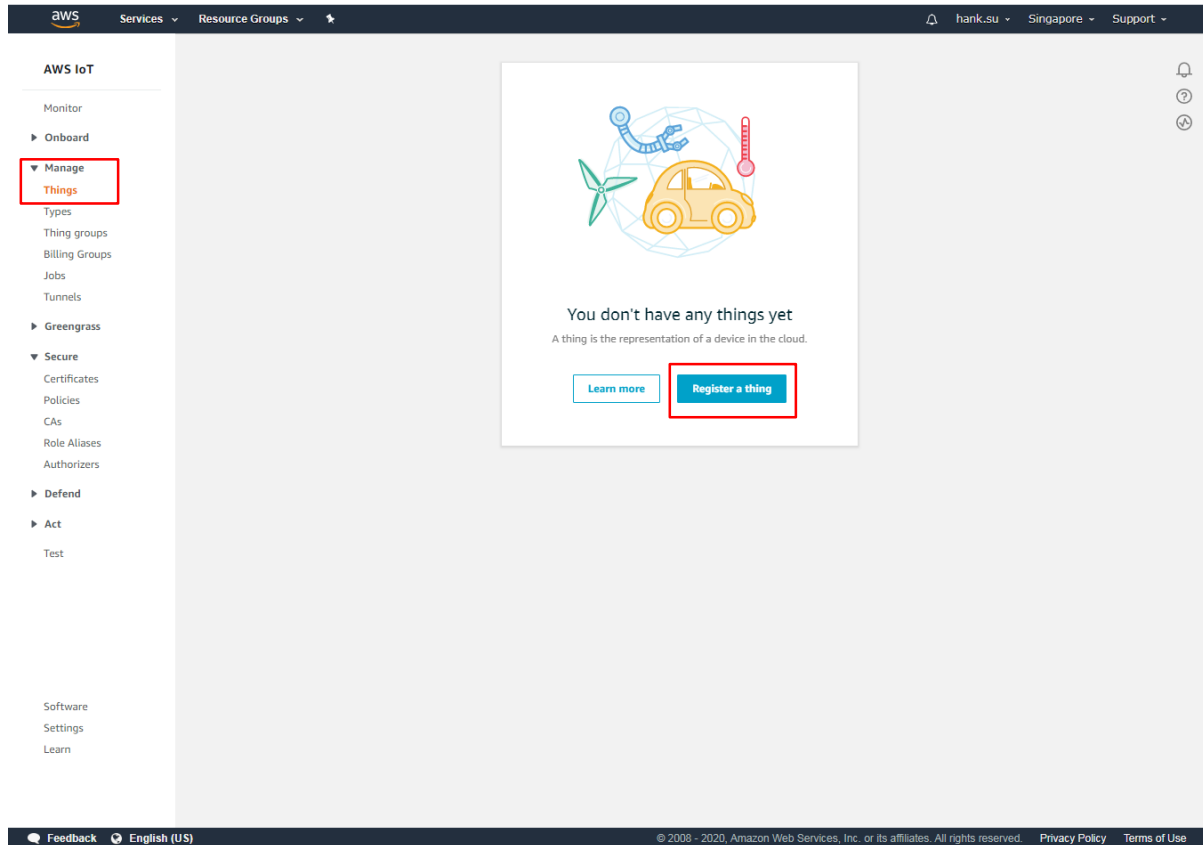


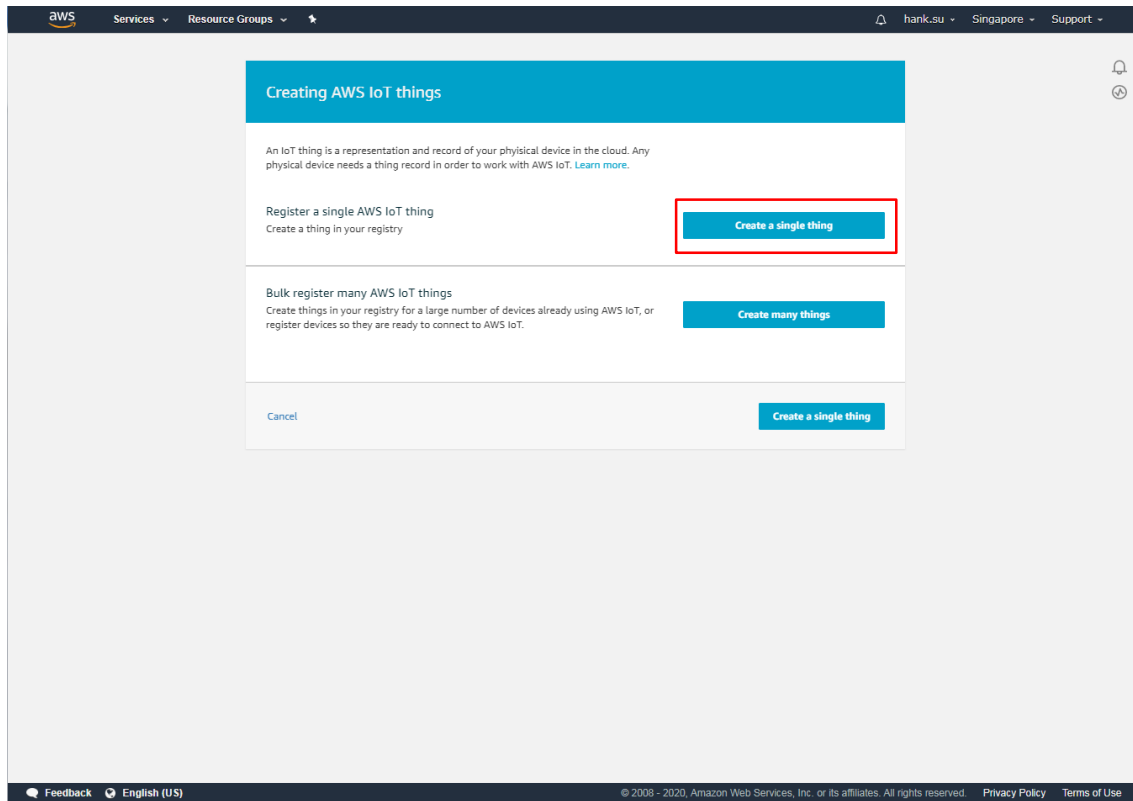
AmebaD Amazon FreeRTOS Getting Started Guide

1 Configure AWS IoT Core

1.1 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

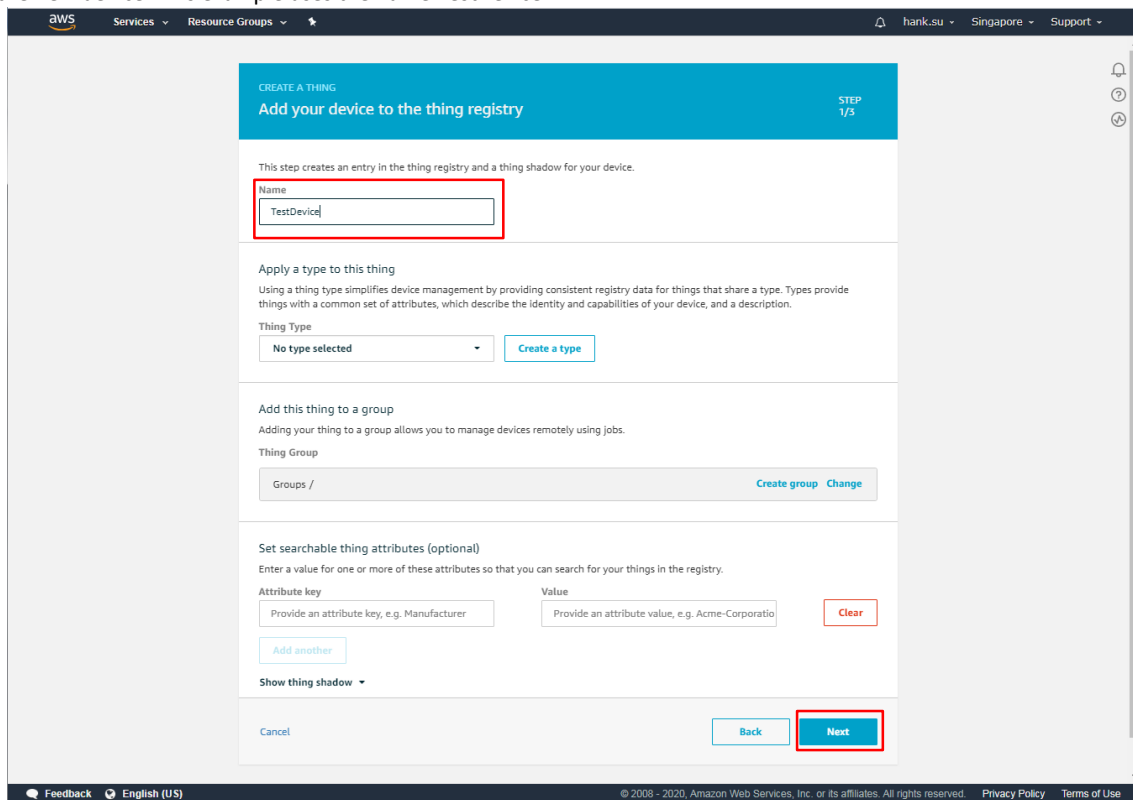
Create a single thing

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**

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.

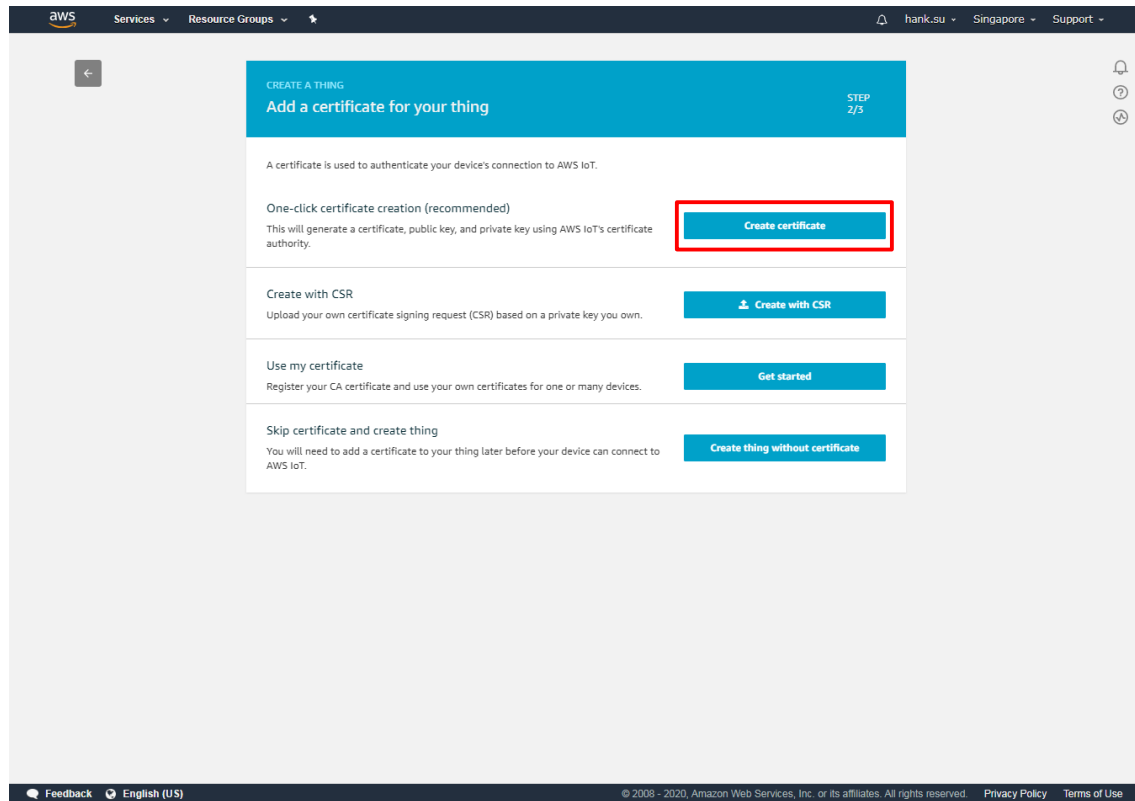
Attribute key
Provide an attribute key, e.g. Manufacturer

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

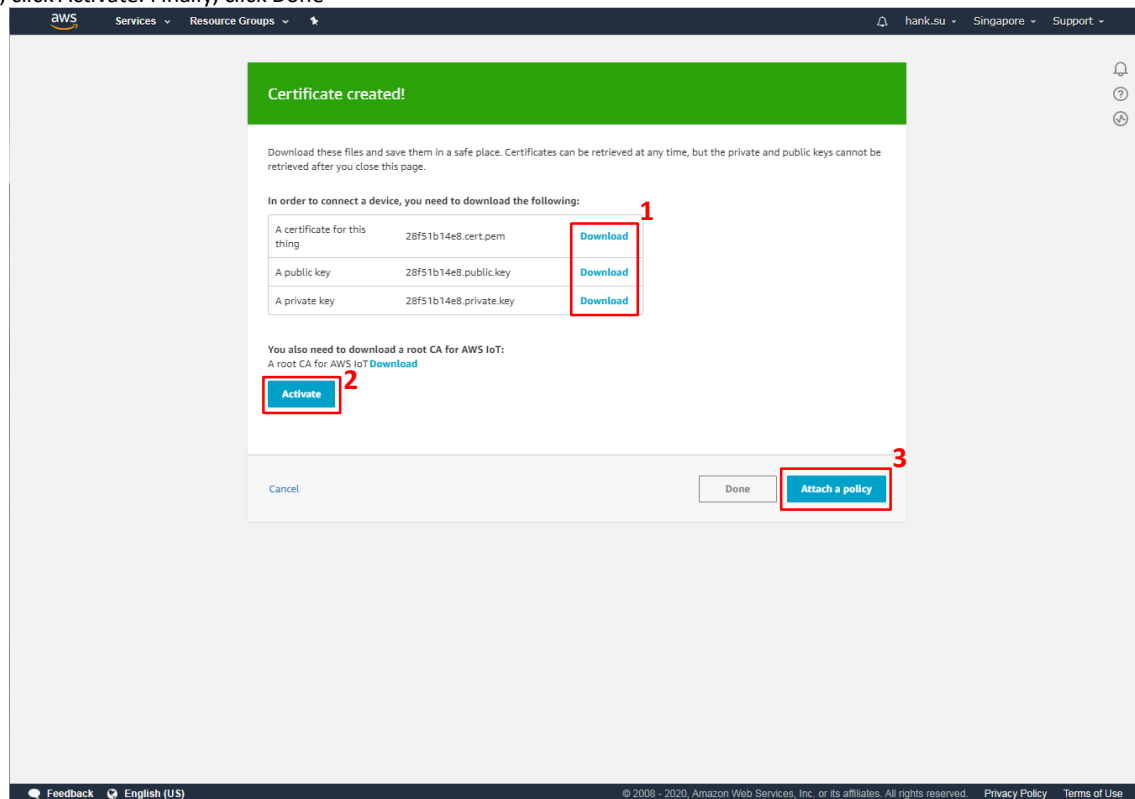
Add another

Show thing shadow ▾

Cancel **Back** **Next**

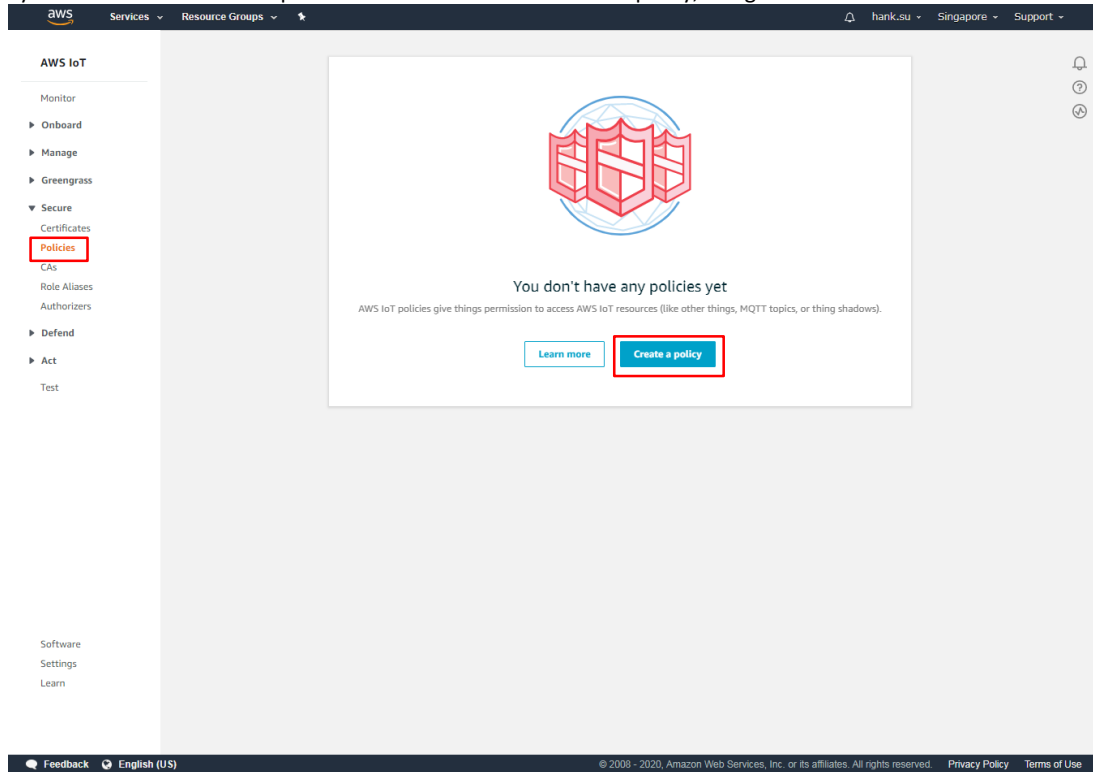


Download the certificate, public key, and private key for the device by clicking Download. Once all the certificate and keys have been downloaded, click Activate. Finally, click Done

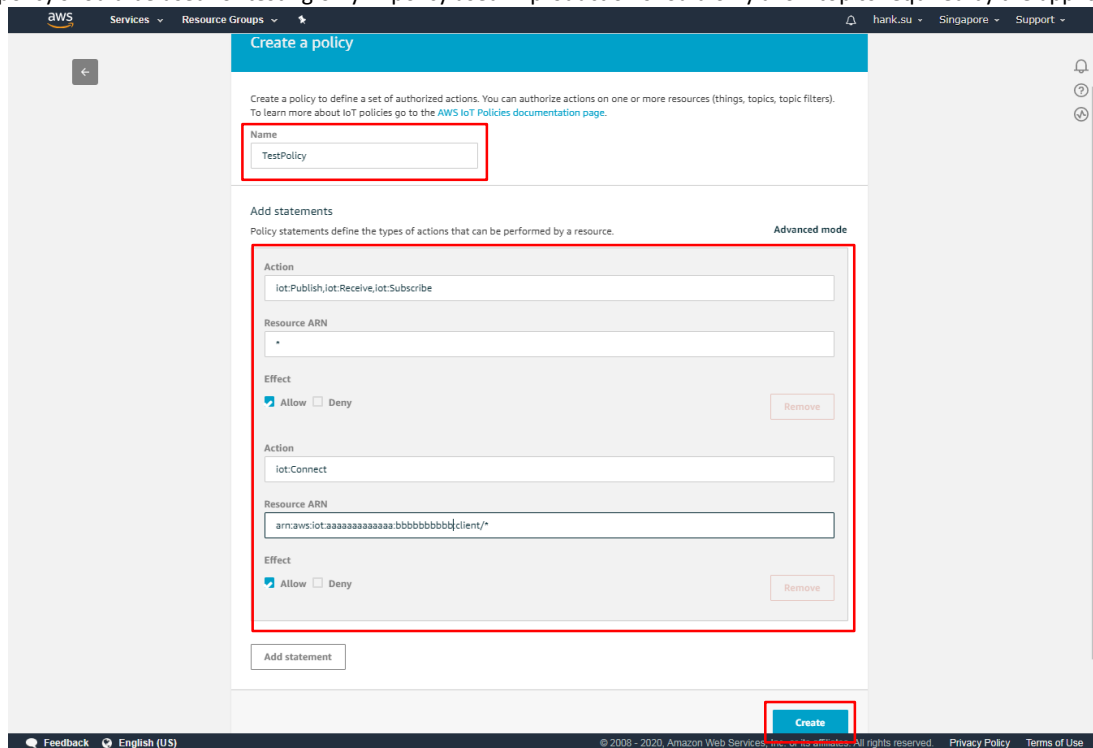


1.2 Create a policy

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

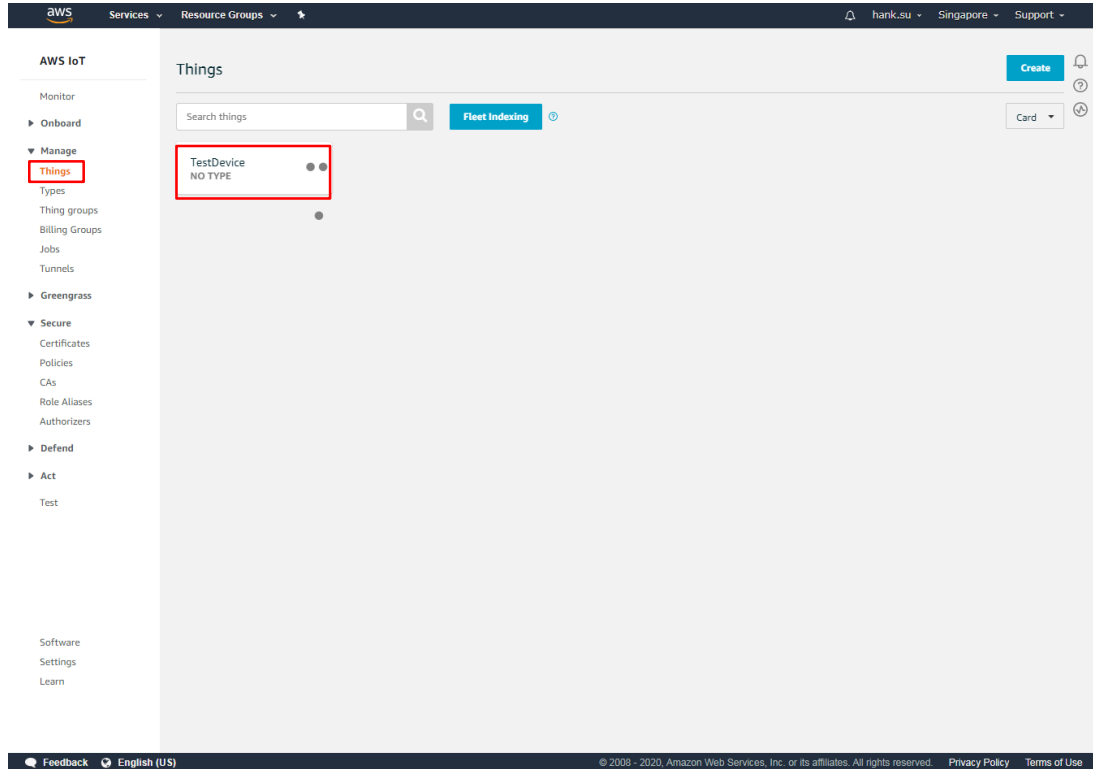


This policy should be used for testing only. A policy used in production should only allow topics required by the application.

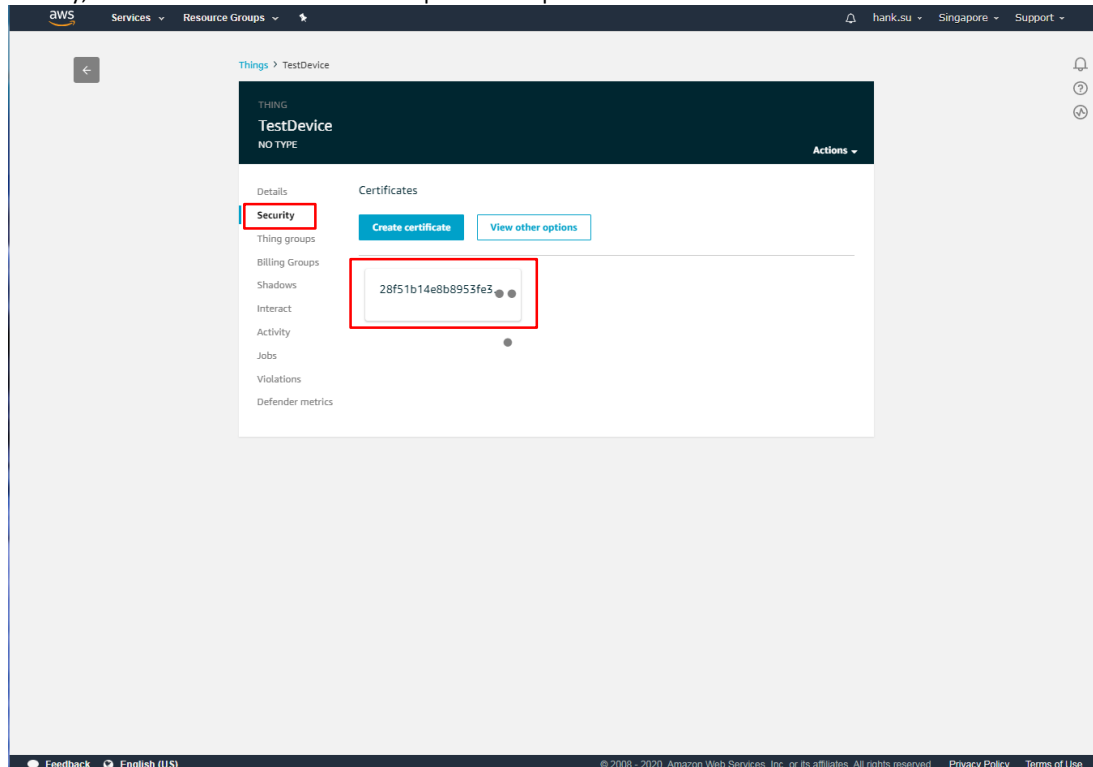


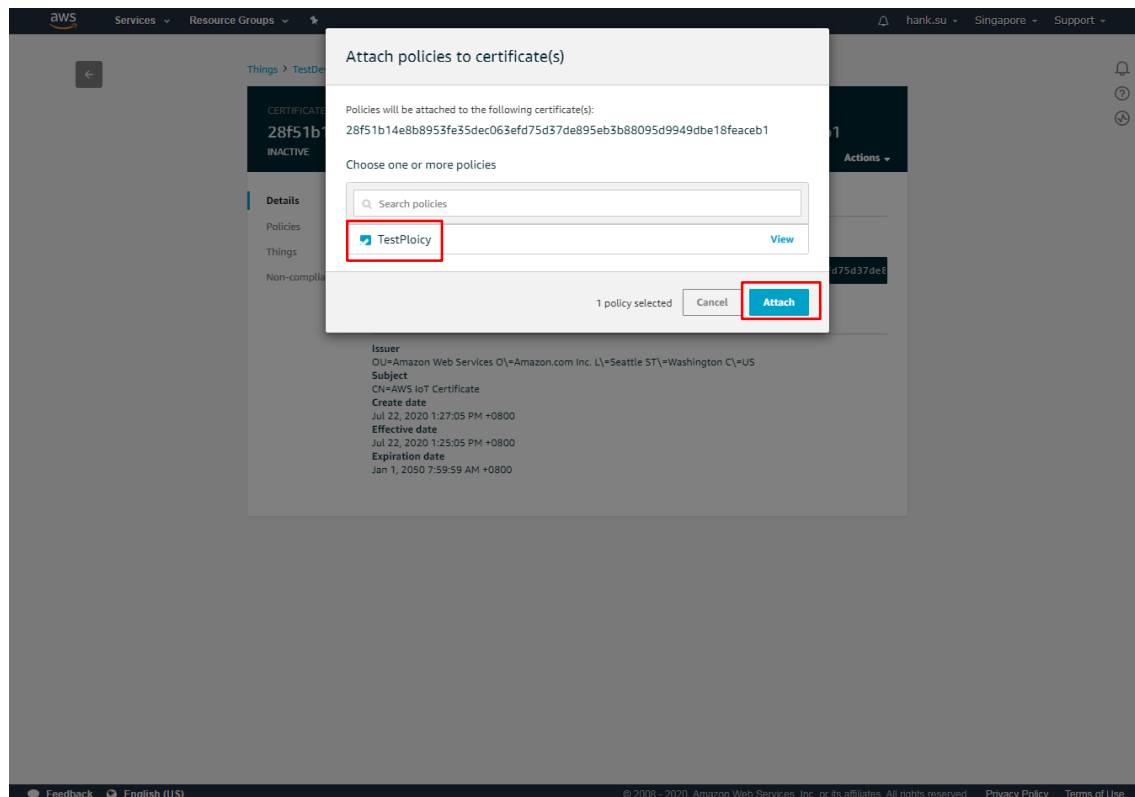
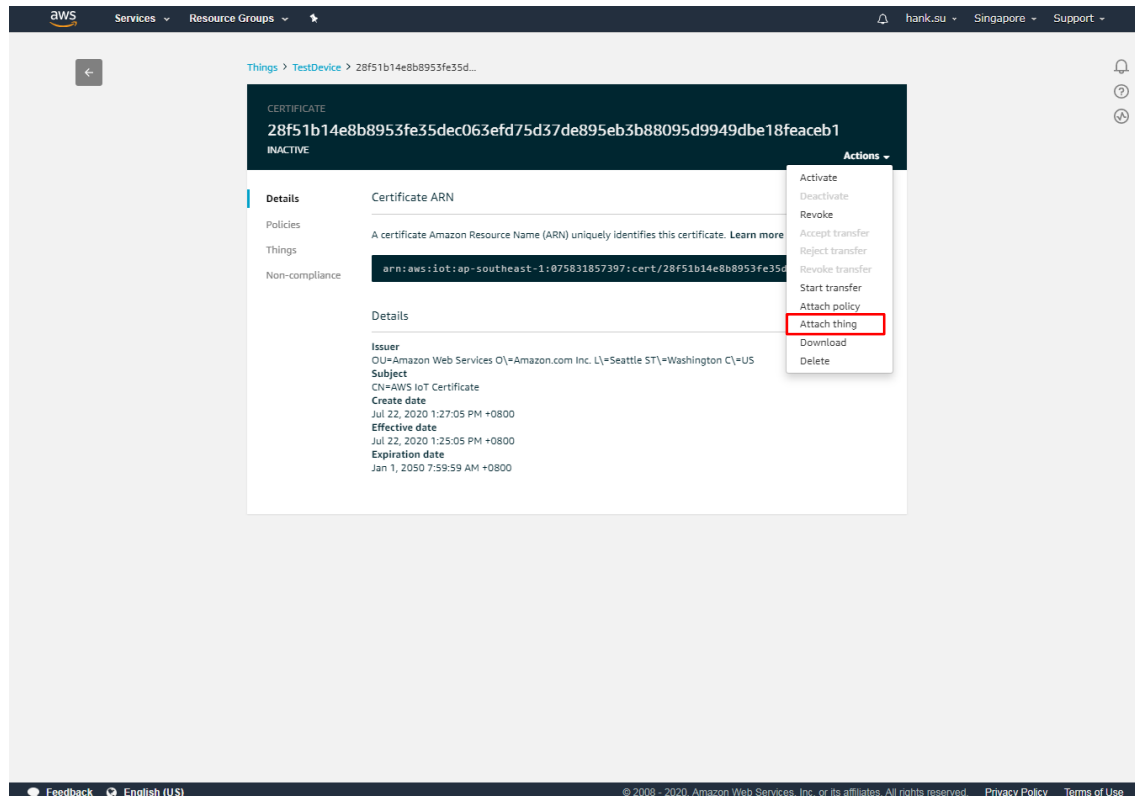
1.3 Attach Policy to Test Device

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.



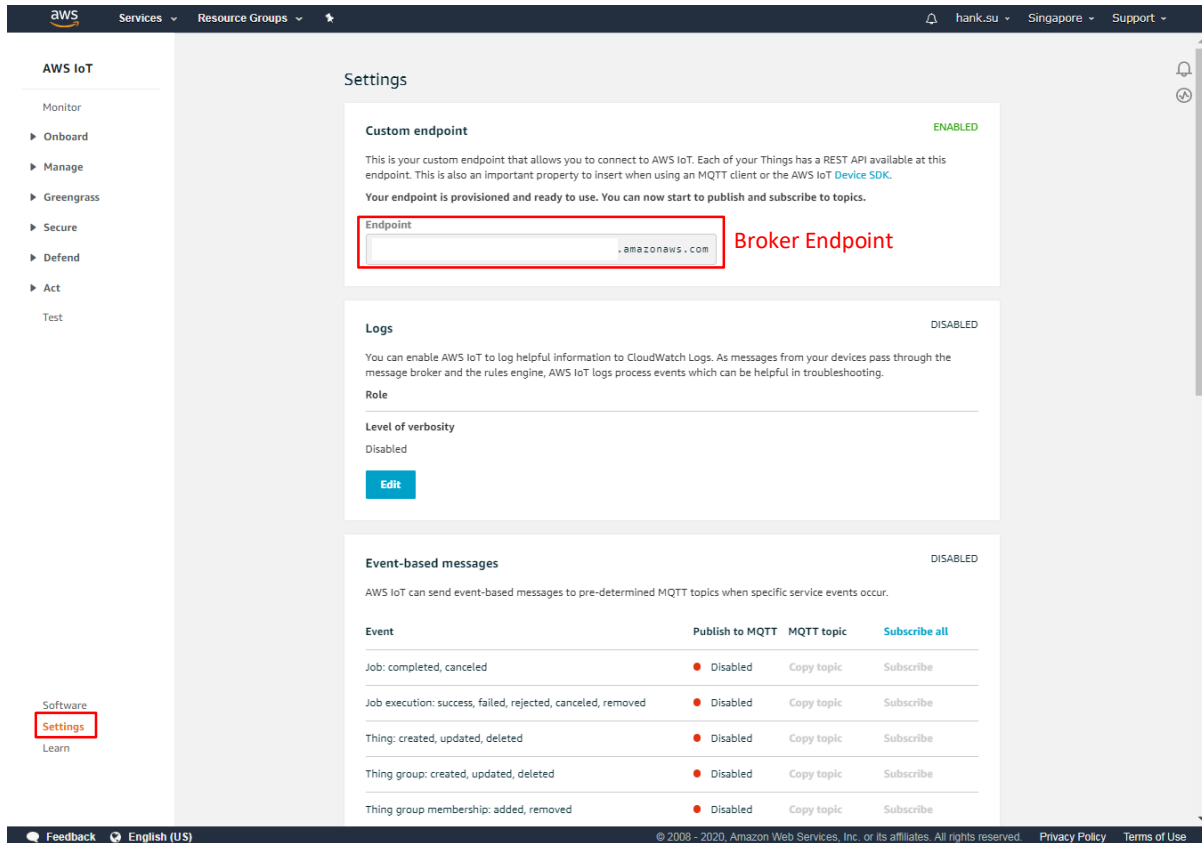
Click Security, then click the certificate create in previous step.





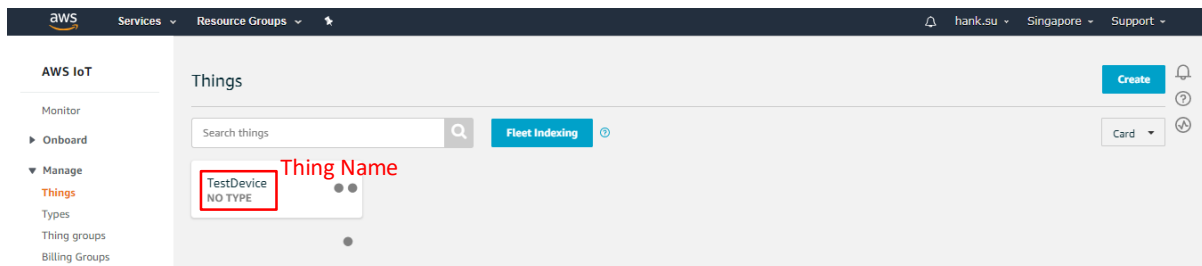
1.4 Setting Test Device with AmebaD Amazon FreeRTOS Source

1.4.1 Get Broker Endpoint



The screenshot shows the AWS IoT Settings page. The 'Custom endpoint' section is highlighted with a red box, showing the endpoint 'amazonaws.com' and a red label 'Broker Endpoint'. The 'Logs' and 'Event-based messages' sections are also visible.

1.4.2 Get Thing Name



The screenshot shows the AWS IoT Things page. A 'Thing' named 'TestDevice' is highlighted with a red box and a red label 'Thing Name'.

1.4.3 Setup IoT Core Information with AmebaD Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in "ambd_amazon-freertos/blob/master/demos/include/aws_clientcredential.h"

```

/*
#define clientcredentialMQTT_BROKER_ENDPOINT      "xxxxxxxxxxxxx.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__ */

```

1.4.4 Setup Thing's Private Key and Certificate

Filled keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in "amdb_amazon-freertos/blob/master/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-certifiacte.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 be done by amdb_amazon-freertos/tools/certificate_configuration/CertificateConfigurator.html

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:

未潤滑任何齒面

⬇️ 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.

```

3/*
4 * @brief PEM-encoded client private key.
5 *
6 * @todo If you are running one of the FreeRTOS demo projects, set this
7 * to the private key that will be used for TLS client authentication.
8 *
9 * @note Must include the PEM header and footer:
10 * "-----BEGIN RSA PRIVATE KEY-----\n"
11 * "...base64 data...\n"
12 * "-----END RSA PRIVATE KEY-----\n"
13 */
14
15 #define keyCLIENT_PRIVATE_KEY_PEM \
16 "-----BEGIN CERTIFICATE-----\n"
17 "MIIDWzCCAaKgAwIBAgIWAIDLSSoG+EARSbBprT4Imiuu8j2vMA0GCSqGSIb3DQEB\n"
18 "-----\n"
19
20 #define keyCLIENT_CERTIFICATE_PEM \
21 "-----BEGIN CERTIFICATE-----\n"
22 "MIIDWzCCAaKgAwIBAgIWAIDLSSoG+EARSbBprT4Imiuu8j2vMA0GCSqGSIb3DQEB\n"
23 "-----\n"
24
25 #define keyCLIENT_PRIVATE_KEY_PEM \
26 "-----BEGIN RSA PRIVATE KEY-----\n"
27 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
28 "-----\n"
29
30 #define keyCLIENT_PRIVATE_KEY_PEM \
31 "-----BEGIN RSA PRIVATE KEY-----\n"
32 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
33 "-----\n"
34
35 #define keyCLIENT_PRIVATE_KEY_PEM \
36 "-----BEGIN RSA PRIVATE KEY-----\n"
37 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
38 "-----\n"
39
40 #define keyCLIENT_PRIVATE_KEY_PEM \
41 "-----BEGIN RSA PRIVATE KEY-----\n"
42 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
43 "-----\n"
44
45 #define keyCLIENT_PRIVATE_KEY_PEM \
46 "-----BEGIN RSA PRIVATE KEY-----\n"
47 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
48 "-----\n"
49
50 #define keyCLIENT_PRIVATE_KEY_PEM \
51 "-----BEGIN RSA PRIVATE KEY-----\n"
52 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
53 "-----\n"
54
55 #define keyCLIENT_PRIVATE_KEY_PEM \
56 "-----BEGIN RSA PRIVATE KEY-----\n"
57 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
58 "-----\n"
59
60 #define keyCLIENT_PRIVATE_KEY_PEM \
61 "-----BEGIN RSA PRIVATE KEY-----\n"
62 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
63 "-----\n"
64
65 #define keyCLIENT_PRIVATE_KEY_PEM \
66 "-----BEGIN RSA PRIVATE KEY-----\n"
67 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
68 "-----\n"
69
70 #define keyCLIENT_PRIVATE_KEY_PEM \
71 "-----BEGIN RSA PRIVATE KEY-----\n"
72 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
73 "-----\n"
74
75 #define keyCLIENT_PRIVATE_KEY_PEM \
76 "-----BEGIN RSA PRIVATE KEY-----\n"
77 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
78 "-----\n"
79
80 #define keyCLIENT_PRIVATE_KEY_PEM \
81 "-----BEGIN RSA PRIVATE KEY-----\n"
82 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
83 "-----\n"
84
85 #define keyCLIENT_PRIVATE_KEY_PEM \
86 "-----BEGIN RSA PRIVATE KEY-----\n"
87 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
88 "-----\n"
89
90 #define keyCLIENT_PRIVATE_KEY_PEM \
91 "-----BEGIN RSA PRIVATE KEY-----\n"
92 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
93 "-----\n"
94
95 #define keyCLIENT_PRIVATE_KEY_PEM \
96 "-----BEGIN RSA PRIVATE KEY-----\n"
97 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
98 "-----\n"
99
100 #define keyCLIENT_PRIVATE_KEY_PEM \
101 "-----BEGIN RSA PRIVATE KEY-----\n"
102 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
103 "-----\n"
104
105 #define keyCLIENT_PRIVATE_KEY_PEM \
106 "-----BEGIN RSA PRIVATE KEY-----\n"
107 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
108 "-----\n"
109
110 #define keyCLIENT_PRIVATE_KEY_PEM \
111 "-----BEGIN RSA PRIVATE KEY-----\n"
112 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
113 "-----\n"
114
115 #define keyCLIENT_PRIVATE_KEY_PEM \
116 "-----BEGIN RSA PRIVATE KEY-----\n"
117 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
118 "-----\n"
119
120 #define keyCLIENT_PRIVATE_KEY_PEM \
121 "-----BEGIN RSA PRIVATE KEY-----\n"
122 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
123 "-----\n"
124
125 #define keyCLIENT_PRIVATE_KEY_PEM \
126 "-----BEGIN RSA PRIVATE KEY-----\n"
127 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
128 "-----\n"
129
130 #define keyCLIENT_PRIVATE_KEY_PEM \
131 "-----BEGIN RSA PRIVATE KEY-----\n"
132 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
133 "-----\n"
134
135 #define keyCLIENT_PRIVATE_KEY_PEM \
136 "-----BEGIN RSA PRIVATE KEY-----\n"
137 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
138 "-----\n"
139
140 #define keyCLIENT_PRIVATE_KEY_PEM \
141 "-----BEGIN RSA PRIVATE KEY-----\n"
142 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
143 "-----\n"
144
145 #define keyCLIENT_PRIVATE_KEY_PEM \
146 "-----BEGIN RSA PRIVATE KEY-----\n"
147 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
148 "-----\n"
149
150 #define keyCLIENT_PRIVATE_KEY_PEM \
151 "-----BEGIN RSA PRIVATE KEY-----\n"
152 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
153 "-----\n"
154
155 #define keyCLIENT_PRIVATE_KEY_PEM \
156 "-----BEGIN RSA PRIVATE KEY-----\n"
157 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
158 "-----\n"
159
160 #define keyCLIENT_PRIVATE_KEY_PEM \
161 "-----BEGIN RSA PRIVATE KEY-----\n"
162 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
163 "-----\n"
164
165 #define keyCLIENT_PRIVATE_KEY_PEM \
166 "-----BEGIN RSA PRIVATE KEY-----\n"
167 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
168 "-----\n"
169
170 #define keyCLIENT_PRIVATE_KEY_PEM \
171 "-----BEGIN RSA PRIVATE KEY-----\n"
172 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
173 "-----\n"
174
175 #define keyCLIENT_PRIVATE_KEY_PEM \
176 "-----BEGIN RSA PRIVATE KEY-----\n"
177 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
178 "-----\n"
179
180 #define keyCLIENT_PRIVATE_KEY_PEM \
181 "-----BEGIN RSA PRIVATE KEY-----\n"
182 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
183 "-----\n"
184
185 #define keyCLIENT_PRIVATE_KEY_PEM \
186 "-----BEGIN RSA PRIVATE KEY-----\n"
187 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
188 "-----\n"
189
190 #define keyCLIENT_PRIVATE_KEY_PEM \
191 "-----BEGIN RSA PRIVATE KEY-----\n"
192 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
193 "-----\n"
194
195 #define keyCLIENT_PRIVATE_KEY_PEM \
196 "-----BEGIN RSA PRIVATE KEY-----\n"
197 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
198 "-----\n"
199
200 #define keyCLIENT_PRIVATE_KEY_PEM \
201 "-----BEGIN RSA PRIVATE KEY-----\n"
202 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
203 "-----\n"
204
205 #define keyCLIENT_PRIVATE_KEY_PEM \
206 "-----BEGIN RSA PRIVATE KEY-----\n"
207 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
208 "-----\n"
209
210 #define keyCLIENT_PRIVATE_KEY_PEM \
211 "-----BEGIN RSA PRIVATE KEY-----\n"
212 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
213 "-----\n"
214
215 #define keyCLIENT_PRIVATE_KEY_PEM \
216 "-----BEGIN RSA PRIVATE KEY-----\n"
217 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
218 "-----\n"
219
220 #define keyCLIENT_PRIVATE_KEY_PEM \
221 "-----BEGIN RSA PRIVATE KEY-----\n"
222 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
223 "-----\n"
224
225 #define keyCLIENT_PRIVATE_KEY_PEM \
226 "-----BEGIN RSA PRIVATE KEY-----\n"
227 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
228 "-----\n"
229
230 #define keyCLIENT_PRIVATE_KEY_PEM \
231 "-----BEGIN RSA PRIVATE KEY-----\n"
232 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
233 "-----\n"
234
235 #define keyCLIENT_PRIVATE_KEY_PEM \
236 "-----BEGIN RSA PRIVATE KEY-----\n"
237 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBNn/AHyhEcozLZC8qECUOn\n"
238 "-----\n"
239
240 #define keyCLIENT_PRIVATE_KEY_PEM \
241 "-----BEGIN RSA PRIVATE KEY-----\n"
242 "MIIEpAIBAAKCAQEAwp96WNucGebARfjD8O+CLsqcBN
```

2 Getting Started with the amebaD

The AmebaD board is able to use the amazon-freertos sdk version 202002.00. The AmebaD Demo board is designed by Realtek and is Wi-Fi ready chip (<https://www.amebaiot.com/ameba-sdk-summary/>).

2.1 Hardware Components

AmebaD Demo Board (<https://www.amebaiot.com/amebad/>).

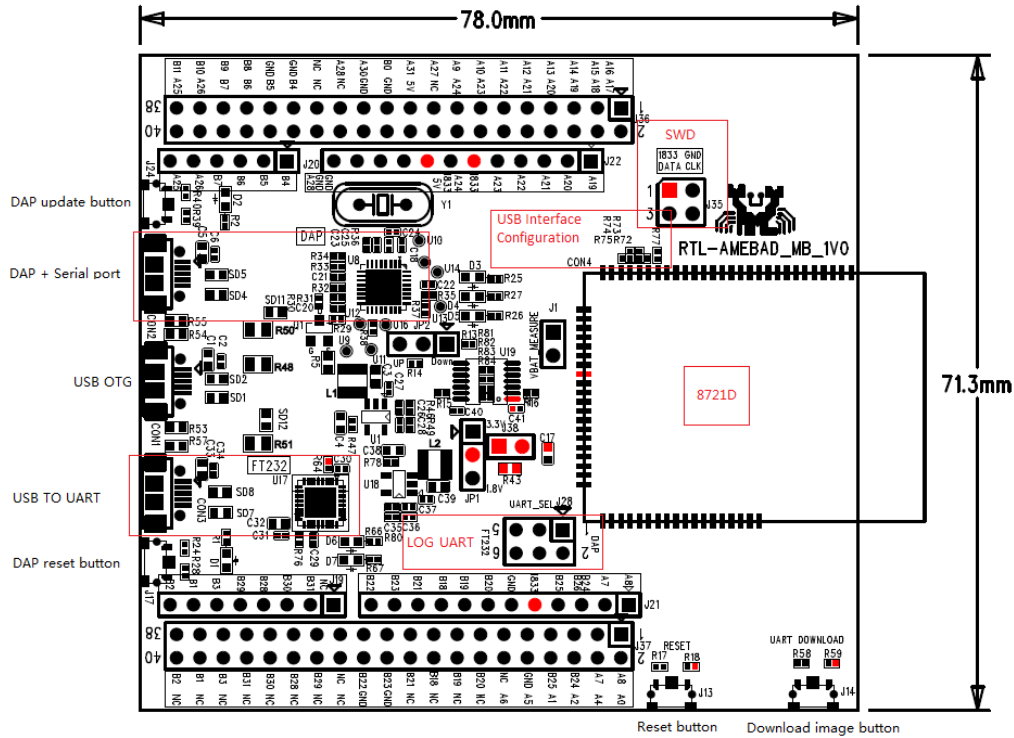


Fig 2-1 AmebaD Demo board

2.2 Supported Development Environment

Currently the amazon-freertos of AmebaD is supported by the IAR Embedded workbench ver.8.30.1. For windows operating system only.

2.3 Pre-Requisite

- Required source code.
- AmebaD Demo board
- IAR Embedded Workbench ver.8.30.1
- Realtek Image Tool

3 IAR Build Environment Setup

This chapter illustrates how to setup IAR development environment for Realtek Ameba-D SDK, including building projects, downloading images and debugging.

3.1 Requirement

3.1.1 IAR Embedded Workbench

IAR provides an IDE environment for code building, downloading, and debugging. Check “IAR Embedded Workbench” on <http://www.iar.com/>, and a trial 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.

3.2 How to Use IAR SDK?

3.2.1 IAR Project Introduction

Because Ameba-D is a dual-core CPU platform, two workspaces are provided to build for each core in `projects\realtek\amebaD\IAR\aws_tests`

- 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

3.2.2 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.

3.2.2.1 Building KM0 Project

The following steps show how to build KM0 project:

- (1) Open `projects\realtek\amebaD\IAR\aws_tests\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 3-1.
- (3) Right click the project and choose “Rebuild All”, as Fig 3-2 shows. The km0_bootloader and km0_application should compile in order.

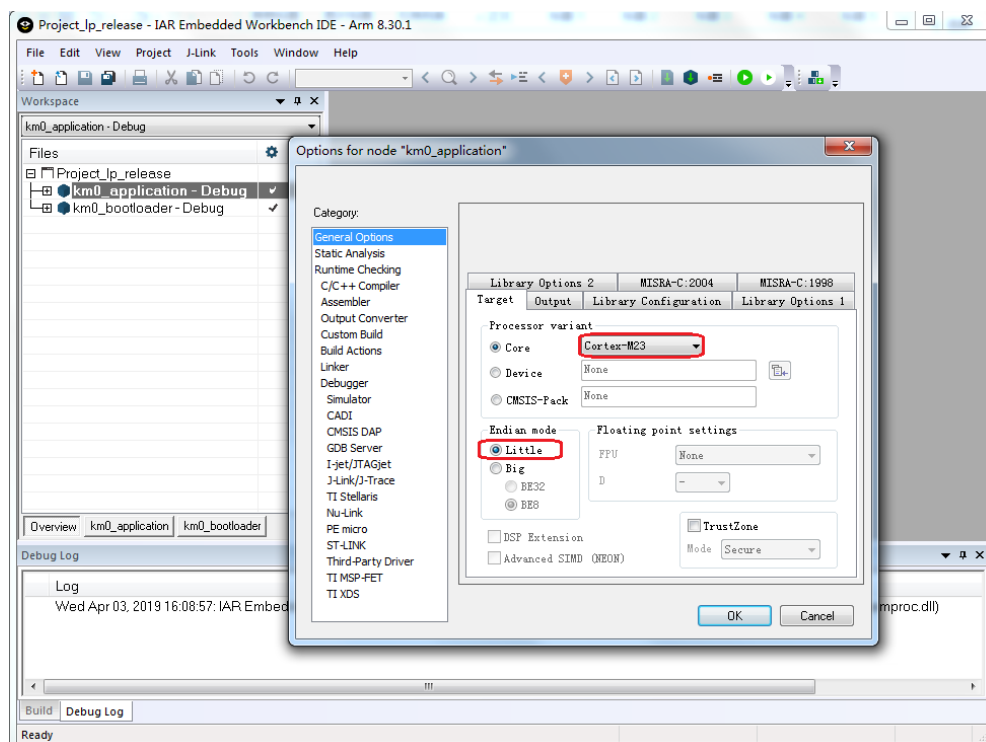


Fig 3-1 KM0 processor options

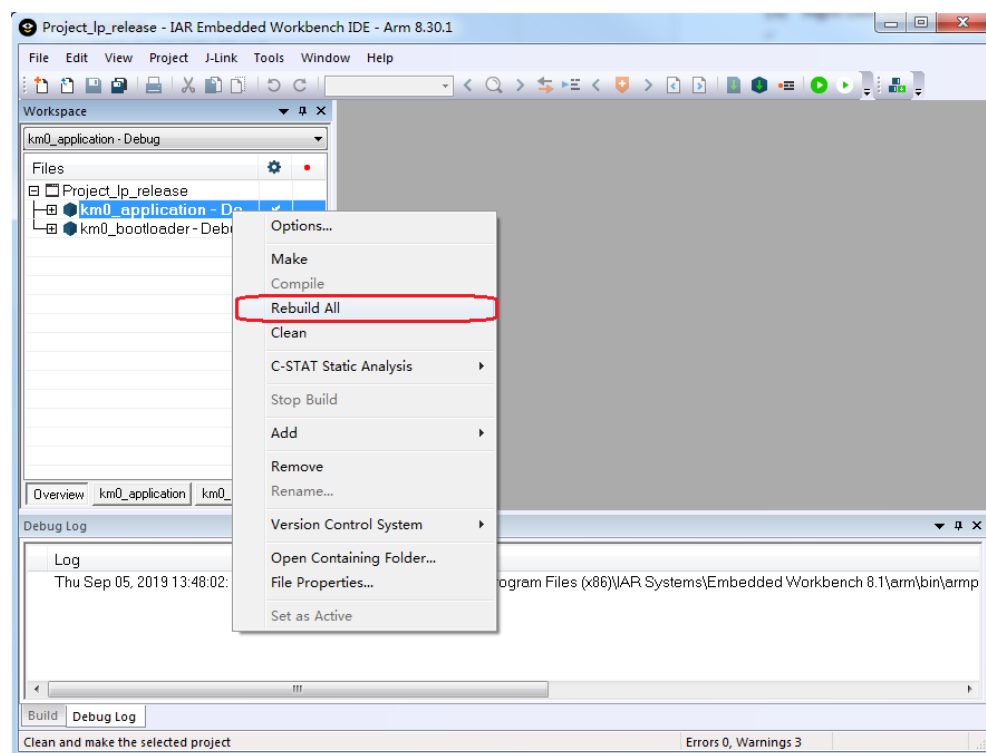


Fig 3-2 Building KM0 project

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. Don't stop it while it is in progress. After post-build action is completed, the window would disappear automatically.

```

C:\Windows\System32\cmd.exe
D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE>cmd /c
""D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE""\..\
..\component\soc\realtek\amebaD_misc\iar_utility\common\tools\nm Debug/Exe/km0
_image/km0_application.axf ! "D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0
_example\EWARM-RELEASE""\..\..\component\soc\realtek\amebaD_misc\iar_utility\c
ommon\tools\sort > Debug/Exe/km0_image/km0_application.map"

D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE>cmd /c
""D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE""\..\
..\component\soc\realtek\amebaD_misc\iar_utility\common\tools\objdump -d Debug
/Exe/km0_image/km0_application.axf > Debug/Exe/km0_image/km0_application.asm"

D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE>for /P
"delims=" %i in ('cmd /c ""D:\Code\AmebaD\03_0903\project\realtek_amebaD_va0_ex
ample\EWARM-RELEASE""\..\..\component\soc\realtek\amebaD_misc\iar_utility\comm
on\tools\grep IMAGE2 Debug/Exe/km0_image/km0_application.map ! "D:\Code\AmebaD\0
3_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE""\..\..\component\soc
\realtek\amebaD_misc\iar_utility\common\tools\grep Base ! "D:\Code\AmebaD\03_09
03\project\realtek_amebaD_va0_example\EWARM-RELEASE""\..\..\component\soc\real
tek\amebaD_misc\iar_utility\common\tools\gawk '{print $1}'"') do set ran2_start=
%xi

```

- (4) After compile, the images km0_boot_all.bin and km0_image2_all.bin can be seen in projects\realtek\amebaD\IAR\aws_tests\Debug\Exe\km0_image.

3.2.2.2 Building KM4 Project

The following steps show how to build KM4 project:

- (1) Open projects\realtek\amebaD\IAR\aws_tests\Project_hp_release.eww.
- (2) Refer to 3.2.1 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 3-3.

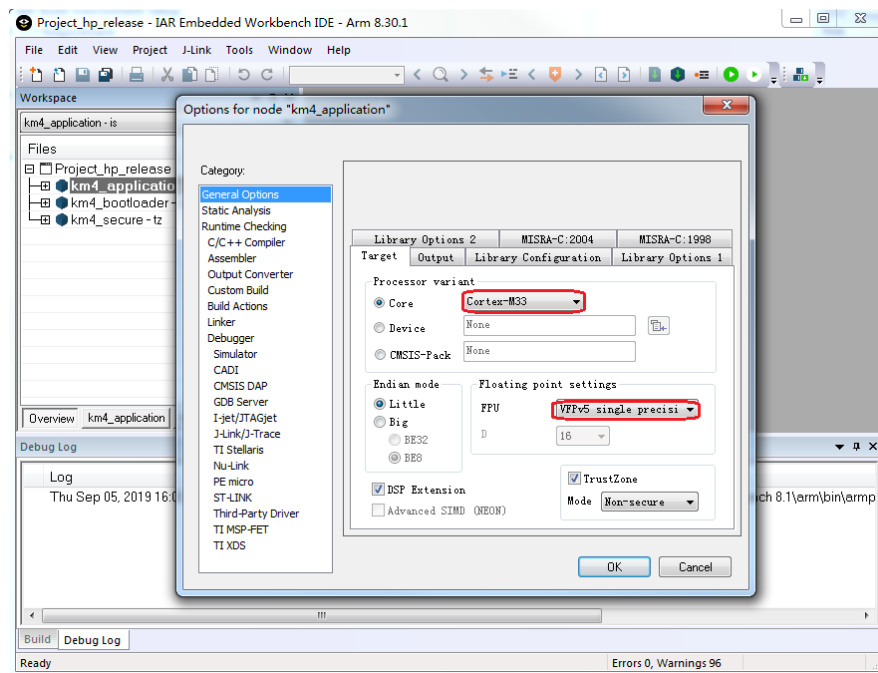


Fig 3-3 KM4 processor options

- (4) Right click the project and choose "Rebuild All", as Fig 3-4 shows. The km4_bootloader, km4_application should compile in order.

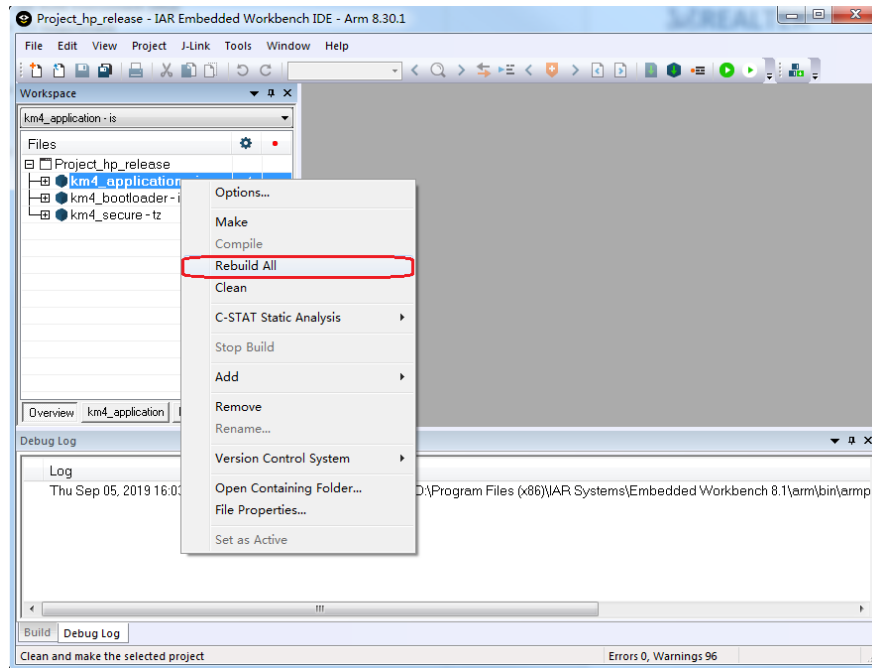
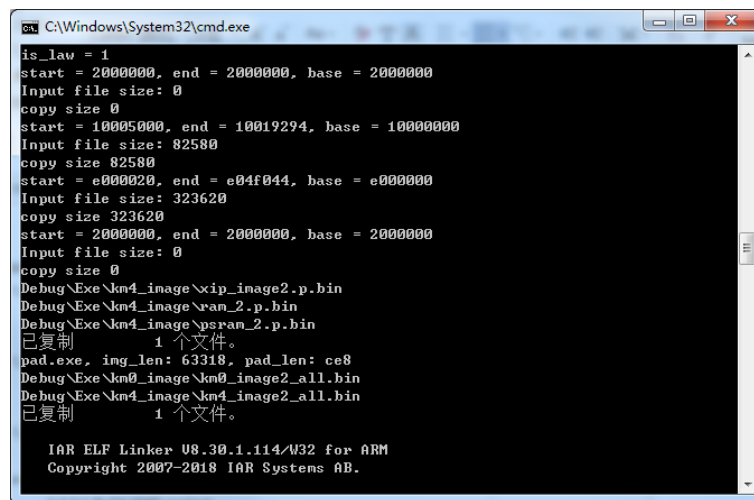


Fig 3-4 Building KM4 project

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. Don't stop it while it is in progress. After post-build action is completed, the window would disappear automatically.



- After compile, the images km4_boot_all.bin and km0_km4_image2.bin can be seen in **projects\realtek\amebaD\IAR\aws_tests\Debug\Exe\km4_image**.
- The generated images can be downloaded by ImageTool:

4 ImageTool

4.1 Introduction

This chapter introduces how to use ImageTool to encrypt, generate and download images. As show in Fig 4-1, ImageTool has four tabpages.

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

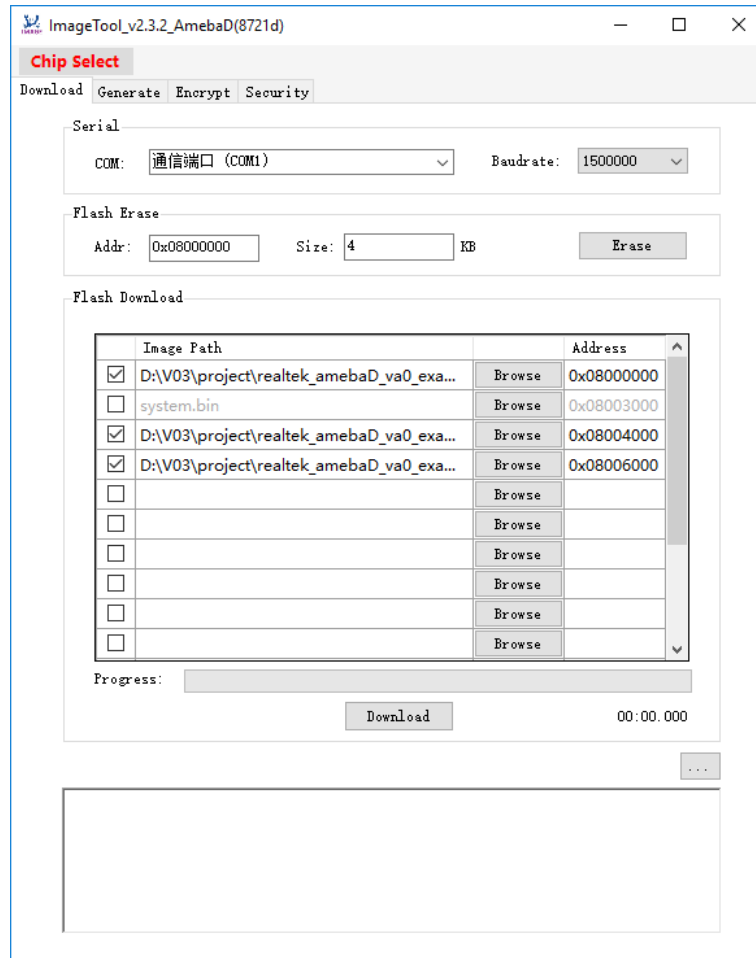


Fig 4-1 ImageTool UI

4.2 Environment Setup

4.2.1 Hardware Setup

The hardware setup is shown in Fig 4-2.

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

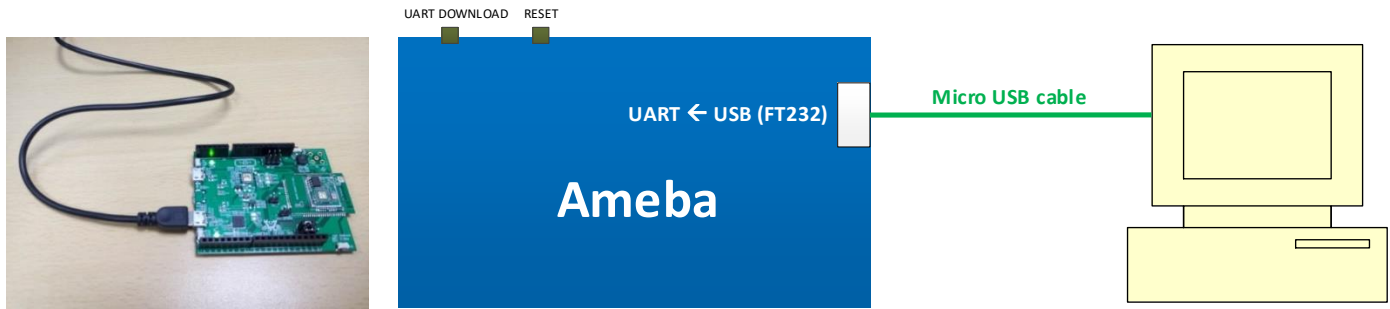


Fig 4-2 Hardware setup

4.2.2 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

4.3 Download

4.3.1 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.

4.3.1.1 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_tests\Debug\Exe\km0_image** and **{path}\projects\realtek\amebaD\IAR\aws_tests\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.

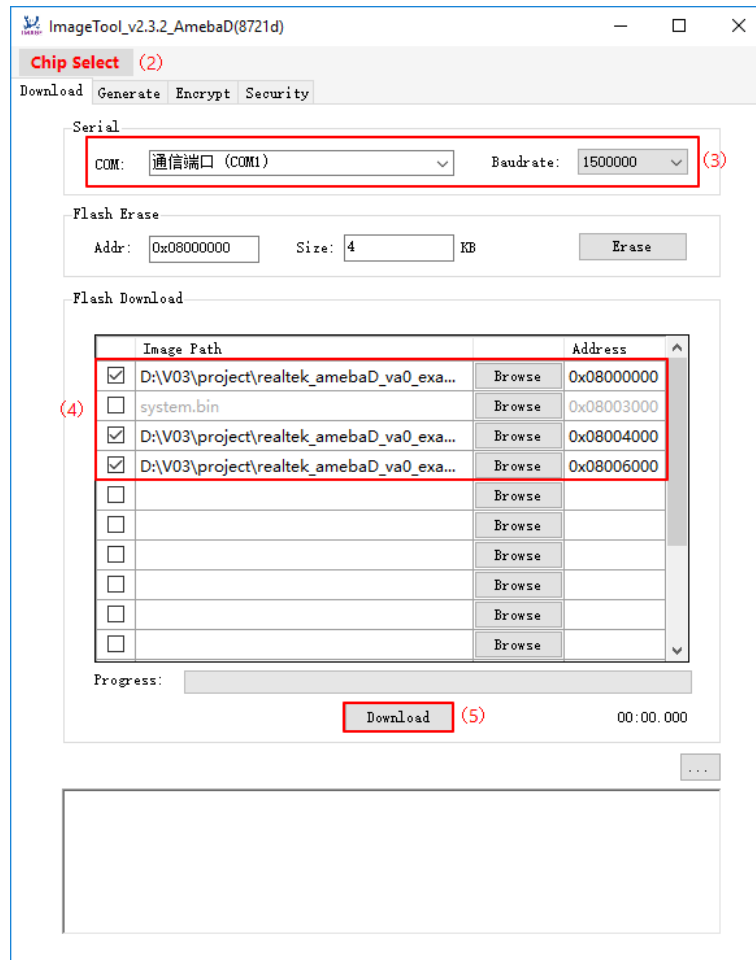


Fig 4-3 ImageTool 'Download' tabpage setting

5 Testing

One the AmebaD has been rebooted the application will automatically start run MQTT demo and communicate to IoT Core.

```
COM26 - Tera Term VT
File Edit Setup Control Window Help

6 12507 [iot_thread] [INFO] [DEMO][12507] Successfully initialized the demo. Network type for the demo: 1
7 12516 [iot_thread] [INFO] [MQTT][12516] MQTT library successfully initialized.
8 12523 [iot_thread] [INFO] [DEMO][12523] MQTT demo client identifier is TestDevice (length 10).
9 14214 [iot_thread] [INFO] [MQTT][14214] Establishing new MQTT connection.
Interface 0 IP address : 192.168.92.13610 14225 [iot_thread] [INFO] [MQTT][14225] Anonymous metrics (SDK language, SDK version) will be provided to AWS IoT. Reconfig
e with AWS_IOT_MQTT_ENABLE_METRICS set to 0 to disable.
11 14244 [iot_thread] [INFO] [MQTT][14244] (MQTT connection 100337e0, CONNECT operation 100339a0) Waiting for operation completion.
12 14359 [iot_thread] [INFO] [MQTT][14358] (MQTT connection 100337e0, CONNECT operation 100339a0) Wait complete with result SUCCESS.
13 14371 [iot_thread] [INFO] [MQTT][14371] New MQTT connection 100379f0 established.
14 14380 [iot_thread] [INFO] [MQTT][14380] (MQTT connection 100337e0) SUBSCRIBE operation scheduled.
15 14389 [iot_thread] [INFO] [MQTT][14389] (MQTT connection 100337e0, SUBSCRIBE operation 100339e0) Waiting for operation completion.
16 14490 [iot_thread] [INFO] [MQTT][14490] (MQTT connection 100337e0, SUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
17 14504 [iot_thread] [INFO] [DEMO][14504] All demo topic filter subscriptions accepted.
18 14512 [iot_thread] [INFO] [DEMO][14512] Publishing messages 0 to 1.
19 14520 [iot_thread] [INFO] [MQTT][14520] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
20 14530 [iot_thread] [INFO] [MQTT][14530] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
21 14539 [iot_thread] [INFO] [DEMO][14539] Waiting for 2 publishes to be received.
22 14634 [iot_thread] [INFO] [DEMO][14634] MQTT PUBLISH 0 successfully sent.
23 14643 [iot_thread] [INFO] [DEMO][14643] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 0!
24 14664 [iot_thread] [INFO] [MQTT][14664] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
25 14674 [iot_thread] [INFO] [DEMO][14674] Acknowledgment message for PUBLISH 0 will be sent.
26 14707 [iot_thread] [INFO] [DEMO][14707] MQTT PUBLISH 1 successfully sent.
27 14732 [iot_thread] [INFO] [DEMO][14732] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 1!
28 14752 [iot_thread] [INFO] [MQTT][14752] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
29 14762 [iot_thread] [INFO] [DEMO][14762] Acknowledgment message for PUBLISH 1 will be sent.
30 14773 [iot_thread] [INFO] [DEMO][14773] 2 publishes received.
31 14778 [iot_thread] [INFO] [DEMO][14778] Publishing messages 2 to 3.
32 14786 [iot_thread] [INFO] [MQTT][14786] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
33 14796 [iot_thread] [INFO] [MQTT][14796] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
34 14805 [iot_thread] [INFO] [DEMO][14805] Waiting for 2 publishes to be received.
35 14920 [iot_thread] [INFO] [DEMO][14920] MQTT PUBLISH 3 successfully sent.
36 14928 [iot_thread] [INFO] [DEMO][14928] MQTT PUBLISH 2 successfully sent.
37 14943 [iot_thread] [INFO] [DEMO][14943] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/3
Publish topic name: iotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 2!
38 14963 [iot_thread] [INFO] [MQTT][14963] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
39 14973 [iot_thread] [INFO] [DEMO][14973] Acknowledgment message for PUBLISH 2 will be sent.
40 14984 [iot_thread] [INFO] [DEMO][14984] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
```

```

COM26 - Tera Term VT
File Edit Setup Control Window Help
Publish payload: Hello world 17!
129 16918 [iot_threa] [INFO] [MQTT][16918] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
130 16928 [iot_threa] [INFO] [DEMO][16928] Acknowledgment message for PUBLISH 17 will be sent.
131 16939 [iot_threa] [INFO] [DEMO][16939] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 16!
132 16960 [iot_threa] [INFO] [MQTT][16960] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
133 16970 [iot_threa] [INFO] [DEMO][16970] Acknowledgment message for PUBLISH 16 will be sent.
134 16980 [iot_threa] [INFO] [DEMO][16980] 2 publishes received.
135 16986 [iot_threa] [INFO] [DEMO][16986] Publishing messages 18 to 19.
136 16993 [iot_threa] [INFO] [MQTT][16993] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
137 17004 [iot_threa] [INFO] [MQTT][17004] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
138 17013 [iot_threa] [INFO] [DEMO][17013] Waiting for 2 publishes to be received.
139 17140 [iot_threa] [INFO] [DEMO][17140] MQTT PUBLISH 19 successfully sent.
140 17148 [iot_threa] [INFO] [DEMO][17148] MQTT PUBLISH 18 successfully sent.
141 17159 [iot_threa] [INFO] [DEMO][17159] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/3
Publish topic name: iotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 18!
142 17180 [iot_threa] [INFO] [MQTT][17180] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
143 17190 [iot_threa] [INFO] [DEMO][17190] Acknowledgment message for PUBLISH 18 will be sent.
144 17203 [iot_threa] [INFO] [DEMO][17203] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
Publish topic name: iotdemo/topic/4
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 19!
145 17223 [iot_threa] [INFO] [MQTT][17223] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
146 17233 [iot_threa] [INFO] [DEMO][17233] Acknowledgment message for PUBLISH 19 will be sent.
147 17242 [iot_threa] [INFO] [DEMO][17242] 2 publishes received.
148 17249 [iot_threa] [INFO] [MQTT][17249] (MQTT connection 100337e0) UNSUBSCRIBE operation scheduled.
149 17259 [iot_threa] [INFO] [MQTT][17258] (MQTT connection 100337e0) UNSUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
150 17446 [iot_threa] [INFO] [MQTT][17446] (MQTT connection 100337e0) Disconnecting connection.
151 17459 [iot_threa] [INFO] [MQTT][17458] (MQTT connection 100337e0) DISCONNECT operation 100339e0) Waiting for operation completion.
152 17471 [iot_threa] [INFO] [MQTT][17470] (MQTT connection 100337e0) DISCONNECT operation 100339e0) Wait complete with result SUCCESS.
153 17483 [iot_threa] [INFO] [MQTT][17482] (MQTT connection 100337e0) Connection disconnected.
154 17495 [iot_threa] [INFO] [MQTT][17495] (MQTT connection 100337e0) Network connection closed.
155 17503 [iot_threa] [INFO] [MQTT][17503] (MQTT connection 100337e0) Network connection destroyed.
156 18747 [iot_threa] [INFO] [MQTT][18747] MQTT library cleanup done.
157 18756 [iot_threa] [INFO] [DEMO][18756] Demo completed successfully.
158 18762 [iot_threa] [INFO] [DEMO][18762] Demo completed successfully.

LwIP DHCP: dhcp stop.
Deinitializing WIFI ...
159 18897 [iot_threa] [INFO] [INIT][18897] SDK cleanup done.
160 18902 [iot_threa] [INFO] [DEMO][18902] -----DEMO FINISHED-----

```

Monitor connection summary.

