Overview
========
This demo demonstrates the follow function:

1. There are two parts: Android APK, AWS Wi-Fi provisioning demo (running on RT1060).
2. With Android APK running on the smart phone, the end user could view and config the Wi-Fi APs via BLE and view the Wi-Fi setting log via AWS.
3. By default, AWS Wi-Fi provisioning demo will start advertising if the Wi-Fi AP is not configured and wait the Wi-Fi AP configuration. After connected to the Android APK, the demo will execute the request from cellphone and reply the response. When the Wi-Fi Ap is configured, the Shadow demo will connect to the AWS via Wi-Fi and publish the configured Wi-Fi AP information.

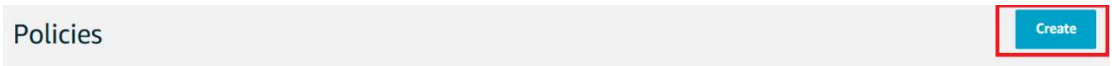
Note: This demo could NOT function with the default setting provided in SDK package because an AWS account is mandatory to run to the demo, the end users must create their owner AWS account and configure the IoT Console before the functionality of the demo could be used. Also, some information specified by the end customers, like Thing name, broker endpoint, etc., must be updated accordingly before the demo would work. Check "Prepare the Demo" to get the detailed guidance of the configuration steps.

Board settings
==============
Refer to Hardware Rework Guide for MIMXRT1060-EVK and AW-AM457-uSD

Prepare the Demo
================
Before running the demo, some steps should be followed to configure AWS IoT Console:

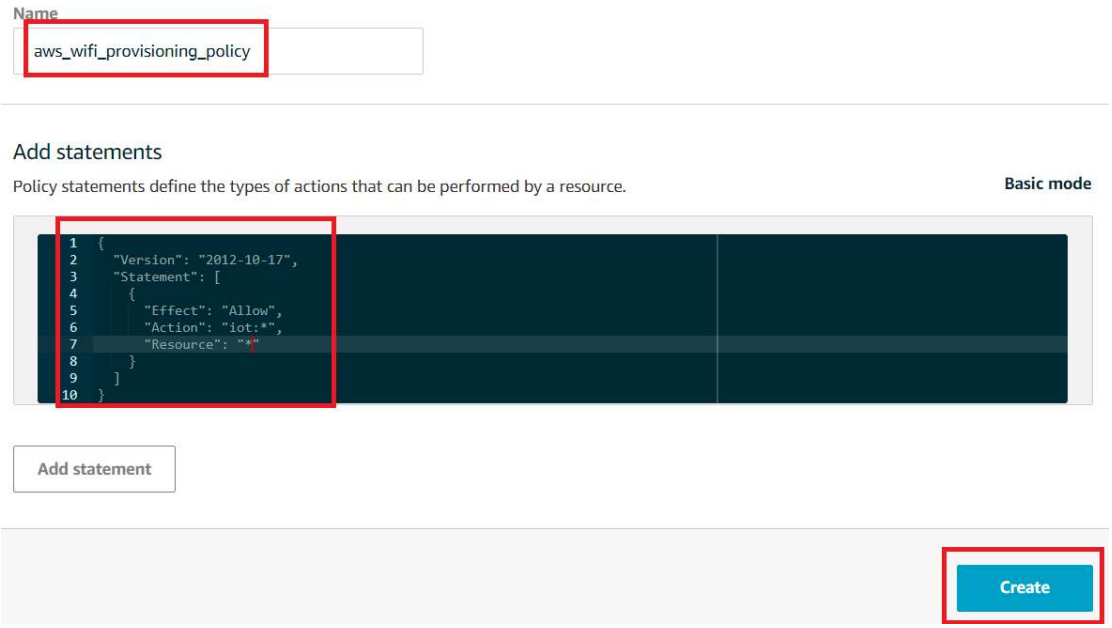
1. Create AWS account: https://console.aws.amazon.com/console/home
2. Configure device in the AWS IoT Console as follow:
   (1) Create one policy as the follow steps (for example: the policy name is "aws_wifi_provisioning_policy").
      a) Click "Policies"


      b) Click "Create"



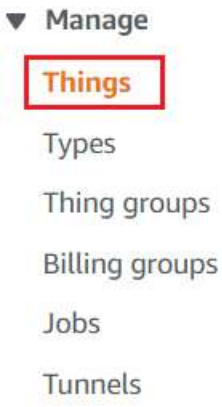

      c) File the policy name and switch to "Advanced mode" fill the follow content.

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

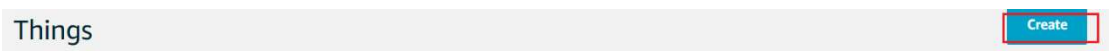



(2) Create one "Thing" as follow (for example: the name is "aws_wifi_provisioning"):

a) Click "Things"

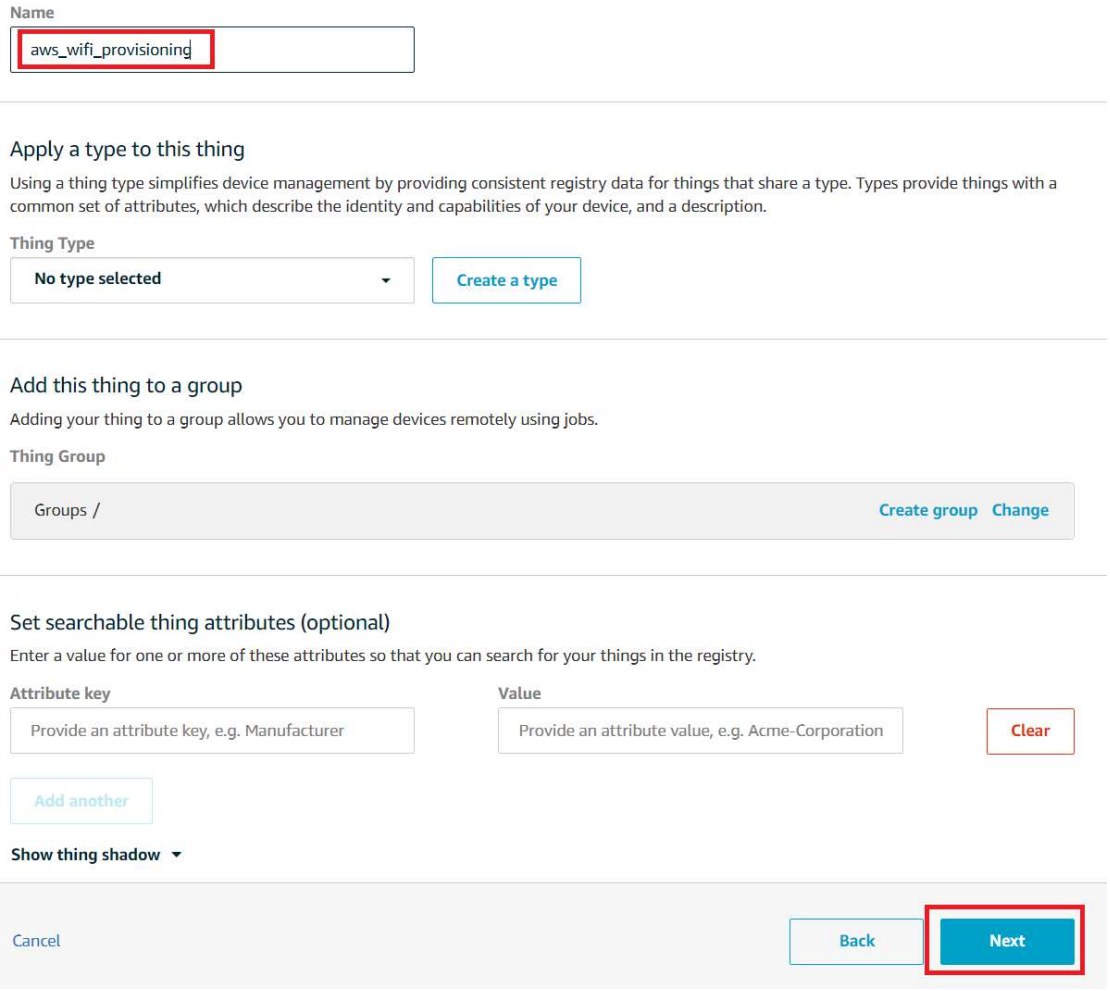



b) Click "Create"

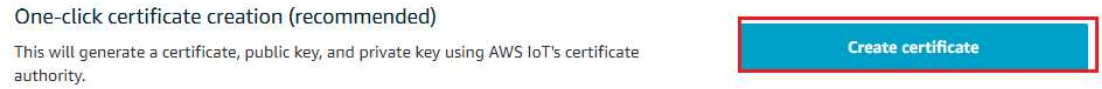



c) Click "Create a single thing"

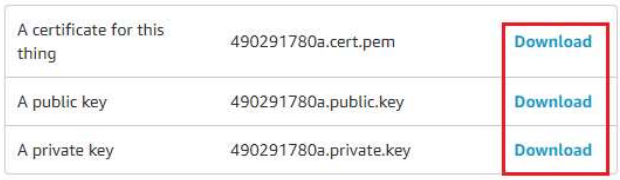

d) Fill the name and click “Next”

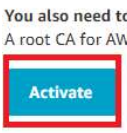



e) Click “Create certificate”




f) Download keys, click “Active” and then click “Attach a policy”





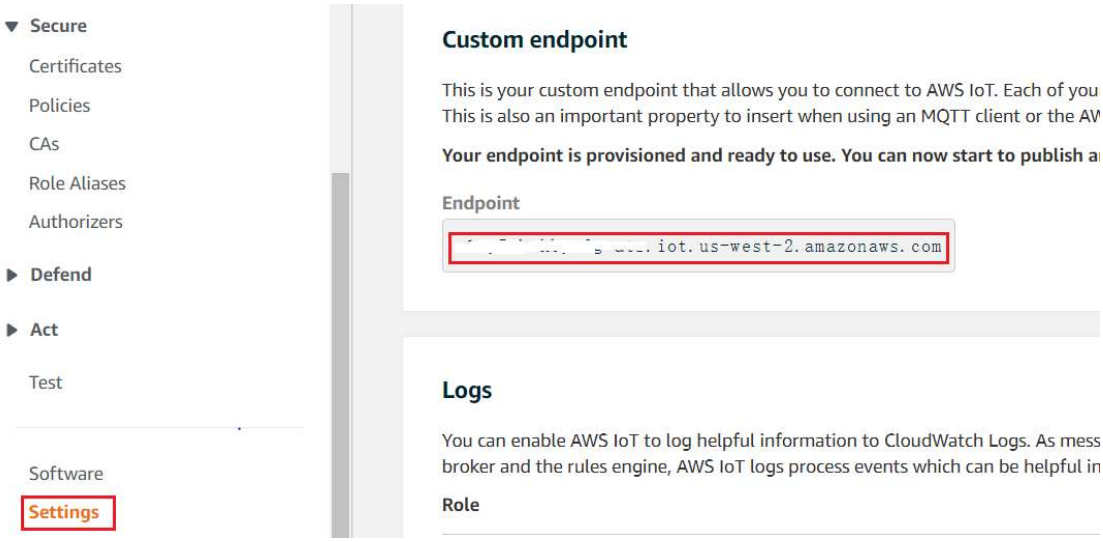
g) Select the policy and click "Register Thing"



3. Configure the demo's codes.

(1) Open aws_clientcredential.h, Fill `clientcredentialMQTT_BROKER_ENDPOINT`, `clientcredentialIOT_THING_NAME` to configure the AWS thing.

a) The `clientcredentialMQTT_BROKER_ENDPOINT` can be got as follow.



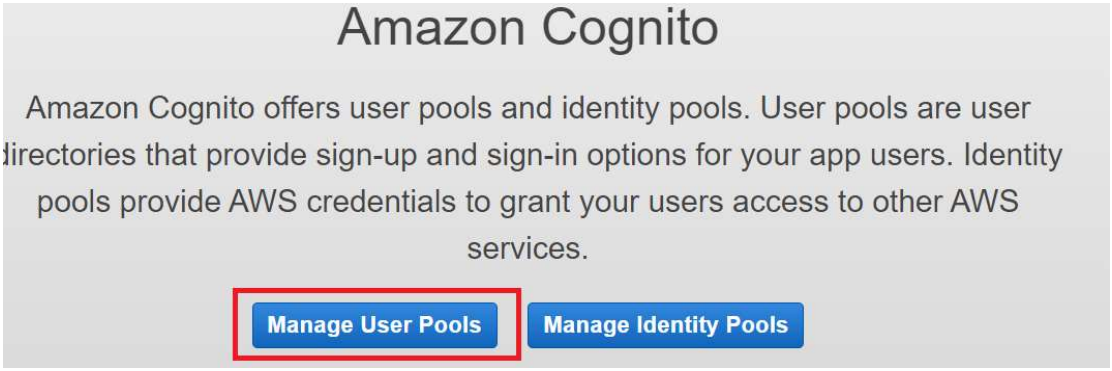

b) The `clientcredentialIOT_THING_NAME` is the Thing Name that is created in above steps.

```
For example:
#define  clientcredentialMQTT_BROKER_ENDPOINT                "xxxxx-
ats.iot.us-west-2.amazonaws.com"
#define  clientcredentialIOT_THING_NAME                       "
aws_wifi_provisioning"
```

(2) Update aws_clientcredential_keys.h, you can use the CertificateConfigurator.html

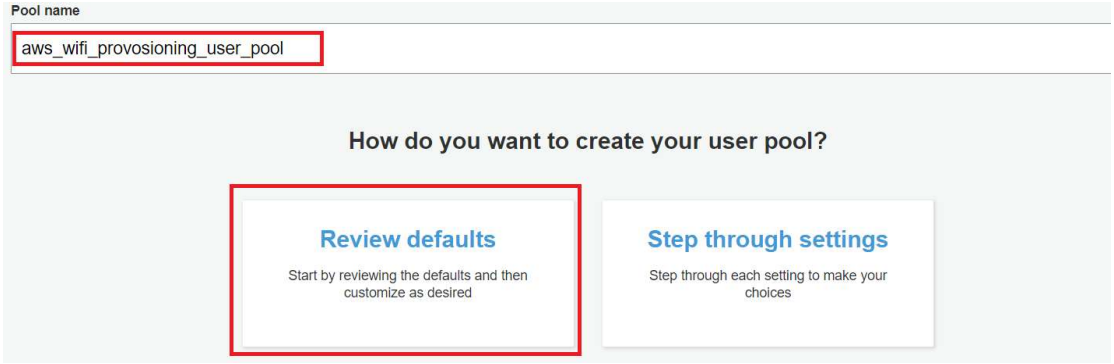
(freertos\tools\certificate_configuration) to generate the "aws_clientcredential_keys.h" and replace this file. It use the key files that are download in the previous step.

4. Install the aws_wifi_provosioning.apk in the android phone.
5. The Android application requires Cognito service to authorize to AWS IoT in order to access device shadows. Use Amazon Cognito to create a new user pool and identity pool.
   a) Open the Amazon Cognito console, click “Manage User Pools”

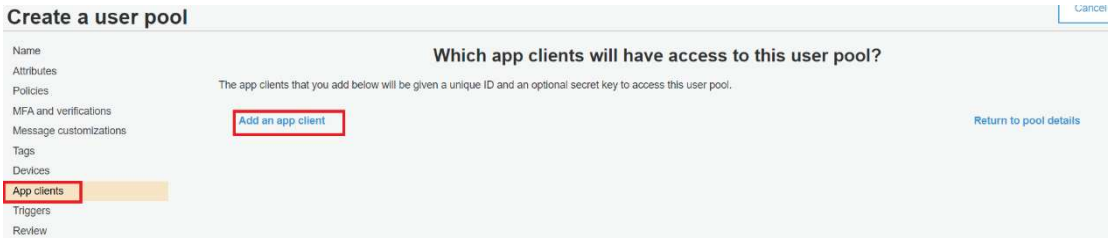



   b) Click “Create a user pool”

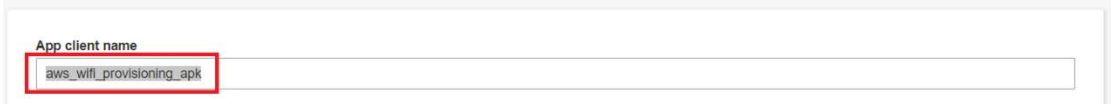

   c) Fill the Identity poll name (for example: aws_wifi_provosioning_user_pool), and select “Review defaults”.

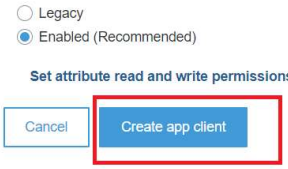



   d) Select “App clients” from the navigation pane, and then choose “Add an app client”

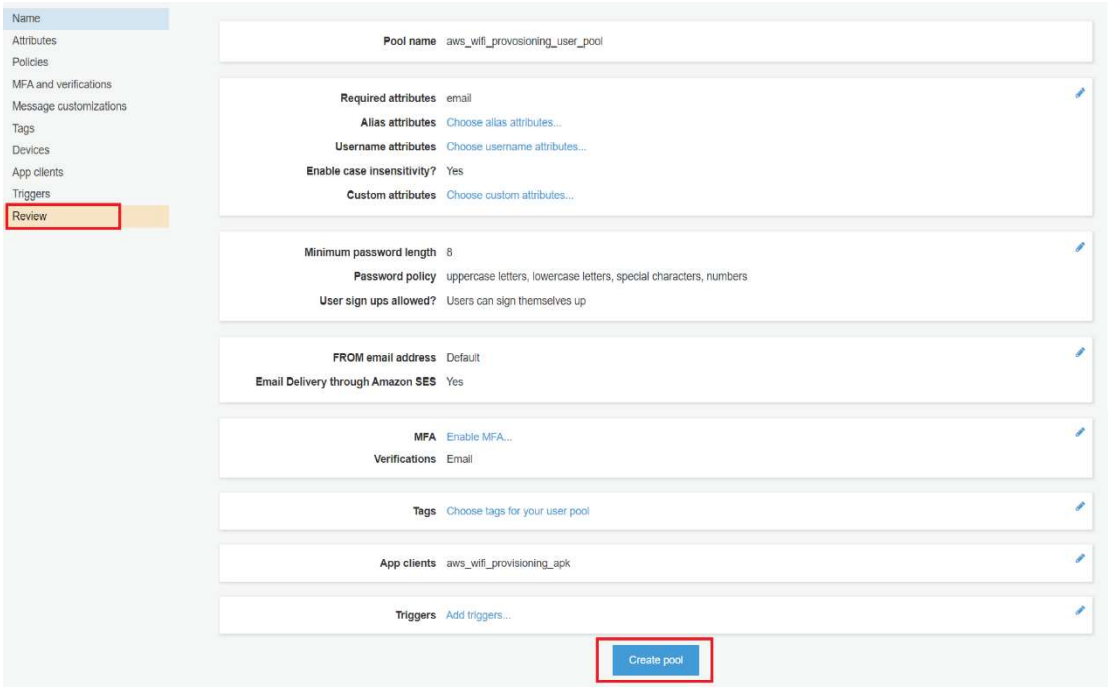



   e) Enter a name for the app client, like as “aws_wifi_provisioning_apk”, and then choose “Create app client”

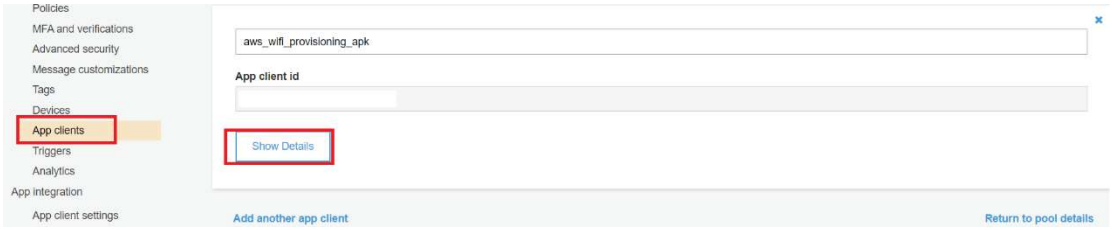





f) From the navigation pane, choose “Review”, and then choose “Create pool”.



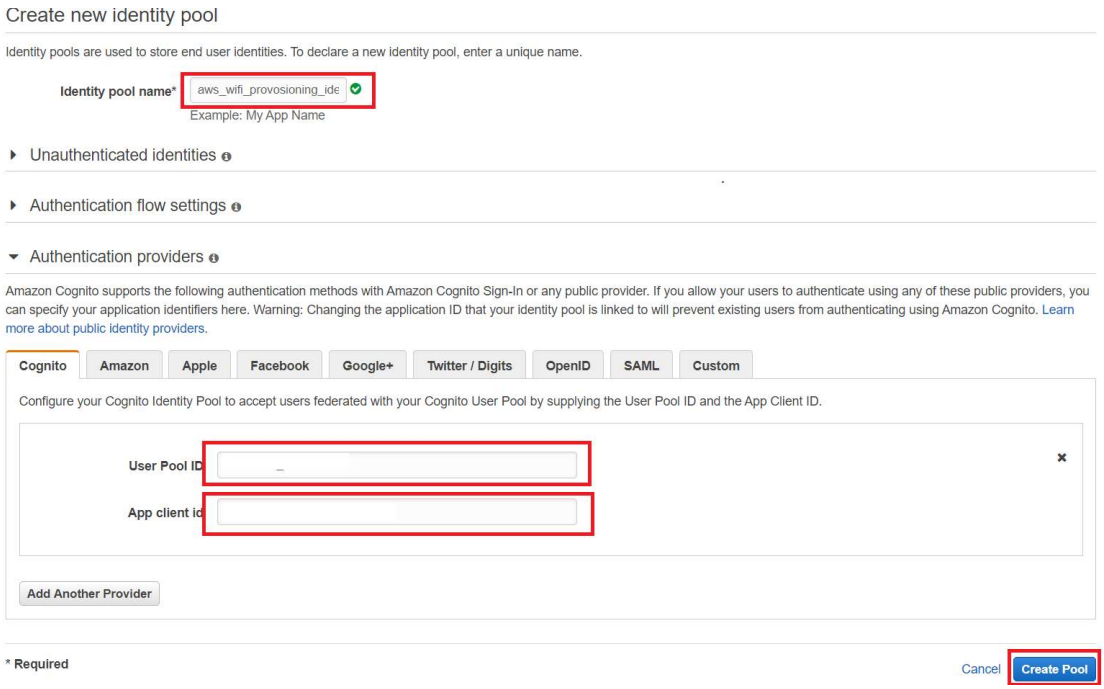

g) From the navigation pane, choose “App clients”, and then choose “Show details”. Make a note of the app client ID and app client secret.




h) Open the Amazon Cognito console, click “Manage Identity Pools”


i) Click “Create new identity pool”


j) Fill the Identity poll name (for example: aws_wifi_provosioning_identity_pool). Expand “Authentication providers”, choose the “Cognito” tab, and then enter your user pool ID and app client ID. Then Choose “Create Pool”

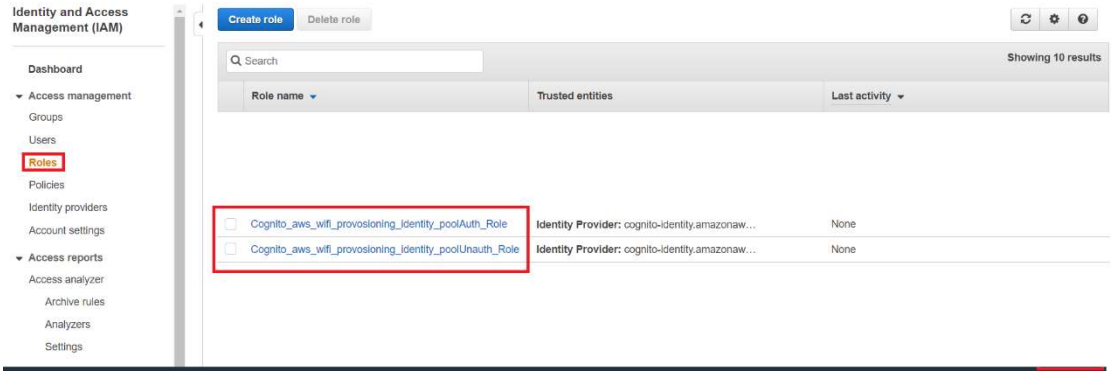



k) Expand “View Details” and make a note of the two IAM role names. Choose “Allow” to create the IAM roles for authenticated and unauthenticated identities to access Amazon Cognito.


l) Copy the Identity pool ID. Make a note of the identity pool ID. It should be of the form us-west-2:12345678-1234-1234-1234-123456789012.

```
// Initialize the Amazon Cognito credentials provider
CognitoCachingCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
    getApplicationContext(),
    "us-west-2:[redacted]", // Identity pool ID
    Regions.US_WEST_2 // Region
);
```

6. Create and attach an IAM policy to the authenticated identity
   a) Open the IAM console, and from the navigation pane, choose “Roles”.

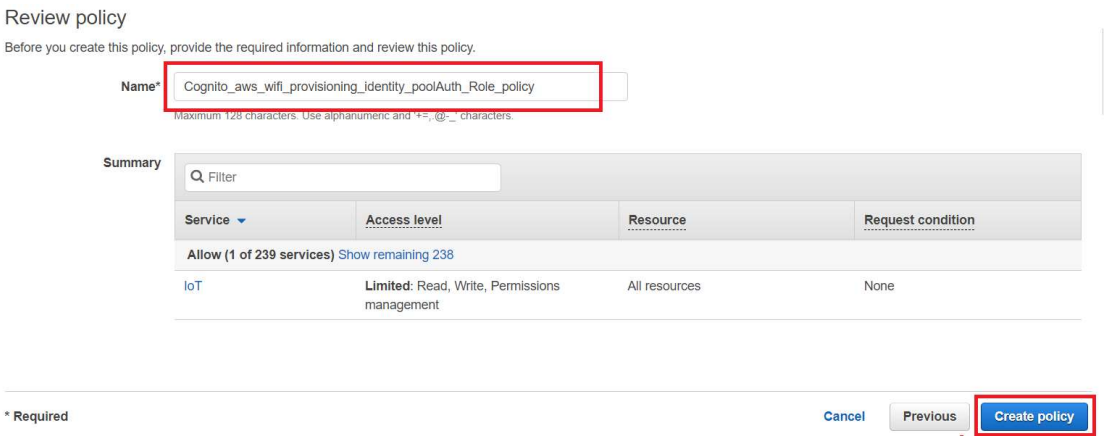



b) Find and choose your authenticated identity's role, choose “Add inline policy”.


c) Choose the “JSON” tab and paste the following JSON. And then choose “Review policy”.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":[
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow"
            ],
            "Resource":[
                "*"
            ]
        }
    ]
}
```


d) Enter a name for the policy, and then choose “Create policy”. This step should be applied for both Roles.




After the above steps. The pool is created successfully.

Prepare one configuration file for the android app.

Prepare "Preferences.properties" file with yours AWS credentials. It's structure looks like this:

customer_specific_endpoint=<REST API ENDPOINT>
cognito_pool_id=<COGNITO POOL ID>
thing_name=<THING NAME>
region=<REGION>
policy_name=<POLICY>

a) customer_specific_endpoint is the endpoint that is configured in aws_clientcredential.h
b) cognito_pool_id is the copied pool id in above step.
c) thing_name is the created Thing name.
d) region is the front part of the endpoint.
e) policy_name is the created Thing policy

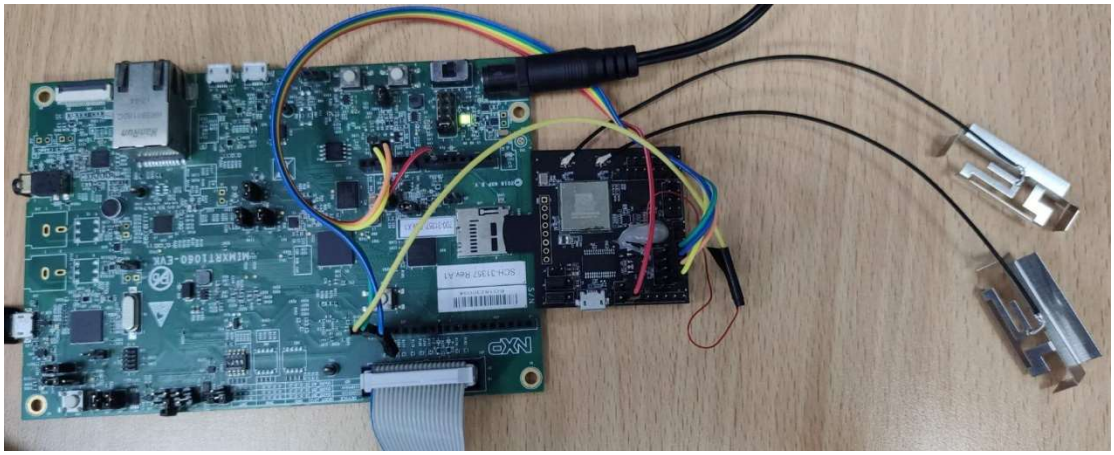
for example:

customer_specific_endpoint=xxxxxx-ats.iot.us-west-2.amazonaws.com
cognito_pool_id=us-west-2:5xxxx7-3xxxx9-4xxx5-axxf-0xxxxxxxxxb
thing_name= aws_wifi_provosioning
region=us-west-2
policy_name=aws_wifi_provisioning_policy

Running the demo

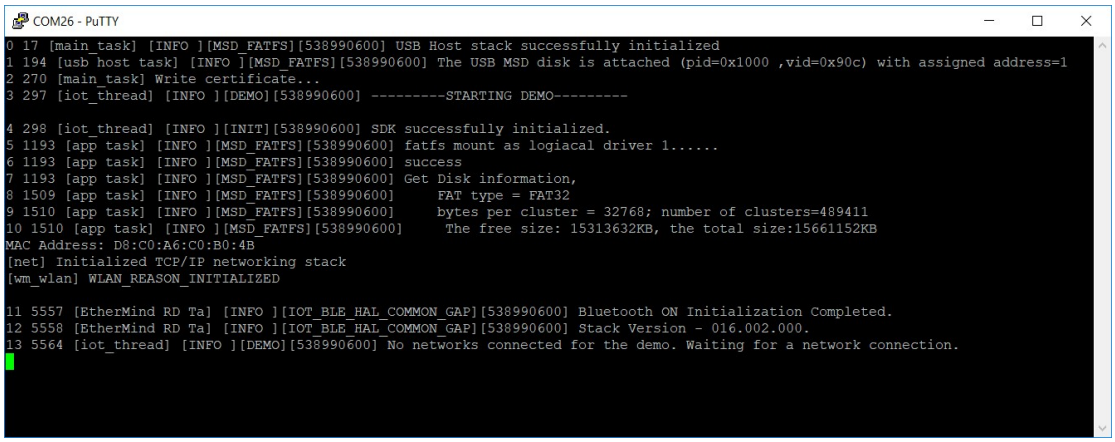
==========

1. Prepare the board as follow picture.



2. Run the demo with rt1060.

   (1) After running the log is as follow:




   (2) Open the android app and load the preferences.properties as the video. And then click “WIFI PROVISIONING” to start aws wifi provisioning test.