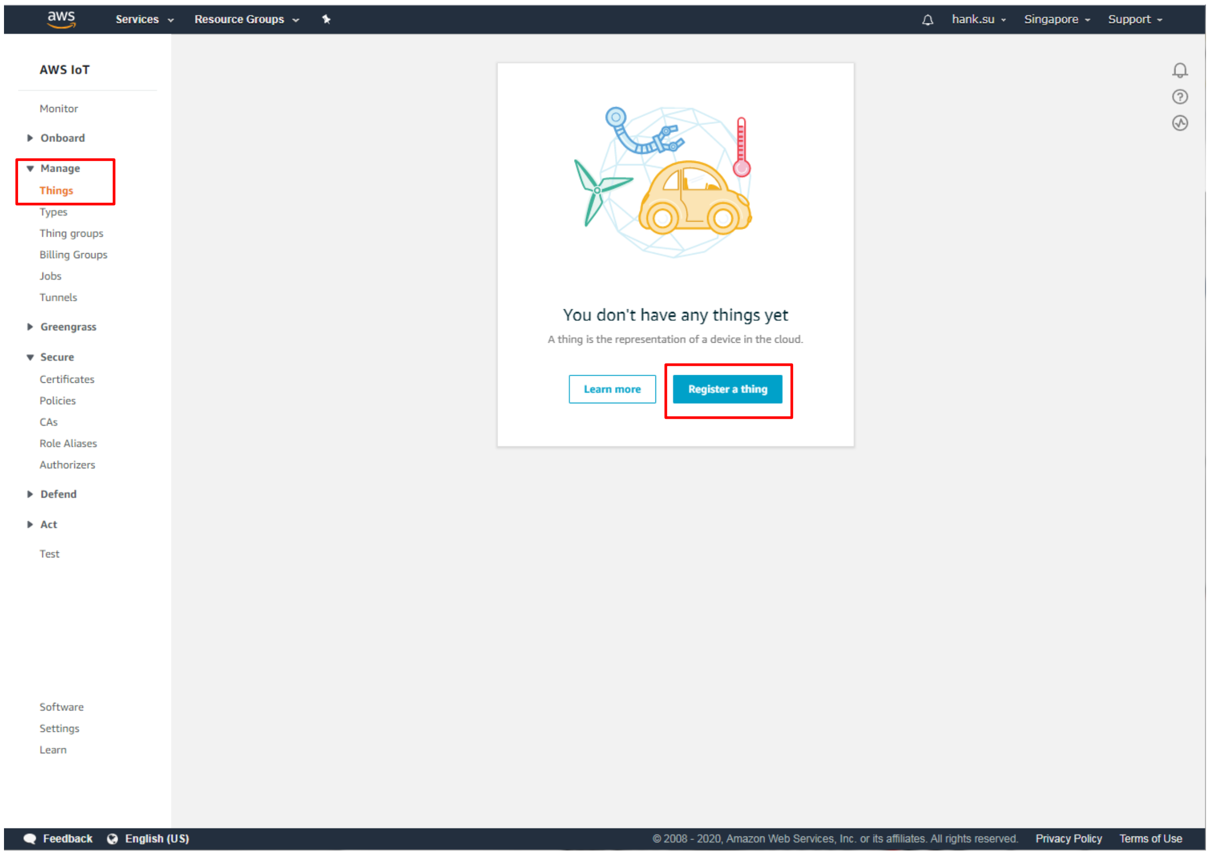
AmebaD Amazon FreeRTOS Getting Started Guide

1 Configure AWS IoT Core

1.1 Create a New Device

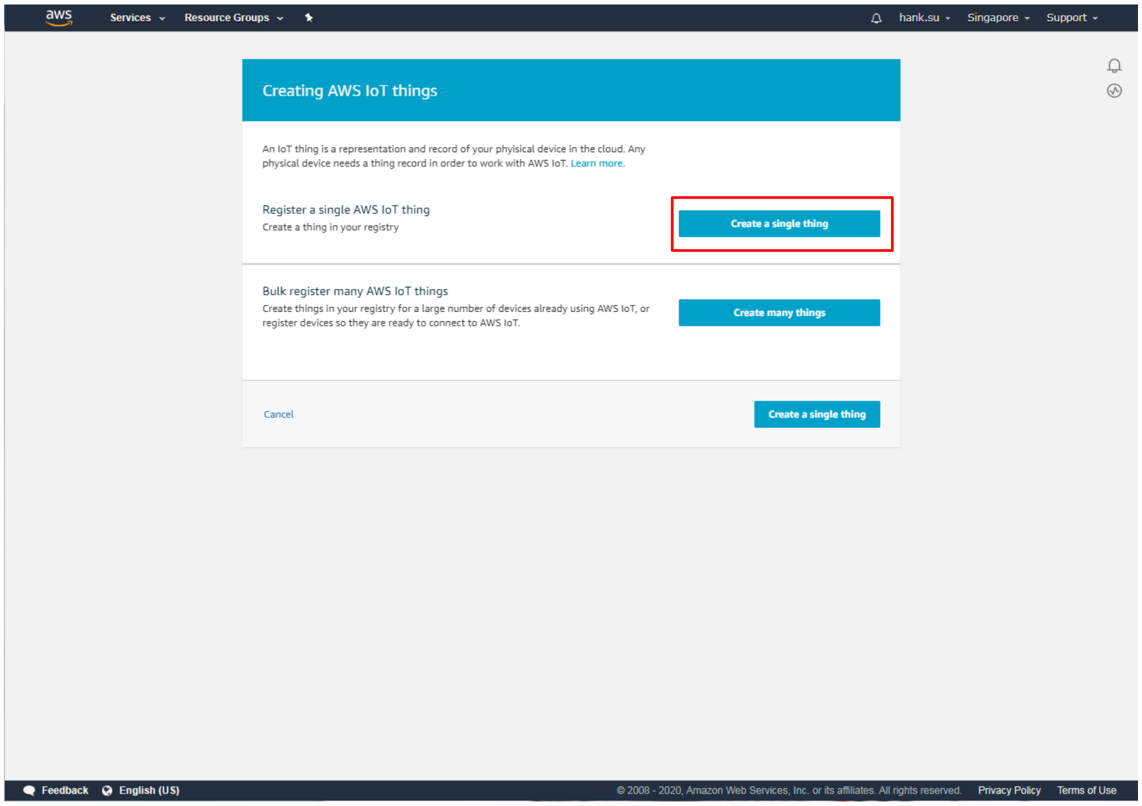
To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click “Register a thing”.



You don't have any things yet

A thing is the representation of a device in the cloud.

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Creating AWS IoT things

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

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

Create a single thing

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

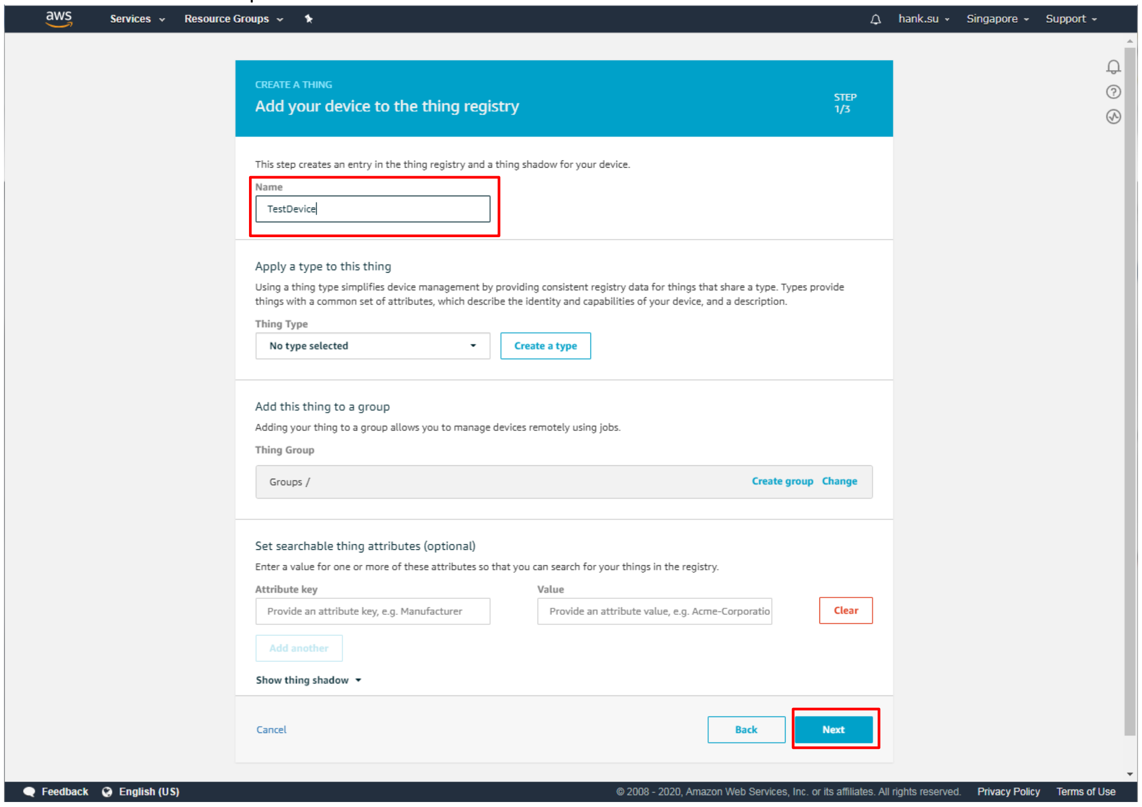
Create many things

Cancel

Create a single thing

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

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



CREATE A THING
Add your device to the thing registry
STEP
1/3

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

Name: TestDevice

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

Thing Type: No type selected

Create a type

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

Thing Group: Groups /

Create group Change

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

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

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

Clear

Add another

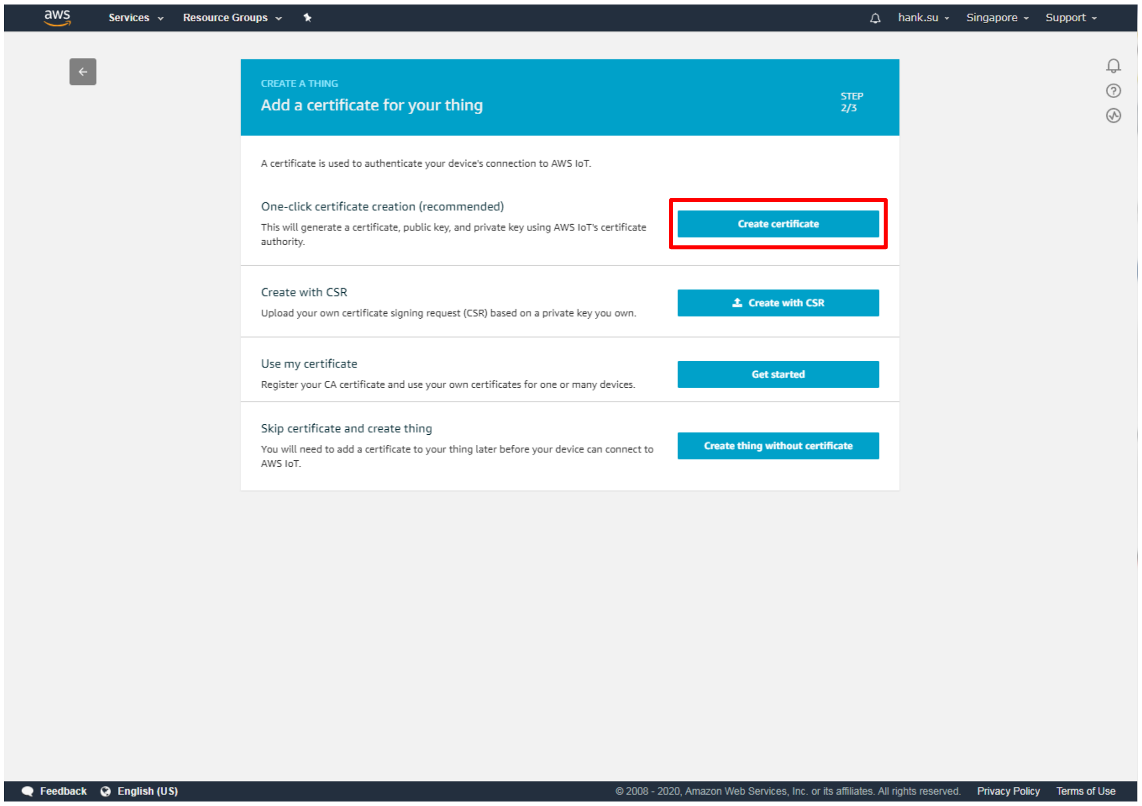
Show thing shadow

Cancel

Back

Next

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



CREATE A THING

Add a certificate for your thing

STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.
Create certificate

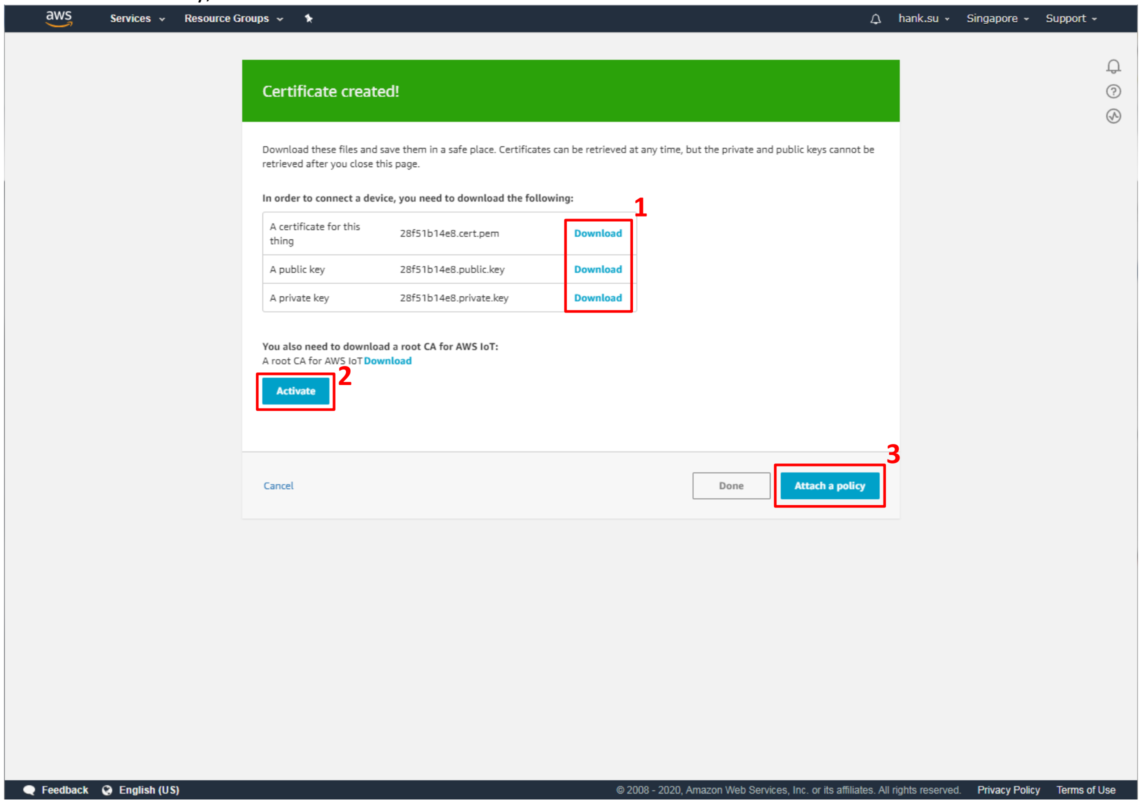
Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.
Create with CSR

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.
Get started

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT.
Create thing without certificate

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

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



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.

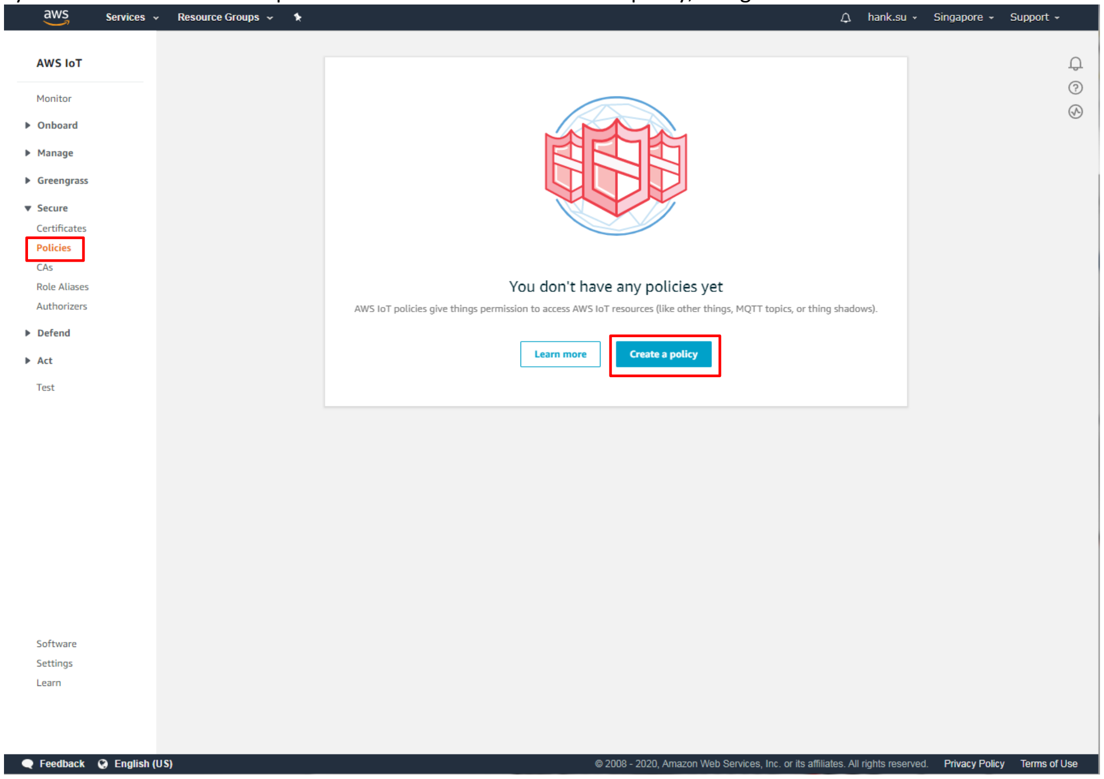
In order to connect a device, you need to download the following:

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**

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"

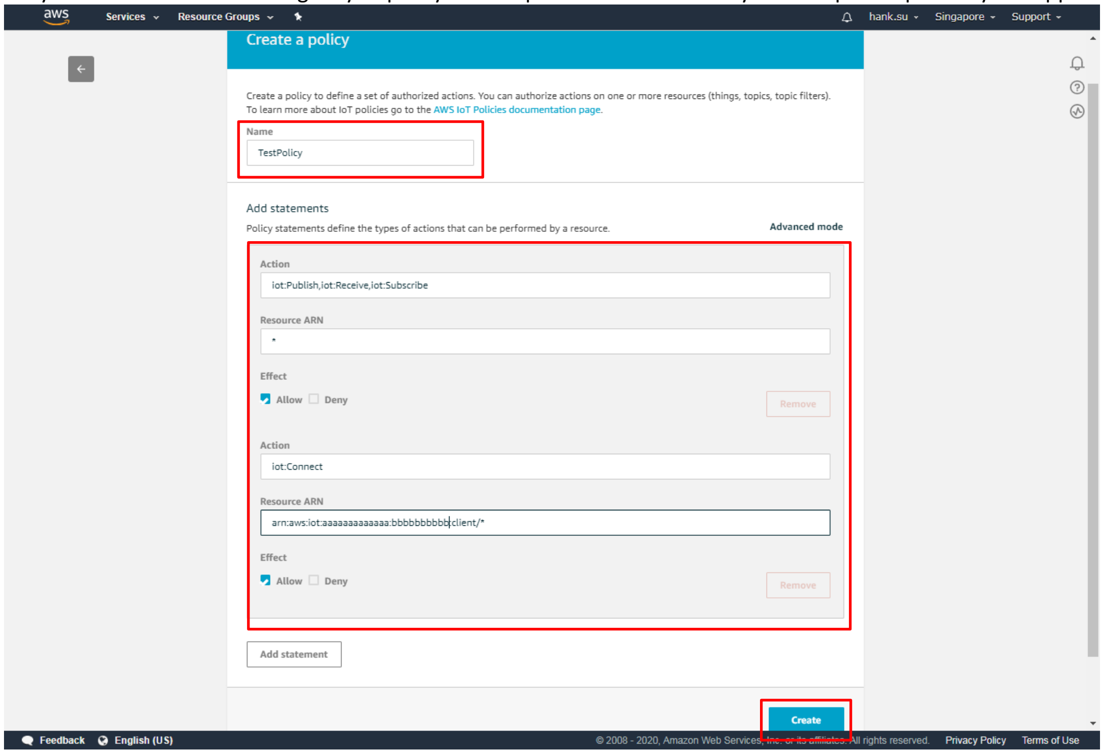


You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows).

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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



Create a policy

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

Name: TestPolicy

Add statements

Policy statements define the types of actions that can be performed by a resource.

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

Resource ARN: *

Effect: [✓ Allow] Deny

Action: iot:Connect

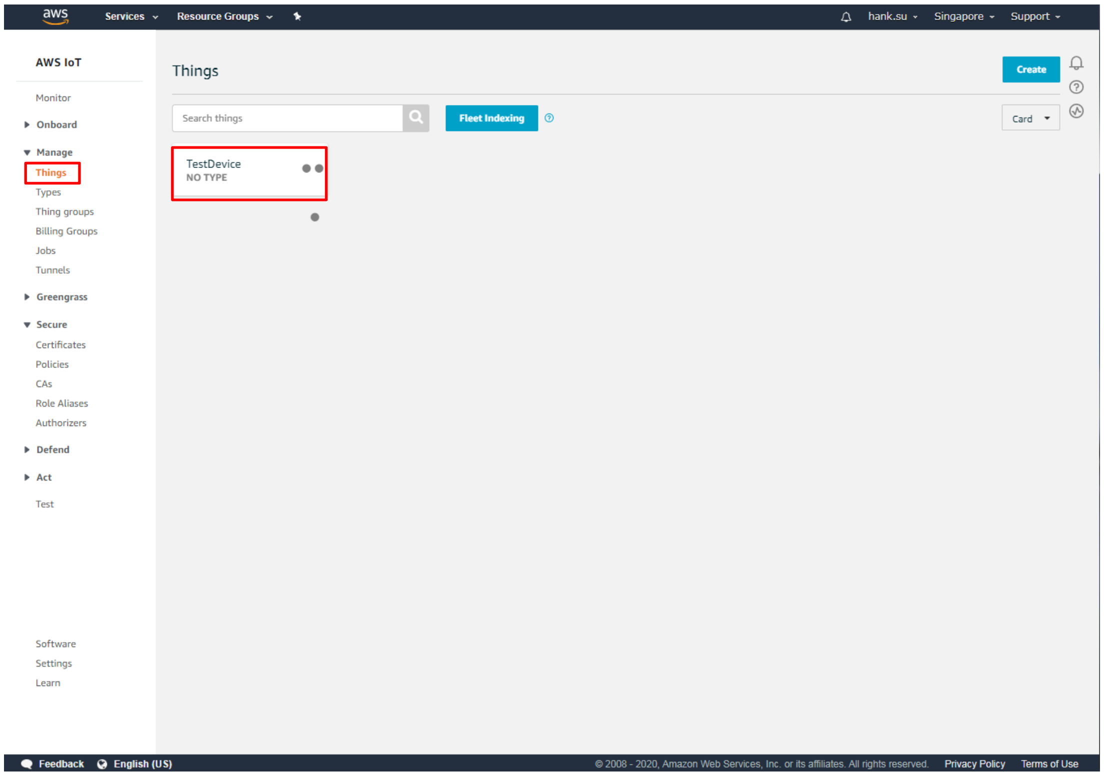
Resource ARN: arn:aws:iot:aaaaaaaaaaaa:bbbbbbbbbb:client/*

Effect: [✓ Allow] Deny

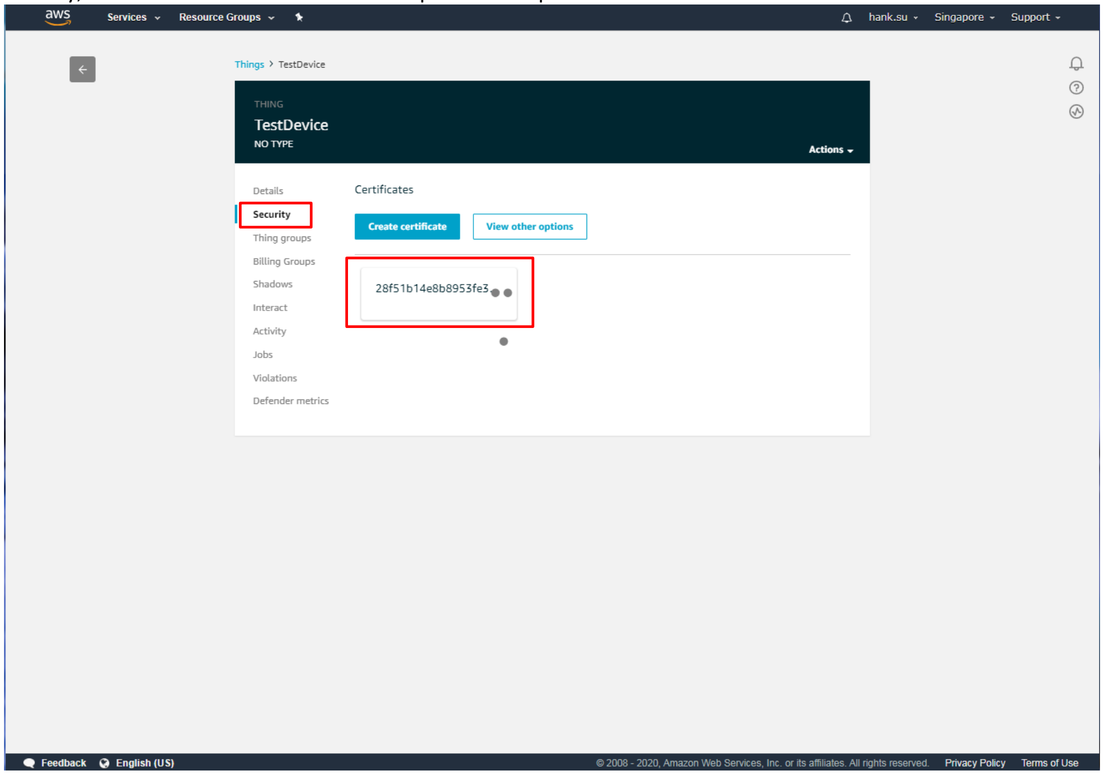
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

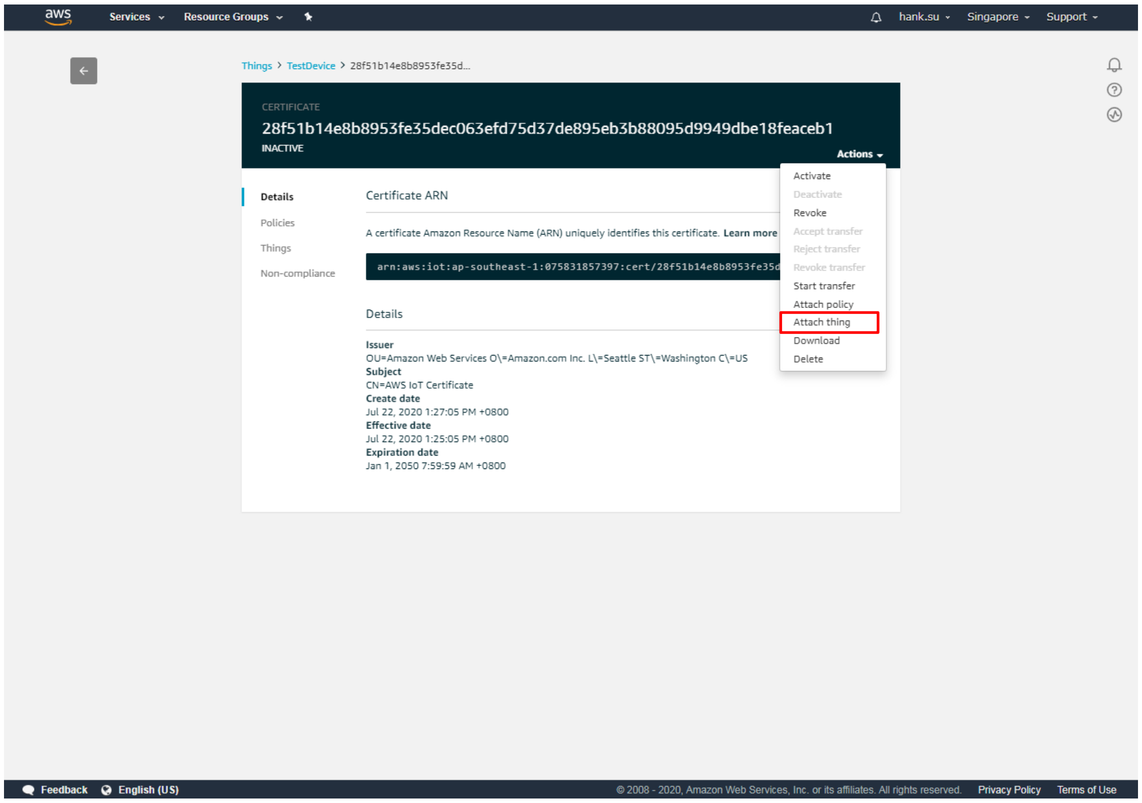
1.3 Attach Policy to Test Device

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.





CERTIFICATE

28f51b14e8b8953fe35dec063efd75d37de895eb3b88095d9949dbe18feaceb1

INACTIVE

Certificate ARN

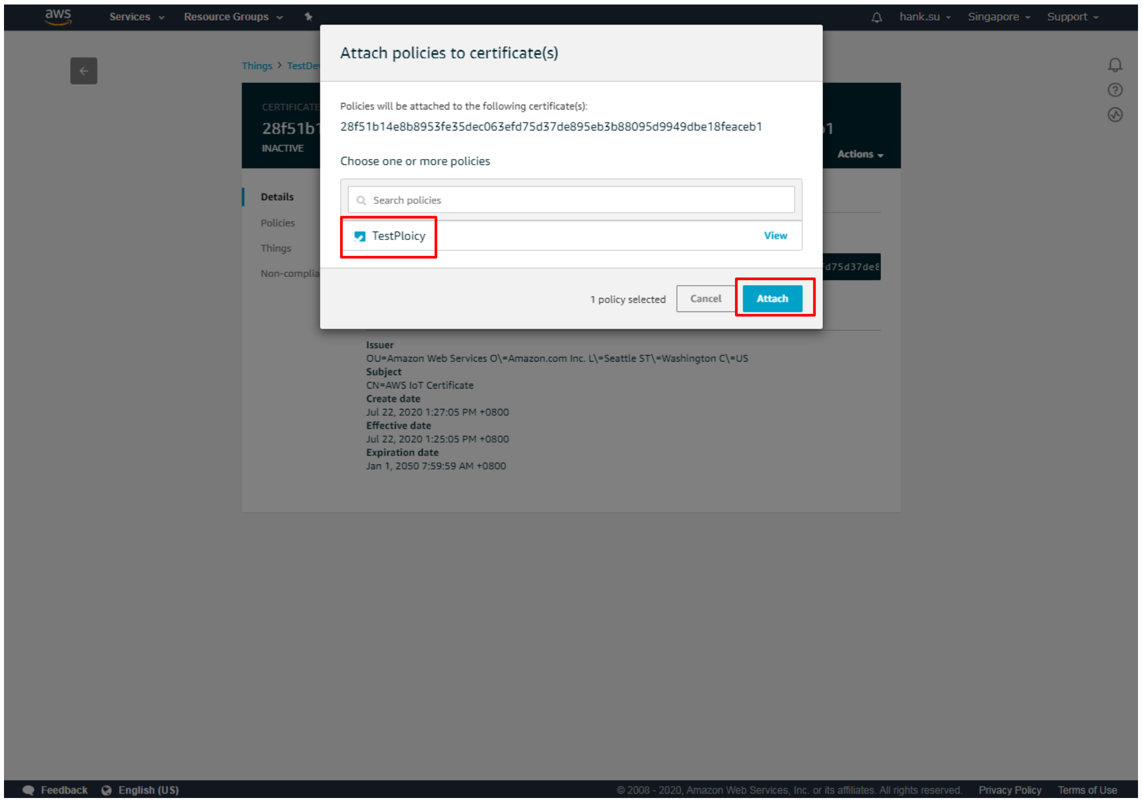
A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:ap-southeast-1:075831857397:cert/28f51b14e8b8953fe35d
```

Details

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Jul 22, 2020 1:27:05 PM +0800
Effective date
Jul 22, 2020 1:25:05 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
28f51b14e8b8953fe35dec063efd75d37de895eb3b88095d9949dbe18feaceb1

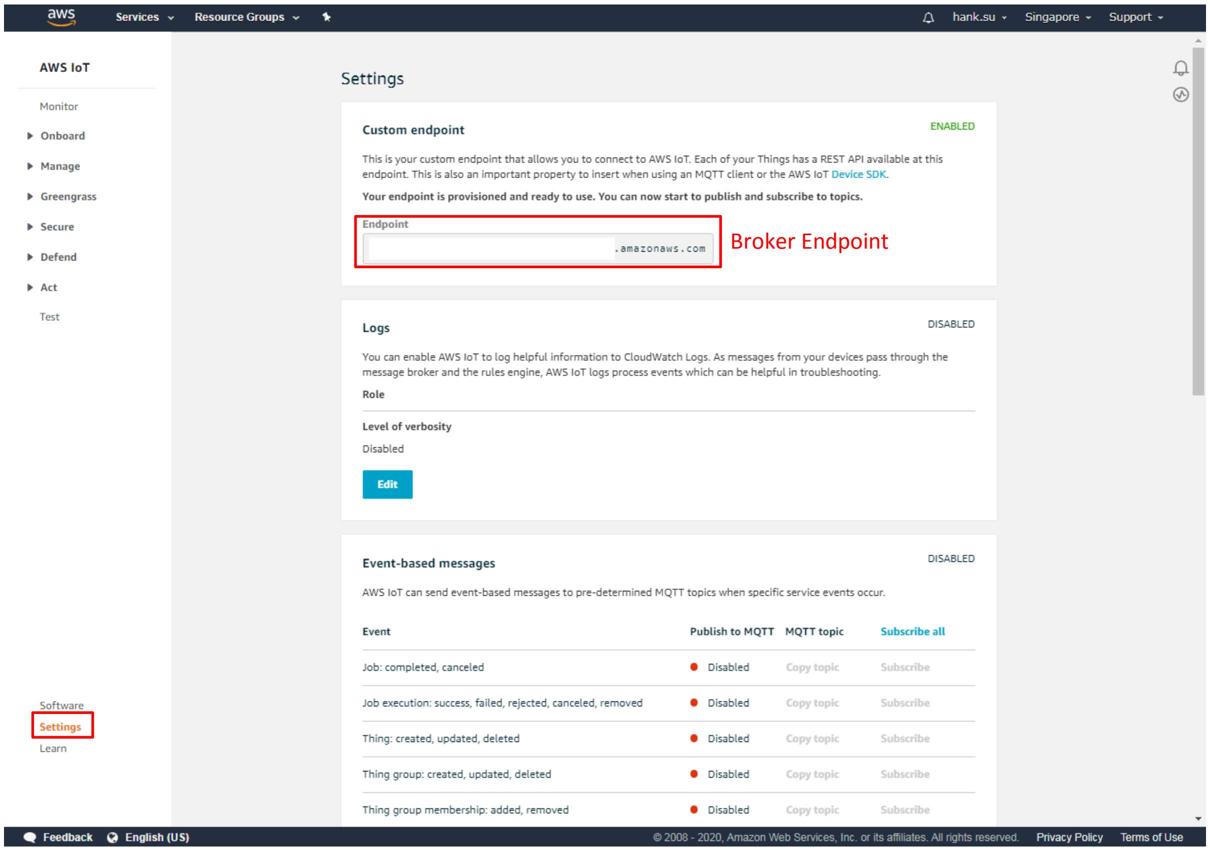
Choose one or more policies

TestPloicy

1 policy selected

1.4 Setting Test Device with AmebaD Amazon FreeRTOS Source

1.4.1 Get Broker Endpoint



Settings

Custom endpoint ENABLED

This is your custom endpoint that allows you to connect to AWS IoT. Each of your Things has a REST API available at this endpoint. This is also an important property to insert when using an MQTT client or the AWS IoT Device SDK.

Your endpoint is provisioned and ready to use. You can now start to publish and subscribe to topics.

Endpoint
.amazonaws.com

Logs DISABLED

You can enable AWS IoT to log helpful information to CloudWatch Logs. As messages from your devices pass through the message broker and the rules engine, AWS IoT logs process events which can be helpful in troubleshooting.

Role

Level of verbosity
Disabled

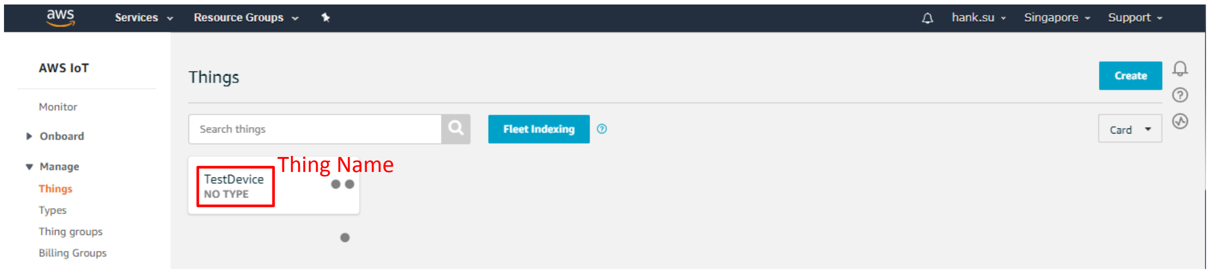
Event-based messages DISABLED

AWS IoT can send event-based messages to pre-determined MQTT topics when specific service events occur.

Event	Publish to MQTT	MQTT topic	Subscribe all
Job: completed, canceled	● Disabled	Copy topic	Subscribe
Job execution: success, failed, rejected, canceled, removed	● Disabled	Copy topic	Subscribe
Thing: created, updated, deleted	● Disabled	Copy topic	Subscribe
Thing group: created, updated, deleted	● Disabled	Copy topic	Subscribe
Thing group membership: added, removed	● Disabled	Copy topic	Subscribe

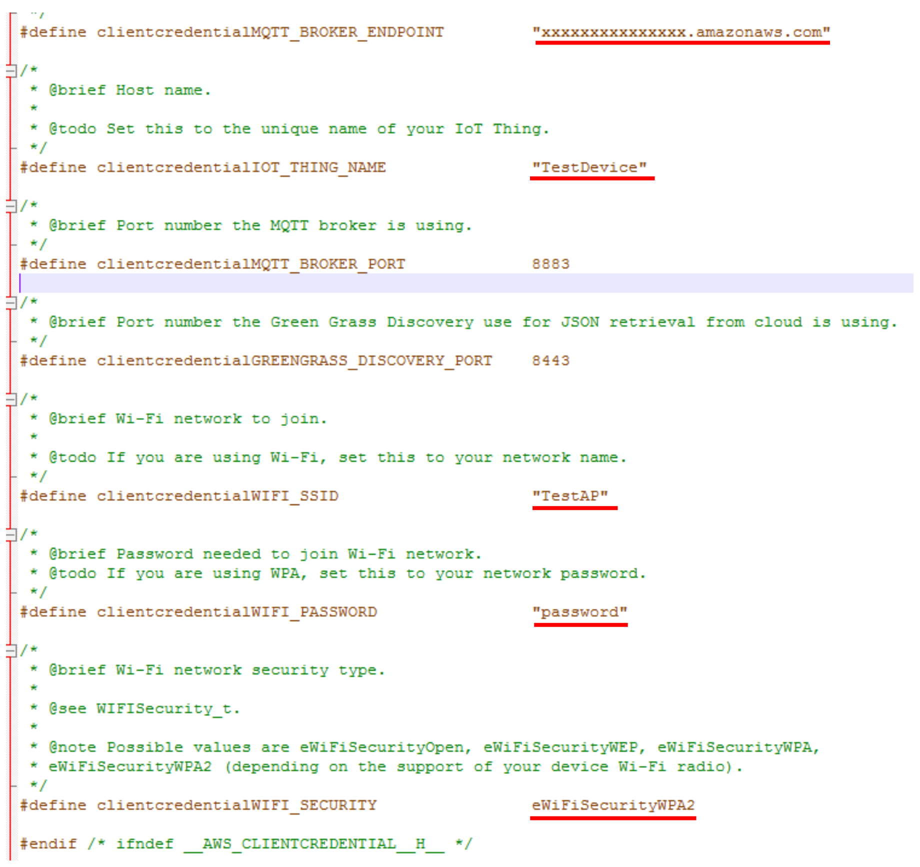
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

1.4.2 Get Thing Name



1.4.3 Setup IoT Core Information with AmebaD Amazon FreeRTOS

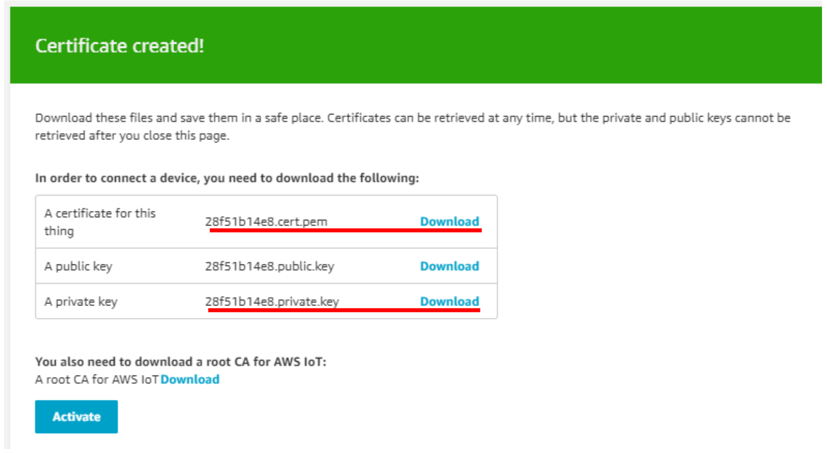
Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in “ambd_amazon-freertos/blob/master/demos/include/aws_clientcredential.h”



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

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

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

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

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

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

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

#endif /* ifndef __AWS_CLIENTCREDENTIAL__H__ */
```

1.4.4 Setup Thing's Private Key and Certificate

Filled keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in "ambd_amazon-freertos/blob/master/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-certifiacte.pem and xxxxxxxx-private.pem.key



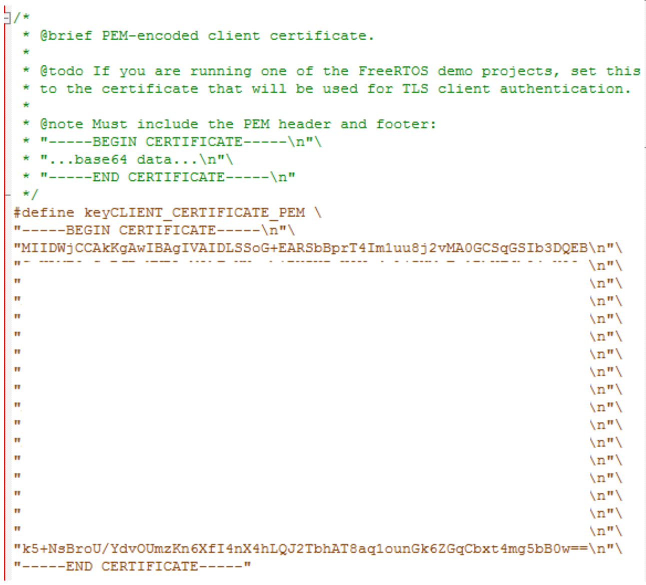
Certificate created!

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.

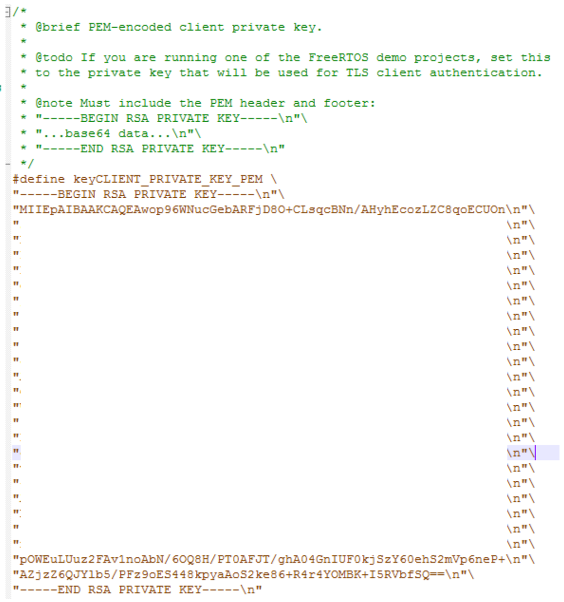
In order to connect a device, you need to download the following:

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



```
/*
 * @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"\
"[redacted]                                                      \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"                                                                \n"\
"k5+NsBroU/YdvOUmzKn6XfI4nX4hLQJ2TbhAT8aq1ounGk6ZGqCbxt4mg5bB0w==\n"\
"-----END CERTIFICATE-----"
```



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

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/).

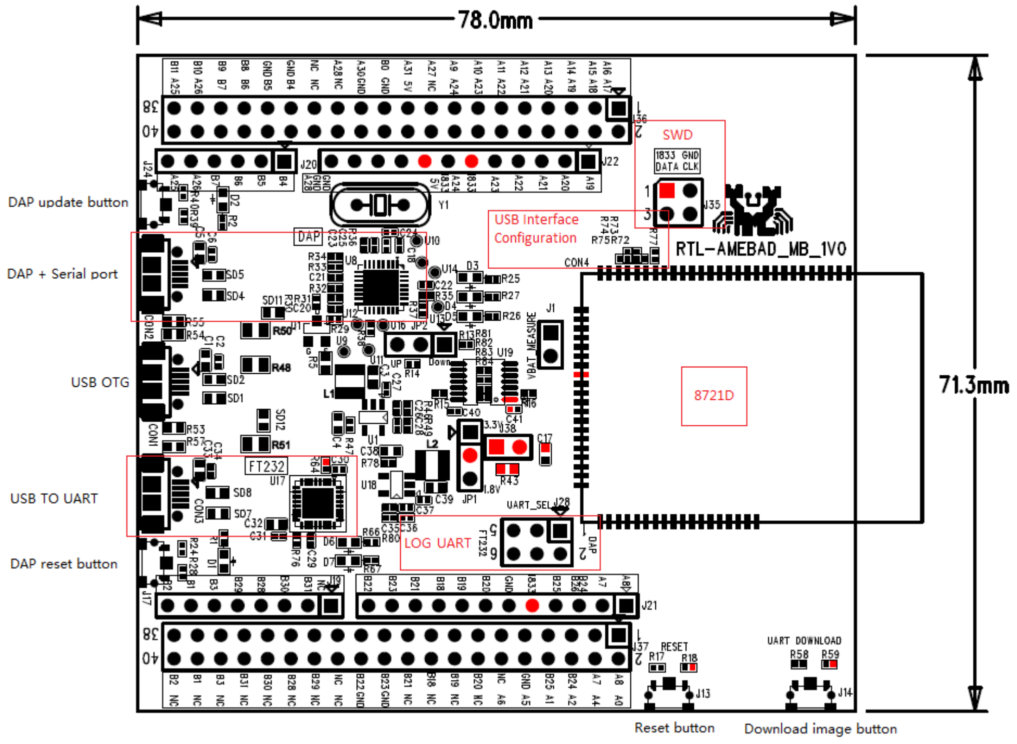


Fig 2-1 AmebaD Demo board

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 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.

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

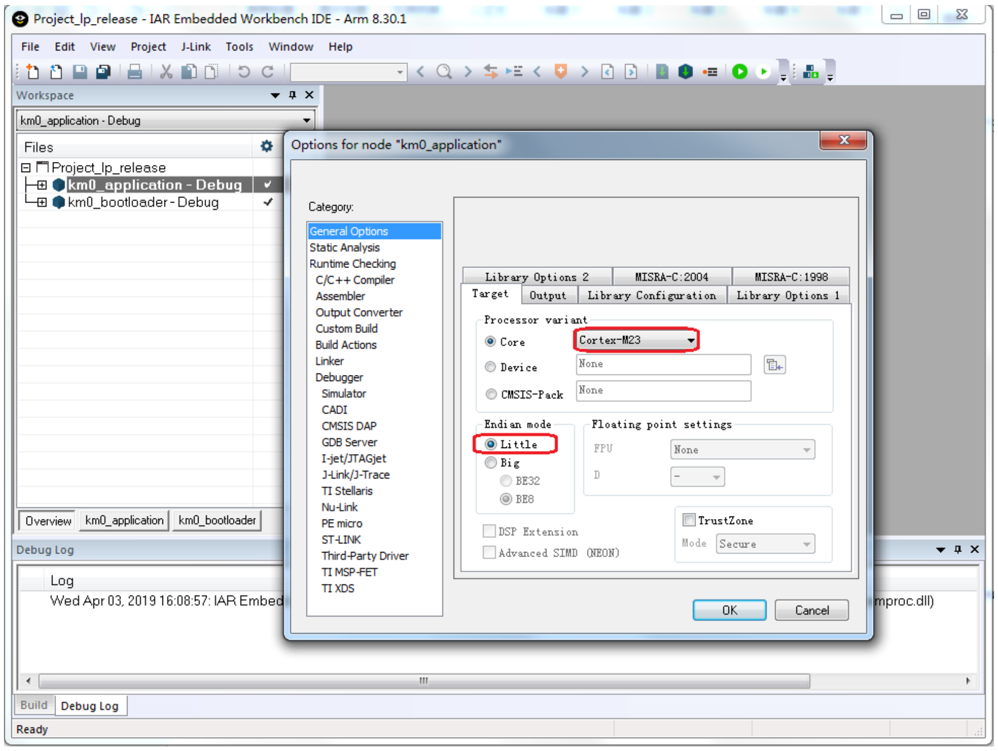
3.2.2 IAR Build

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

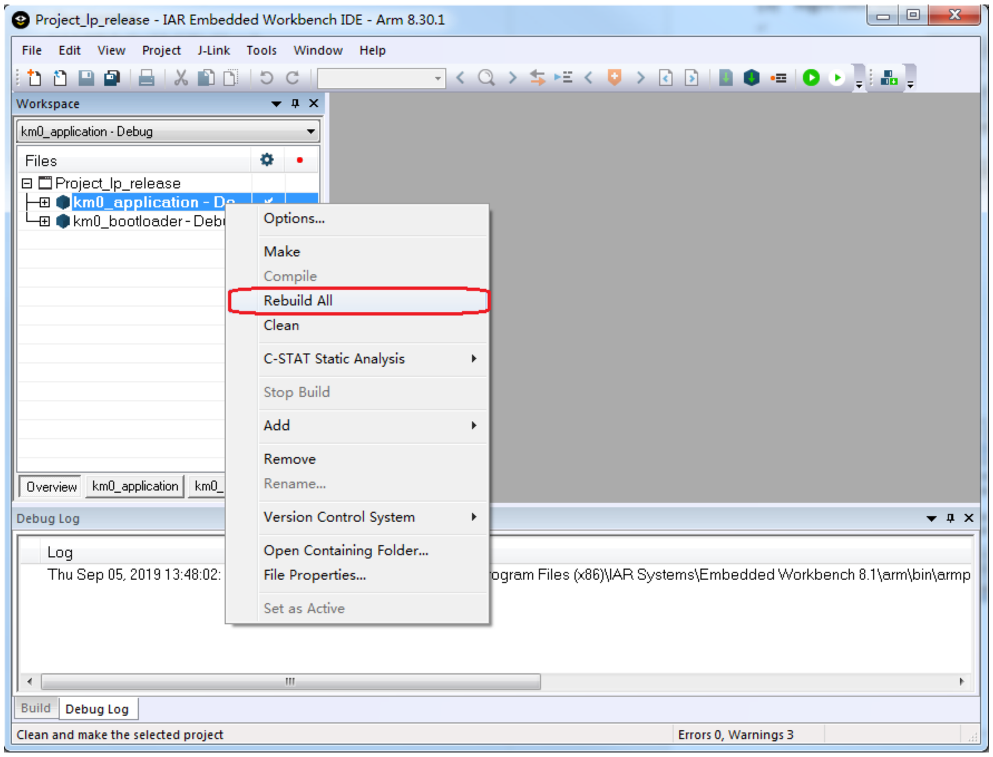
The following steps show how to build KM0 project:

(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.



```
Log
Wed Apr 03, 2019 16:08:57: IAR Embed... mproc.dll)
```

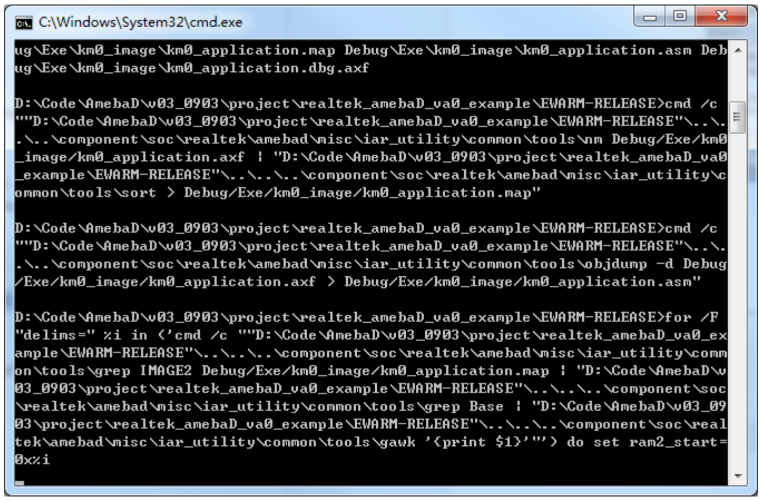
Fig 3-1 KM0 processor options



```
Log
Thu Sep 05, 2019 13:48:02: ... ogram Files (x86)\IAR Systems\Embedded Workbench 8.1\arm\bin\armp
```

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
ug\Exe\km0_image\km0_application.map Debug\Exe\km0_image\km0_application.asm Deb
ug\Exe\km0_image\km0_application.dbg.axf

D:\Code\AmebaD\v03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE>cmd /c
""D:\Code\AmebaD\v03_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\v03_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\v03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE>cmd /c
""D:\Code\AmebaD\v03_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\v03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE>for /F
"delims=" %i in ('cmd /c ""D:\Code\AmebaD\v03_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\v
03_0903\project\realtek_amebaD_va0_example\EWARM-RELEASE"\..\..\..\component\soc
\realtek\amebad\misc\iar_utility\common\tools\grep Base | "D:\Code\AmebaD\v03_09
03\project\realtek_amebaD_va0_example\EWARM-RELEASE"\..\..\..\component\soc\real
tek\amebad\misc\iar_utility\common\tools\gawk '{print $1}'"') do set ram2_start=
0x%i
```

(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**.

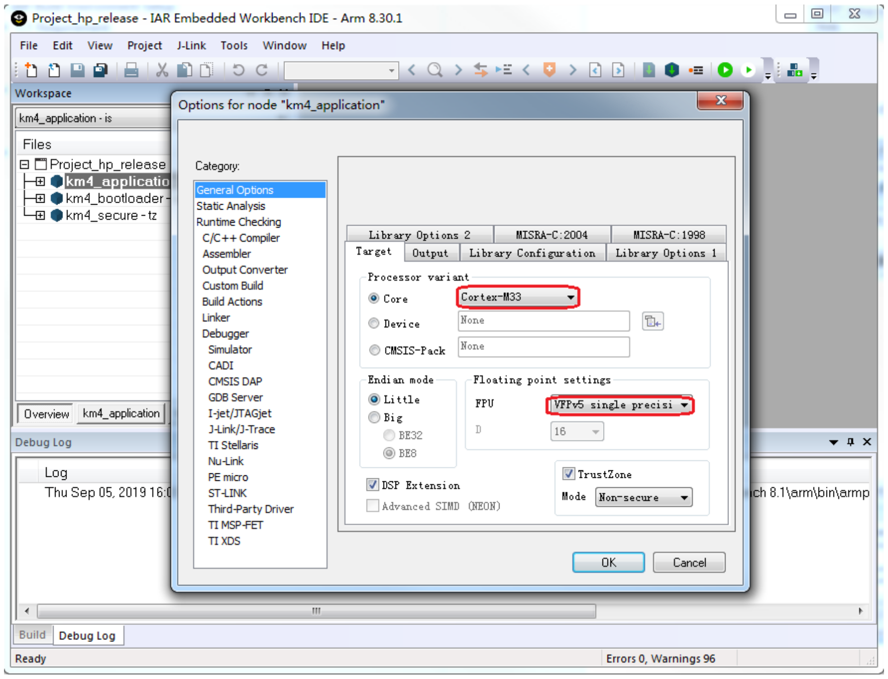
3.2.2.2 Building KM4 Project

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.



Project_hp_release - IAR Embedded Workbench IDE - Arm 8.30.1

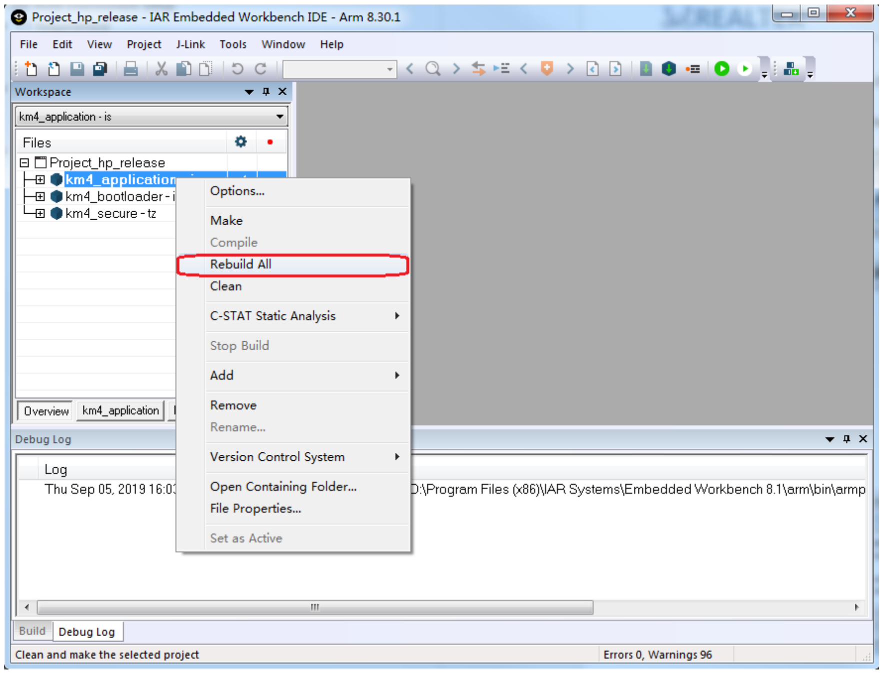
Options for node "km4_application"

Category: General Options

Target tab:
- Processor variant: Core [✓] Cortex-M33; Device None; CMSIS-Pack None
- Endian mode: [✓ Little] Big
- Floating point settings: FPU VFPv5 single precisi; D 16
- [✓] DSP Extension
- [ ] Advanced SIMD (NEON)
- [✓] TrustZone, Mode Non-secure

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.



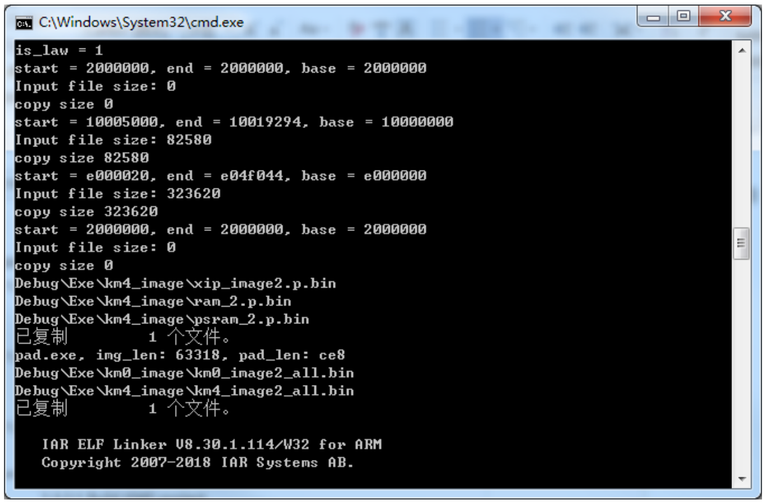
```
Thu Sep 05, 2019 16:0 ... D:\Program Files (x86)\IAR Systems\Embedded Workbench 8.1\arm\bin\armp
```

Clean and make the selected project   Errors 0, Warnings 96

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.


C:\Windows\System32\cmd.exe

```
is_law = 1
start = 2000000, end = 2000000, base = 2000000
Input file size: 0
copy size 0
start = 10005000, end = 10019294, base = 10000000
Input file size: 82580
copy size 82580
start = e000020, end = e04f044, base = e000000
Input file size: 323620
copy size 323620
start = 2000000, end = 2000000, base = 2000000
Input file size: 0
copy size 0
Debug\Exe\km4_image\xip_image2.p.bin
Debug\Exe\km4_image\ram_2.p.bin
Debug\Exe\km4_image\psram_2.p.bin
已复制         1 个文件。
pad.exe, img_len: 63318, pad_len: ce8
Debug\Exe\km0_image\km0_image2_all.bin
Debug\Exe\km4_image\km4_image2_all.bin
已复制         1 个文件。

   IAR ELF Linker V8.30.1.114/W32 for ARM
   Copyright 2007-2018 IAR Systems AB.
```

(5) 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**.

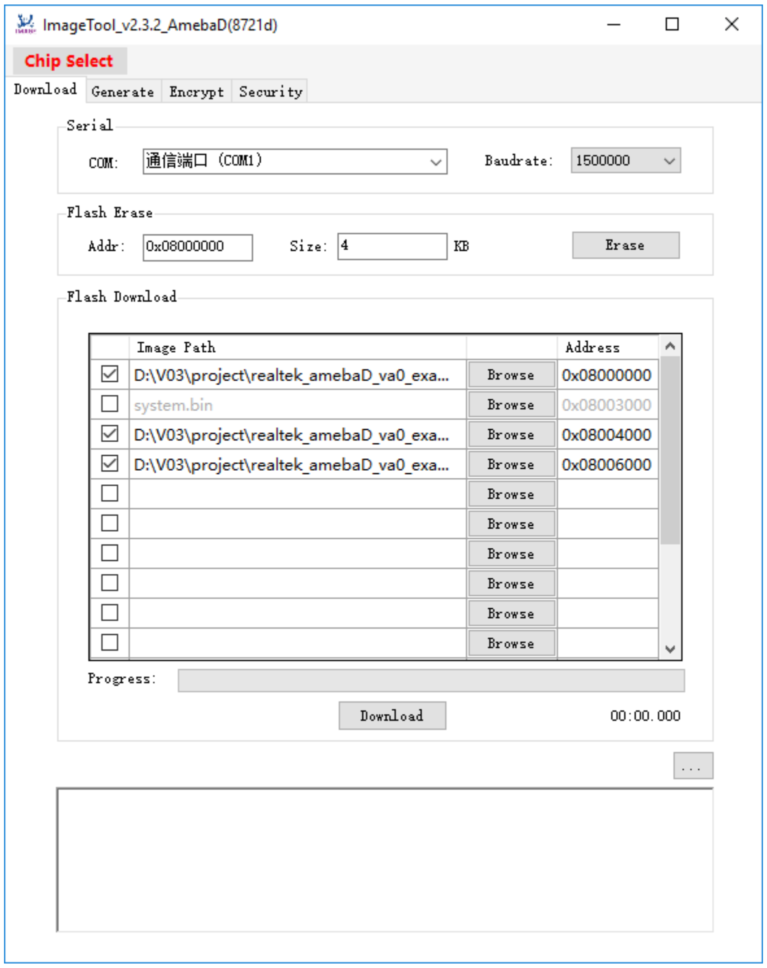
(6) The generated images can be downloaded by ImageTool:

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.



COM: 通信端口 (COM1)　Baudrate: 1500000

Addr: 0x08000000　Size: 4 KB

	Image Path		Address
☑	D:\V03\project\realtek_amebaD_va0_exa...	Browse	0x08000000
☐	system.bin	Browse	0x08003000
☑	D:\V03\project\realtek_amebaD_va0_exa...	Browse	0x08004000
☑	D:\V03\project\realtek_amebaD_va0_exa...	Browse	0x08006000
☐		Browse	
☐		Browse	
☐		Browse	
☐		Browse	
☐		Browse	
☐		Browse	

00:00.000

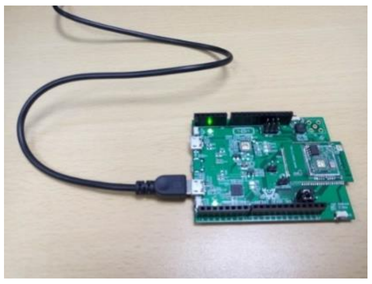
Fig 4-1 ImageTool UI

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.



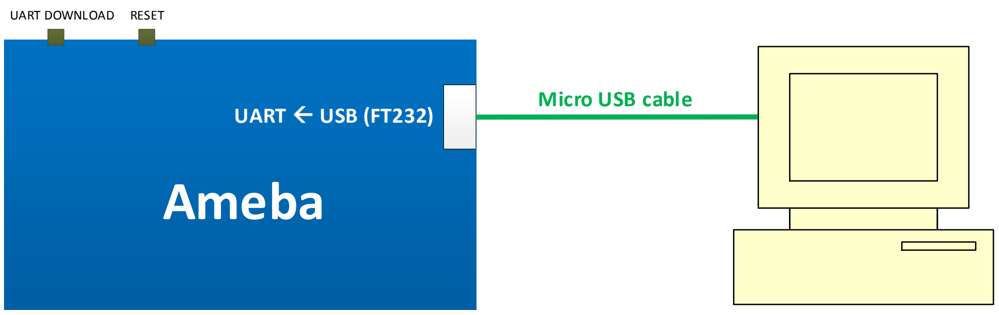
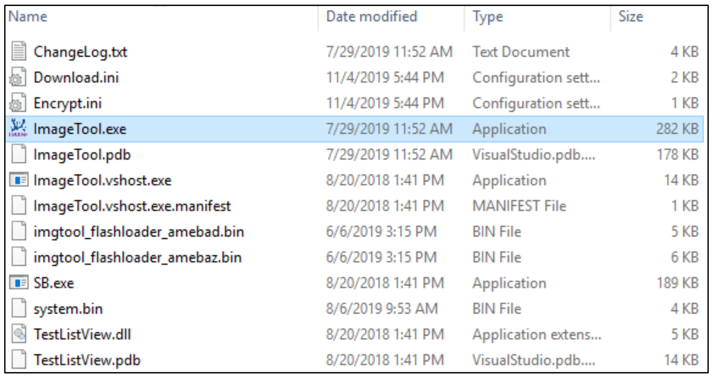


Fig 4-2 Hardware setup

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.

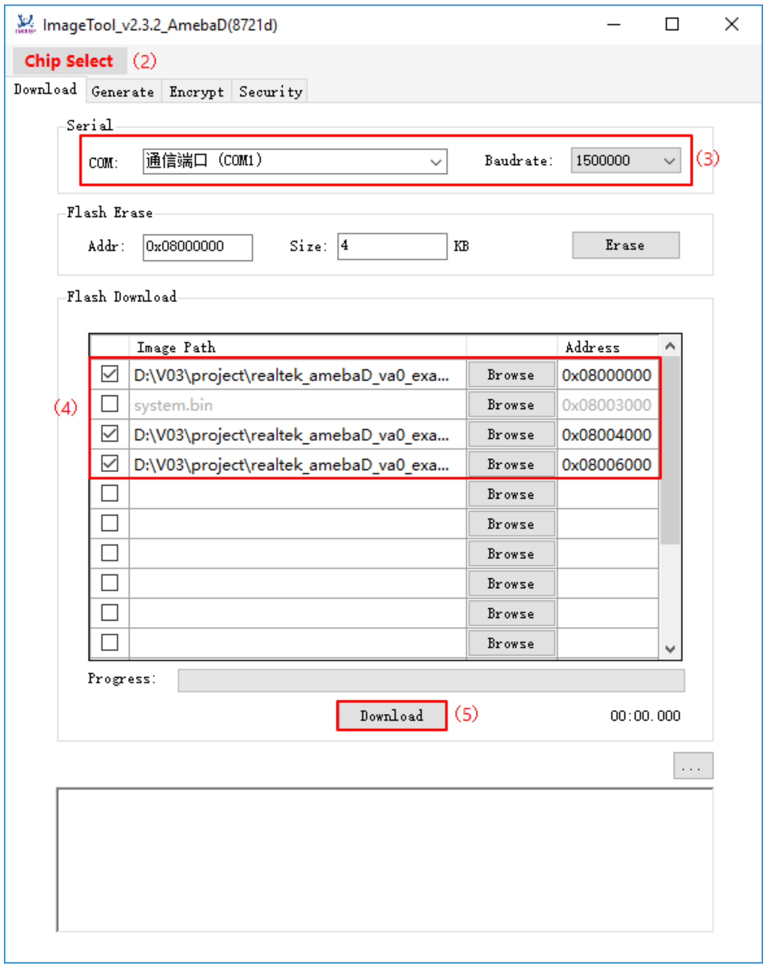
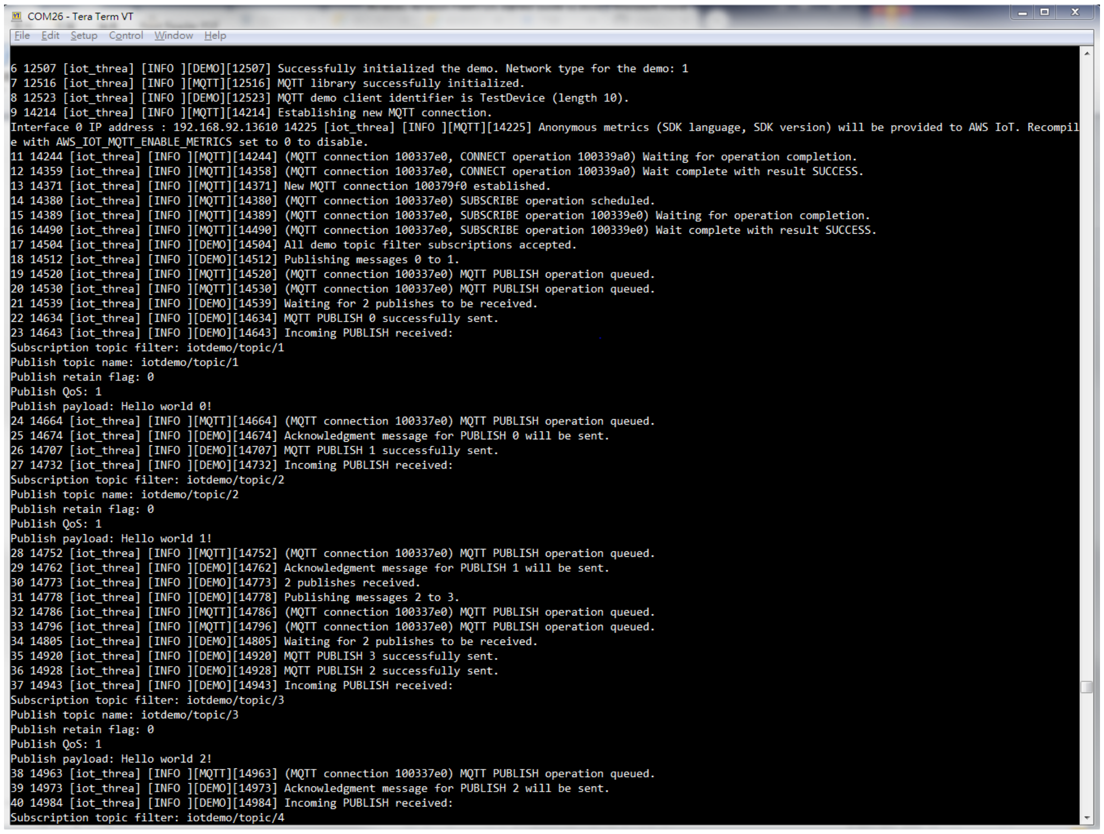


	Image Path		Address
☑	D:\V03\project\realtek_amebaD_va0_exa...	Browse	0x08000000
☐	system.bin	Browse	0x08003000
☑	D:\V03\project\realtek_amebaD_va0_exa...	Browse	0x08004000
☑	D:\V03\project\realtek_amebaD_va0_exa...	Browse	0x08006000
☐		Browse	
☐		Browse	
☐		Browse	
☐		Browse	
☐		Browse	
☐		Browse	

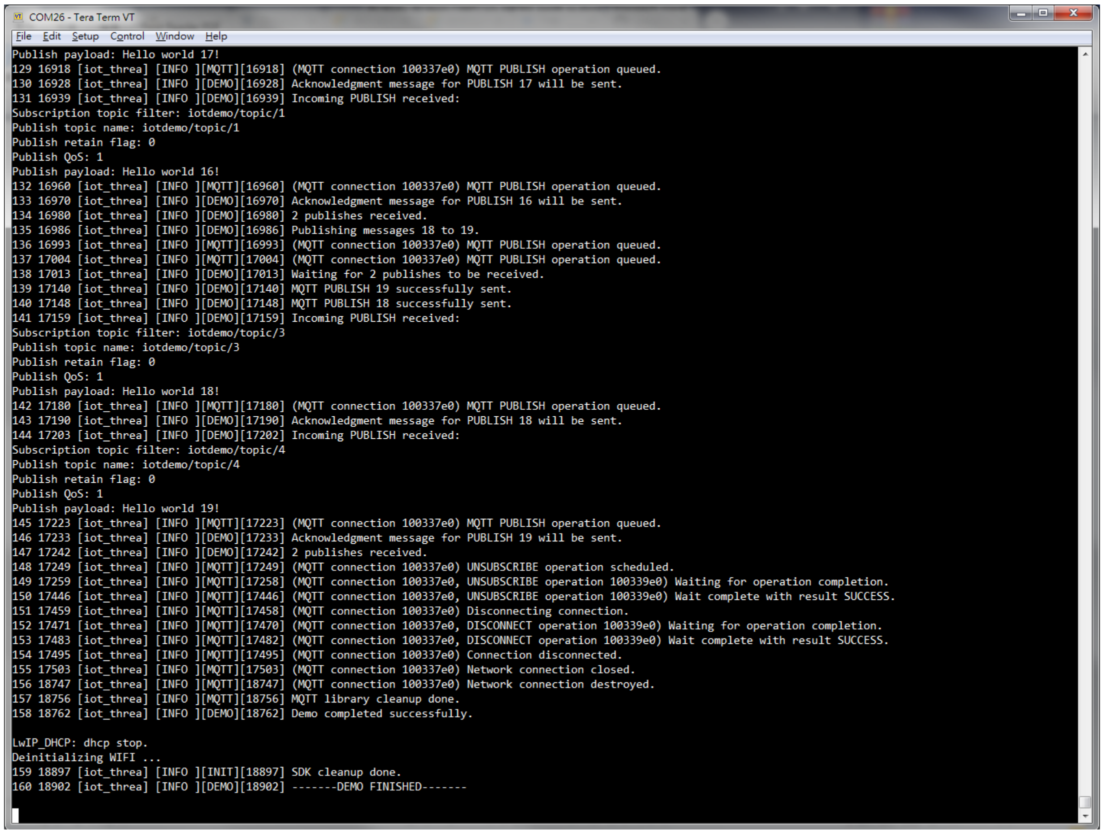
Fig 4-3 ImageTool ‘Download’ tabpage setting

5 Testing

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

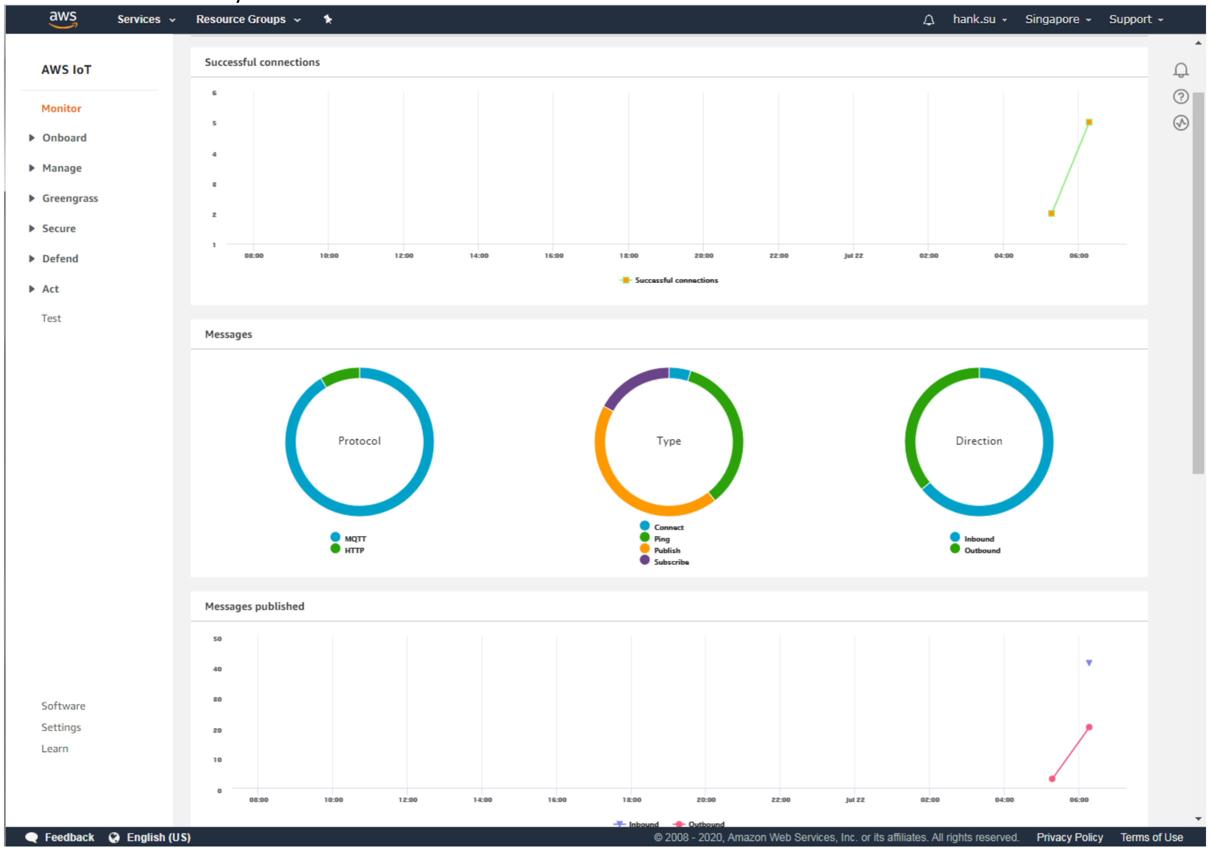


```
6 12507 [iot_threa] [INFO ][DEMO][12507] Successfully initialized the demo. Network type for the demo: 1
7 12516 [iot_threa] [INFO ][MQTT][12516] MQTT library successfully initialized.
8 12523 [iot_threa] [INFO ][DEMO][12523] MQTT demo client identifier is TestDevice (length 10).
9 14214 [iot_threa]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



```
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][17202] 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) Waiting for operation completion.
150 17446 [iot_threa] [INFO ][MQTT][17446] (MQTT connection 100337e0, UNSUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
151 17459 [iot_threa] [INFO ][MQTT][17458] (MQTT connection 100337e0) Disconnecting connection.
152 17471 [iot_threa] [INFO ][MQTT][17470] (MQTT connection 100337e0, DISCONNECT operation 100339e0) Waiting for operation completion.
153 17483 [iot_threa] [INFO ][MQTT][17482] (MQTT connection 100337e0, DISCONNECT operation 100339e0) Wait complete with result SUCCESS.
154 17495 [iot_threa] [INFO ][MQTT][17495] (MQTT connection 100337e0) Connection disconnected.
155 17503 [iot_threa] [INFO ][MQTT][17503] (MQTT connection 100337e0) Network connection closed.
156 18747 [iot_threa] [INFO ][MQTT][18747] (MQTT connection 100337e0) Network connection destroyed.
157 18756 [iot_threa] [INFO ][MQTT][18756] MQTT library cleanup done.
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.


aws
Services
Resource Groups
hank.su
Singapore
Support
AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Test
Software
Settings
Learn
Successful connections
Successful connections
Messages
Protocol
MQTT
HTTP
Type
Connect
Ping
Publish
Subscribe
Direction
Inbound
Outbound
Messages published
Inbound
Outbound
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use