Talaria TWO(INP2045)

Low Power Multi-Protocol Wireless Platform SoC

IEEE 802.11 b/g/n, BLE 5.0

Application Note

AWS IoT Device SDK- Secure MQTT, Device Shadow and Jobs Service

Release: 08-10-2021

InnoPhase, Inc.
6815 Flanders Drive
San Diego, CA 92121
innophaseinc.com

Revision History

Version Number	Date	Comments
1.0	07-04-2020	First release.
2.0	04-23-2021	Enhanced application outputs to print SDK version.
3.0	06-29-2021	ELF paths updated.
3.1	08-10-2021	Updated for SDK 2.3 release.

Contents

1 Figures

2 Tables

3 Terms & Definitions

AWS	Amazon Web Services
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
SDK	Software Development Kit
VM	Virtual Machine

4 Introduction

The applications discussed in this document provides a brief on using Talaria TWO board and the SDK with Amazon Web Services (AWS) IoT.

More information on the AWS IoT developer guide can be found at: https://docs.aws.amazon.com/iot/latest/developerguide/.

5 AWS IoT Device SDK Embedded C

AWS IoT C SDK - `aws-iot-device-sdk-embedded-C` is ported onto Talaria TWO. The code accompanying this application has codes combined with AWS IOT lib, external code used by AWS(TLS and json) and Talaria TWO port specific codes.

For more information on `aws-iot-device-sdk-embedded-C` can be found here:

https://github.com/aws/aws-iot-device-sdk-embedded-C

In the accompanying code, `talaria_t2` platform specific porting and implementation changes are housed in folder `talaria_t2`.

Following is the folder structure of the accompanying code:

1. Folder `aws_core`: contains `aws_core` client code from `aws-iot-device-sdk-embedded-C`
2. Folder `external`: contains third party lib used in this example jsmn
3. Folder `talaria_t2`: `talaria_t2` platform specific implementation
4. `libaws_iot_t2.a`: the library generated for `aws_core` client code from `aws-iot-device-sdk-embedded-`
5. Folder `cert`: contains the client certificate
6. Folder `sample`: Contains AWS samples like subscribe, publish, job and shadow
   Files in the `sample` folder are also from AWS Device SDK Github with minor changes in WCM related APIs used to connect to network.

6 Sample Applications

1. Sample application 1: `sample_pub_sub`
   Provides details on how to publish/subscribe to MQTT topics and send/receive messages.
2. Sample application 2: `shadow_sample`
   Provides details on how to use the `AWS IoT Device Shadow` service, to update the shadow of a device.
3. Sample application 2: `jobs_sample`
   Provides details on how to create a job in AWS IoT and have the device execute it.

7 AWS Set-up

1. Create an AWS IoT account
   An AWS account is needed to run the sample applications. AWS accounts include twelve months of Free Tier Access.
   More information on: https://portal.aws.amazon.com/billing/signup#/start

2. Create and register device/thing
   Device/thing must be registered onto the AWS IoT registry.
   Use the following link to AWS IoT user guide to download the necessary certificates and private key: https://docs.aws.amazon.com/iot/latest/developerguide/create-iot-resources.html.

   Note:
   - Ensure the downloaded certificates and private key are saved in a secure location as it provides only for a one-time download.
   - To determine your custom AWS, download location, go to AWS IoT Console -> Settings

3. Save Certificate and Private Key onto the device
   There are four certificates that will be downloaded from AWS for the created Thing. Out of which `Public Key` will not be used in this example.
   Save the certificates (as there is a need to install these in the device) and rename them as per the following table to create file system and write it into Talaria TWO using the download tool:

File Name	Rename
private.pem.key	aws_device_pkey
device.pem.crt	aws_device_cert
amazon-root-CA-1.pem	aws_root_ca
Public Key	Not used in these examples

Table 1: AWS Certificates

4. Create and attach a Policy to the certificate associated with the device/thing. To allow interaction with all the topics and other resources used in the example codes, a wildcard policy is set and attached to the thing's certificate. Please edit and update the policy to the following as shown:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
```



```
      "Resource": "*"

    }

  ]

}
```


8 Programming VM-based applications

8.1 Programming Talaria TWO board with certificates

The default path for AWS should be: `/root/certs/aws/app`.

8.1.1 Show File System Contents

Click on `Show File System Contents` to see the current available files in the file system.

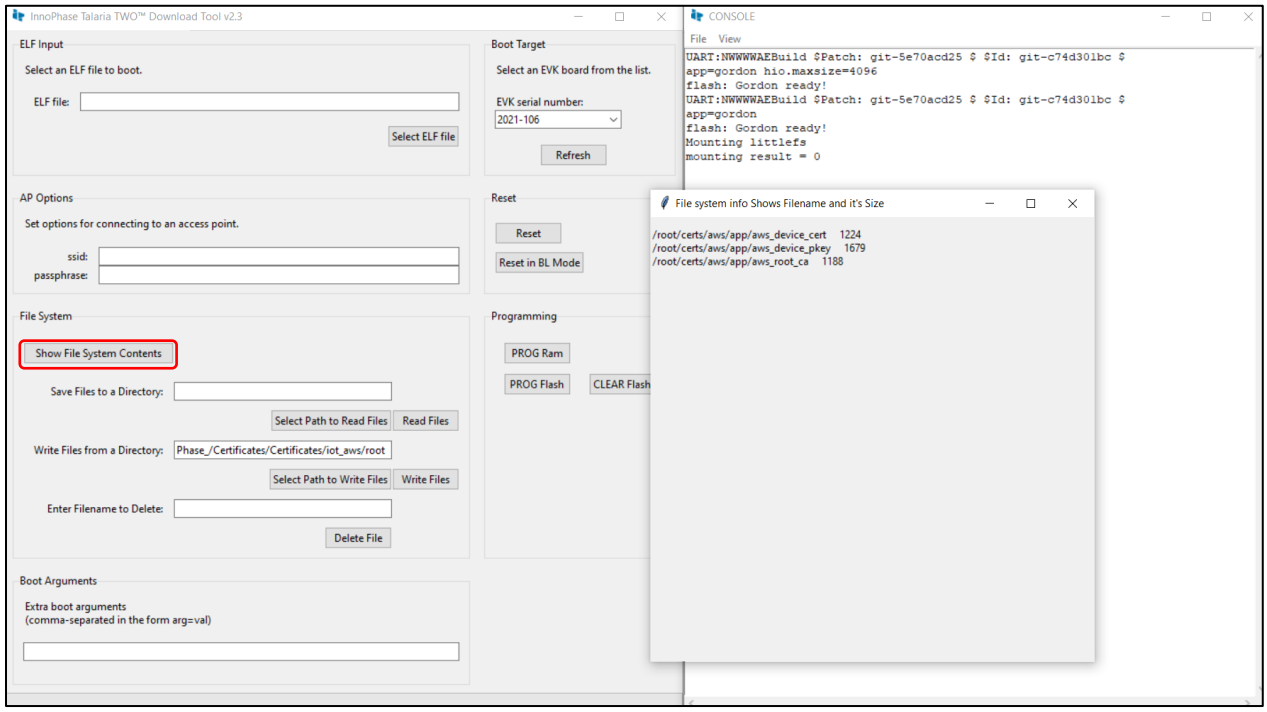

Figure 1: Show File System Contents

8.1.2 Write Files

To write files into Talaria TWO, user must create a folder with the name `root` and place all certificates either directly into the root or they can create multiple subfolders (for example: `/root/iot_aws`) and place the certificates inside the sub-directory and update the path as per the file system in the `.c` file.

The default path is `/root/certs/aws/app`. If user writes into `root/iot_aws/cert_names` then the path should be updated in the `.c` file accordingly. Any number of files/folders inside `root` will be written.

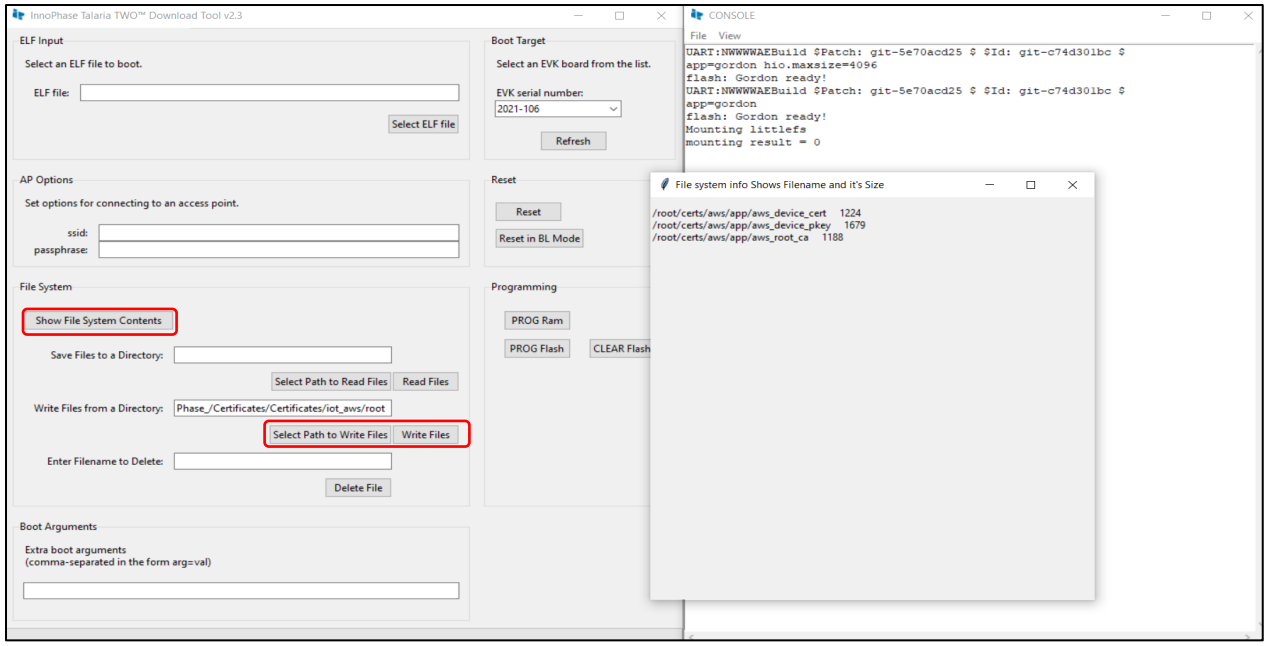

Figure 2: Write certificates to Talaria TWO

8.2 Programming Talaria TWO board with ELF

Program the ELFs onto Talaria TWO using the Download tool. Launch the Download tool provided with InnoPhase Talaria TWO SDK. In the GUI window, select the appropriate EVK from the drop-down and load the appropriate ELF. Click on `Prog Flash`.

For details on using the Download tool, refer to the document: UG_Download_Tool.pdf.

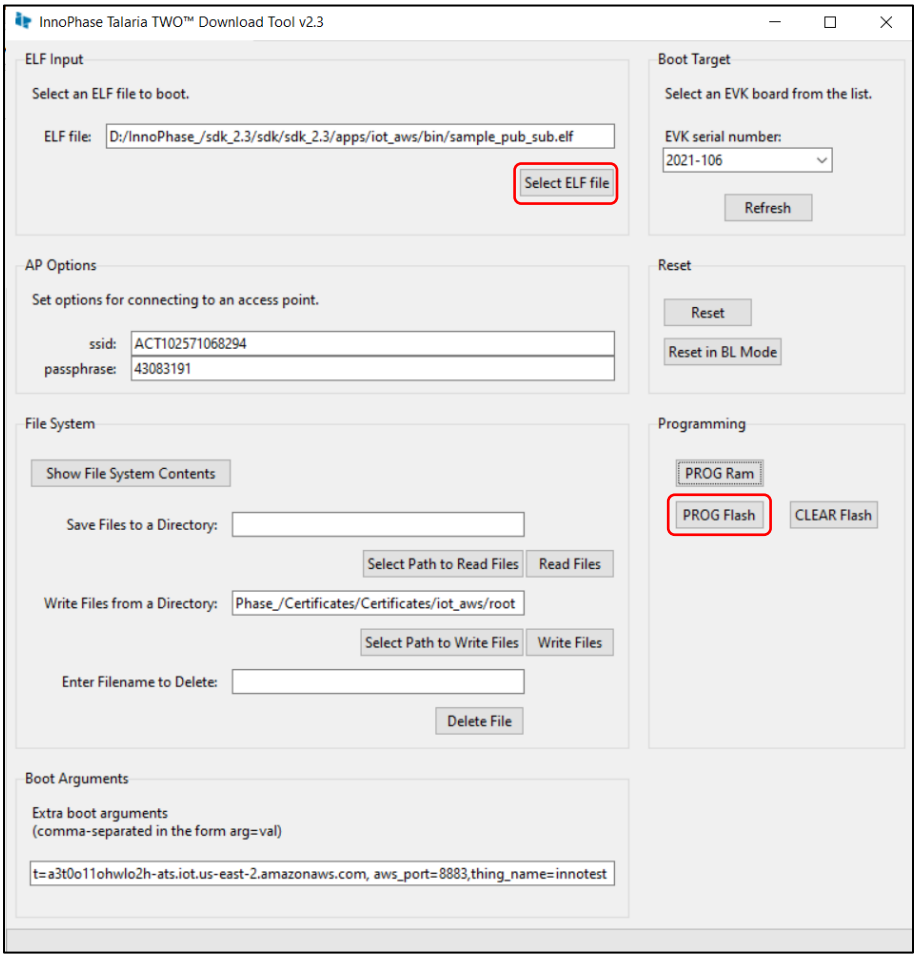

Figure 3: Programming Talaria TWO - Download tool output

Boot Arguments:

```
aws_host=xxxxxx.amazonaws.com, aws_port=8883,thing_name=xxxxx
```

Note: Replace the xxxxxx with the appropriate details.

Ensure correct boot parameters are supplied to your Wi-Fi network and the information from the device/thing created previously on AWS.

1. `aws_host` is the custom AWS location.
2. `thing_name` is the name of the device/thing we created earlier.

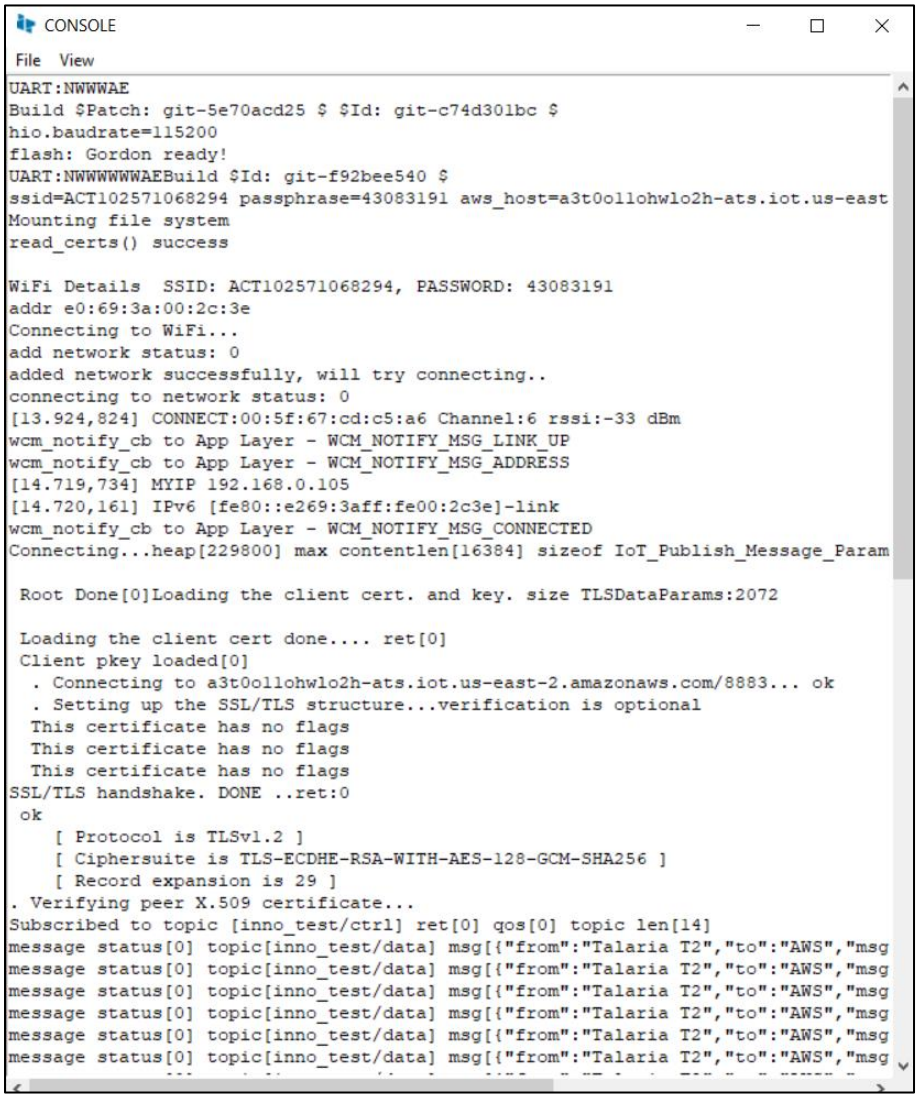

Figure 4: Console output

9 MQTT Publish and Subscribe

9.1 Subscribe

1. In the AWS IoT Console, go to `Test`.
2. In the Subscription topic text box, type `inno_test/data` and click on `Subscribe`.

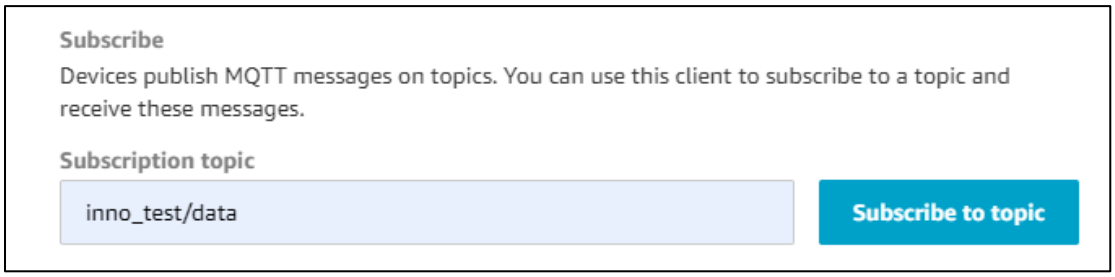

Figure 5: Subscribe to topic

3. In the Subscriptions tab, click on `inno_test/data`.

Figure 6: Subscriptions – inno_test/data

9.2 Running the sample application

1. Program the Talaria TWO board with `sample_pub_sub.elf` available at: `sdk_2.3/apps/iot_aws/bin` using the process described in section 8.2.

2. Upon successful execution, the following console output will be provided:

```
UART:NWWWWWWAEBuild $Id: git-f92bee540 $

ssid=ACT102571068294 passphrase=43083191 aws_host=a3t0o11ohwlo2h-

ats.iot.us-east-2.amazonaws.com aws_port=8883 thing_name=innotest

Mounting file system

read_certs() success



WiFi Details  SSID: ACT102571068294, PASSWORD: 43083191


addr e0:69:3a:00:2c:3e

Connecting to WiFi...

add network status: 0

added network successfully, will try connecting..

connecting to network status: 0

[13.924,824] CONNECT:00:5f:67:cd:c5:a6 Channel:6 rssi:-33 dBm

wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_LINK_UP

wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_ADDRESS

[14.719,734] MYIP 192.168.0.105

[14.720,161] IPv6 [fe80::e269:3aff:fe00:2c3e]-link

wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_CONNECTED

Connecting...heap[229800] max contentlen[16384] sizeof

IoT_Publish_Message_Params (16)


 Root Done[0]Loading the client cert. and key. size TLSDataParams:2072
```



```
 Loading the client cert done.... ret[0]
 Client pkey loaded[0]
  . Connecting to a3t0o11ohwlo2h-ats.iot.us-east-
2.amazonaws.com/8883... ok
  . Setting up the SSL/TLS structure...verification is optional
  This certificate has no flags
  This certificate has no flags
  This certificate has no flags
SSL/TLS handshake. DONE ..ret:0
 ok
    [ Protocol is TLSv1.2 ]
    [ Ciphersuite is TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 ]
    [ Record expansion is 29 ]
. Verifying peer X.509 certificate...
Subscribed to topic [inno_test/ctrl] ret[0] qos[0] topic len[14]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":1}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":2}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":3}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":4}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":5}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":6}]
```



```
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":7}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":8}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":9}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":10}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":11}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":12}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":13}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":14}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":15}]
```


3. The AWS IoT dashboard will appear as in Figure 7.

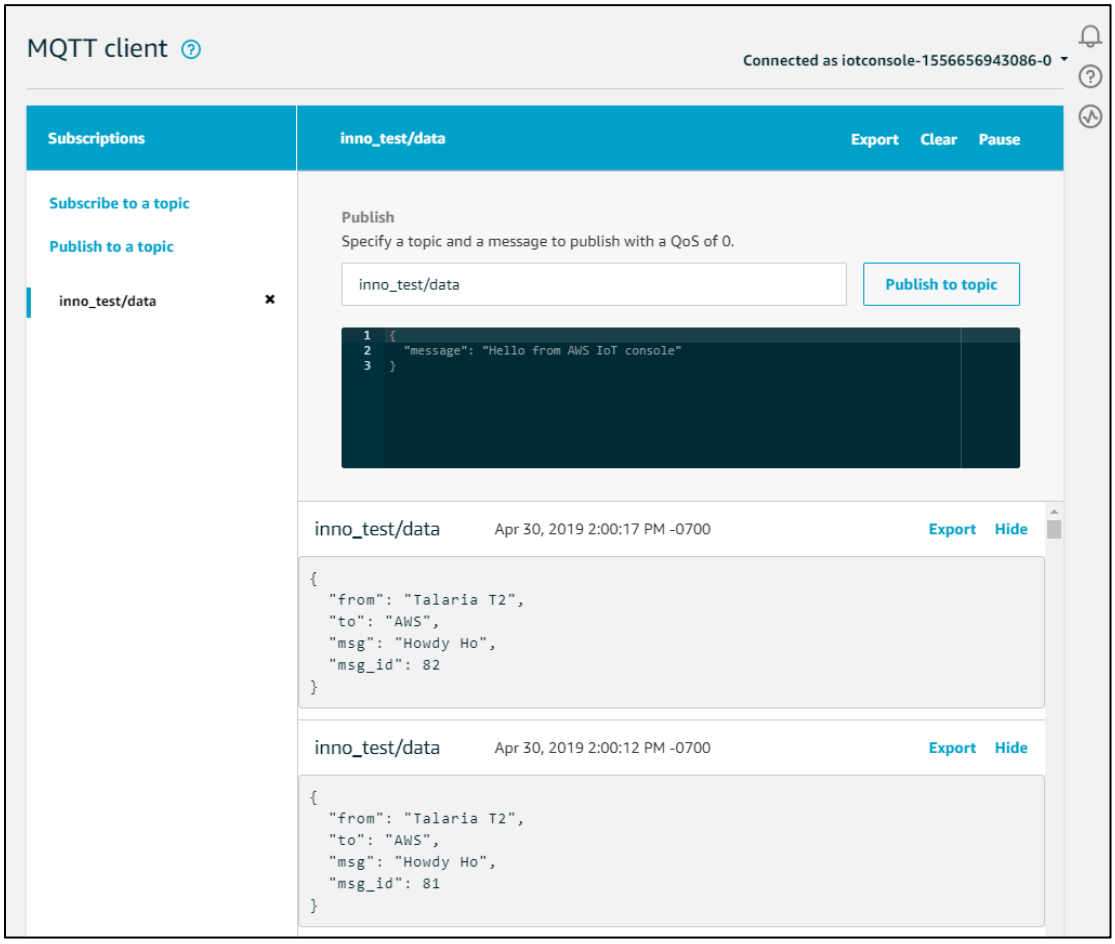

Figure 7: AWS IoT Dashboard

9.3 Publish

1. Program the Talaria TWO board with `sample_pub_sub.elf` available at: `sdk_2.3/apps/iot_aws/bin` using the process described in section 8.2.
2. In the AWS IoT Console, go to `Test`.
3. On the Publish topic text box, enter `inno_test/ctrl`.

Figure 8: Publish to topic

4. Copy and paste the following json formatted text into the colored console as shown in Figure 8.

```
{
  "from": "AWS IoT console"
  "to": "T2"
  "msg": "Hello from AWS IoT console"
}
```


5. On clicking Publish to topic, the following output is displayed in the console:

```
UART:NWWWWWWAEBuild $Id: git-f92bee540 $
ssid=ACT102571068294 passphrase=43083191 aws_host=a3t0o11ohwlo2h-
ats.iot.us-east-2.amazonaws.com aws_port=8883 thing_name=innotest
Mounting file system
read_certs() success



WiFi Details  SSID: ACT102571068294, PASSWORD: 43083191


addr e0:69:3a:00:2c:3e
Connecting to WiFi...
add network status: 0
added network successfully, will try connecting..
connecting to network status: 0
[13.924,824] CONNECT:00:5f:67:cd:c5:a6 Channel:6 rssi:-33 dBm
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_LINK_UP
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_ADDRESS
[14.719,734] MYIP 192.168.0.105
[14.720,161] IPv6 [fe80::e269:3aff:fe00:2c3e]-link
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_CONNECTED
Connecting...heap[229800] max contentlen[16384] sizeof
IoT_Publish_Message_Params (16)


 Root Done[0]Loading the client cert. and key. size TLSDataParams:2072


 Loading the client cert done.... ret[0]
 Client pkey loaded[0]
```



```
  . Connecting to a3t0o11ohwlo2h-ats.iot.us-east-
2.amazonaws.com/8883... ok
  . Setting up the SSL/TLS structure...verification is optional
  This certificate has no flags
  This certificate has no flags
  This certificate has no flags
SSL/TLS handshake. DONE ..ret:0
 ok
    [ Protocol is TLSv1.2 ]
    [ Ciphersuite is TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 ]
    [ Record expansion is 29 ]
. Verifying peer X.509 certificate...
Subscribed to topic [inno_test/ctrl] ret[0] qos[0] topic len[14]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":1}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":2}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":3}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":4}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":5}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":6}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":7}]
```



```
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":8}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":9}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy
- from: AWS IoT console
- to: T2
- message: Hello from AWS IoT console
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":10}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":11}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":12}]
- from: AWS IoT console
- to: T2
- message: Hello from AWS IoT console
Ho","msg_id":13}]
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":14}]
- from: AWS IoT console
- to: T2
- message: Hello from AWS IoT console
message status[0] topic[inno_test/data] msg[{"from":"Talaria
T2","to":"AWS","msg":"Howdy Ho","msg_id":15}]
```


10 Device Shadow

10.1 Running the sample application

1. In the AWS IoT Console, go to Manage -> Things ->`YourThingName` -> Shadow.
2. Program the Talaria TWO board with `shadow_sample.elf` available at: `sdk_2.3/apps/iot_aws/bin` using the process described in section 8.2.
3. On successful execution, the following console output will be provided:

```
UART:NWWWWWWAEBuild $Id: git-f92bee540 $
ssid=ACT102571068294    passphrase=43083191    aws_host=a3t0o11ohwlo2h-
ats.iot.us-east-2.amazonaws.com aws_port=8883 thing_name=innotest
Mounting file system
read_certs() success



WiFi Details  SSID: ACT102571068294, PASSWORD: 43083191


addr e0:69:3a:00:2c:3e
Connecting to WiFi...
add network status: 0
added network successfully, will try connecting..
connecting to network status: 0
[13.939,715] CONNECT:00:5f:67:cd:c5:a6 Channel:6 rssi:-31 dBm
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_LINK_UP
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_ADDRESS
[14.835,011] MYIP 192.168.0.105
[14.835,173] IPv6 [fe80::e269:3aff:fe00:2c3e]-link
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_CONNECTED
Shadow Connect
```



```
 Root Done[0]Loading the client cert. and key. size TLSDataParams:2072


 Loading the client cert done.... ret[0]
 Client pkey loaded[0]
  .        Connecting        to        a3t0o11ohwlo2h-ats.iot.us-east-
2.amazonaws.com/8883... ok
  . Setting up the SSL/TLS structure...  This certificate has no flags
  This certificate has no flags
  This certificate has no flags
SSL/TLS handshake. DONE ..ret:0
 ok
    [ Protocol is TLSv1.2 ]
    [ Ciphersuite is TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 ]
    [ Record expansion is 29 ]
. Verifying peer X.509 certificate...
 ok
Shadow Connected
init_and_connect_aws_iot. ret:0
Update                                                          Shadow:
{"state":{"reported":{"temperature":26,"windowOpen":true}},
"clientToken":"innotest-0"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":27,"windowOpen":true}},
"clientToken":"innotest-1"}
Update Accepted !!
```



```
Update                                                          Shadow:
{"state":{"reported":{"temperature":28,"windowOpen":true}},
"clientToken":"innotest-2"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":29,"windowOpen":true}},
"clientToken":"innotest-3"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":30,"windowOpen":true}},
"clientToken":"innotest-4"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":31,"windowOpen":true}},
"clientToken":"innotest-5"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":32,"windowOpen":true}},
"clientToken":"innotest-6"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":31,"windowOpen":true}},
"clientToken":"innotest-7"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":30,"windowOpen":true}},
"clientToken":"innotest-8"}
Update Accepted !!
```

```
Update                                                          Shadow:
{"state":{"reported":{"temperature":29,"windowOpen":true}},
"clientToken":"innotest-9"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":28,"windowOpen":true}},
"clientToken":"innotest-10"}
Update Accepted !!
Update                                                          Shadow:
{"state":{"reported":{"temperature":27,"windowOpen":true}},
"clientToken":"innotest-11"}
```


4. The AWS IoT dashboard will appear as shown in Figure 9.

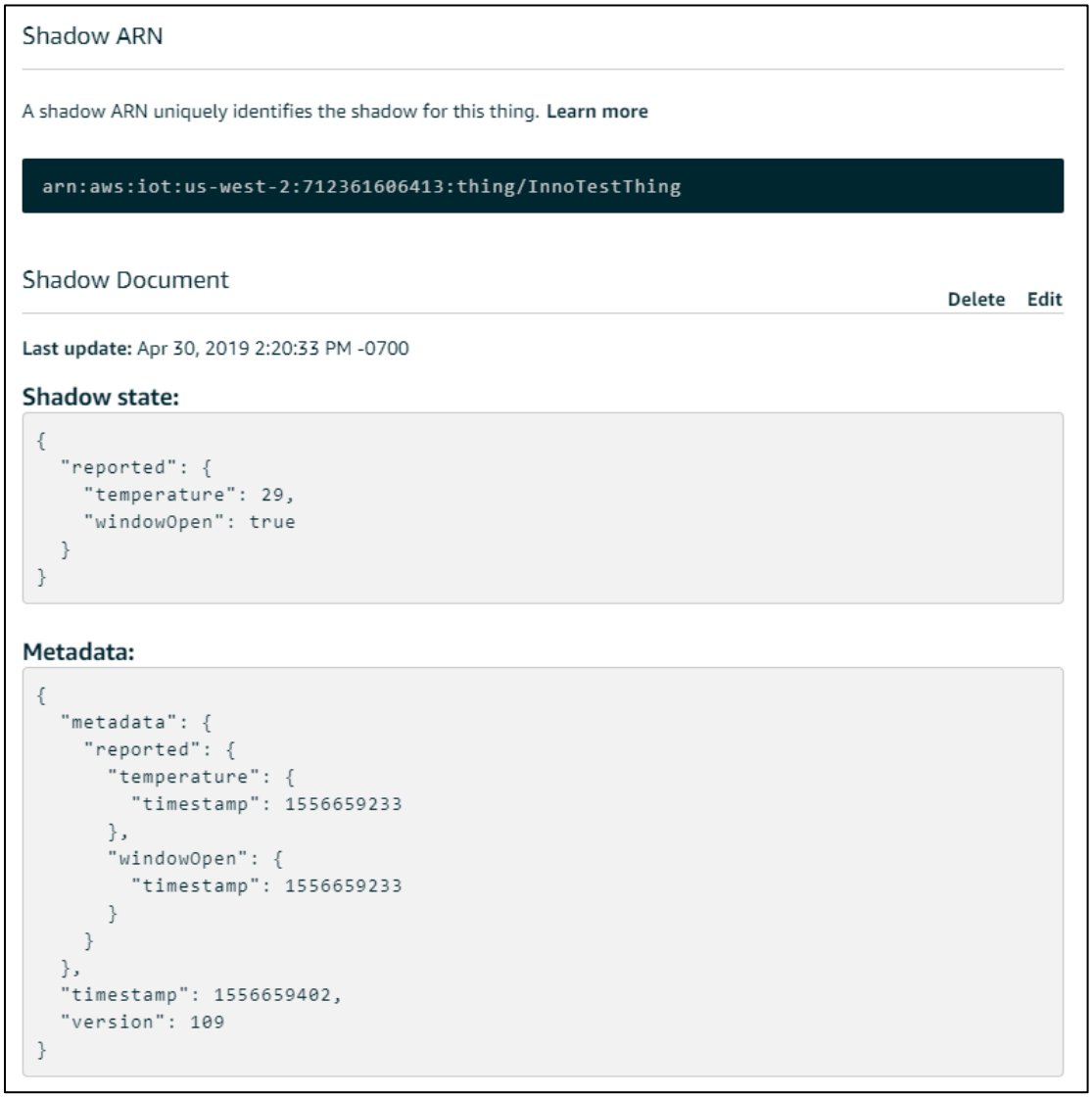

Figure 9: AWS IoT Dashboard

11 Running Jobs

11.1 Creating a job in AWS

1. Create a new .json file.

```
{
    "operation": "customJob",
    "otherInfo": "someValue"
}
```

2. Create a bucket to store files on your Amazon Simple Storage Service (Amazon S3). More information on creating buckets on the Amazon S3 can be found here: https://s3.console.aws.amazon.com.

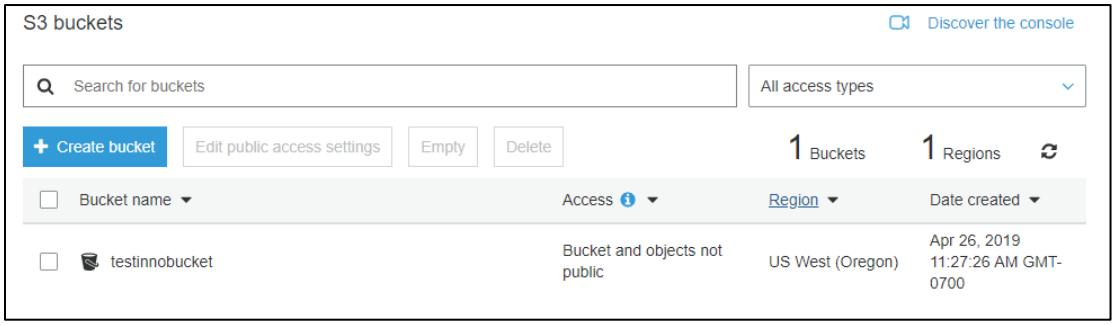

Figure 10: Creating a bucket to store files on Amazon S3

3. Upload the new .json file onto the Amazon S3 bucket.

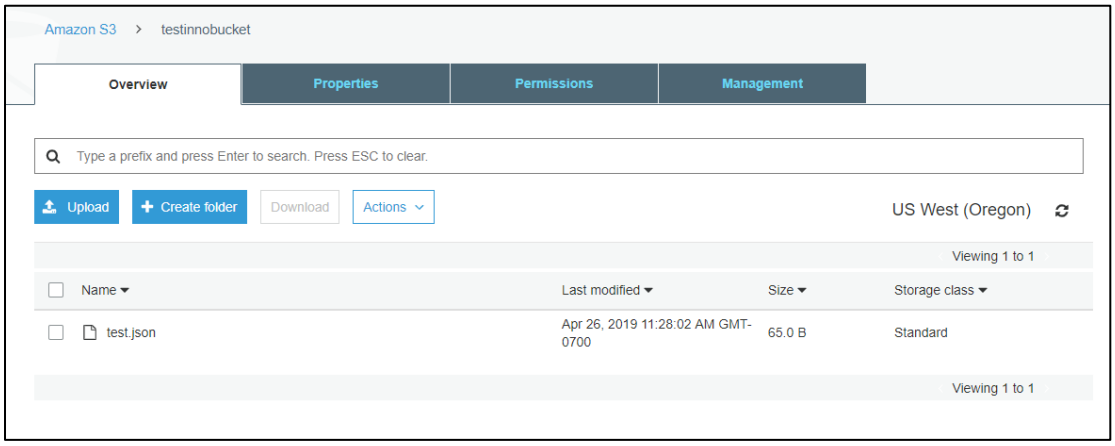

Figure 11: Uploading .json file onto the Amazon S3 bucket

4. In the AWS IoT Console, go to Manage -> Jobs.
5. Click on `Create` and then on `Create custom job`.

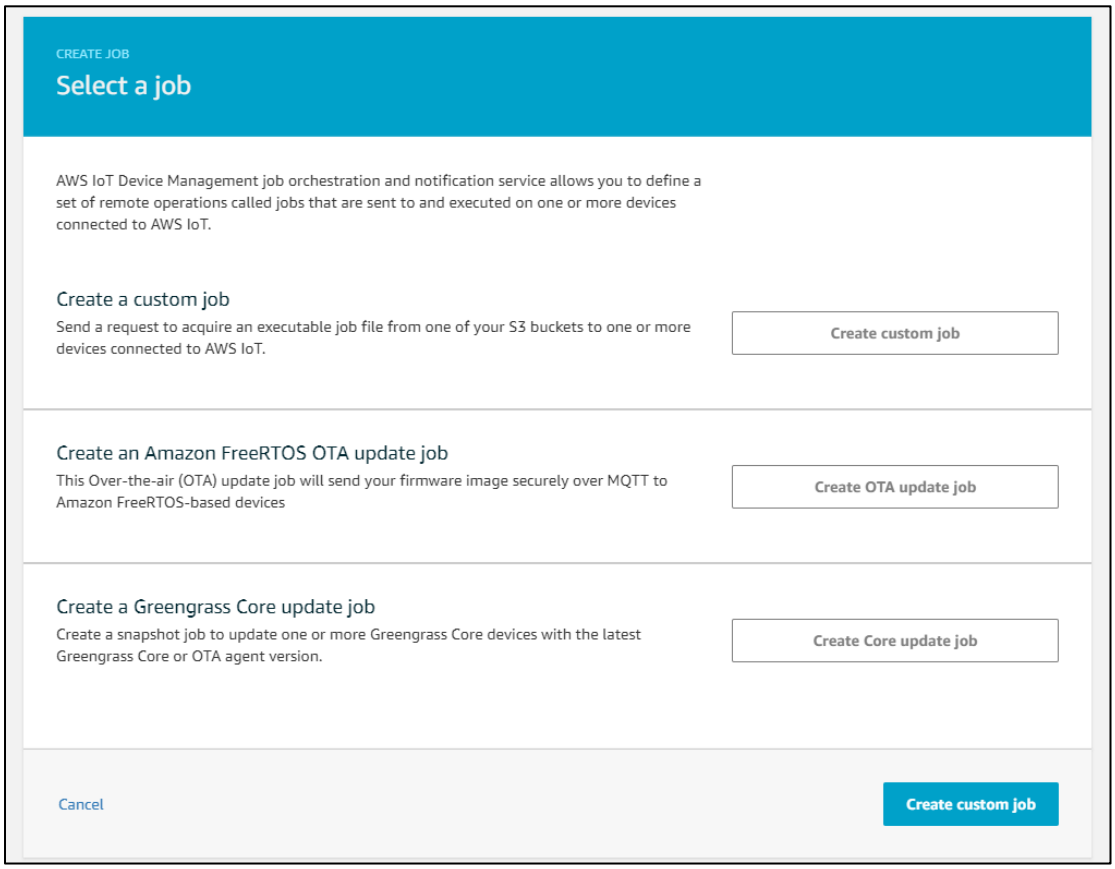

Figure 12: Creating a custom job

6. Fill the `Job ID` and `Description` as per your requirement.

7. In `Select devices to update`, select your thing as the device to be included in the job.

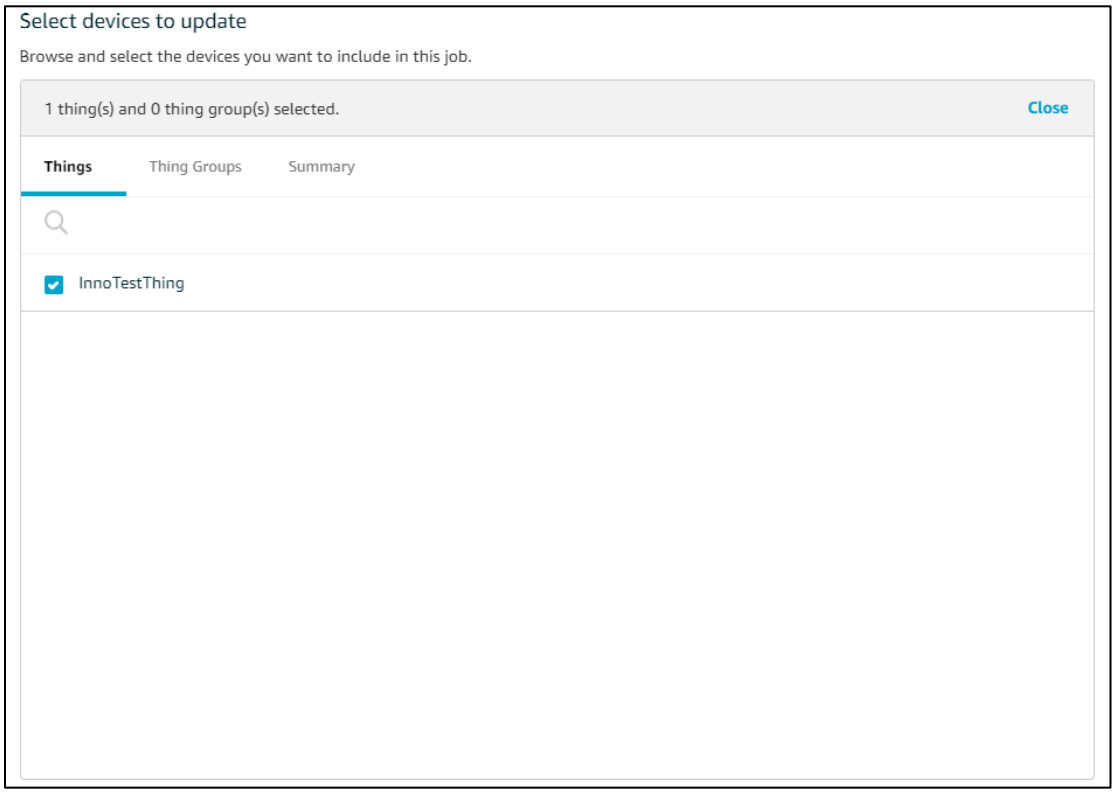

Figure 13: Selecting devices to update

8. In `Add a job file`, go ahead, and select the job file uploaded into your S3 bucket.

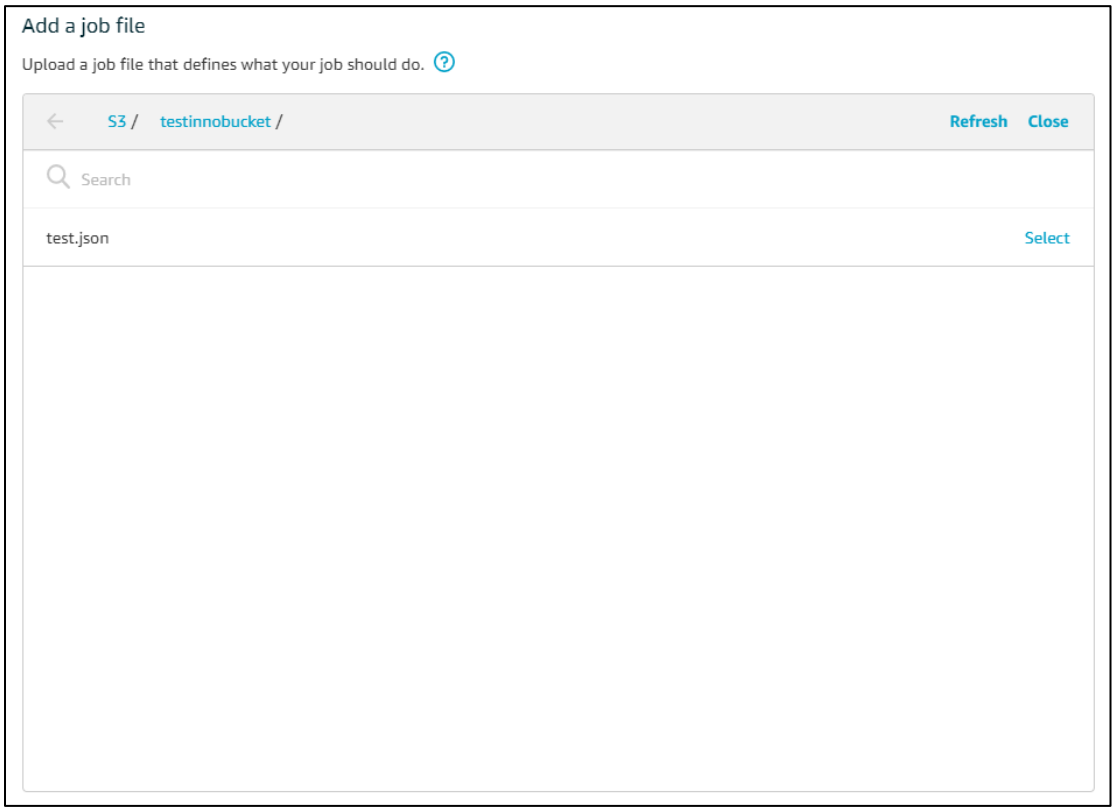

Figure 14: Adding a job file

9. Click on `Next`. In the next window, click on `Create`.
10. The new job you created will now appear on the AWS IoT Console.

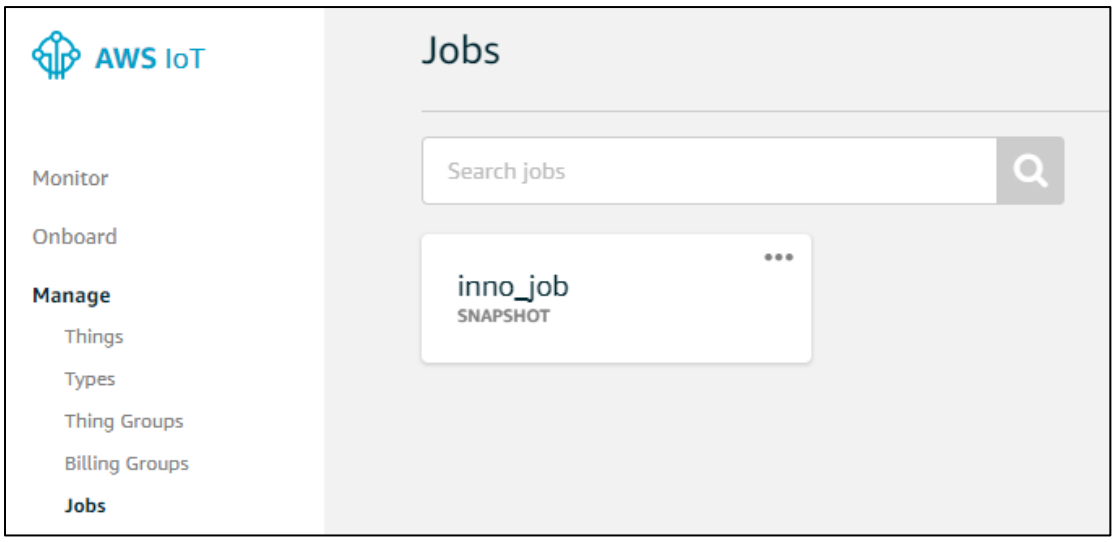

Figure 15: AWS IoT Console – new job created

11.2 Running the sample application

1. Program the Talaria TWO board with `jobs_sample.elf` available at: `sdk_2.3/apps/iot_aws/bin` using the process described in section 8.2.
2. On successful execution, the following console output will be provided:

```
UART:NWWWWWWAEBuild $Id: git-f92bee540 $
ssid=ACT102571068294    passphrase=43083191    aws_host=a3t0o11ohwlo2h-
ats.iot.us-east-2.amazonaws.com aws_port=8883 thing_name=innotest
Mounting file system
read_certs() success



WiFi Details  SSID: ACT102571068294, PASSWORD: 43083191


addr e0:69:3a:00:2c:3e
Connecting to WiFi...
add network status: 0
added network successfully, will try connecting..
connecting to network status: 0
[13.968,534] CONNECT:00:5f:67:cd:c5:a6 Channel:6 rssi:-27 dBm
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_LINK_UP
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_ADDRESS
[14.771,379] MYIP 192.168.0.105
[14.771,541] IPv6 [fe80::e269:3aff:fe00:2c3e]-link
wcm_notify_cb to App Layer - WCM_NOTIFY_MSG_CONNECTED
Connecting...heap[229040]       max       contentlen[16384]       sizeof
IoT_Publish_Message_Params (16)


 Root Done[0]Loading the client cert. and key. size TLSDataParams:2072
```



```
 Loading the client cert done.... ret[0]
 Client pkey loaded[0]
  .        Connecting        to        a3t0o11ohwlo2h-ats.iot.us-east-
2.amazonaws.com/8883... ok
  . Setting up the SSL/TLS structure...verification is optional
  This certificate has no flags
  This certificate has no flags
  This certificate has no flags
SSL/TLS handshake. DONE ..ret:0
 ok
    [ Protocol is TLSv1.2 ]
    [ Ciphersuite is TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 ]
    [ Record expansion is 29 ]
. Verifying peer X.509 certificate...
AWS Connection is done ret:0
Success subscribing JOB_GET_PENDING_TOPIC: 0
Success subscribing JOB_NOTIFY_NEXT_TOPIC: 0
Success subscribing JOB_DESCRIBE_TOPIC ($next): 0
Success subscribing JOB_UPDATE_TOPIC/accepted: 0
Success subscribing JOB_UPDATE_TOPIC/rejected: 0
aws_iot_jobs_send_query: 0
Success aws_iot_jobs_describe: 0
JOB_GET_PENDING_TOPIC callback
topic: $aws/things/innotest/jobs/get/accepted
payload: {"timestamp":1628590744,"inProgressJobs":[],"queuedJobs":[]}
inProgressJobs: []
queuedJobs: []
```



```
JOB_NOTIFY_NEXT_TOPIC / JOB_DESCRIBE_TOPIC($next) callback

topic: $aws/things/innotest/jobs/$next/get/accepted

payload: {"timestamp":1628590744}

execution property not found, nothing to do

aws_iot_mqtt_yield: 0

aws_iot_mqtt_yield: 0

aws_iot_mqtt_yield: 0
```

3. The AWS IoT Console will display as completed once the job is completed.

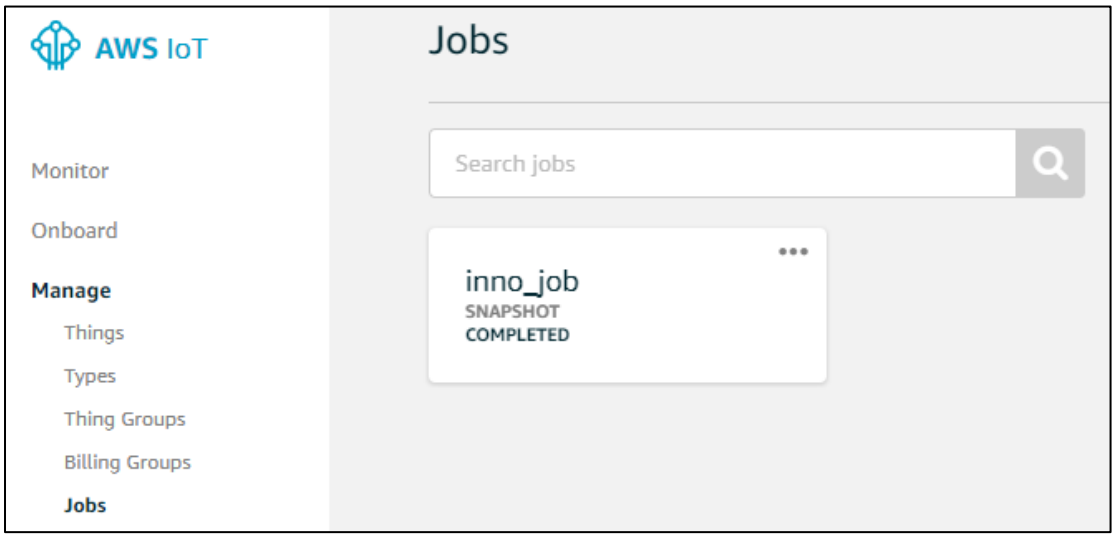

Figure 16: AWS IoT Console – Job Completed

4. You can continue creating new jobs which will be executed by your device/thing.

12 Support

1. Sales Support: Contact an InnoPhase sales representative via email – sales@innophaseinc.com
2. Technical Support:
    a. Visit: https://innophaseinc.com/contact/
    b. Also Visit: https://innophaseinc.com/talaria-two-modules
    c. Contact: support@innophaseinc.com

InnoPhase is working diligently to provide outstanding support to all customers.

13 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, InnoPhase Incorporated does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and assumes no liability associated with the use of such information. InnoPhase Incorporated takes no responsibility for the content in this document if provided by an information source outside of InnoPhase Incorporated.

InnoPhase Incorporated disclaims liability for any indirect, incidental, punitive, special or consequential damages associated with the use of this document, applications and any products associated with information in this document, whether or not such damages are based on tort (including negligence), warranty, including warranty of merchantability, warranty of fitness for a particular purpose, breach of contract or any other legal theory. Further, InnoPhase Incorporated accepts no liability and makes no warranty, express or implied, for any assistance given with respect to any applications described herein or customer product design, or the application or use by any customer's third-party customer(s).

Notwithstanding any damages that a customer might incur for any reason whatsoever, InnoPhase Incorporated' aggregate and cumulative liability for the products described herein shall be limited in accordance with the Terms and Conditions of identified in the commercial sale documentation for such InnoPhase Incorporated products.

Right to make changes — InnoPhase Incorporated reserves the right to make changes to information published in this document, including, without limitation, changes to any specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — InnoPhase Incorporated products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an InnoPhase Incorporated product can reasonably be expected to result in personal injury, death or severe property or environmental damage. InnoPhase Incorporated and its suppliers accept no liability for inclusion and/or use of InnoPhase Incorporated products in such equipment or applications and such inclusion and/or use is at the customer's own risk.