IOT SECURITY AND PRIVACY
ASSIGNMENT 6 – AWS IOT
By:- Anmol Sureshkumar Panchal

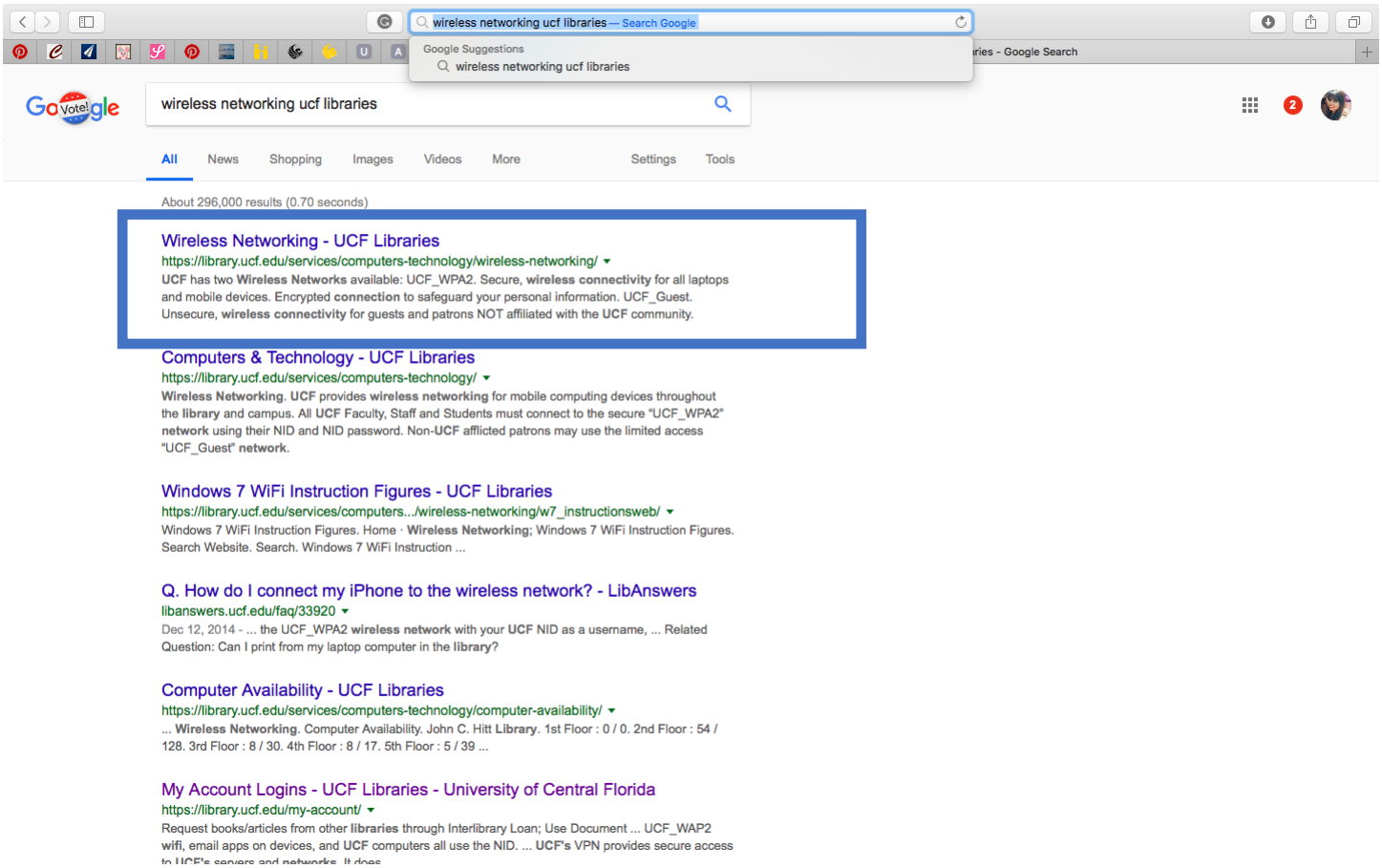
Questions:

In this assignment, students are required to connect ESP32 through UCF_WPA2 or eduroam to Amazon AWS IoT and update the state of the connected sensor (DHT11) continuously with the AWS IoT thing shadow. The data such as the state generated by the sensor should be written into Amazon DynamoDB. Note: be careful not to exceed the free account quota of data for AWS IoT. Please refer to the references [1][2][3][4][5][6][7][8][9], particularly [1] [2], if necessary for this assignment.

Requirements:

1. Document in detail the procedures connecting ESP32 through *UCF_WPA2* or *eduroam* to Amazon AWS IoT, including configuring Amazon AWS IoT and writing code connecting to Amazon AWS IoT. (8 points)

- Go to https://library.ucf.edu/services/computers-technology/wireless-networking/ and download the certificate.



Pages

Computer Availability
Computer Policy
Computer Specifications
LibTech
Software
Study Areas with Tech
Technology Help
Technology Lending
Wireless Networking

Wireless Networking

UCF has two Wireless Networks available:

- **UCF_WPA2**
  - Secure, wireless connectivity for all laptops and mobile devices
  - Encrypted connection to safeguard your personal information.
  - Uses your NID & NID Password for authentication
  - Faculty, Staff, and UCF Students should always connect to UCF_WPA2.
- **UCF_Guest**
  - Unsecure, wireless connectivity for guests and patrons NOT affiliated with the UCF community.
  - No encryption; Do NOT use for web browsing that requires submitting personal information.
  - Requires authentication via Acceptable Use Policy page before fully connected.
  - Only university guest may connect to UCF_Guest.

Use the tabs below to view directions on how to connect your device to the UCF_WPA2 network.

Additional wireless networking support can be found at wireless.ucf.edu

Windows 7 | Windows 8 / 8.1 / 10 | Mac OS X | iOS (iPhone/iPad) | Android

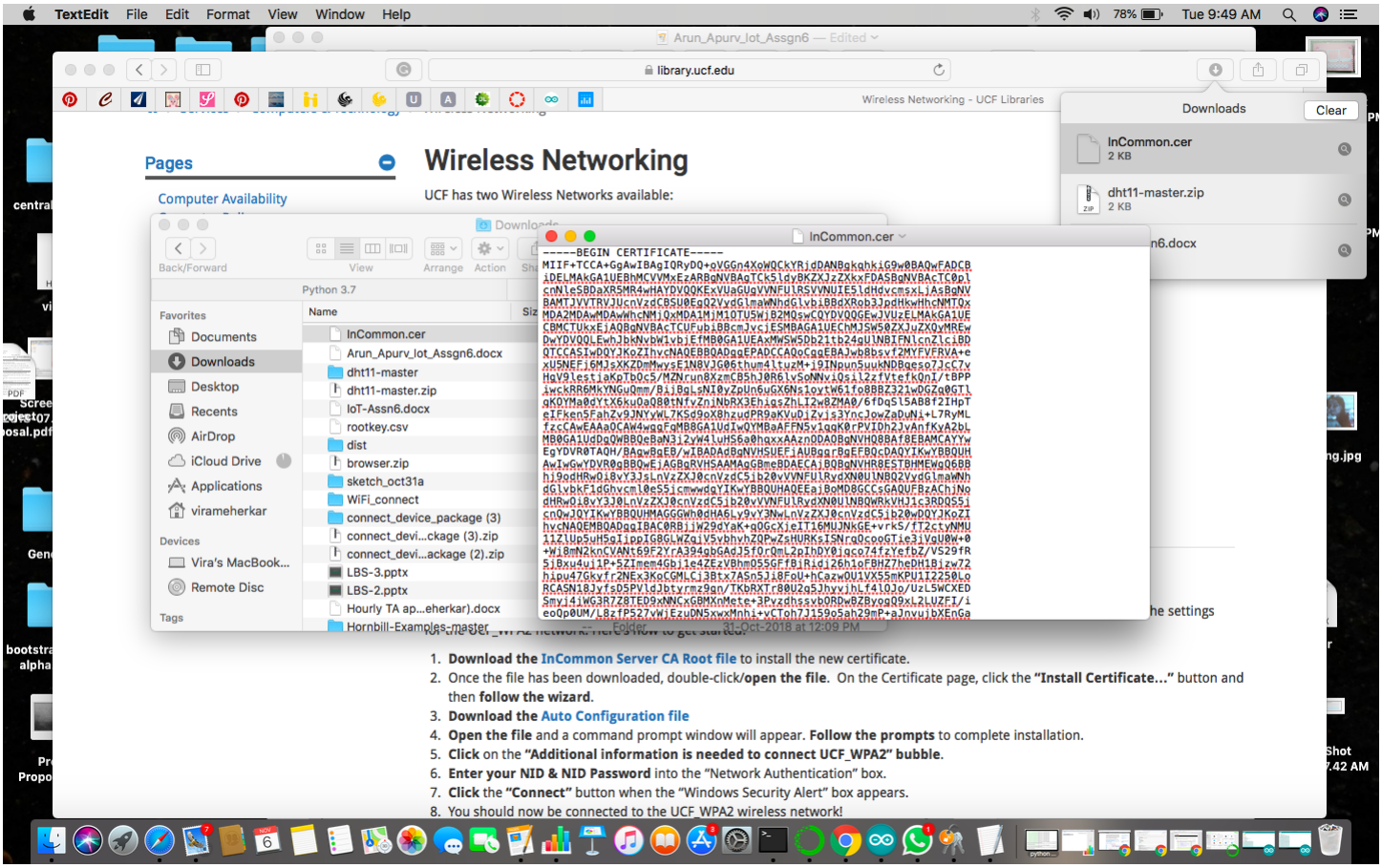
Windows 7

Windows 7 requires you to first install a security certificate and then run a special script that automatically sets up all of the settings for the UCF_WPA2 network. Here's how to get started:

1. **Download the InCommon Server CA Root file** to install the new certificate.
2. Once the file has been downloaded, double-click**/open the file**. On the Certificate page, click the **"Install Certificate..."** button and then **follow the wizard**.
3. **Download the Auto Configuration file**
4. **Open the file** and a command prompt window will appear. **Follow the prompts** to complete installation.
5. **Click** on the **"Additional information is needed to connect UCF_WPA2" bubble**.
6. **Enter your NID & NID Password** into the "Network Authentication" box.
7. **Click** the **"Connect"** button when the "Windows Security Alert" box appears.
8. You should now be connected to the UCF_WPA2 wireless network!

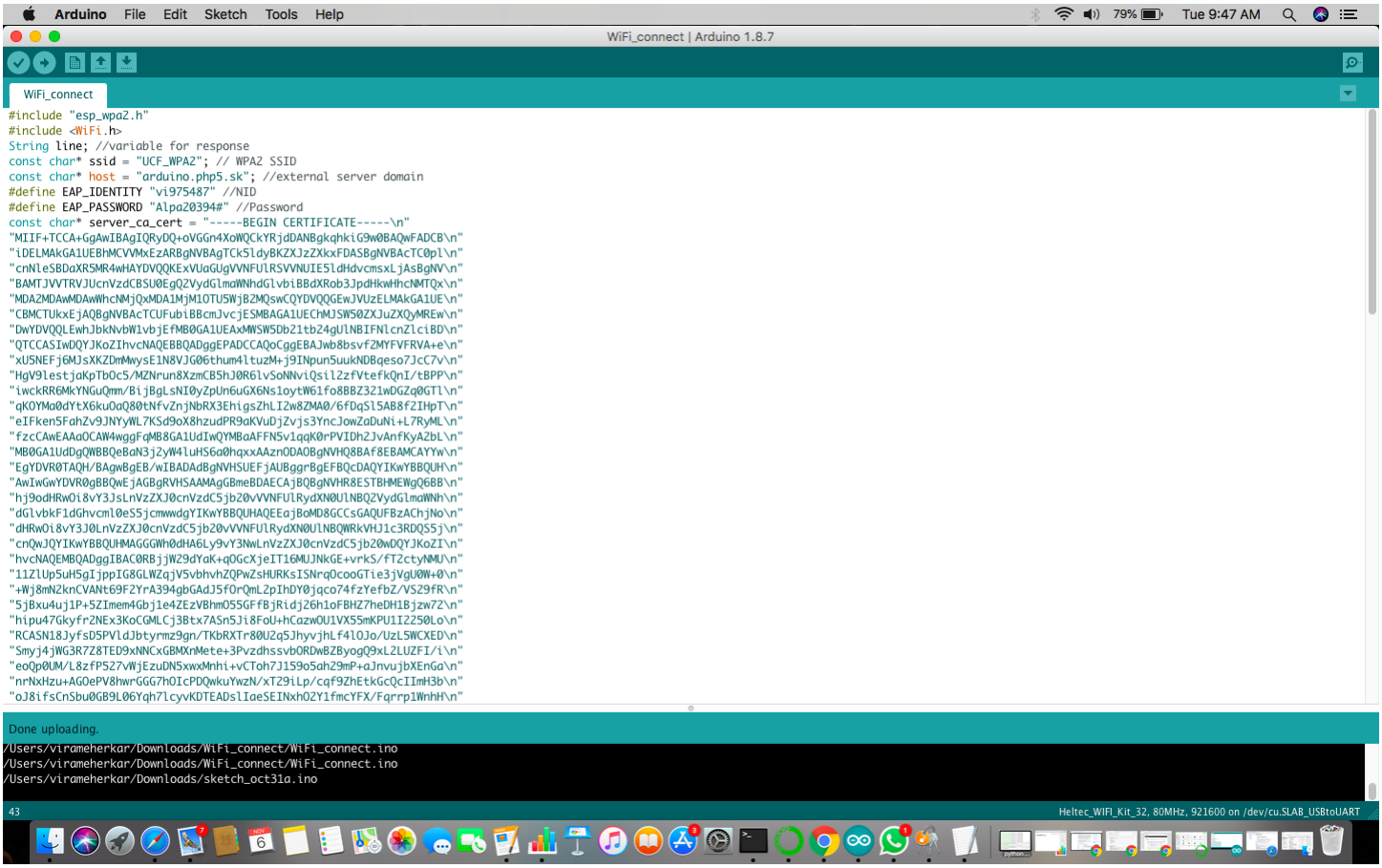
Still having trouble? Stop by the LibTech desk on the 3rd floor for friendly assistance.

- Once downloaded the certificate open it in text Editor and copy paste the certificate into the code as a

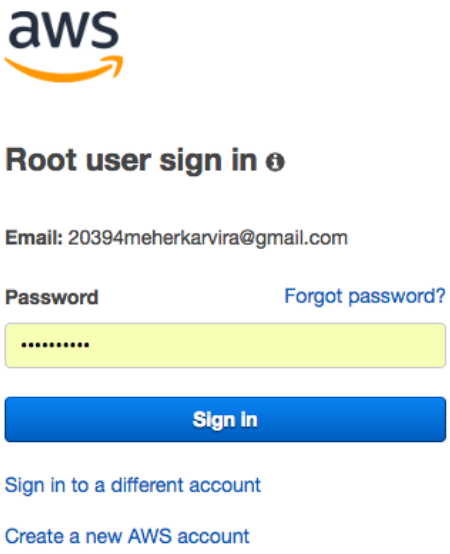


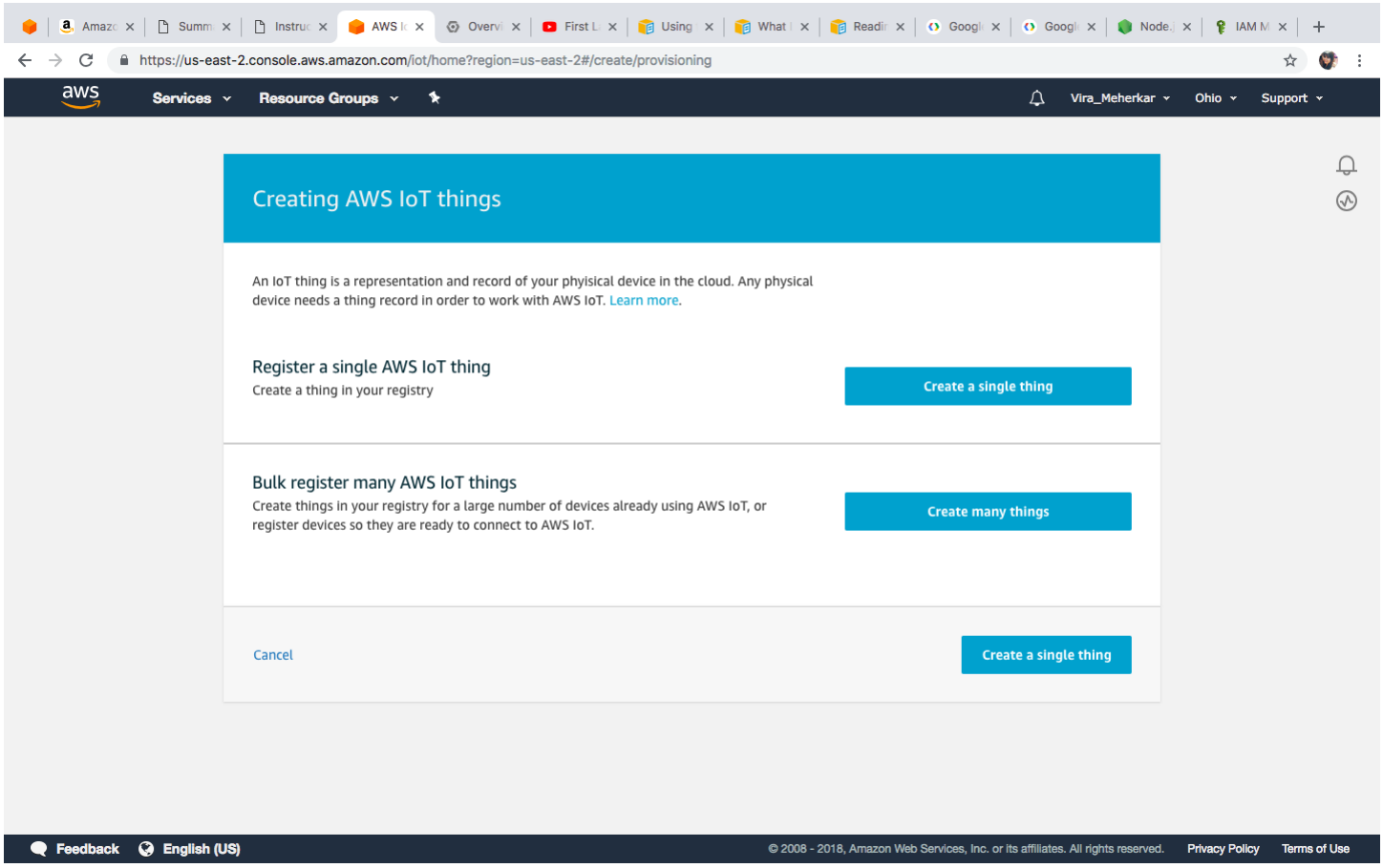
array of character.

- Then use the WPA2Enterprise library to configure the certificate with you UCFID and Password.

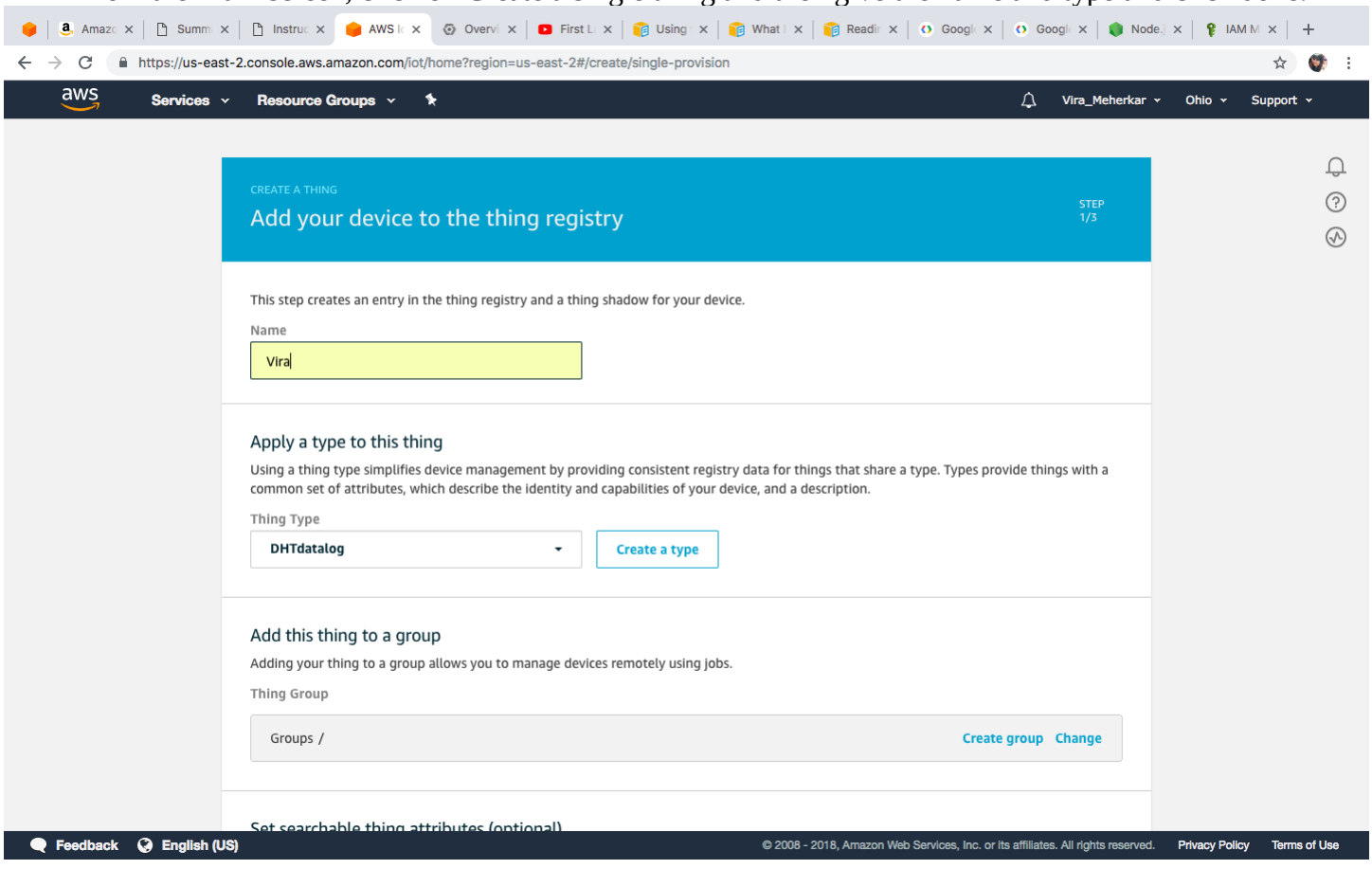


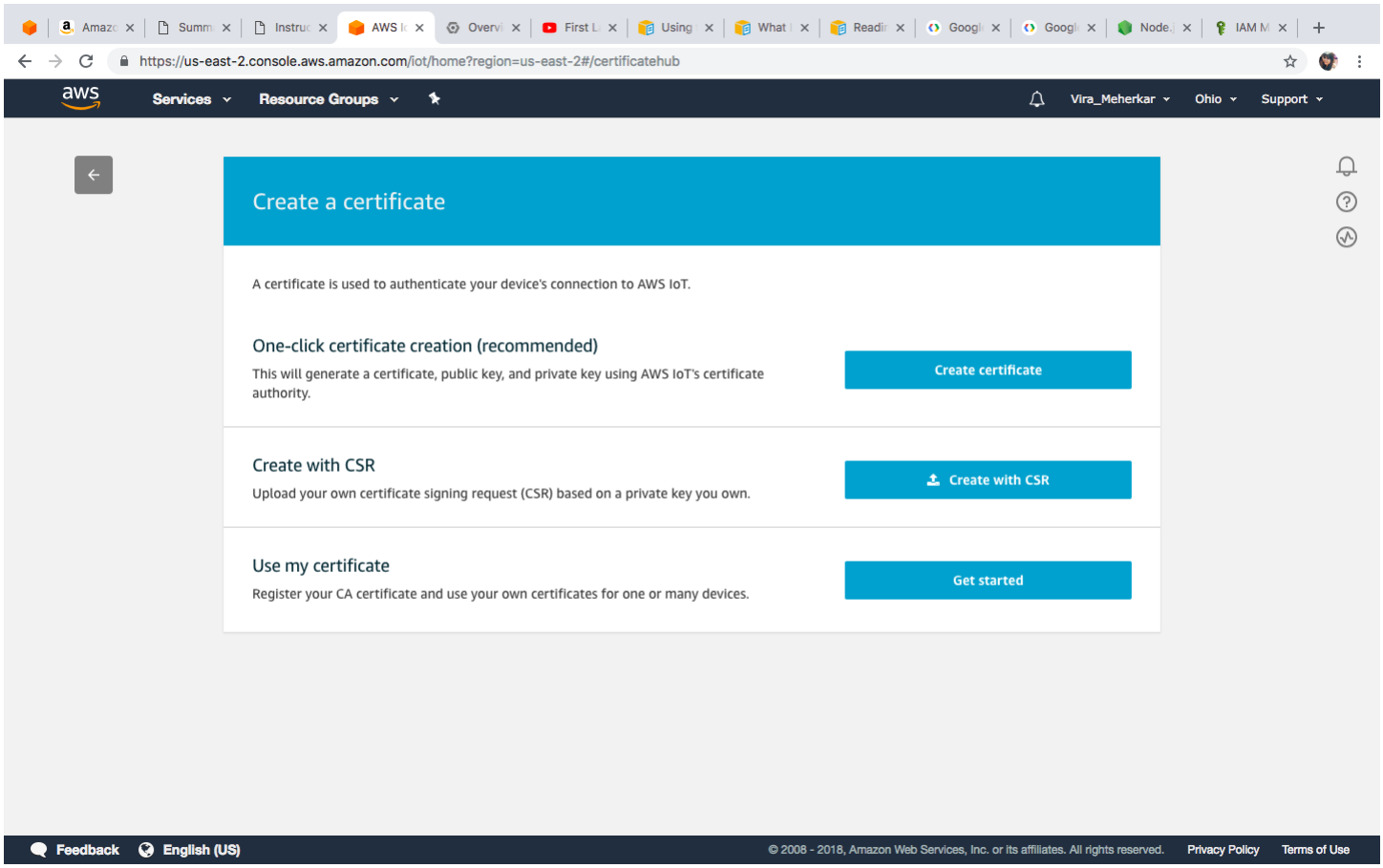
- Create a free AWS account, from the AWS console go to IoT Core and then on the left side bar click on Manage.



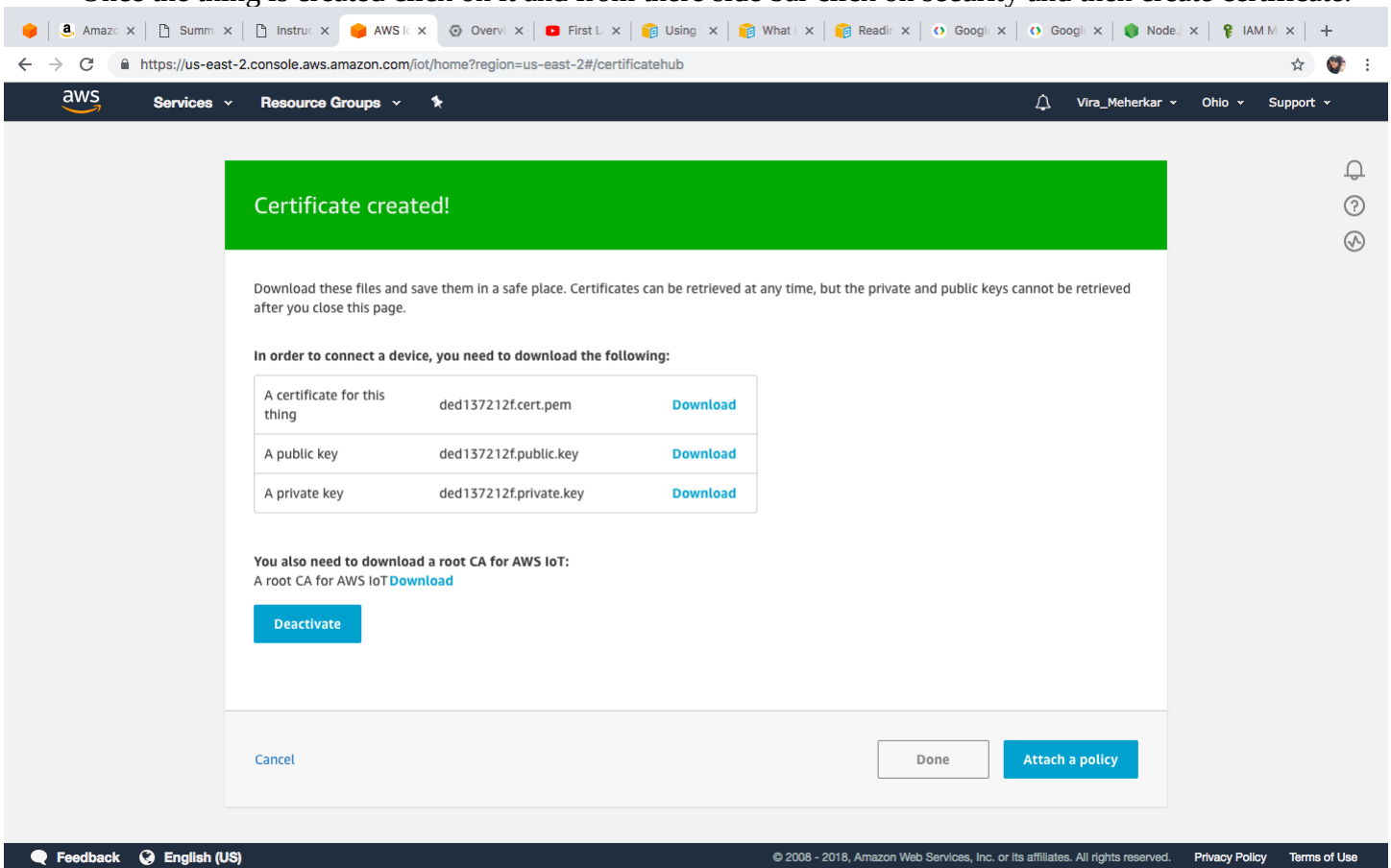


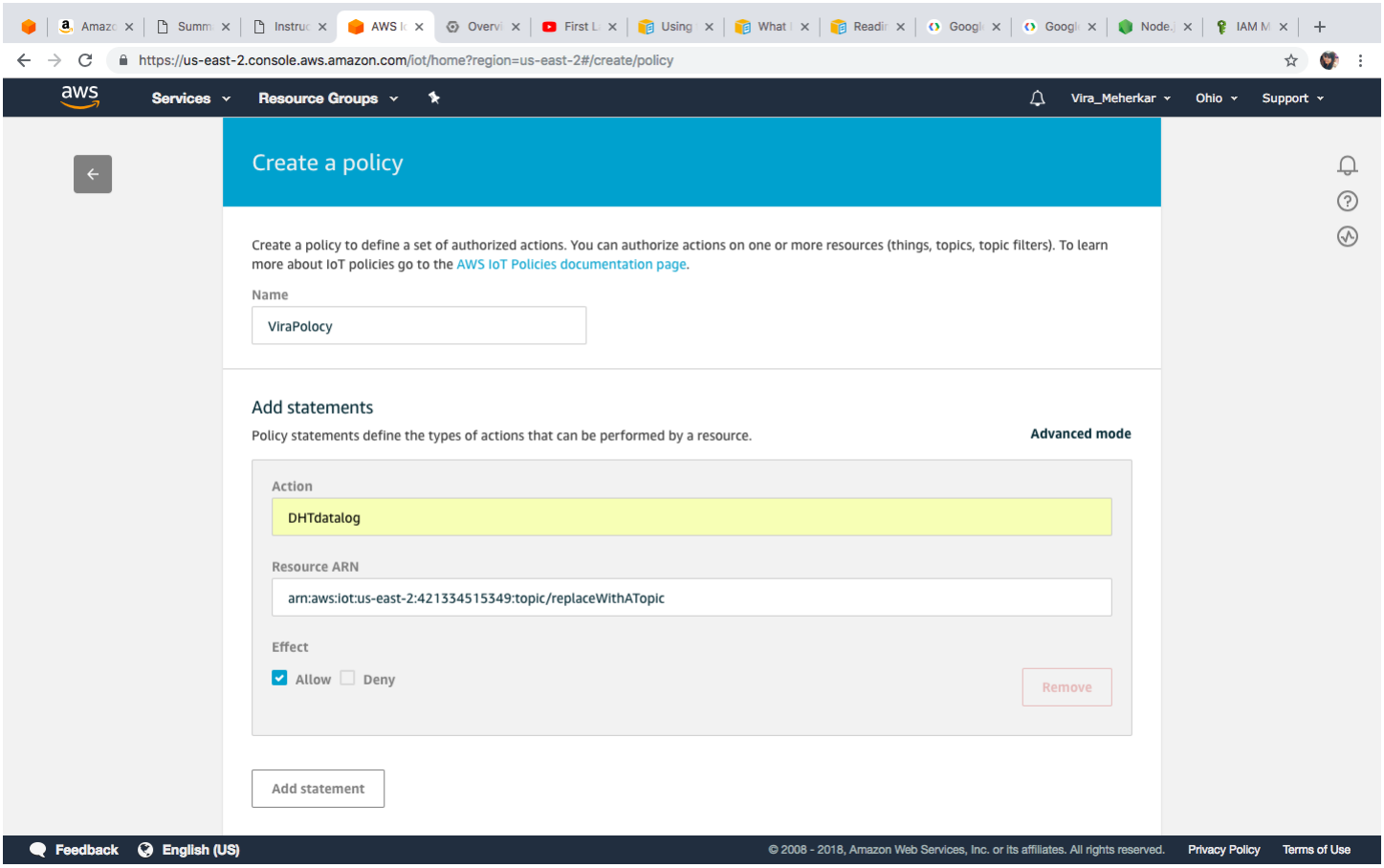
- From the Main screen, click on Create a single thing and then give the name and type and click done.



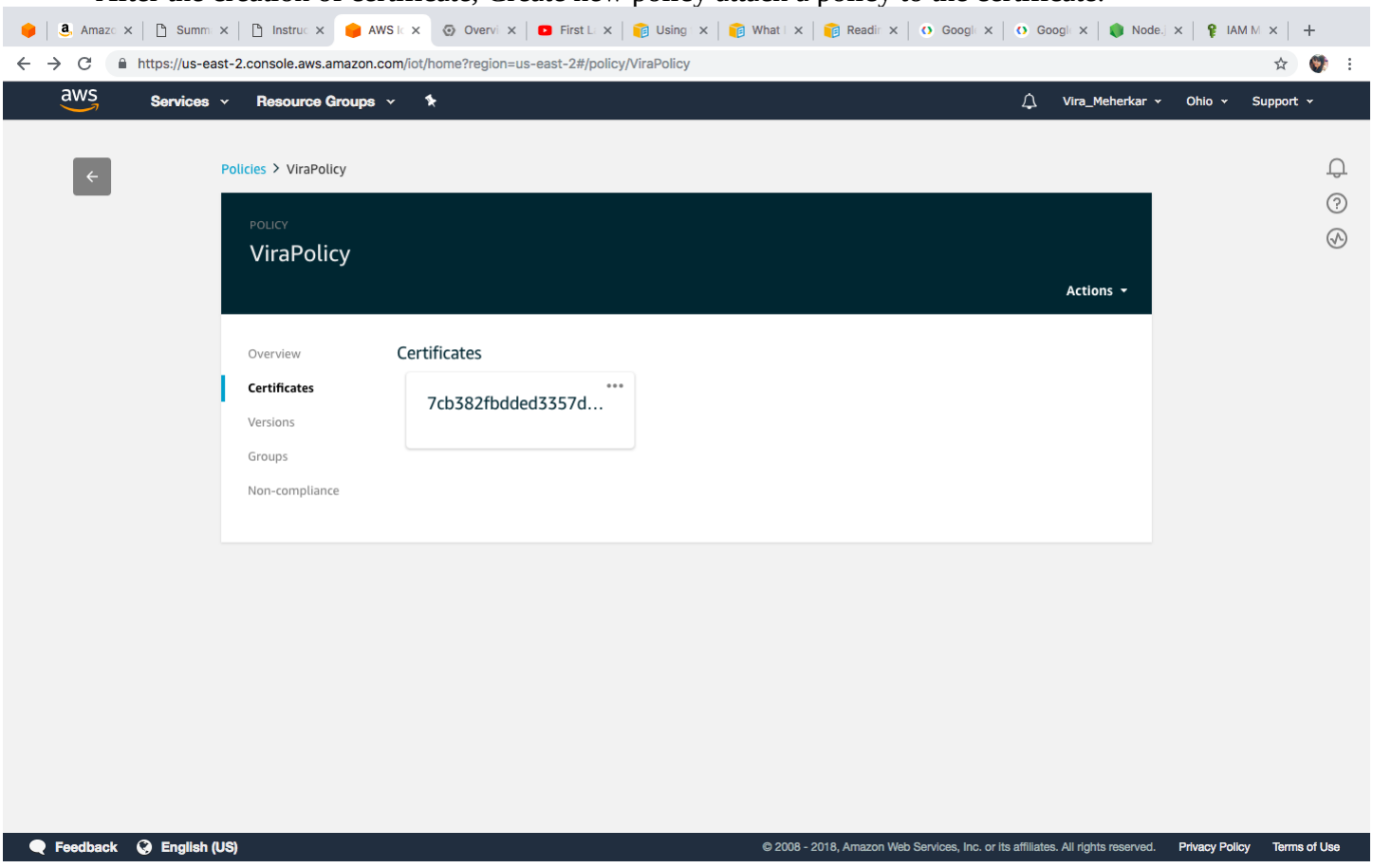


- Once the thing is created click on it and from there side bar click on security and then create certificate.

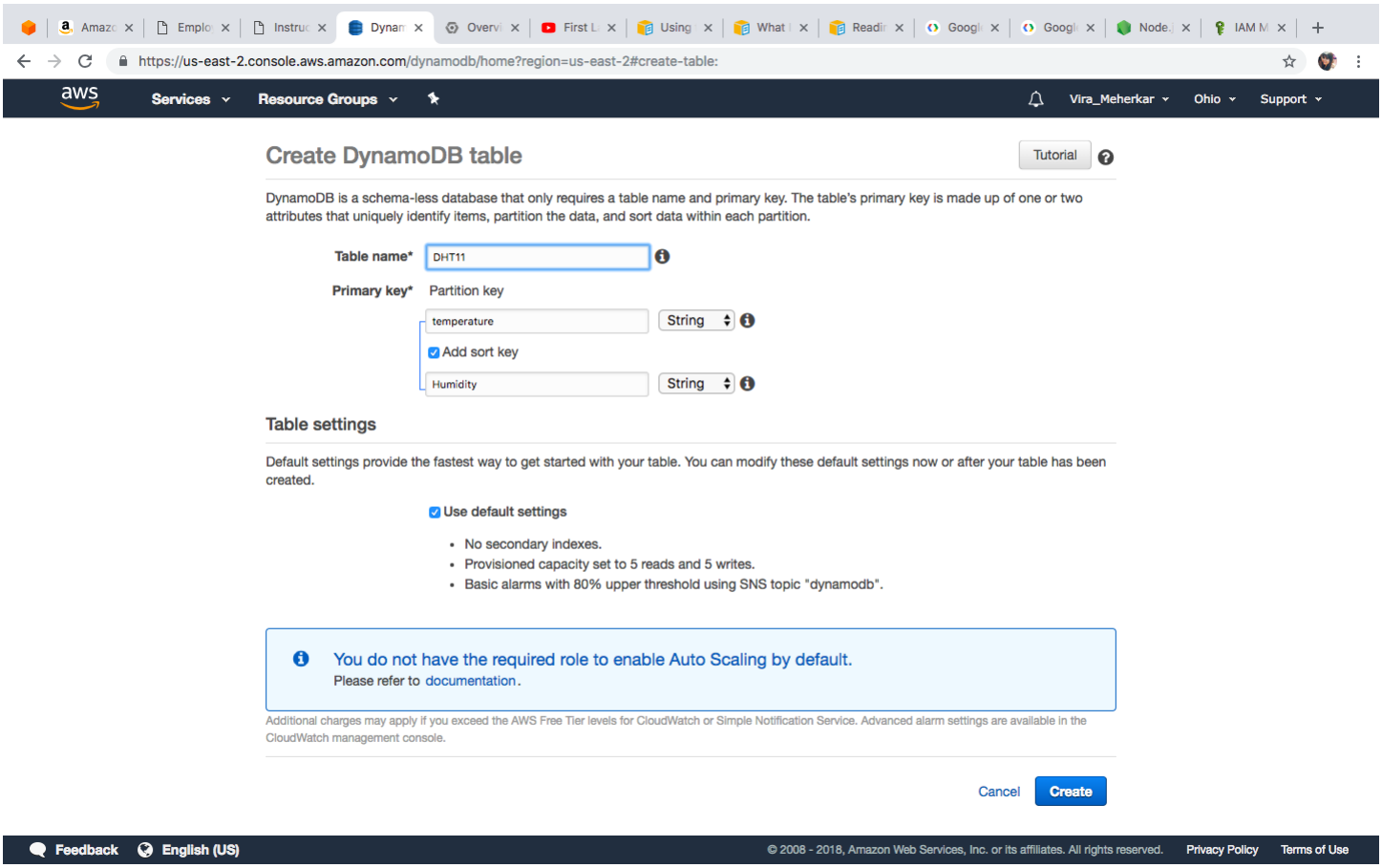




- After the creation of certificate, Create new policy attach a policy to the certificate.



- Now from dynamo DB create a table to insert data from the IOT thing.



- From the Side Bar click on ACT and then create rule for the insertion action to select the data from the IOT thing and insert in the specific table.

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

Data_Insert

Description

enter humidity and temperature

Rule query statement

Indicate the source of the messages you want to process with this rule.

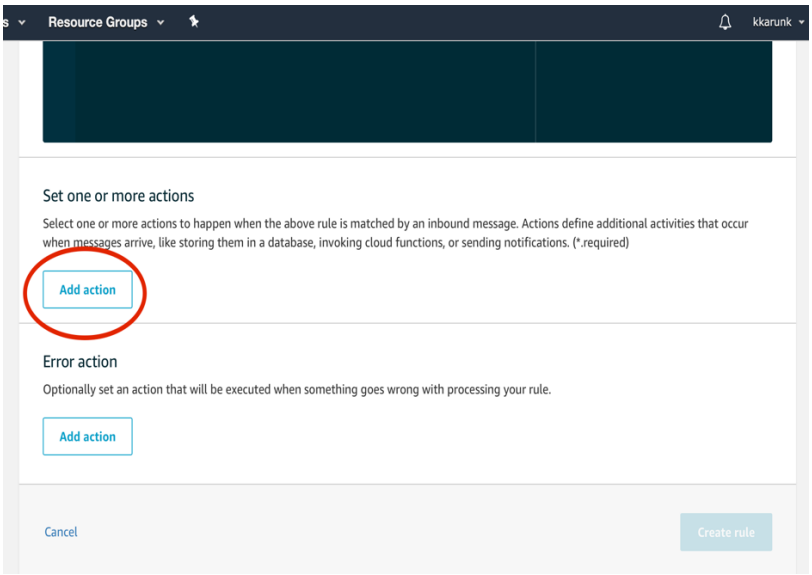
Using SQL version

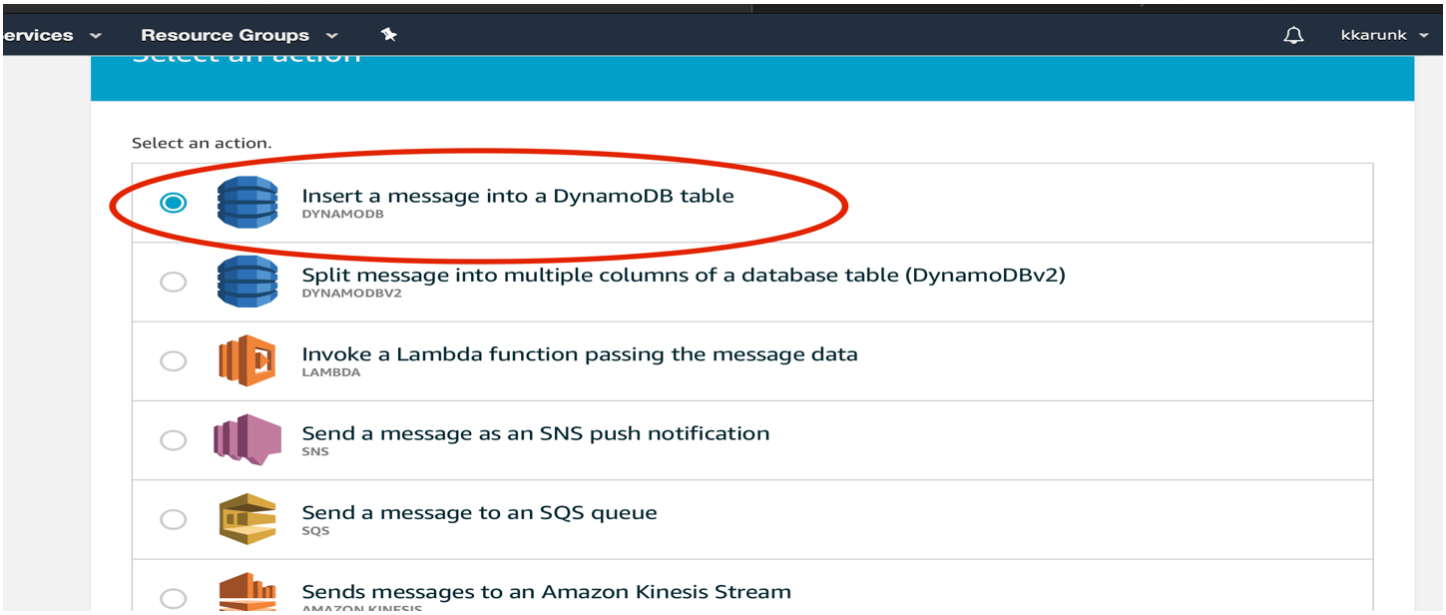
2016-03-23

You don't have any rules yet

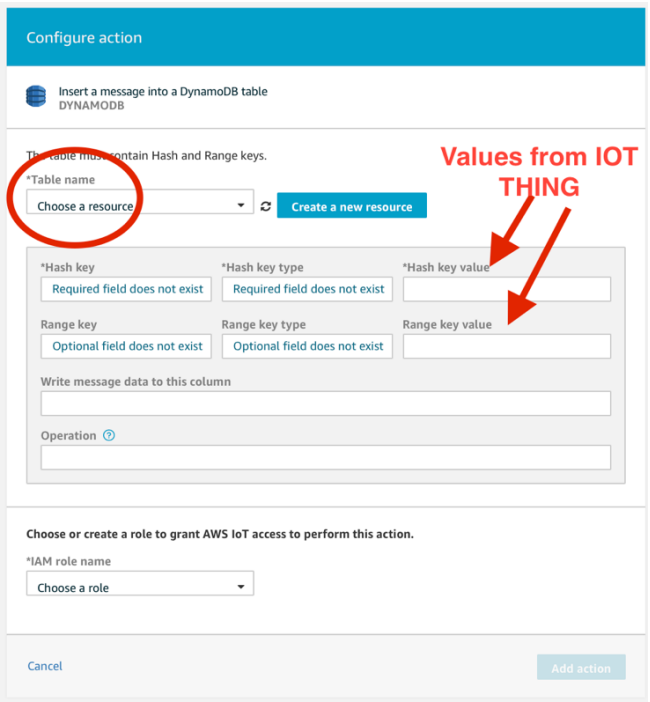
Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.

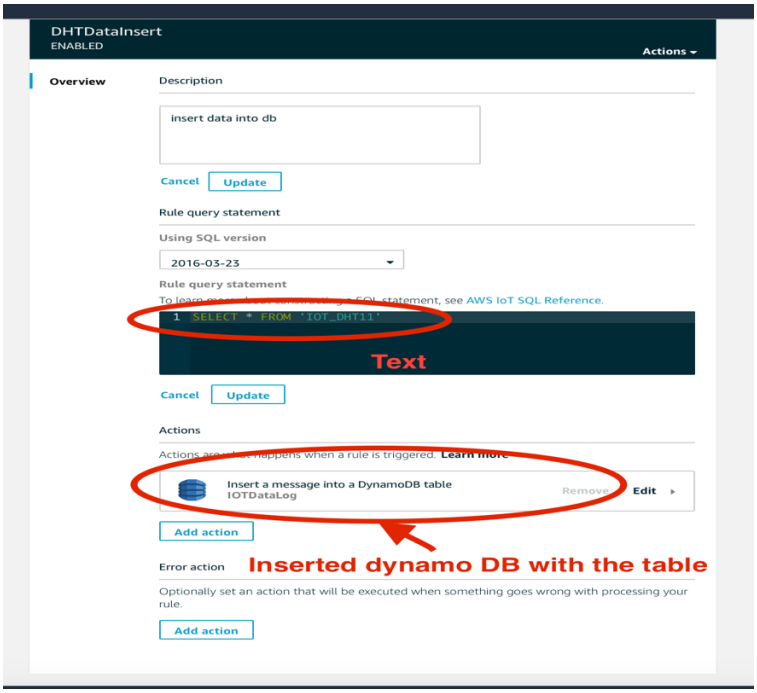
Learn more | Create a rule



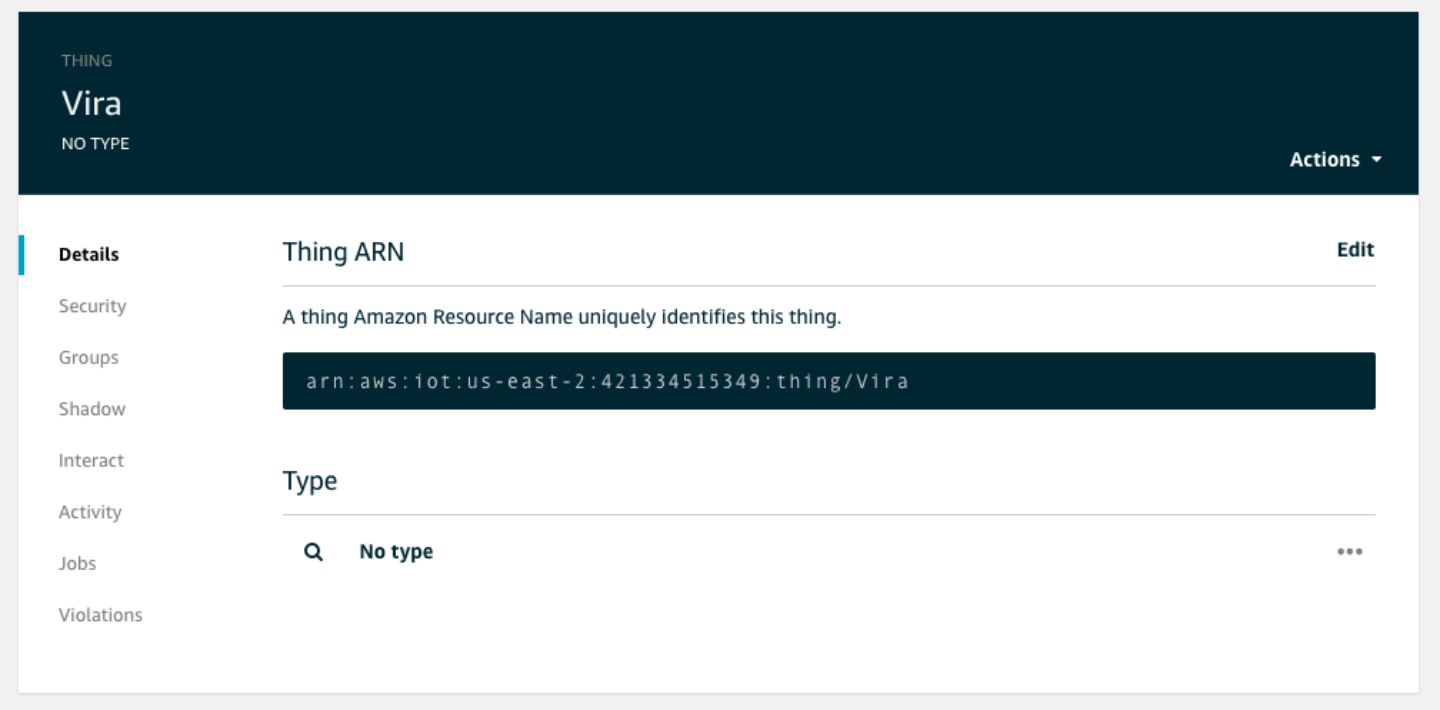


- Select the table name in you dynamo DB and the IOT Thing name for the insertion operation.



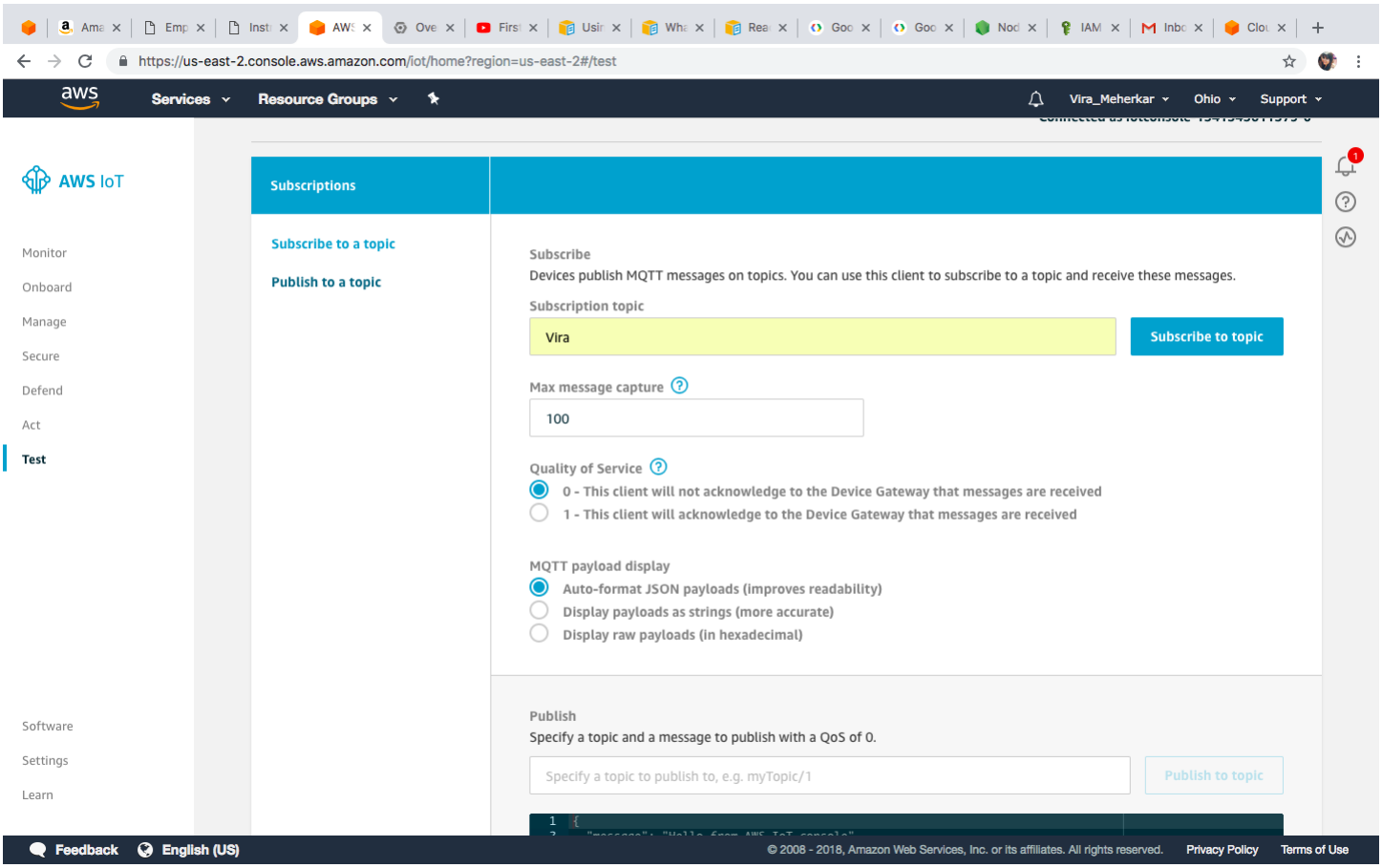


- In the ESP32 code, we use the AWS IOT library to connect to the AWS, which uses the HTTPS AWS



endpoint to connect to the IOT

- For the Publish and Subscribe it uses the MQTT shadow updates, also we should JSON format to ensure

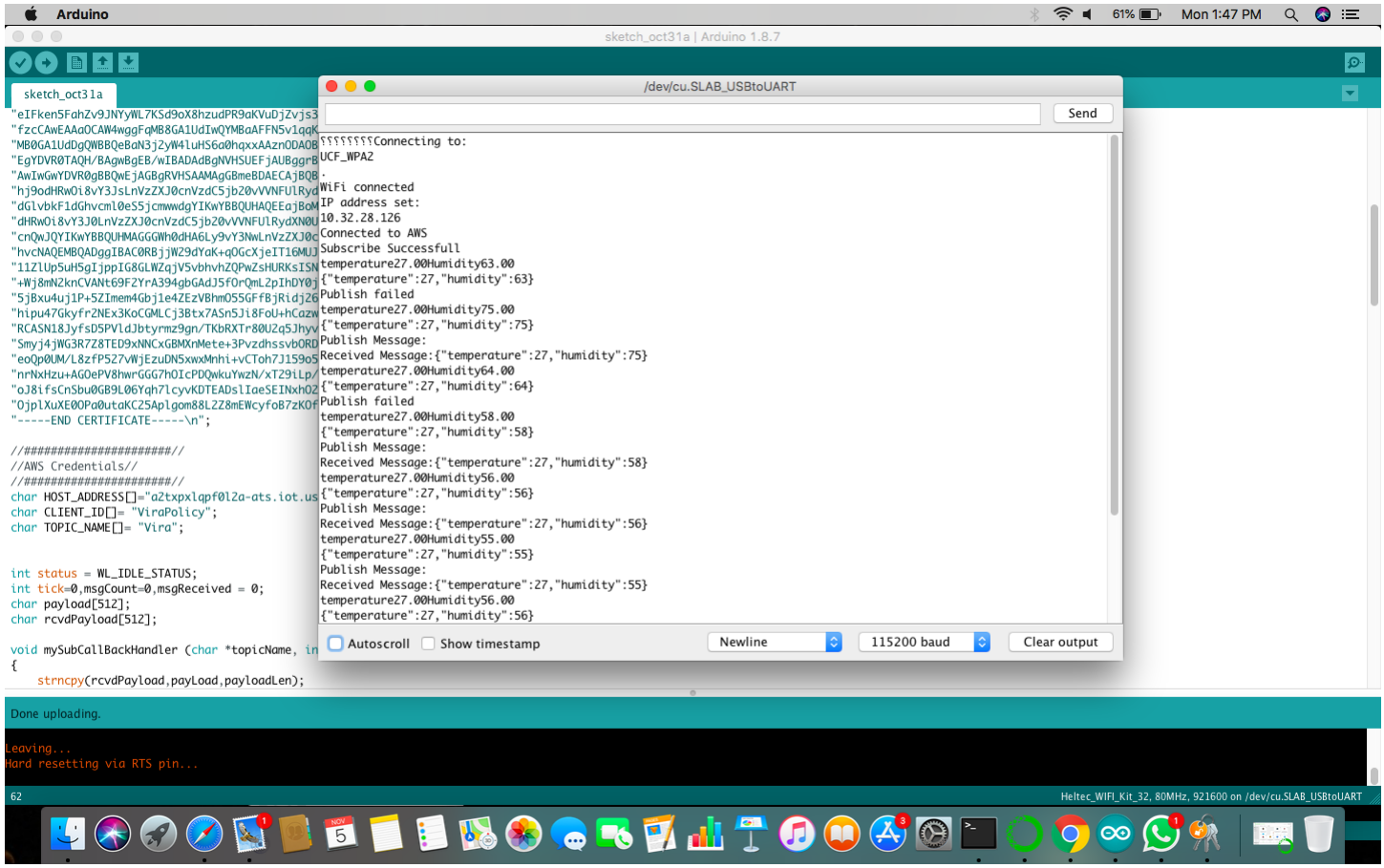


the data read and insertion in the Dynamo DB table, that we have created.

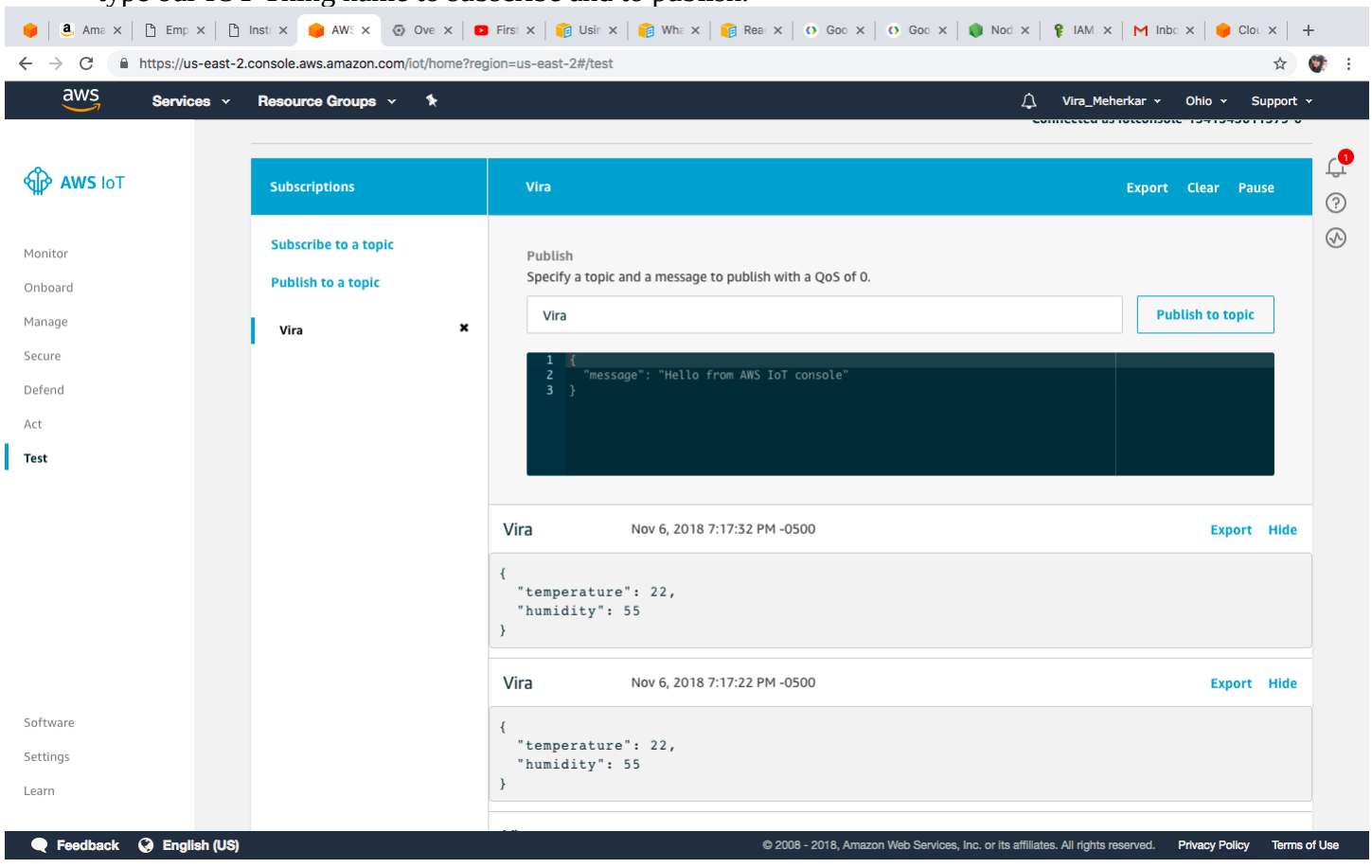
- Also we need to update the certificates and key in a ‘C’ file to ensure connection to the IOT.


```
//#######################//
//AWS Credentials//
//#######################//
char HOST_ADDRESS[]="a2txpxlqpf0l2a-ats.iot.us-east-2.amazonaws.com";
char CLIENT_ID[]= "ViraPolicy";
char TOPIC_NAME[]= "Vira";
```

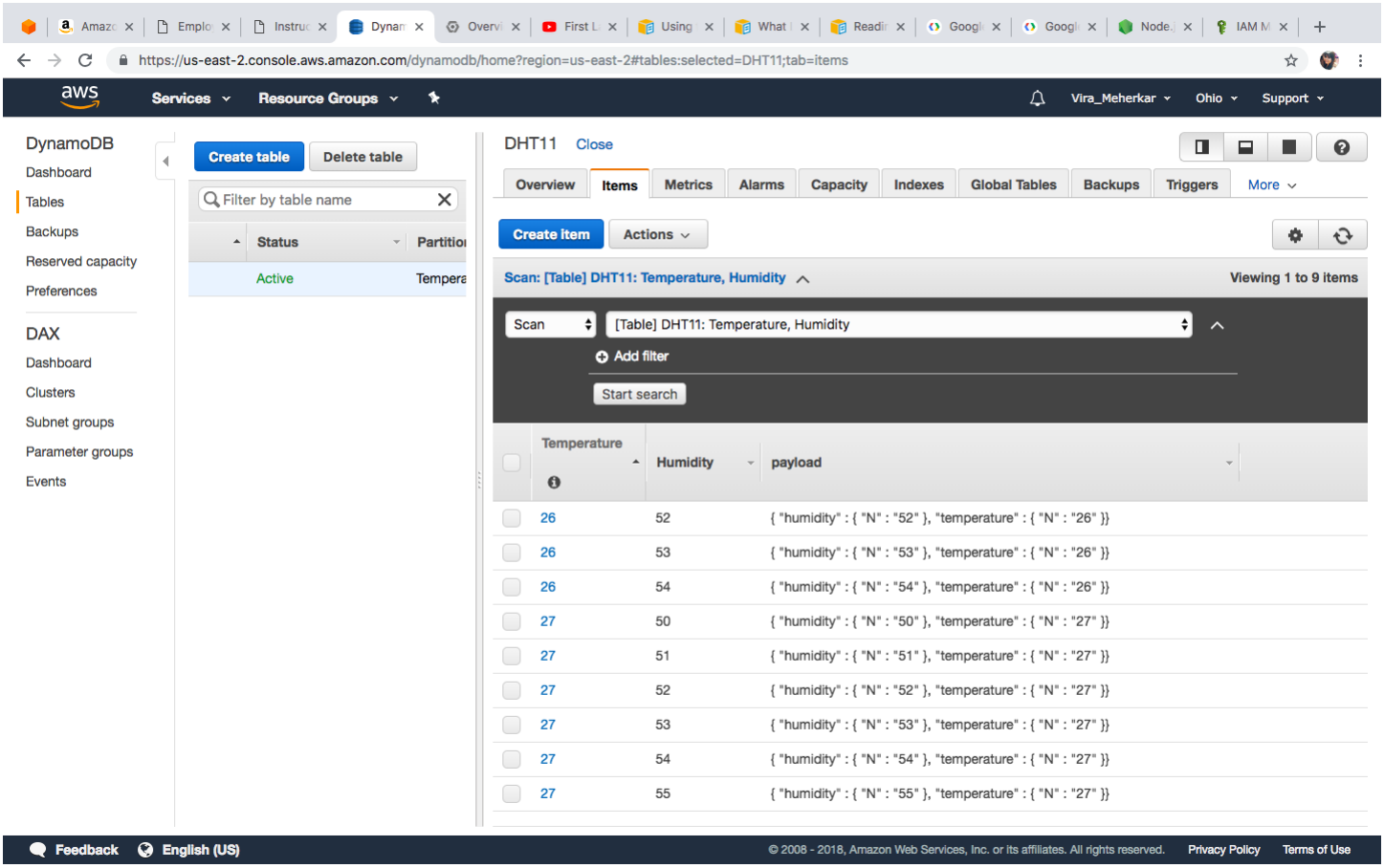
- To test the operation of publish and subscribe, we can go to Test section in the Side Bar and We can



type our IOT Thing name to subscribe and to publish.



- From the above screen shot we can see that the Values are same in Dynamo DB as the Value from the ESP32 DHT sensor read.
- Once we can see the data in the PubSub Test page, we head to the dynamo DB Table and From the Item



Tab we can see the data there on the table.

Q.5:- For processing real time data
Download and install required libraries for GPS to work in Arduino IDE
Connect Arduino/ESP32 with GPS sensor and store the latitude longitude along like we stored temperature and humidity in aws in DynamoDB table.
Write the code for the same and upload it to Arduino? ESP32 board and let it run.
You can build a variety of real-time data processing systems using AWS Lambda, Amazon Kinesis, Amazon S3, and Amazon DynamoDB. We will create a web page where we will use Google Map API key generated to embed the webpage using coordinates from DynamoDB table. And we will be able to show the data in real time then on Google Maps.

ESP32 CODE

```
#include <AWS_IOT.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include "esp_wpa2.h"
AWS_IOT hornbill;

#include "DHTesp.h"
#include "Ticker.h"
DHTesp dht;

//######################//
//UCF login credentials//
//######################//
const char* ssid ="UCF_WPA2";
const char* host = "arduino.php5.sk";
#define EAP_IDENTITY "virameherkar@knights.ucf.edu"
#define EAP_PASSWORD "Alpa20394#"
#define EAP_USERNAME "vi975487"

//############################//
// UCF SERVER CERTIFICATE  //
//############################//
const char* UCF_server_ca_cert = "-----BEGIN CERTIFICATE-----\n"
"MIIF+TCCA+GgAwIBAgIQRyDQ+oVGGn4XoWQCkYRjdDANBgkqhkiG9w0BAQwFADCB\n"
"iDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCk5ldyBKZXJzZXkxFDASBgNVBAcTC0pl\n"
"cnNleSBDaXR5MR4wHAYDVQQKExVUaGUgVVNFUlRSVVNUIE5ldHdvcmsxLjAsBgNV\n"
"BAMTJVVTRVJUcnVzdCBSU0EgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkwHhcNMTQx\n"
"MDA2MDAwMDAwWhcNMjQxMDA1MjM1OTU5WjB2MQswCQYDVQQGEwJVUzELMAkGA1UE\n"
"CBMCTUkxEjAQBgNVBAcTCUFubiBBcmJvcjESMBAGA1UEChMJSW50ZXJuZXQyMREw\n"
"DwYDVQQLEwhJbkNvbW1vbjEfMB0GA1UEAxMWSW5Db21tb24gUlNBIFNlcnZlciBD\n"
"QTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJwb8bsvf2MYFVFRVA+e\n"
"xU5NEFj6MJsXKZDmMwysE1N8VJG06thum4ltuzM+j9INpun5uukNDBqeso7JcC7v\n"
"HgV9lestjaKpTbOc5/MZNrun8XzmCB5hJ0R6lvSoNNviQsil2zfVtefkQnI/tBPP\n"
"iwckRR6MkYNGuQmm/BijBgLsNI0yZpUn6uGX6Ns1oytW61fo8BBZ321wDGZq0GTl\n"
"qKOYMa0dYtX6kuOaQ80tNfvZnjNbRX3EhigsZhLI2w8ZMA0/6fDqSl5AB8f2IHpT\n"
"eIFken5FahZv9JNYyWL7KSd9oX8hzudPR9aKVuDjZvjs3YncJowZaDuNi+L7RyML\n"
"fzcCAwEAAaOCAW4wggFqMB8GA1UdIwQYMBaAFFN5v1qqK0rPVIDh2JvAnfKyA2bL\n"
"MB0GA1UdDgQWBBQeBaN3j2yW4luHS6a0hqxxAAznODAOBgNVHQ8BAf8EBAMCAYYw\n"
"EgYDVR0TAQH/BAgwBgEB/wIBADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH\n"
"AwIwGwYDVR0gBBQwEjAGBgRVHSAAMAgGBmeBDAECAjBQBgNVHR8ESTBHMEWgQ6BB\n"
"hj9odHRwOi8vY3JsLnVzZXJ0cnVzdC5jb20vVVNFUlRydXN0UlNBQ2VydGlmaWNh\n"
"dGlvbkF1dGhvcml0eS5jcmwwdgYIKwYBBQUHAQEEajBoMD8GCCsGAQUFBzAChjNo\n"
"dHRwOi8vY3J0LnVzZXJ0cnVzdC5jb20vVVNFUlRydXN0UlNBQWRkVHJ1c3RDQS5j\n"
"cnQwJQYIKwYBBQUHMAGGGWh0dHA6Ly9vY3NwLnVzZXJ0cnVzdC5jb20wDQYJKoZI\n"
```

```
"hvcNAQEMBQADggIBAC0RBjjW29dYaK+qOGcXjeIT16MUJNkGE+vrkS/fT2ctyNMU\n"
"11ZlUp5uH5gIjppIG8GLWZqjV5vbhvhZQPwZsHURKsISNrqOcooGTie3jVgU0W+0\n"
"+Wj8mN2knCVANt69F2YrA394gbGAdJ5fOrQmL2pIhDY0jqco74fzYefbZ/VS29fR\n"
"5jBxu4uj1P+5ZImem4Gbj1e4ZEzVBhmO55GFfBjRidj26h1oFBHZ7heDH1Bjzw72\n"
"hipu47Gkyfr2NEx3KoCGMLCj3Btx7ASn5Ji8FoU+hCazwOU1VX55mKPU1I2250Lo\n"
"RCASN18JyfsD5PVldJbtyrmz9gn/TKbRXTr80U2q5JhyvjhLf4lOJo/UzL5WCXED\n"
"Smyj4jWG3R7Z8TED9xNNCxGBMXnMete+3PvzdhssvbORDwBZByogQ9xL2LUZFI/i\n"
"eoQp0UM/L8zfP527vWjEzuDN5xwxMnhi+vCToh7J159o5ah29mP+aJnvujbXEnGa\n"
"nrNxHzu+AGOePV8hwrGGG7hOIcPDQwkuYwzN/xT29iLp/cqf9ZhEtkGcQcIImH3b\n"
"oJ8ifsCnSbu0GB9L06Yqh7lcyvKDTEADslIaeSEINxhO2Y1fmcYFX/Fqrrp1WnhH\n"
"OjplXuXE0OPa0utaKC25Aplgom88L2Z8mEWcyfoB7zKOfD759AN7JKZWCYwk\n"
"-----END CERTIFICATE-----\n";

//########################//
//AWS Credentials//
//########################//
char HOST_ADDRESS[]="a2txpxlqpf0l2a-ats.iot.us-east-2.amazonaws.com";
char CLIENT_ID[]= "ViraPolicy";
char TOPIC_NAME[]= "Vira";


int status = WL_IDLE_STATUS;
int tick=0,msgCount=0,msgReceived = 0;
char payload[512];
char rcvdPayload[512];

void mySubCallBackHandler (char *topicName, int payloadLen, char *payLoad)
{
   strncpy(rcvdPayload,payLoad,payloadLen);
   rcvdPayload[payloadLen] = 0;
   msgReceived = 1;
}

void setup()
{
   Serial.begin(115200);
   int dhtPin = 17;
   dht.setup(dhtPin, DHTesp::DHT11);
   Serial.println();



//################################//
// WIFI CONNECTION STARTS HERE //
//################################//
  byte error = 0;
  Serial.begin(115200);
```

```
  delay(10);
  Serial.println("Connecting to: ");
  Serial.println(ssid);
  WiFi.disconnect(true);  //disconnect from wifi to set new wifi connection
  WiFi.mode(WIFI_STA);
  error += esp_wifi_sta_wpa2_ent_set_ca_cert((const unsigned char*)UCF_server_ca_cert,
strlen(UCF_server_ca_cert));
  error += esp_wifi_sta_wpa2_ent_set_identity((uint8_t *)EAP_USERNAME, strlen(EAP_USERNAME));
  error += esp_wifi_sta_wpa2_ent_set_username((uint8_t *)EAP_USERNAME, strlen(EAP_USERNAME));
  error += esp_wifi_sta_wpa2_ent_set_password((uint8_t *)EAP_PASSWORD, strlen(EAP_PASSWORD));
  if (error != 0)
  {
   Serial.println("Error setting WPA properties.");
  }
  WiFi.enableSTA(true);
  esp_wpa2_config_t config = WPA2_CONFIG_INIT_DEFAULT();
  if (esp_wifi_sta_wpa2_ent_enable(&config) != ESP_OK)
  {
   Serial.println("WPA2 Settings Not OK");
  }
  WiFi.begin(ssid); //connect to Eduroam function
  WiFi.setHostname("RandomHostname"); //set Hostname for your device - not neccesary
  while (WiFi.status() != WL_CONNECTED)
  {
   delay(5000);
   Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address set: ");
  Serial.println(WiFi.localIP()); //print LAN IP

//################################//
// WIFI CONNECTION ENDS HERE //
//################################//

//################################//
// AWS CONNECTION STARTS HERE //
//################################//

    if (hornbill.connect(HOST_ADDRESS,CLIENT_ID)== 0)
    {
       Serial.println("Connected to AWS");
       delay(1000);

       if(0==hornbill.subscribe(TOPIC_NAME,mySubCallBackHandler))
       {
```

```
      Serial.println("Subscribe Successfull");
    }
    else
    {
      Serial.println("Subscribe Failed, Check the Thing Name and Certificates");
      while(1);
    }
  }
  else
  {
    Serial.println("AWS connection failed, Check the HOST Address");
    while(1);
  }

  delay(2000);

}
//##############################//
// AWS CONNECTION ENDS HERE //
//##############################//
void loop()
{

//##############################//
// DHT DATA READ SATRTS HERE //
//##############################//
  delay(1000);
  TempAndHumidity newValues = dht.getTempAndHumidity();

  float heatIndex = dht.computeHeatIndex(newValues.temperature, newValues.humidity);
  float dewPoint = dht.computeDewPoint(newValues.temperature, newValues.humidity);

//############################//
// DHT DATA READ ENDS HERE //
//############################//

//######################################//
// AWS PUBLISH AND RECEIVE STARTS HERE //
//######################################//
  if(msgReceived == 1)
  {
    msgReceived = 0;
    Serial.print("Received Message:");
    Serial.println(rcvdPayload);
  }
```

```
   if(tick >= 5)   // publish to topic every 5seconds
   {
      tick=0;
      StaticJsonBuffer<300> JSONbuffer;
      JsonObject& JSONencoder = JSONbuffer.createObject();
      JSONencoder["temperature"] = newValues.temperature;
      JSONencoder["humidity"] = newValues.humidity;
      Serial.println("temperature" + String(newValues.temperature) +  "Humidity" +
String(newValues.humidity));


      char JSONmessageBuffer[100];
      JSONencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));
      Serial.println(JSONmessageBuffer);


      if(hornbill.publish(TOPIC_NAME,JSONmessageBuffer) == 0)
      {
         Serial.print("Publish Message:");
         Serial.println(payload);
      }
      else
      {
         Serial.println("Publish failed");
      }

   }
   vTaskDelay(1000 / portTICK_RATE_MS);
   tick++;
}


————————————————————————————-
```