# **Appendix I: Firmware Code**

#include "HX711.h"

#include "Queue.h"

// HX711 circuit wiring//Libraries

#include <SPI.h>

#include <WiFiNINA.h>

#include <HX711.h>

#define DT 2

#define SCK 3

#define DT1 4

#define SCK1 5

float childsenscalibration\_factor = -7050;

float driversenscalibration\_factor = 7050;

HX711 childseatsens;

HX711 driverseatsens;

char networkSsid[] = "My Home";

char networkPass[] = "trishok4";

//Time allowance for driver to return to vehicle or remove child before an alert is sent

static int checkTime = 30000;

int status = WL\_IDLE\_STATUS;

IPAddress server(192,168,1,3); // Google

//Sensor check variables

struct sysStatus{

bool driver = false;

bool child = false;

};

// Initialize the client library

WiFiClient client;

void setup() {

// initialize LED

pinMode(LED\_BUILTIN, OUTPUT);

Serial.begin(9600);

while (!Serial)

Serial.println("Starting Program.");

//Connect to the server using WiFi

connectWifi();

childseatsens.begin(DT, SCK);

driverseatsens.begin(DT1, SCK1);

childseatsens.set\_scale();

childseatsens.tare(); //Reset the scale to 0

childseatsens.set\_scale(childsenscalibration\_factor);

driverseatsens.set\_scale();

driverseatsens.tare(); //Reset the scale to 0

driverseatsens.set\_scale(driversenscalibration\_factor);

}

void connectWifi(){

//Connect to the network and server

while (status != WL\_CONNECTED) {

Serial.print("Attempting connection to network: ");

Serial.println(networkSsid);

status = WiFi.begin(networkSsid, networkPass);

// Delay to establish connection

delay(10000);

IPAddress ardAddr = WiFi.localIP();

Serial.println(ardAddr);

//if connected, blink the LED

digitalWrite(LED\_BUILTIN, HIGH);

delay(1000);

digitalWrite(LED\_BUILTIN, LOW);

delay(1000);

}

}

void dispWifiStatus() {

// display network information

Serial.print("Network: ");

Serial.println(WiFi.SSID());

// display Arduino's IP

IPAddress ipAddr = WiFi.localIP();

Serial.print("IP Address: ");

Serial.println(ipAddr);

}

void checkWifi(){

if(!client.connected()){

Serial.println("Connection Lost. Trying to Reestablish Connection.");

connectWifi();

}

}

//Check for the presence of the driver if driver is not there for more than 1 min then send status as false

//Inputs: None

//Outputs:

// bool driver:

// true: Driver is present in vehicle

// false: Driver is not in vehicle

bool checkDriver(){

//int driverseatavg=0;

int driverseatval=255;

int count=0; /\* driver not present count 1ms\*60=60 sec \*/

if(driverseatsens.is\_ready()){

Serial.print("Driver Scale Reading: ");

driverseatval=(int)driverseatsens.get\_units()\*0.453592;

//driverseatsensqueue.dequeue();

if(driverseatval == 0)

{

count += 1;

}

else

{

count=0; /\*reset timer \*/

}

Serial.print(driverseatsens.get\_units()\*0.453592, 3);

Serial.print(" kg"); //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person

Serial.println( );

if(count == 60)

{

return false;

}

}

else{

Serial.println("Driver Scale Not reading .");

}

return true;

}

//Check for the presence of the child

//Inputs: None

//Outputs:

// bool child:

// true: Child is present in vehicle

// false: Child is not in vehicle

bool checkChild(){

// int childseatavg=0;

int childseatval=255;

int count=0; /\* driver not present count 1ms\*60=60 sec \*/

if(childseatsens.is\_ready()){

Serial.print("Child Scale Reading: ");

childseatval=(int)childseatsens.get\_units()\*0.453592;

//childseatsensqueue.dequeue();

//childseatsensqueue.enqueue(childseatval);

Serial.print(childseatsens.get\_units()\*0.453592, 3);

Serial.print("kg"); //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person

Serial.println( );

if(childseatval == 0)

{

count += 1;

}

else

{

count=0; /\*reset timer \*/

}

if(count == 60)

{

return false;

}

}

else{

Serial.println("Child scale Not reading .");

}

return true;

}

// Function to check the sensor.

//Inputs:

// delayTime - controls how long to check for driver presence before sending an alert

//Outputs:

// bool alert - returns the status of the alert check.

// 1 = no alert (driver present or child is not)

// 2= alert because child is alone and driver has not reentered vehicle over

int checkSensors(int delayTime){

int alert = 1;

//Create the structure to keep track of sensor status and initialize to false

sysStatus sensStatus;

sensStatus.driver = false;

sensStatus.child = false;

bool check = false;

//Loop to check the status of the system

while(check == false){

//if there's no child, immediately stop checking since there's no reason to proceed

if(checkChild() == false){

check == true;

alert == 1;

}

//otherwise check for the driver

else if(checkDriver() == true){

check == true;

alert == 1;

}

//otherwise, child is alone in the vehicle. Use delayTime to delay for a set amount of

//time to give driver time to get back in vehicle or remove child

else{

delay(delayTime);

if(checkChild() == false){

check == true;

alert == 1;

}

//otherwise check for the driver

else if(checkDriver() == true){

check == true;

alert == 1;

}

else{

check == false;

alert == 2;

}

}

}

return alert;

}

//Send an alert that the child is alone

//Inputs: None

//Outputs: None

void sendAlert(void){

Serial.println("Sending alert.");

if (!client.connect(server, 80)) {

Serial.println("Connection to host failed");

return;

}

Serial.println("Connected to server successful!");

client.print("Alert");

client.stop();

}

void loop() {

//Check to make sure the connection still exists

//checkWifi();

// Make a HTTP request:

int alertval=0;

alertval=checkSensors(checkTime);

//sendAlert(alertval);

if(alertval == 2)

{

sendAlert();

}

delay(1000);

}

# **Appendix II: Backend Software Code**

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 11 13:39:12 2019

@author: Trishok

"""

from \_\_future\_\_ import print\_function

import sys

import ssl

import time

import os

import logging, traceback

import paho.mqtt.client as mqtt

import socket

fileDir = os.path.dirname(os.path.realpath('\_\_file\_\_'))

IoT\_protocol\_name = "x-amzn-mqtt-ca"

aws\_iot\_endpoint = "a2gw161u0ey957-ats.iot.us-west-2.amazonaws.com" # <random>.iot.<region>.amazonaws.com

url = "https://{}".format(aws\_iot\_endpoint)

ca = os.path.join(fileDir, 'VeriSign-Class-3-Public-Primary-Certification-Authority-G5.pem')

cert = os.path.join(fileDir, 'a407f17540-certificate.pem')

private = os.path.join(fileDir, 'a407f17540-private.pem')

logger = logging.getLogger()

logger.setLevel(logging.DEBUG)

handler = logging.StreamHandler(sys.stdout)

log\_format = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

handler.setFormatter(log\_format)

logger.addHandler(handler)

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.bind(('192.168.1.3', 80))

s.listen(0)

def ssl\_alpn():

try:

#debug print openssl version

logger.info("open ssl version:{}".format(ssl.OPENSSL\_VERSION))

ssl\_context = ssl.create\_default\_context()

ssl\_context.set\_alpn\_protocols([IoT\_protocol\_name])

ssl\_context.load\_verify\_locations(cafile=ca)

ssl\_context.load\_cert\_chain(certfile=cert, keyfile=private)

return ssl\_context

except Exception as e:

print("exception ssl\_alpn()")

raise e

if \_\_name\_\_ == '\_\_main\_\_':

topic = "ECE574\_Project"

try:

mqttc = mqtt.Client()

ssl\_context= ssl\_alpn()

mqttc.tls\_set\_context(context=ssl\_context)

logger.info("start connect")

mqttc.connect(aws\_iot\_endpoint, port=443)

logger.info("connect success")

mqttc.loop\_start()

message = "Hey you left you Child in the Car please Hurry"

msgflag = False

while True:

client, addr = s.accept()

print("Got a connection from %s" % str(addr))

if client:

content = client.recv(32)

alertval=int(content)

if alertval == 2:

print("Child alert:")

mqttc.publish(topic, message)

time.sleep(1)

except Exception as e:

logger.error("exception main()")

logger.error("e obj:{}".format(vars(e)))

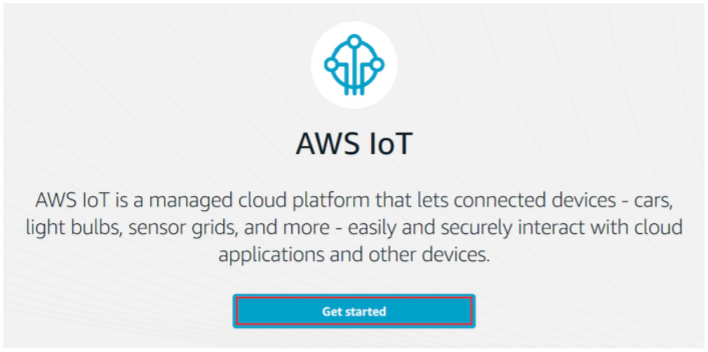
logger.error("message:{}".format(e.message))

traceback.print\_exc(file=sys.stdout)

# **Amazon AWS IoT Full User Manual**

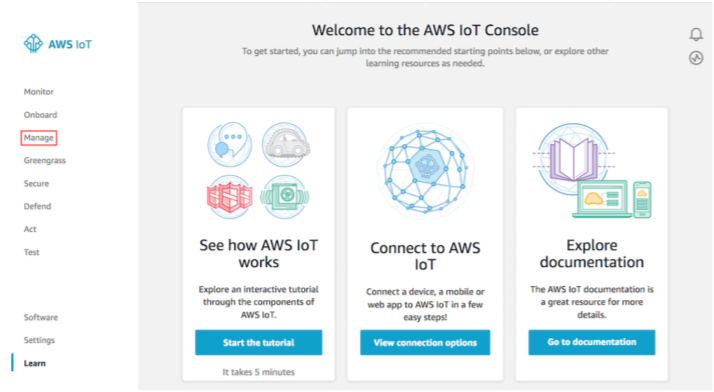
1. Sign into the AWS IoT Console

* Open the [AWS home page](https://aws.amazon.com/) and choose Sign into the Console.
* Follow the online instructions. Part of the sign-up procedure involves providing a valid Email address and password.
* Sign into the AWS Management Console and open the AWS IoT console.
* On the Welcome page, choose Get started.

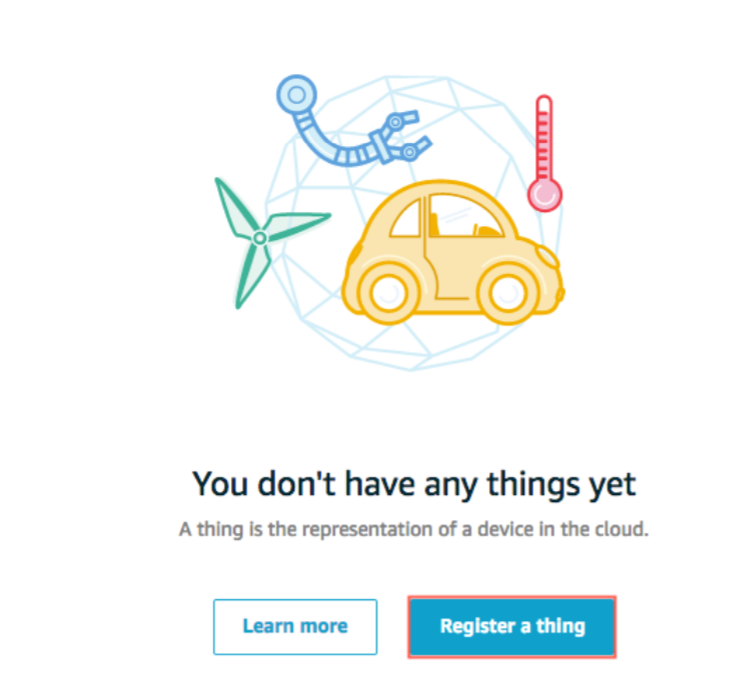


1. Register a Device(Thing) and Policy

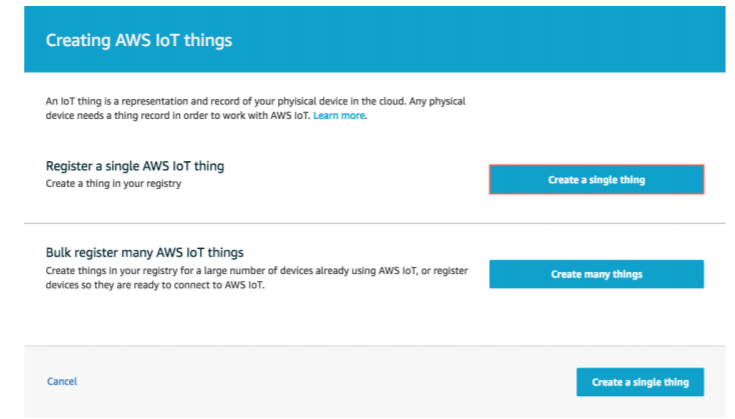
* On the Welcome to the AWS IoT Console page, in the navigation pane, choose Manage.



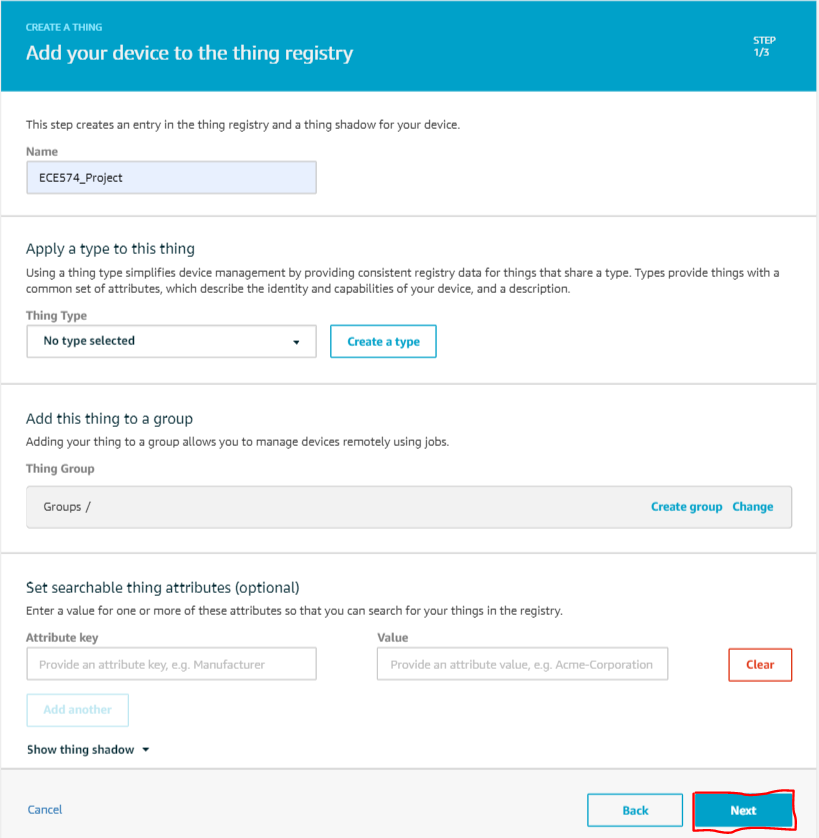
* On the You don't have any things yet page, choose Register a thing.



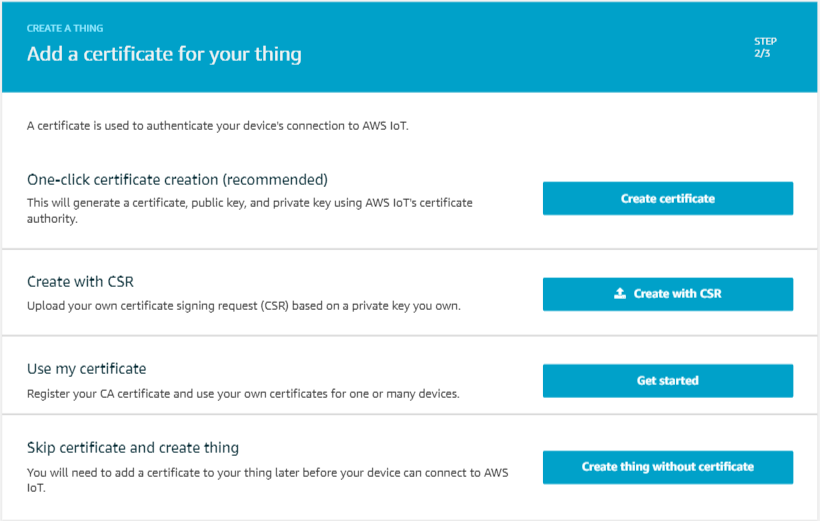
* On the Creating AWS IoT things page, choose Create a single thing.



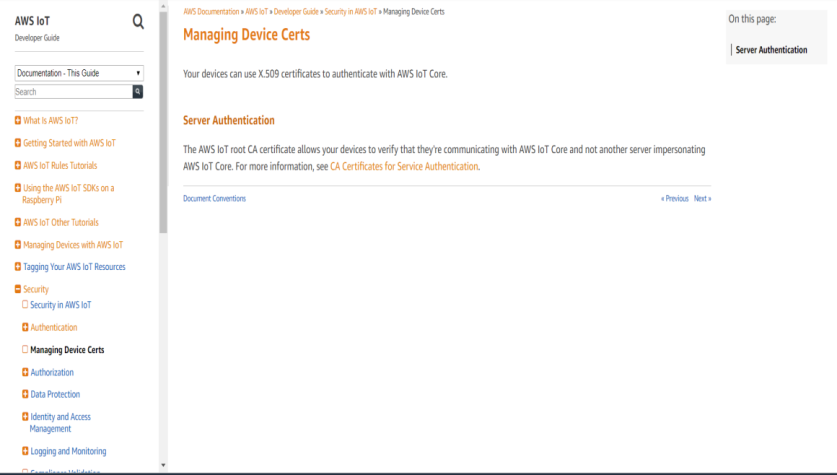
* On the Create a thing page, in the Name field, enter a name for your thing, such as ECE574\_Project. Choose Next.



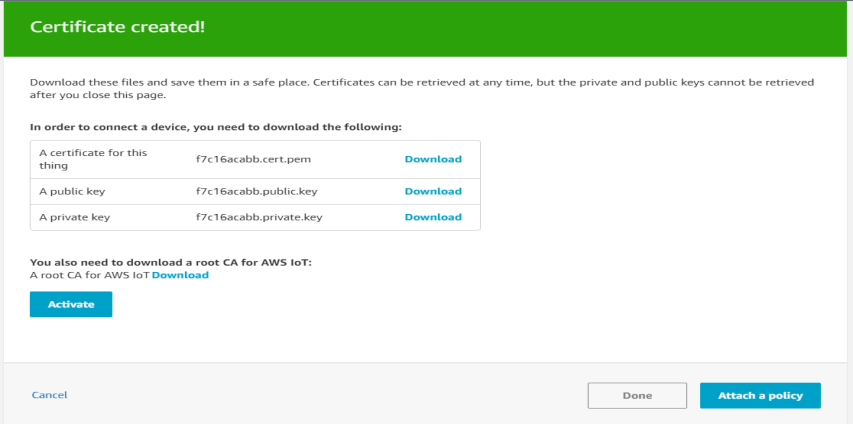
* On the Add a certificate for your thing page, choose Create certificate(one-click certificate creation). This generates an X.509 certificate and key pair.



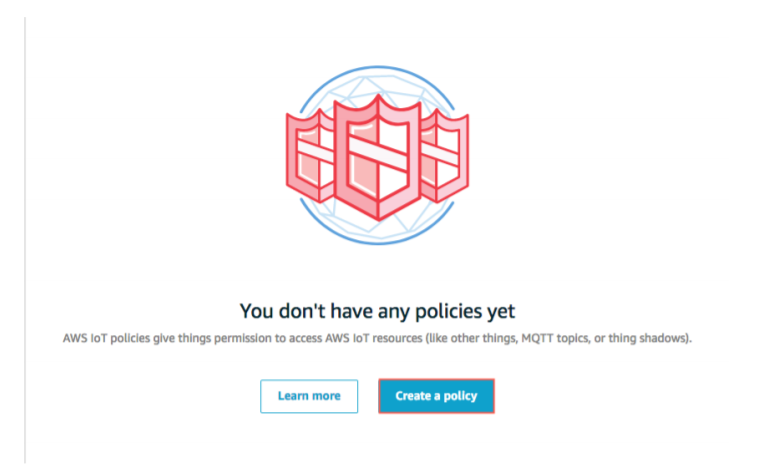
* On the Certificate created! page,
  + Choose Download A certificate for your Thing
  + Choose Download Public and Private key.
  + Choose Download A root certificate authority (CA) for AWS:
    - A new webpage is displayed.



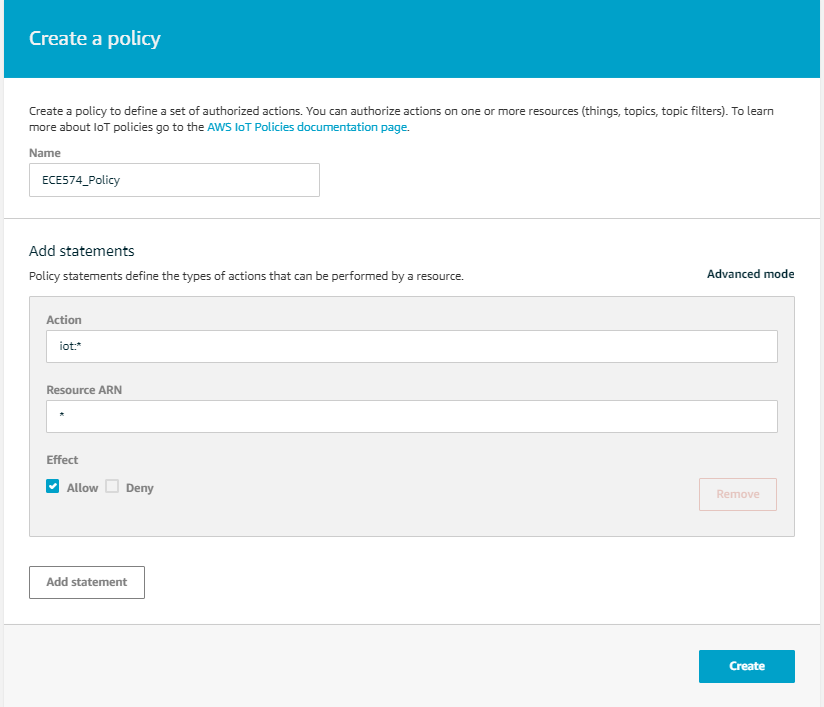
* + - Choose CA certificates for Server Authentication
    - Choose RSA 2048 bit key: VeriSign Class 3 Public Primary G5 root CA Certificate. This will display a page with the text of the root CA certificate. Copy this text and paste it into a file named VeriSign Class 3 Public Primary G5 root CA Certificate.pem.
* Choose Activate to activate the X.509 certificate, and then choose Attach a policy.



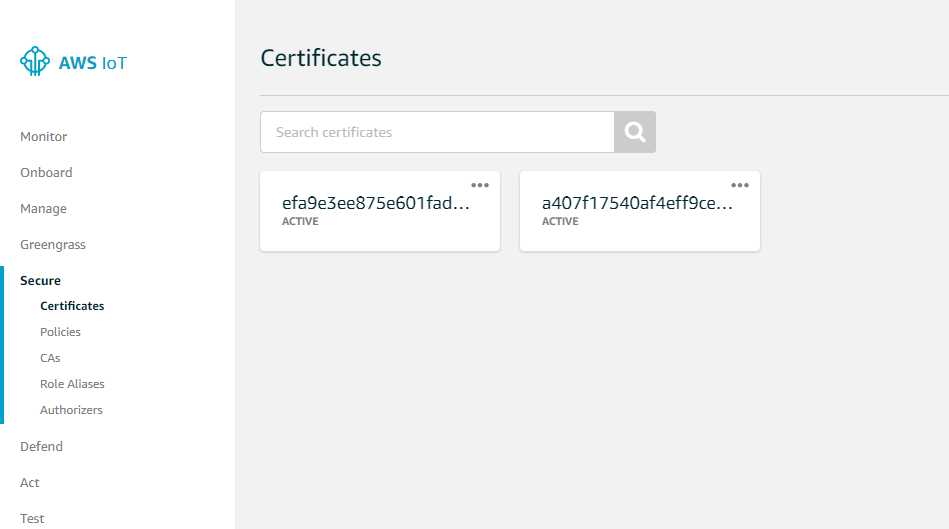
* On the Add a policy for your thing page, choose Register Thing.
* On the AWS IoT console, in the navigation pane, choose Secure, and then choose Policies. Choose Create.



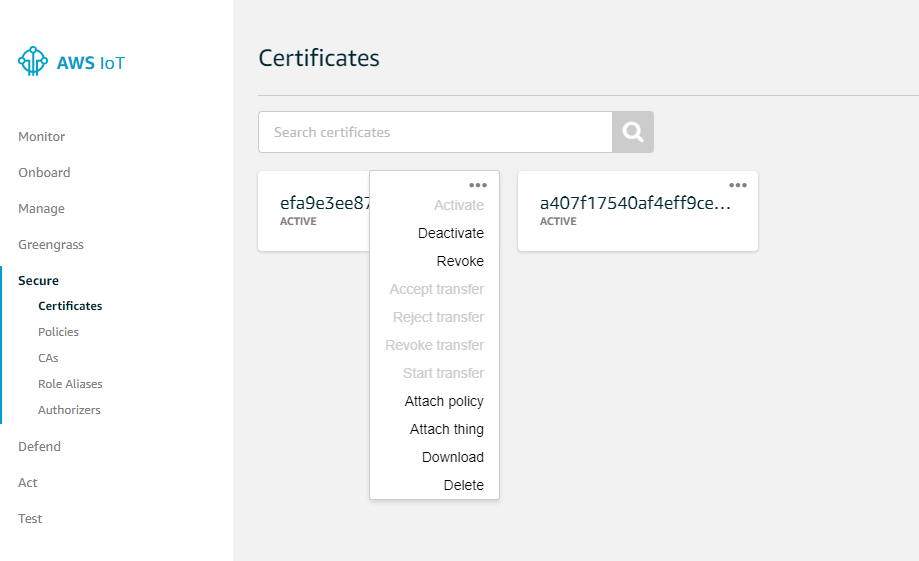
* On the Create a policy page:
  + Enter a Name for the policy, such as ECE574\_Policy.
  + For Action, enter iot:\*. For Resource ARN, enter \*.
  + Under Effect, choose Allow, and then choose Create.



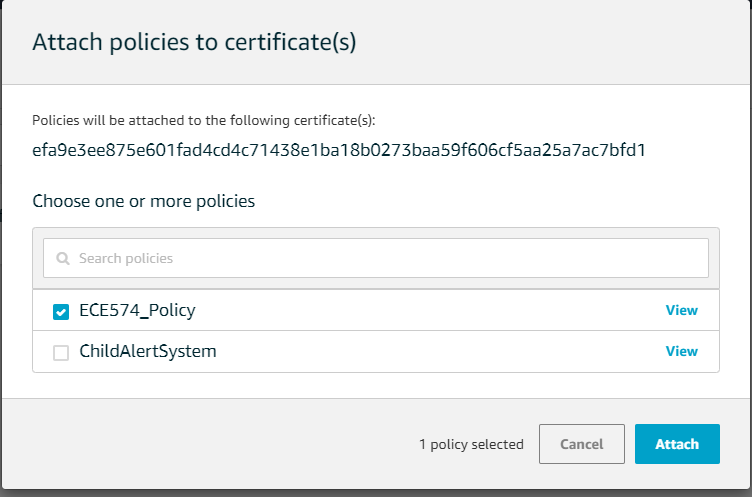
1. Attach an AWS IoT Thing and Policy to a Device Certificate.
   1. In the left navigation pane, choose Secure, and then choose Certificates.



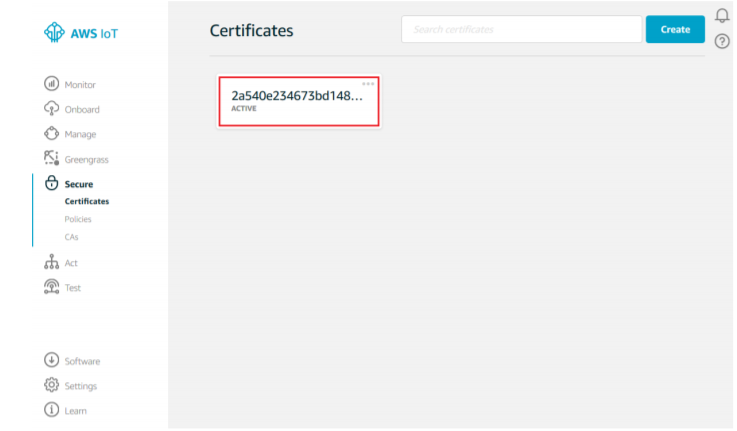
* 1. In the box for the certificate you created, choose ... to open a drop-down menu, and then choose Attach policy.



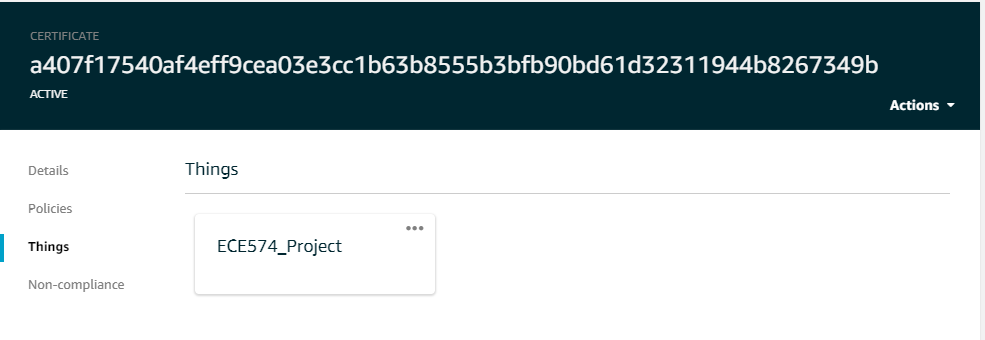
* 1. In Attach policies to certificate(s), select the check box next to the policy you created in the previous step, and then choose Attach.



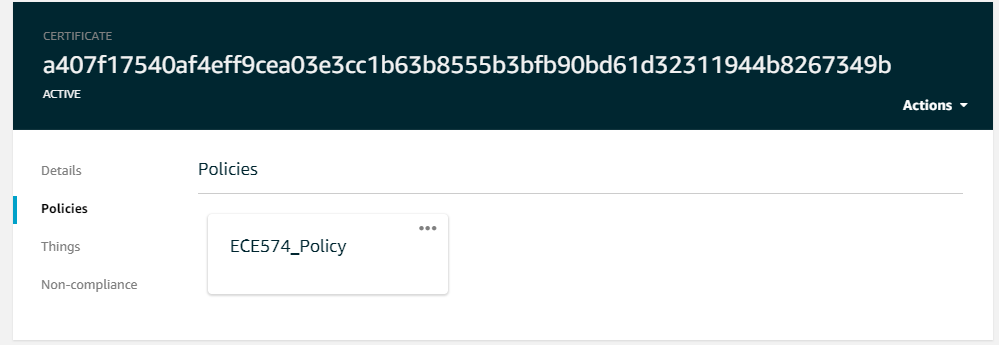
* 1. Repeat the above two steps and attach the Thing instead of Policy.
  2. To verify the thing is attached, select the box for the certificate.



* 1. On the Details page for the certificate, in the left navigation pane, choose Things.



* 1. To verify the policy is attached, on the Details page for the certificate, in the left navigation pane, choose Policies.

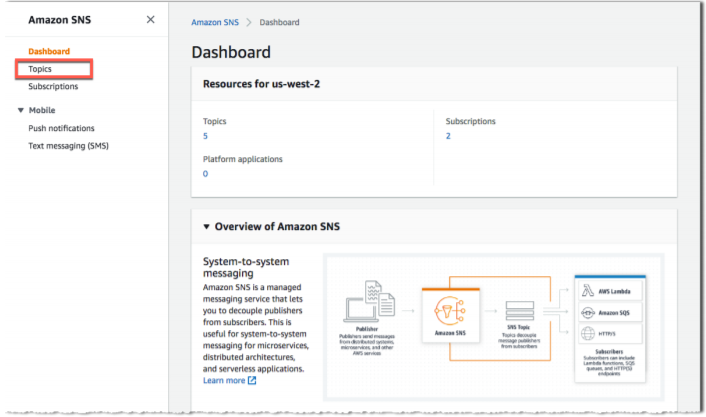


* 1. Save the End point of the Thing you created for communication purpose.
     + Click on the Thing box you created in the previous step.
     + In the left pane, click on Interact.
     + Save the HTTPS API Endpoint for further Communication.

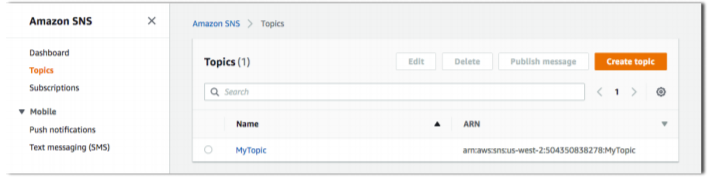
AWS SNS

1. Create an SNS Topic

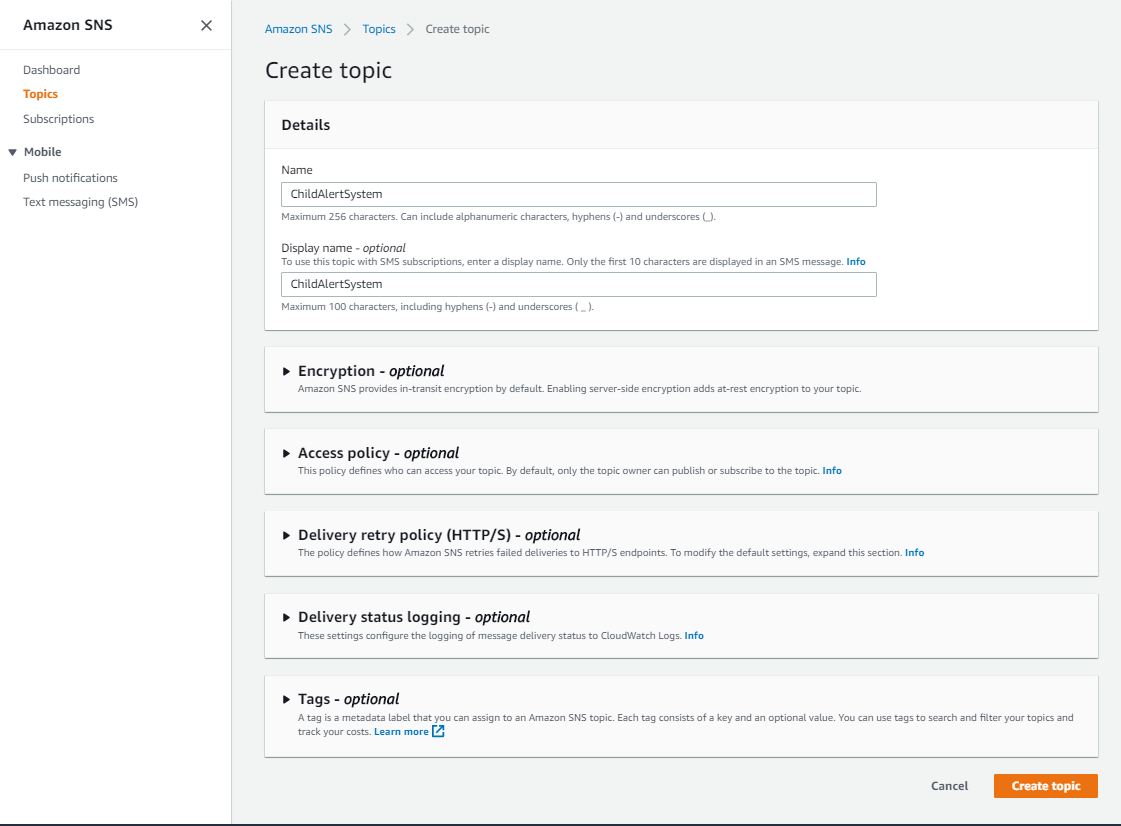
* Use the Amazon SNS console to create an Amazon SNS topic. (Note Amazon SNS is not available in all AWS Regions.)
  + Open the Amazon SNS console.
  + On the left pane, choose Topics.



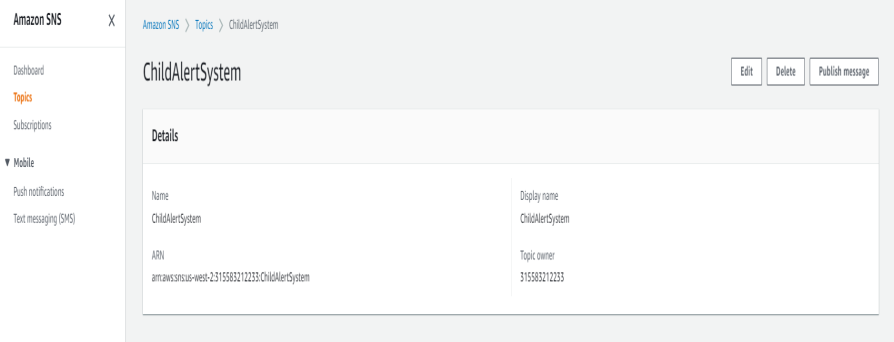
* Choose Create topic.



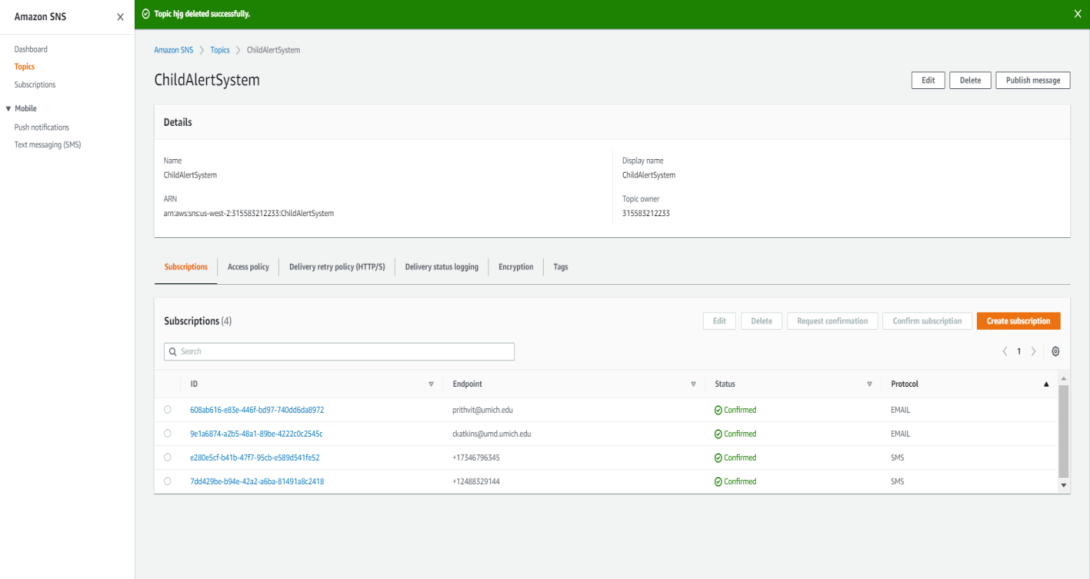
* Enter a topic name and a display name, and then choose Create topic.



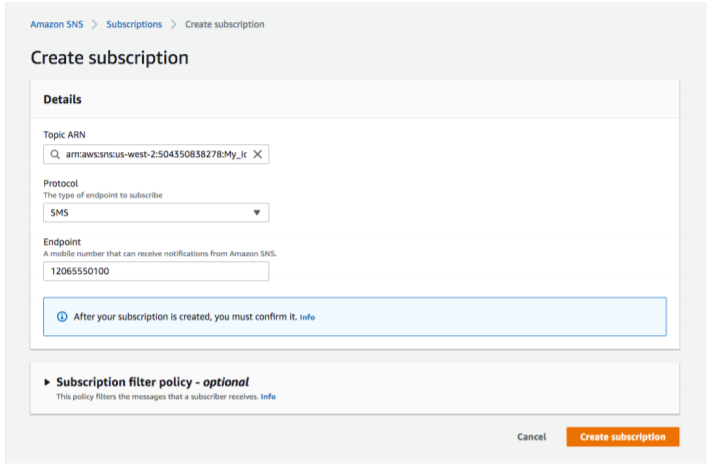
* Make a note of the ARN for the topic you just created.



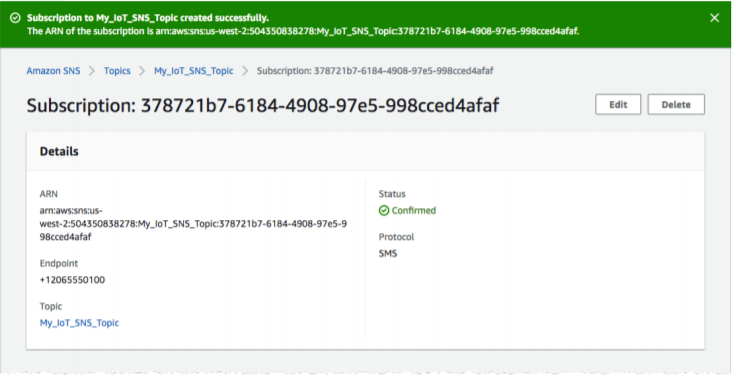
1. Subscribe to an Amazon SNS Topic
   * In the Amazon SNS console, select the check box next to the topic you just created. Under the Topic, choose Subscribe to topic.



* + On Create subscription, from the Protocol drop-down list, choose SMS



* + The Amazon SNS console displays the following message, but you might not receive a confirmation message.

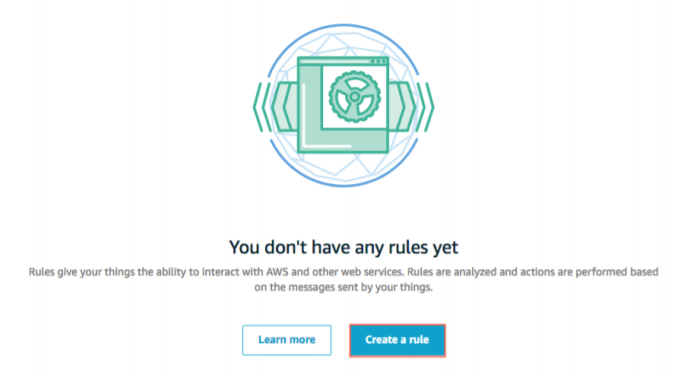


Rule

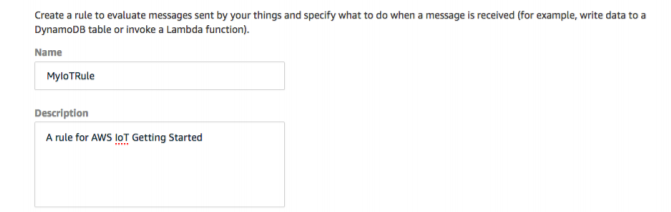
1. Create a Rule
   * In the AWS IoT console, in the left navigation pane, choose Act.



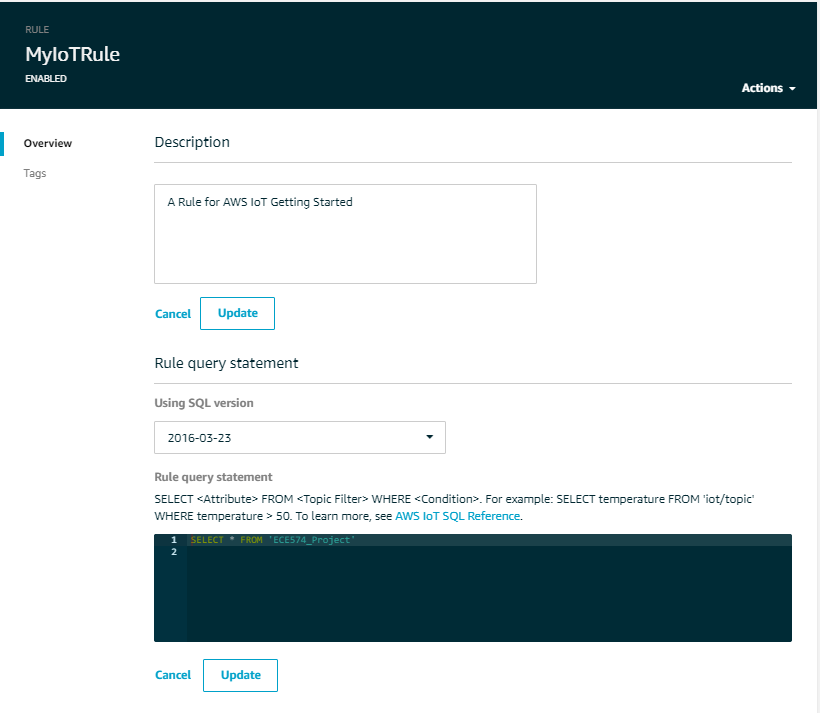
* + On the Act page, choose Create a rule.



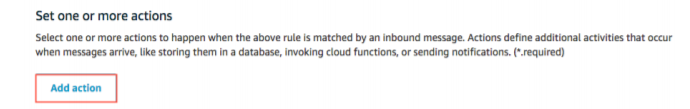
* + On the Create a rule page, in the Name field, enter a name for your rule. In the Description field, enter a description for the rule.



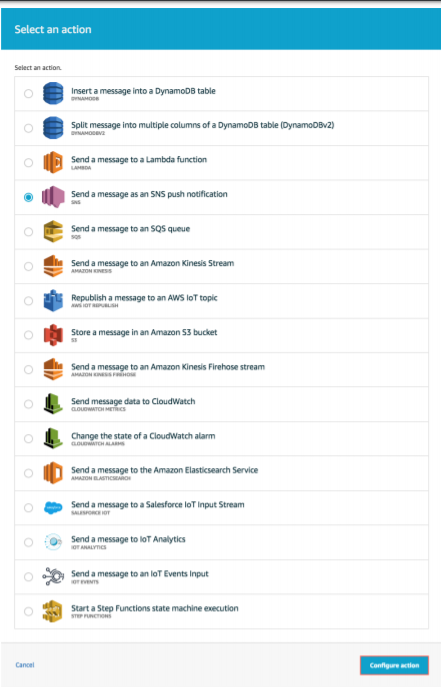
* + Scroll down to Rule query statement. Choose the latest version from the Using SQL version dropdown list. In the Rule query statement field, enter SELECT \* FROM 'my/topic'.



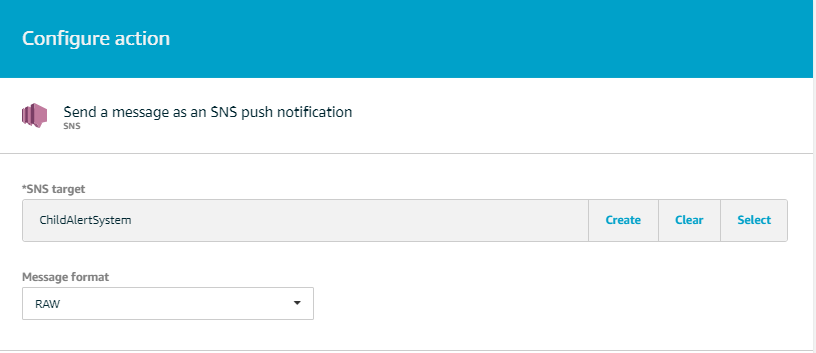
* + In Set one or more actions, choose Add action.



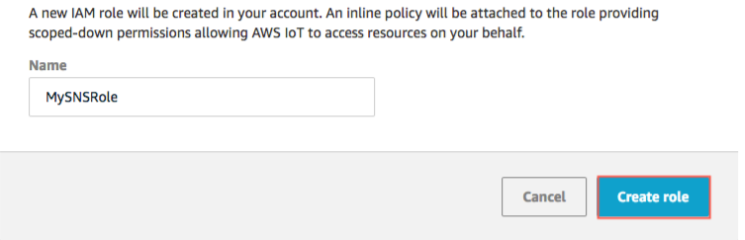
* + On the Select an action page, choose Send a message as an SNS push notification, and then choose Configure action.



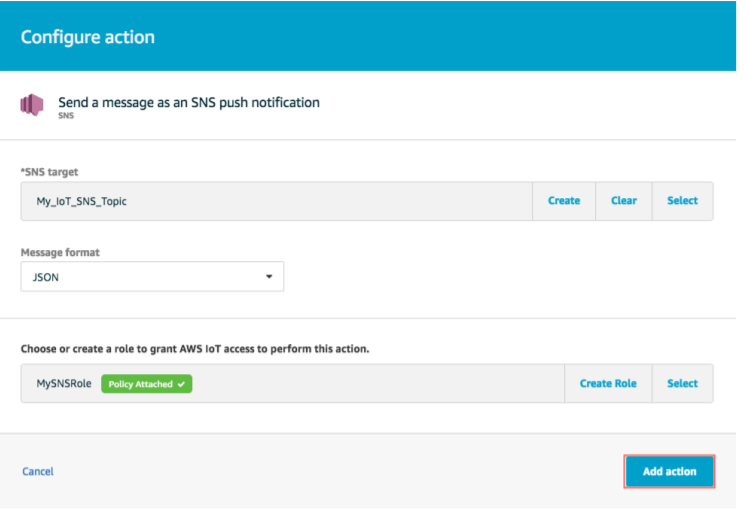
* + On the Configure action page, under SNS target, choose Select to expand the SNS topic. Then choose Select next to the Amazon SNS topic you created earlier. Under Message format, choose RAW.



* + Choose Create a new role. In IAM role name, enter a name for your new role, and then choose Create a new role.



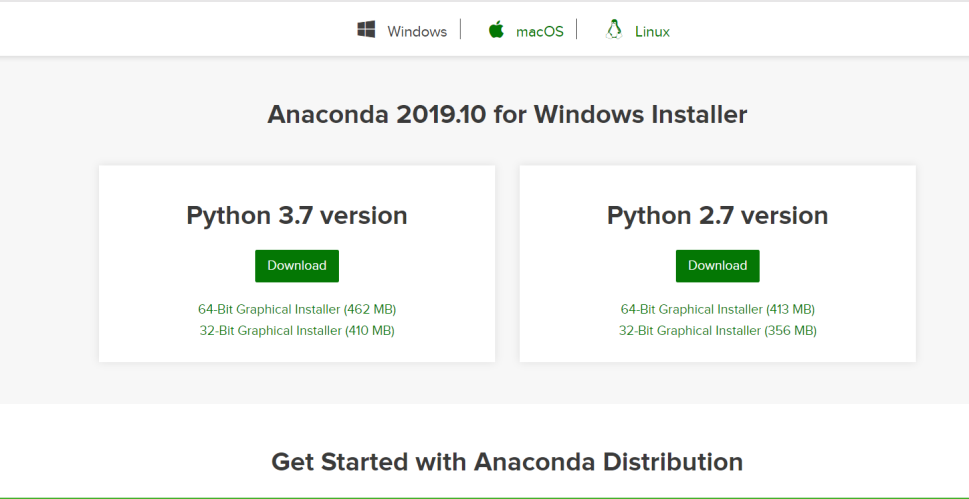
* + Under IAM role name, choose Update role to apply the permissions to the newly created role. Choose the role, and then choose Add action.



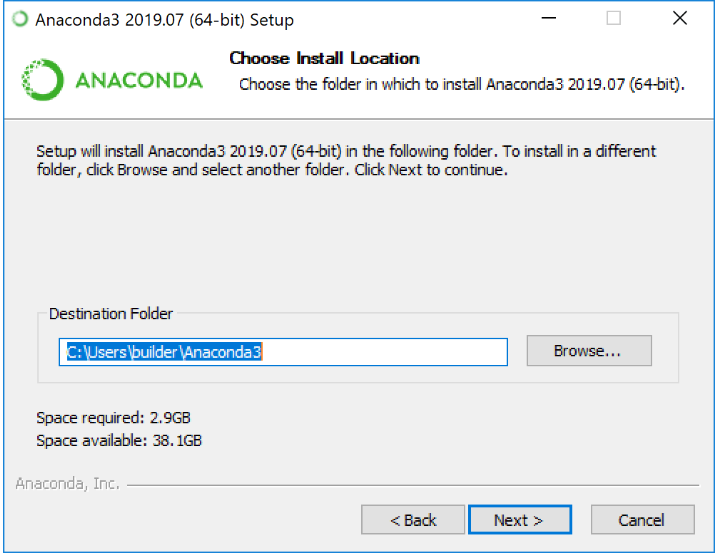
* + On the Create a Rule page, choose Create rule.

**Central Module**

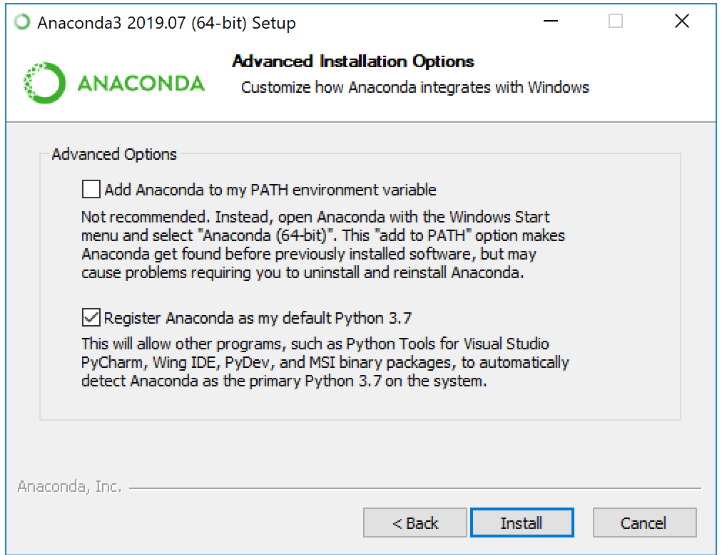
1. Installation of Anaconda Python
   * Open [Anaconda Distributions](https://www.anaconda.com/distribution/) Website.
   * Choose Windows Python 3.7 version 64-bit Graphical Installer and Download the Installer.



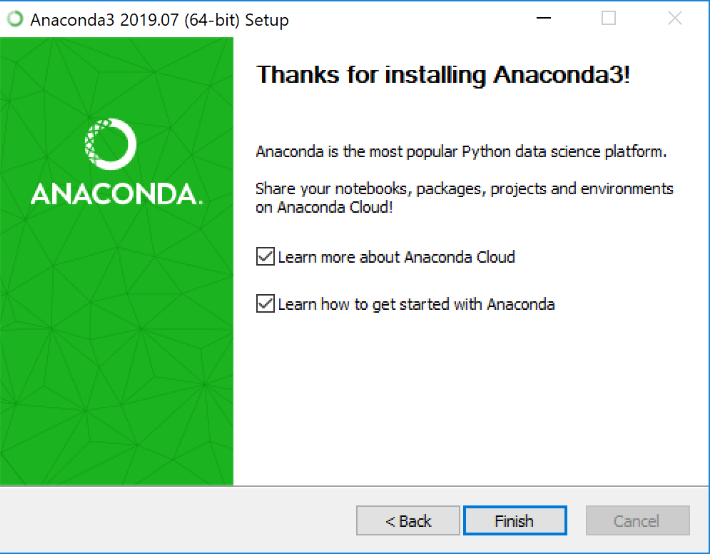
* + Open the Location where you Downloaded, right click on Installer and click on Run as Administration.
  + Click on Next.
  + Read the licensing terms and click “I Agree”.
  + Select an install for “Just Me”.
  + Select a destination folder to install Anaconda and click the Next button.



* + Choose Register Anaconda as my Default Python 3.7. Click on Install.



* + Click on Next.
  + After a successful installation, Click on Finish.



* + In Computer, go to Search, type Spyder and open the Spyder.
  + In Spyder IDE, Click on File and open a New File.
  + Save the File with .py extension.
  + Write the Python Algorithm (Code).
  + Click on Green Run Button.
  + If the Details given are Correct, it makes a connection with AWS IoT Thing ECE574\_Project. If not Check the Details and retry the connection.

