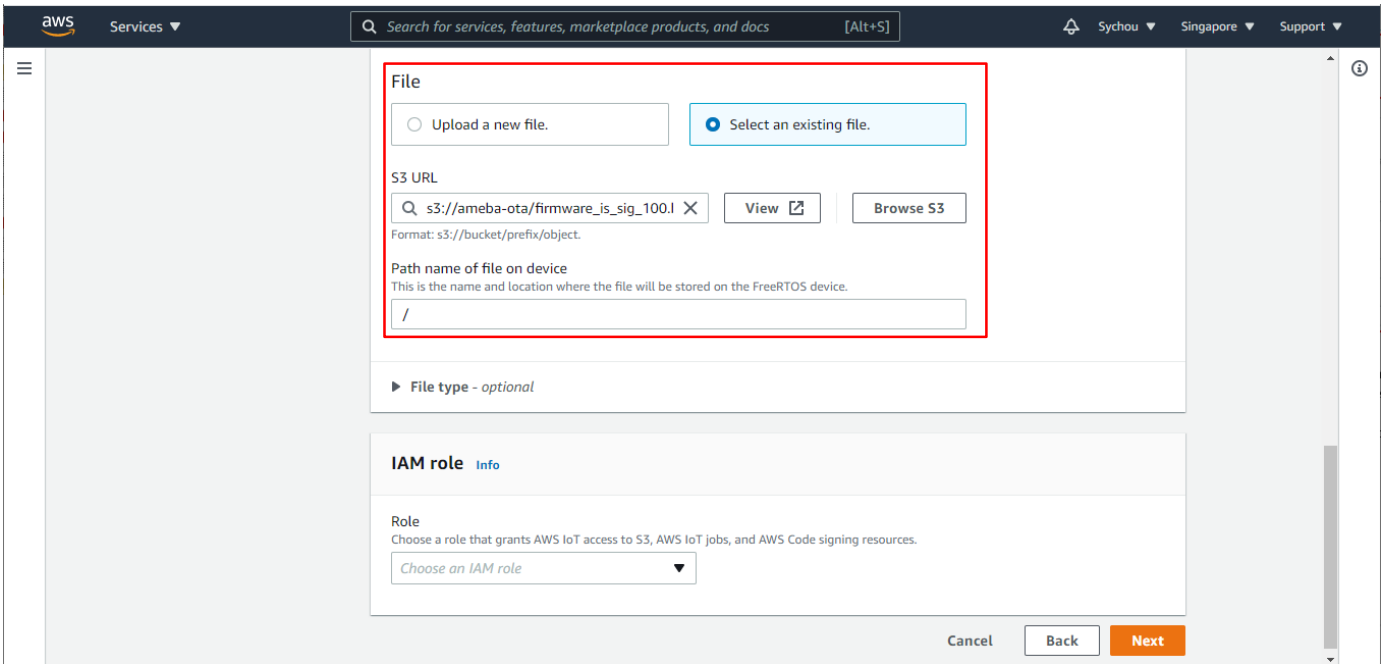
Step 3. In the following page, choose your device to update and select the protocol for file transfer



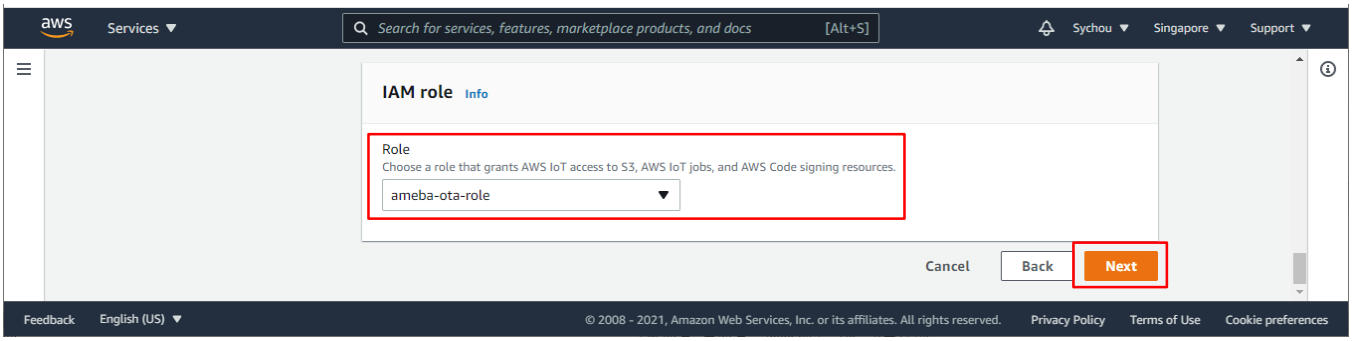
Step 4. In the following page, choose the option 'Use my custom signed firmware image'.
In the signature field just enter '/'. Choose hash algorithm as 'SHA-256'. Choose encryption algorithm as 'ECDSA'.
In "pathname of code signing certificate", enter '/'



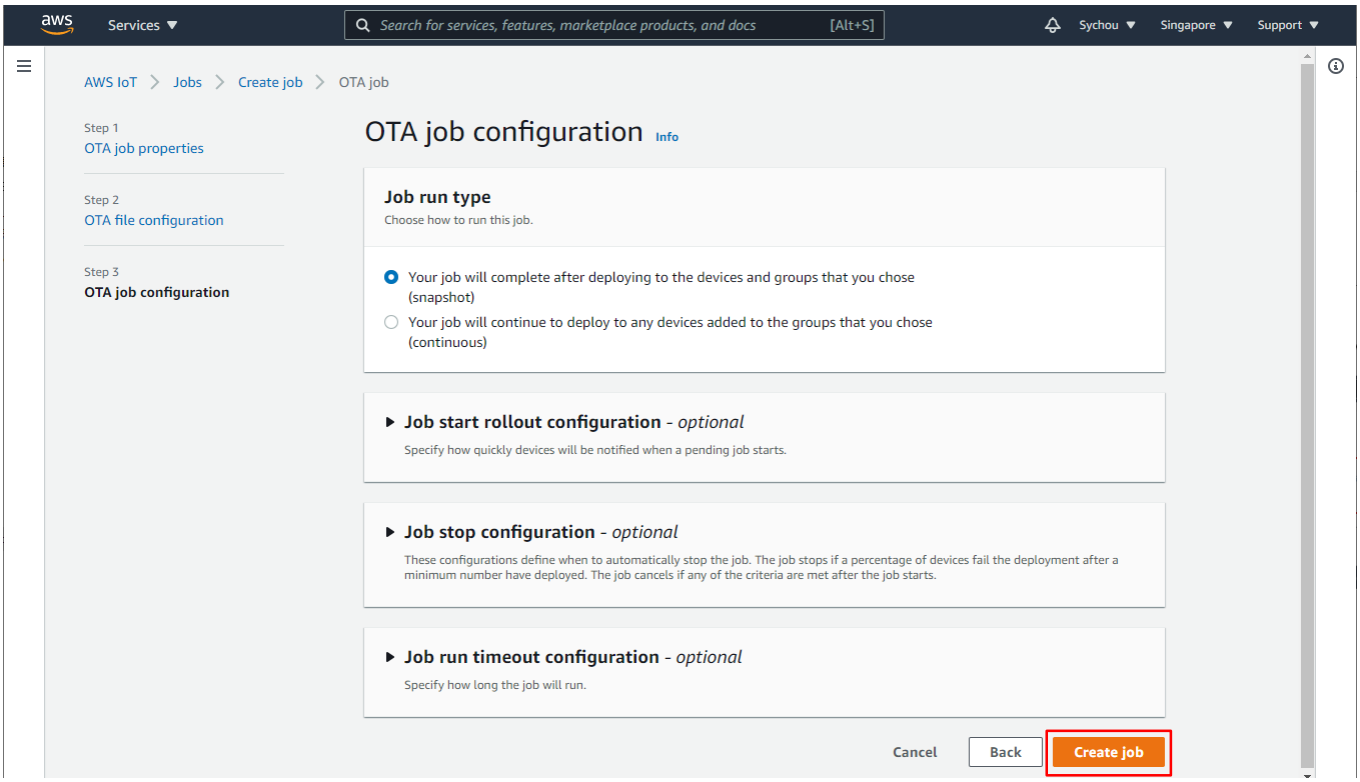
Step 5. Choose your custom signed firmware binary that was generated by the python script from S3 bucket.
In "Pathname of file on device", enter '/'



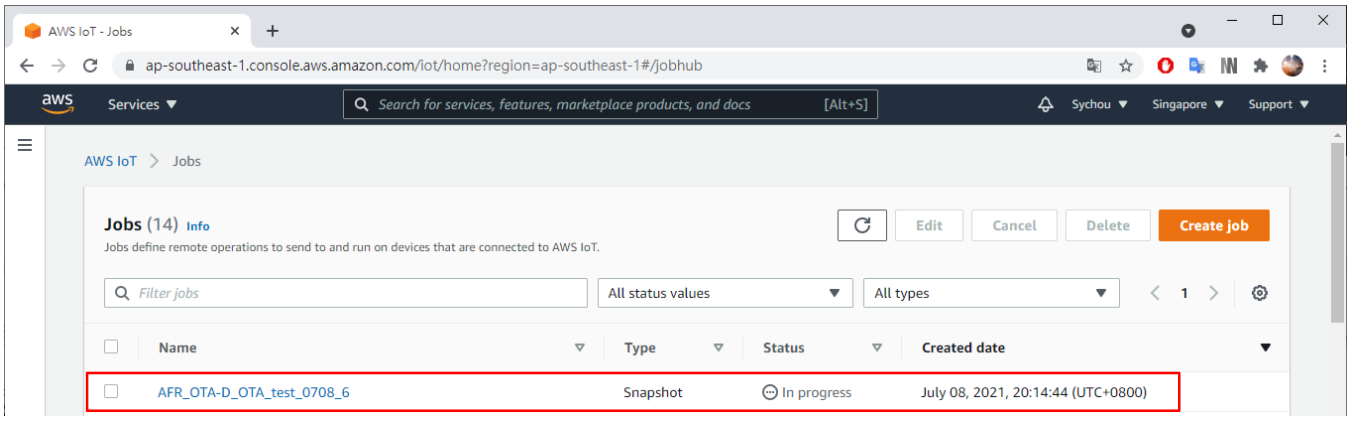
Step 6. Choose the IAM role for OTA update job. (This is the same IAM role as any OTA update job)



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



...



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 AmebaD to request the update.

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

In the beginning, we can check the app version (0.9.2) of this running firmware, and the OTA process by the job ID:

```
Interface 0 IP address : 192.168.0.115 10615 [iot_threa] [INFO ][DEMO][10615] Successfully initialized the demo. Network type for the demo: 1
6 10625 [iot_threa] [INFO ][MQTT][10625] MQTT library successfully initialized.
7 10633 [iot_threa] [INFO ][DEMO][10633] OTA demo version 0.9.2

8 10639 [iot_threa] [INFO ][DEMO][10639] Connecting to broker...

9 10646 [iot_threa] [INFO ][DEMO][10646] MQTT demo client identifier is ameba-ota (length 9).
10 13604 [iot_threa] [INFO ][MQTT][13604] Establishing new MQTT connection.
11 13614 [iot_threa] [INFO ][MQTT][13614] (MQTT connection 1003f400, CONNECT operation 1003f5a0) Waiting for operation completion.
12 13751 [NetRecv] [INFO] [MQTT] [core_mqtt_serializer.c:970] 13 13757 [NetRecv] CONNACK session present bit not set.14 13762 [NetRecv]
15 13763 [NetRecv] [INFO] [MQTT] [core_mqtt_serializer.c:912] 16 13769 [NetRecv] Connection accepted.17 13772 [NetRecv]
18 13774 [iot_threa] [INFO ][MQTT][13774] (MQTT connection 1003f400, CONNECT operation 1003f5a0) Wait complete with result SUCCESS.
19 13789 [iot_threa] [INFO ][MQTT][13789] New MQTT connection 1001e918 established.
20 13798 [iot_threa] [OTA_AgentInit_internal] OTA Task is Ready.
ota_imagestate = AWS_OTA_IMAGE_STATE_FLAG_IMG_NEW
21 13808 [OTA Agent] [prvPAL_GetPlatformImageState_rtl8721d] Image current state (0x02)
22 13816 [OTA Agent] [prvExecuteHandler] Called handler. Current State [Ready] Event [Start] New state [RequestingJob]
23 13829 [OTA Agent] [INFO ][MQTT][13829] (MQTT connection 1003f400) SUBSCRIBE operation scheduled.
24 13838 [OTA Agent] [INFO ][MQTT][13838] (MQTT connection 1003f400, SUBSCRIBE operation 100405e0) Waiting for operation completion.
25 14110 [OTA Agent] [INFO ][MQTT][14110] (MQTT connection 1003f400, SUBSCRIBE operation 100405e0) Wait complete with result SUCCESS.
26 14122 [OTA Agent] [prvSubscribeToJobNotificationTopics] OK: $aws/things/ameba-ota/jobs/$next/get/accepted
27 14133 [OTA Agent] [INFO ][MQTT][14133] (MQTT connection 1003f400) SUBSCRIBE operation scheduled.
28 14142 [OTA Agent] [INFO ][MQTT][14142] (MQTT connection 1003f400, SUBSCRIBE operation 1003f780) Waiting for operation completion.
29 14274 [OTA Agent] [INFO ][MQTT][14274] (MQTT connection 1003f400, SUBSCRIBE operation 1003f780) Wait complete with result SUCCESS.
30 14286 [OTA Agent] [prvSubscribeToJobNotificationTopics] OK: $aws/things/ameba-ota/jobs/notify-next
31 14295 [OTA Agent] [prvRequestJob_Mqtt] Request #0
32 14302 [OTA Agent] [INFO ][MQTT][14302] (MQTT connection 1003f400) MQTT PUBLISH operation queued.
33 14311 [OTA Agent] [INFO ][MQTT][14311] (MQTT connection 1003f400, PUBLISH operation 1003f780) Waiting for operation completion.
34 14546 [OTA Agent] [INFO ][MQTT][14546] (MQTT connection 1003f400, PUBLISH operation 1003f780) Wait complete with result SUCCESS.
35 14558 [OTA Agent] [prvExecuteHandler] Called handler. Current State [RequestingJob] Event [RequestJobDocument] New state [WaitingForJob]
36 14576 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ clientToken: 0:ameba-ota ]
37 14584 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ timestamp: 1625746566 ]
38 14592 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ jobId: AFR_OTA-D_OTA_test_0708_6 ]
39 14601 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ protocols: ["MQTT"] ]
40 14609 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ streamname: AFR_OTA-4b43e2fc-fa3f-4d97-ae88-1f9069ccbd6c ]
41 14620 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ filepath: / ]
42 14627 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ filesize: 808009 ]
43 14634 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ fileid: 0 ]
44 14641 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ certfile: / ]
45 14648 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: /"}]}}}}... ]
46 14657 [OTA Agent] [prvParseJobDoc] Job was accepted. Attempting to start transfer.
ota_imagestate = AWS_OTA_IMAGE_STATE_FLAG_IMG_NEW
47 14669 [OTA Agent] [prvPAL_GetPlatformImageState_rtl8721d] Image current state (0x02)
```

We can see that the OTA process start!

```
[OTA] Erase sector @ 0x1fd000
[OTA] Erase sector @ 0x1fe000
[OTA] Erase sector @ 0x1ff000
ota_imagestate = AWS_OTA_IMAGE_STATE_FLAG_IMG_NEW
48 15330 [OTA Agent] [prvPAL_GetPlatformImageState_rtl8721d] Image current state (0x02)
49 15338 [OTA Agent] [prvSetDataInterface] Data interface is set to MQTT.
50 15344 [OTA Agent] [prvProcessJobHandler] Setting OTA data inerface.
51 15351 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForJob] Event [ReceivedJobDocument] New state [CreatingFile]
52 15366 [OTA Agent] [INFO ][MQTT][15365] (MQTT connection 1003f400) SUBSCRIBE operation scheduled.
53 15374 [OTA Agent] [INFO ][MQTT][15374] (MQTT connection 1003f400, SUBSCRIBE operation 10040680) Waiting for operation completion.
54 15387 [iot_threa] [INFO ][DEMO][15387] State: RequestingJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0

55 15491 [OTA Agent] [INFO ][MQTT][15491] (MQTT connection 1003f400, SUBSCRIBE operation 10040680) Wait complete with result SUCCESS.
56 15503 [OTA Agent] [prvInitFileTransfer_Mqtt] OK: $aws/things/ameba-ota/streams/AFR_OTA-4b43e2fc-fa3f-4d97-ae88-1f9069ccbd6c/data/cbor
57 15515 [OTA Agent] [prvExecuteHandler] Called handler. Current State [CreatingFile] Event [CreateFile] New state [RequestingFileBlock]
58 15531 [OTA Agent] [INFO ][MQTT][15531] (MQTT connection 1003f400) MQTT PUBLISH operation queued.
59 15540 [OTA Agent] [prvRequestFileBlock_Mqtt] OK: $aws/things/ameba-ota/streams/AFR_OTA-4b43e2fc-fa3f-4d97-ae88-1f9069ccbd6c/get/cbor
60 15552 [OTA Agent] [prvExecuteHandler] Called handler. Current State [RequestingFileBlock] Event [RequestFileBlock] New state [WaitingForFi
61 15906 [OTA Agent] [prvIngestDataBlock] Received file block 1, size 1024
[OTA][prvPAL_WriteBlock_rtl8721d] iFileSize 808009, iOffset: 0x400: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x1063e0 (0x1003e0)
62 15924 [OTA Agent] [prvIngestDataBlock] Remaining: 789
63 15930 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForF
64 16069 [OTA Agent] [prvIngestDataBlock] Received file block 2, size 1024
[65 16077 [iot_threa] Error: No OTA data buffers available.
66 16082 [iot_threa] Error: No OTA data buffers available.
67 16088 [iot_threa] Error: No OTA data buffers available.
OTA][prvPAL_WriteBlock_rtl8721d] iFileSize 808009, iOffset: 0x800: iBlockSize: 0x400
[OTA] Write 1024 bytes @ 0x68 16114 [iot_threa] Error: No OTA data buffers available.
1067e0 (0x1007e0)
69 16122 [OTA Agent] [prvIngestDataBlock] Remaining: 788
70 16127 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForF
71 16141 [OTA Agent] [prvIngestDataBlock] Received file block 4, size 1024
[OTA][prvPAL_WriteBlock_rtl8721d] iFileSize 808009, iOffset: 0x1000: iBlockSize: 0x400
```

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.

```
3012 97410 [OTA Agent] [prvIngestDataBlock] Received final expected block of file.
3013 97418 [OTA Agent] [prvStopRequestTimer] Stopping request timer.
[OTA] Authenticating and closing file.
3014 97427 [OTA Agent] [prvPAL_CloseFile] Authenticating and closing file.
3015 97434 [OTA Agent] [prvPAL_CheckFileSignature_rtl8721d] Started sig-sha256-ecdsa signature verification, file: /
3016 97444 [OTA Agent] Assume Cert - No such file: /. Using header file
3017 98370 [OTA Agent] [prvIngestDataBlock] File receive complete and signature is valid.
3018 98378 [OTA Agent] [prvStopRequestTimer] Stopping request timer.
3019 98384 [OTA Agent] [prvPublishStatusMessage] Msg: {"status":"IN_PROGRESS","statusDetails":{"self_test":"ready","updatedBy":"0x90002"}}
3020 98400 [OTA Agent] [INFO ][MQTT][98400] (MQTT connection 1003f400) MQTT PUBLISH operation queued.
3021 98410 [OTA Agent] [INFO ][MQTT][98409] (MQTT connection 1003f400, PUBLISH operation 100407e0) Waiting for operation completion.
3022 98421 [iot_threa] [INFO ][DEMO][98421] State: WaitingForFileBlock  Received: 791   Queued: 0   Processed: 0   Dropped: 454

3023 98750 [OTA Agent] [INFO ][MQTT][98750] (MQTT connection 1003f400, PUBLISH operation 100407e0) Wait complete with result SUCCESS.
3024 98762 [OTA Agent] [prvPublishStatusMessage] 'IN_PROGRESS' to $aws/things/ameba-ota/jobs/AFR_OTA-D_OTA_test_0708_6/update
3025 98774 [OTA Agent] [INFO ][DEMO][98774] Received eOTA_JobEvent_Activate callback from OTA Agent.

3026 98783 [OTA Agent] [INFO ][MQTT][98783] (MQTT connection 1003f400) Disconnecting connection.
3027 98793 [OTA Agent] [INFO] [MQTT] [core_mqtt.c:2149] 3028 98798 [OTA Agent] Disconnected from the broker.3029 98803 [OTA Agent]
3030 98805 [OTA Agent] [INFO ][MQTT][98805] (MQTT connection 1003f400) Network connection closed.
3031 98814 [OTA Agent] [INFO ][DEMO][98814] Mqtt disconnected due to invoking diconnect function.

3032 99434 [iot_threa] [INFO ][DEMO][99434] State: WaitingForFileBlock  Received: 791   Queued: 0   Processed: 0   Dropped: 454

3033 99447 [iot_threa] [prvStopRequestTimer] Stopping request timer.
3034 99793 [OTA Agent] [INFO ][MQTT][99793] (MQTT connection 1003f400) Network connection destroyed.
[OTA] [prvPAL_ActivateNewImage_rtl8721d] Download new firmware 806912 bytes completed @ 0x8106000
[OTA] signature: 81958711, size = 806912, OtaTargetHdr.FileImgHdr.ImgLen = 806912

OTA image(08106000) checksum ok!!!

[change ota signature] Update OTA success![OTA] [prvPAL_ActivateNewImage_rtl8721d] 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 0.9.3, which is bigger than old one 0.9.2.

```
47 14119 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ certfile: / ]
48 14126 [OTA Agent] [prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: /"}]}}}}... ]
49 14135 [OTA Agent] [prvParseJobDoc] In self test mode.
50 14140 [OTA Agent] [prvValidateUpdateVersion] The update version is newer than the version on device.
51 14150 [OTA Agent] [prvParseJobDoc] Setting image state to Testing for file ID 0
52 14157 [OTA Agent] [prvSetImageState_rtl8721d] Testing image.
53 14163 [OTA Agent] [prvPublishStatusMessage] Msg: {"status":"IN_PROGRESS","statusDetails":{"self_test":"active","updatedBy":"0x90003"}}
54 14178 [OTA Agent] [INFO ][MQTT][14178] (MQTT connection 1003f400) MQTT PUBLISH operation queued.
55 14187 [OTA Agent] [INFO ][MQTT][14187] (MQTT connection 1003f400, PUBLISH operation 100405a0) Waiting for operation completion.
56 14426 [OTA Agent] [INFO ][MQTT][14426] (MQTT connection 1003f400, PUBLISH operation 100405a0) Wait complete with result SUCCESS.
57 14440 [OTA Agent] [prvPublishStatusMessage] 'IN_PROGRESS' to $aws/things/ameba-ota/jobs/AFR_OTA-D_OTA_test_0708_6/update
58 14451 [OTA Agent] [prvExecuteHandler] Called handler. Current State [WaitingForJob] Event [ReceivedJobDocument] New state [CreatingFile]
59 14463 [OTA Agent] [prvInSelfTestHandler] prvInSelfTestHandler, platform is in self-test.
ota_imagestate = AWS_OTA_IMAGE_STATE_FLAG_PENDING_COMMIT
60 14476 [OTA Agent] [prvPAL_GetPlatformImageState_rtl8721d] Image current state (0x01)
61 14484 [OTA Agent] [INFO ][DEMO][14484] Received eOTA_JobEvent_StartTest callback from OTA Agent.

62 14493 [OTA Agent] [prvSetImageState_rtl8721d] Accepted and committed final image.
63 14501 [OTA Agent] [prvStopSelfTestTimer] Stopping the self test timer.
64 14508 [OTA Agent] [prvPublishStatusMessage] Msg: {"status":"SUCCEEDED","statusDetails":{"reason":"accepted v0.9.3"}}
65 14521 [OTA Agent] [INFO ][MQTT][14521] (MQTT connection 1003f400) MQTT PUBLISH operation queued.
66 14530 [OTA Agent] [INFO ][MQTT][14530] (MQTT connection 1003f400, PUBLISH operation 100405a0) Waiting for operation completion.
67 14542 [iot_threa] [INFO ][DEMO][14542] State: RequestingJob  Received: 1   Queued: 0   Processed: 0   Dropped: 0

68 14779 [OTA Agent] [INFO ][MQTT][14779] (MQTT connection 1003f400, PUBLISH operation 100405a0) Wait complete with result SUCCESS.
69 14791 [OTA Agent] [prvPublishStatusMessage] 'SUCCEEDED' to $aws/things/ameba-ota/jobs/AFR_OTA-D_OTA_test_0708_6/update
70 14802 [OTA Agent] [prvExecuteHandler] Called handler. Current State [CreatingFile] Event [StartSelfTest] New state [WaitingForJob]
71 15554 [iot_threa] [INFO ][DEMO][15554] State: CreatingFile  Received: 1   Queued: 0   Processed: 0   Dropped: 0
```

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

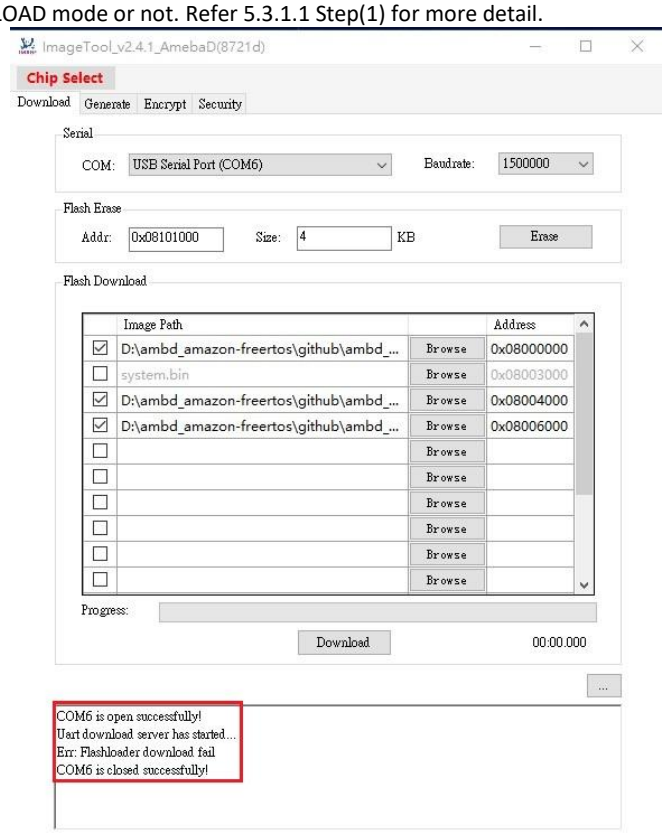
8 Troubleshooting

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 Flashloader download fail

Please check device in UART_DOWNLOAD mode or not. Refer 5.3.1.1 Step(1) for more detail.



8.2 ERROR: Invalid Key

Please check **WIFI_SSID** and **WIFI_PASSWORD** in in amazon-freertos /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 ...
```

8.3 Failed to establish new MQTT connection

Please check **clientcredentialMQTT_BROKER_ENDPOINT** in amazon-freertos /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-------
```

8.4 TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in amazon-freertos/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-------
```