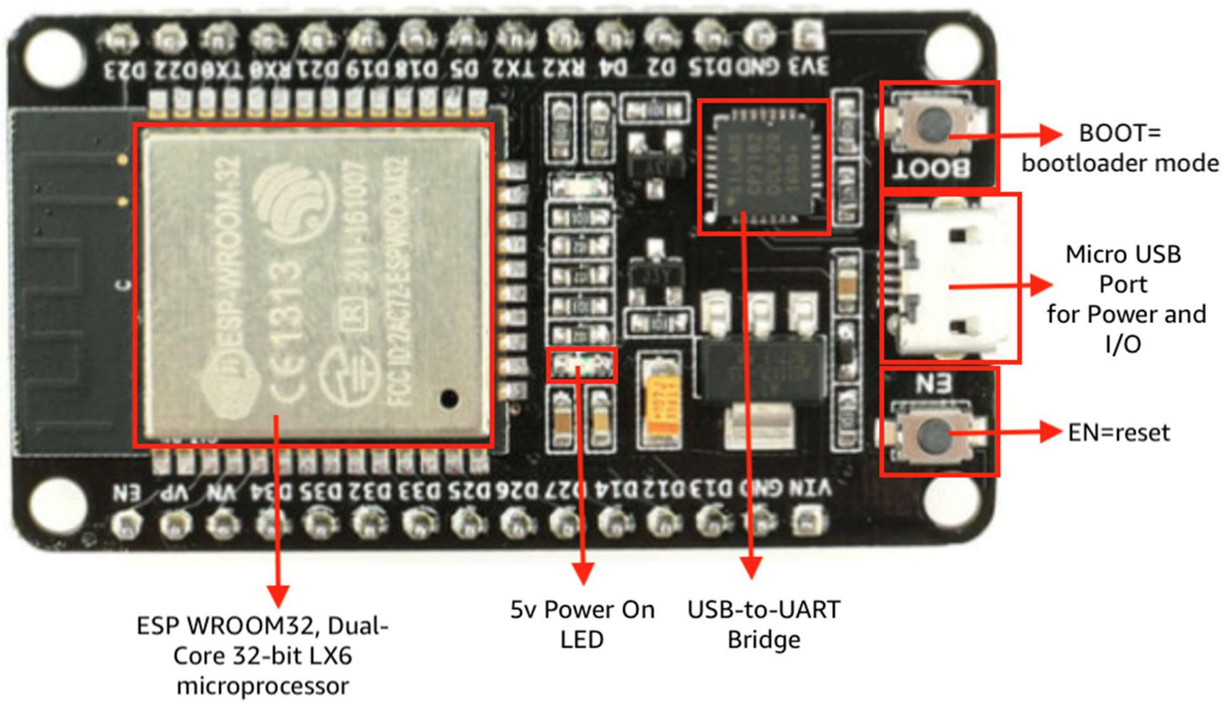
Streaming IOT DATA to DynamoDB

ESP32:

ESP32 is a low-cost, low-power Microcontroller with an integrated Wi-Fi and Bluetooth. It is the successor to the ESP8266


BOOT= bootloader mode
Micro USB Port for Power and I/O
EN=reset
ESP WROOM32, Dual-Core 32-bit LX6 microprocessor
5v Power On LED
USB-to-UART Bridge

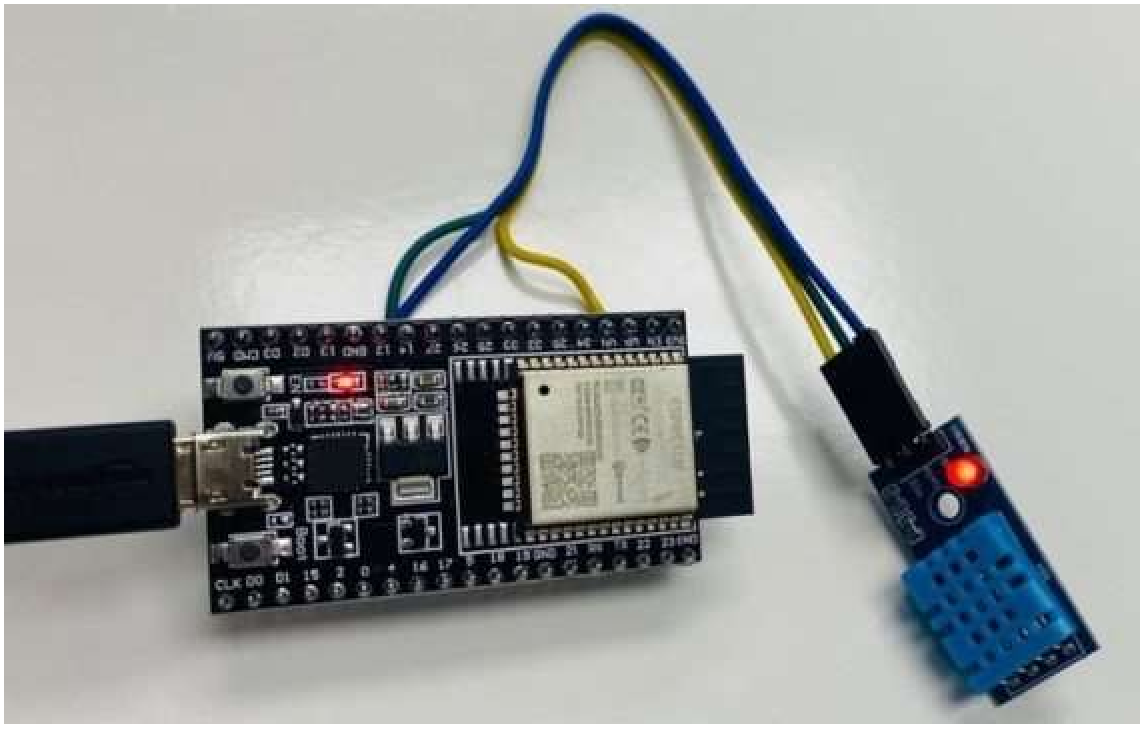
Prerequisites:

- ⇨ AWS Free tier account. AWS services.(IOT Core, DynamoDB)
- ⇨ ESP32 Controller
- ⇨ DHT11
- ⇨ Python/C ++
- ⇨ Arduino IDE

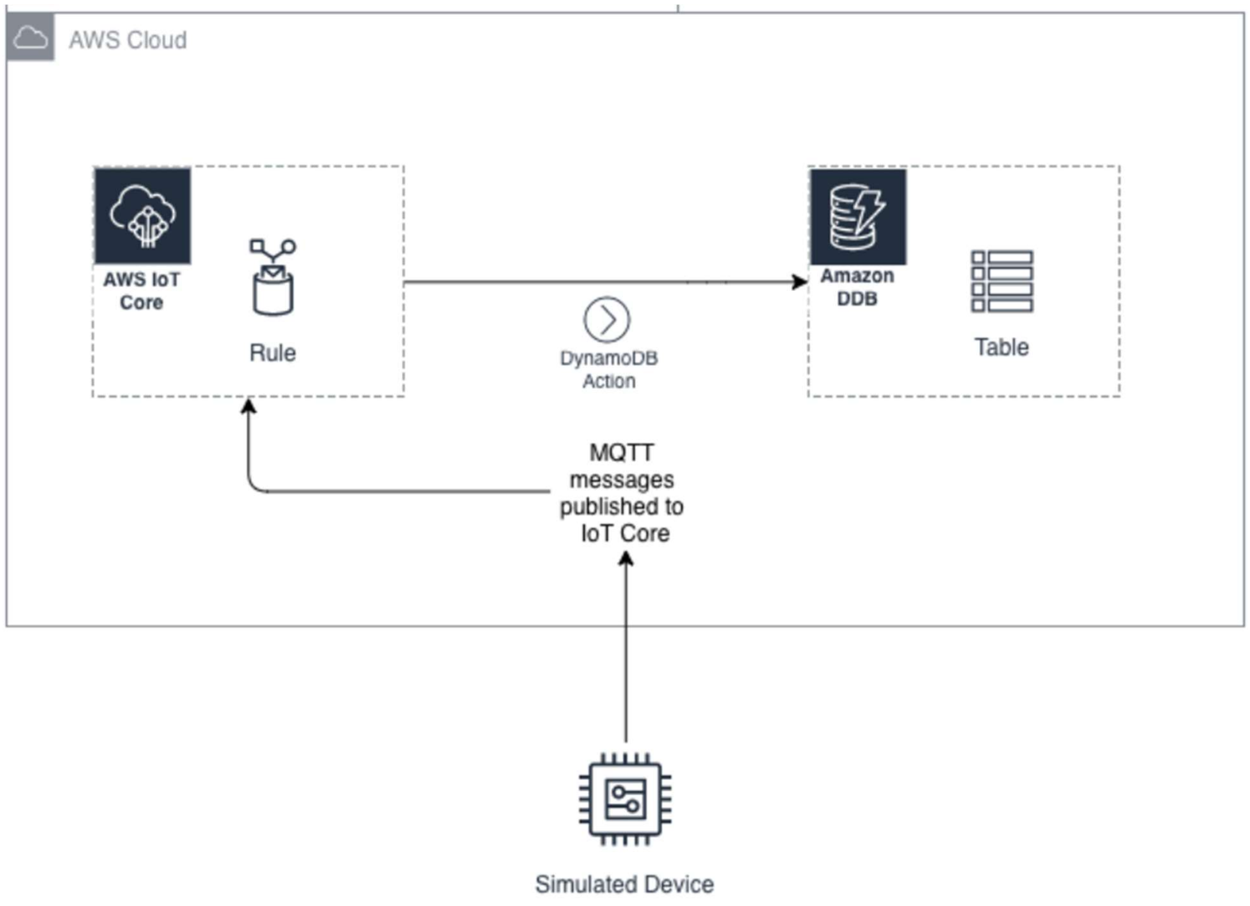
The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code. It connects to the Arduino hardware to upload programs and communicate with them.

Connection

DHT11	ESP32
+	3.3V
-	GND
Data(Middle Port)	D4 or any one (mentioned in code)

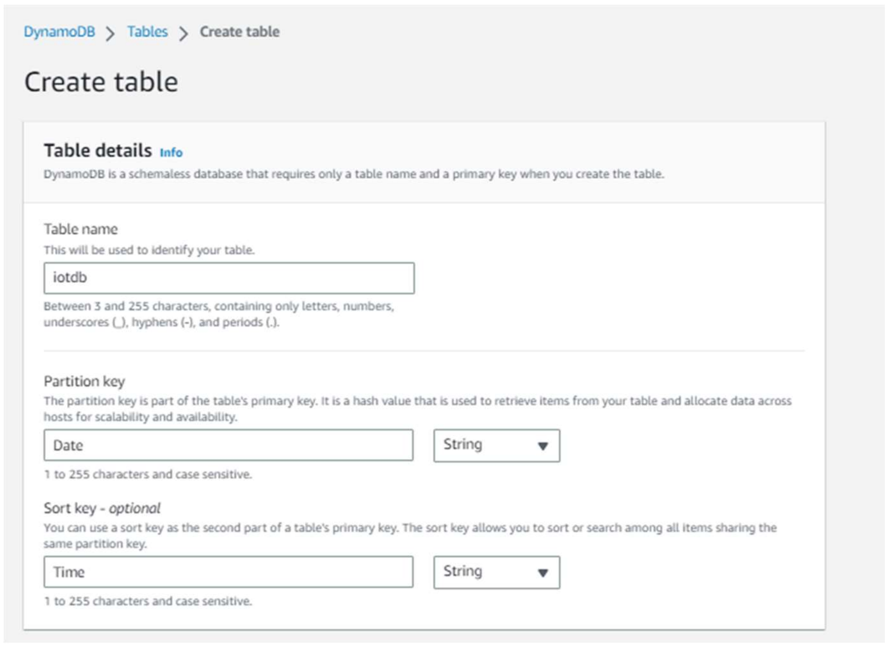


Architecture Diagram:



Task: Read the Temperature and humidity readings from DHT11 sensor and push it to AWS IOT core to and rule will then route it to the dynamo DB.

Create a DynamoDB table.



DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

iotdb

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

Date | String

1 to 255 characters and case sensitive.

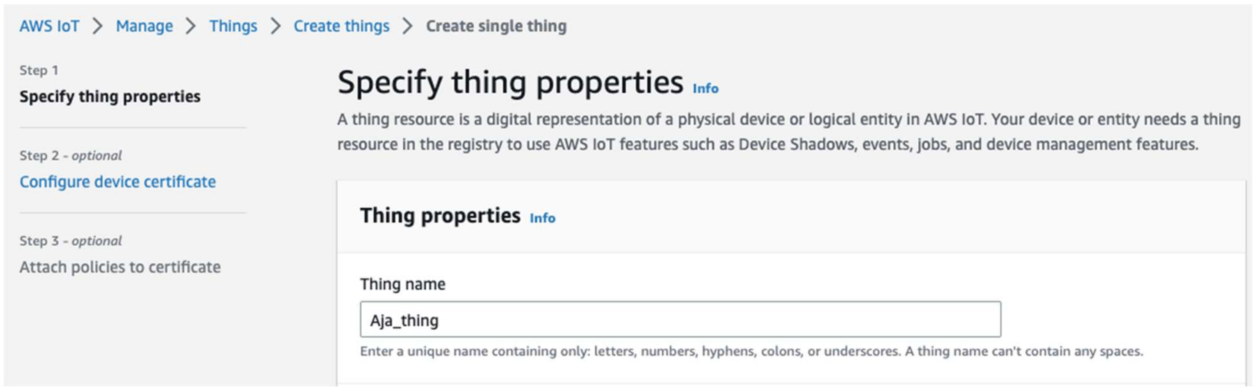
Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Time | String

1 to 255 characters and case sensitive.

Create a Thing in the AWS IOT Core.



AWS IoT > Manage > Things > Create things > Create single thing

Step 1
Specify thing properties

Step 2 - *optional*
Configure device certificate

Step 3 - *optional*
Attach policies to certificate

Specify thing properties Info

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties Info

Thing name

Aja_thing

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Create a policy that accepts all the topics(*), Create topic, publish topic, Subscribe topic.

End of thing creation. Download the certificates.

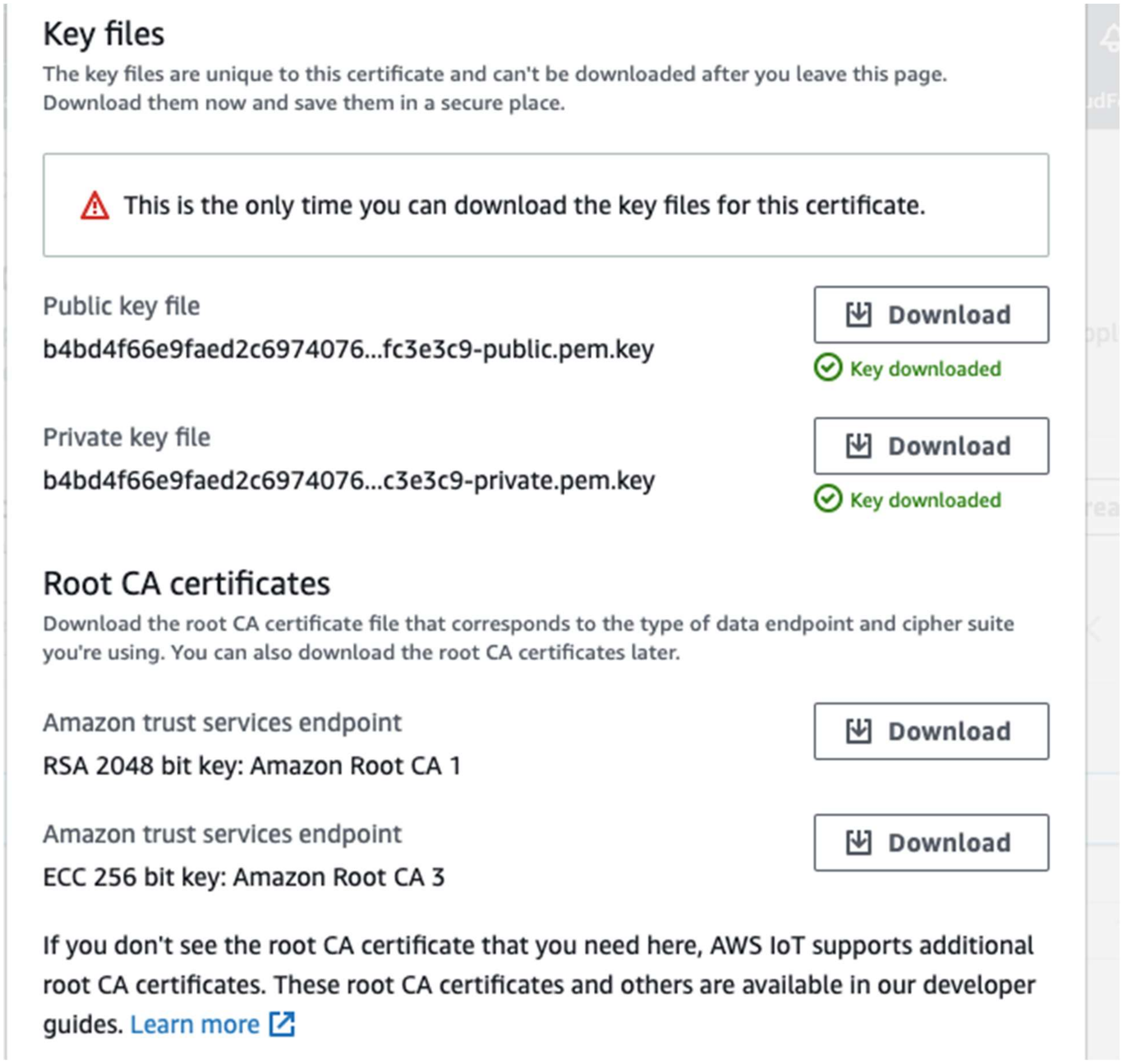
Details you needed for IOT device to publish message from code .

Device Data Endpoint: a3tdsfddfdfdfdf-ats.iot.us-east-1.amazonaws.com (AWS IOT → Settings)

Thing Name: XYZ

ESP32 Needs your internet user name and password. It gets a private ip from the modem .

Certificates



Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file
b4bd4f66e9faed2c6974076...fc3e3c9-public.pem.key

Private key file
b4bd4f66e9faed2c6974076...c3e3c9-private.pem.key

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

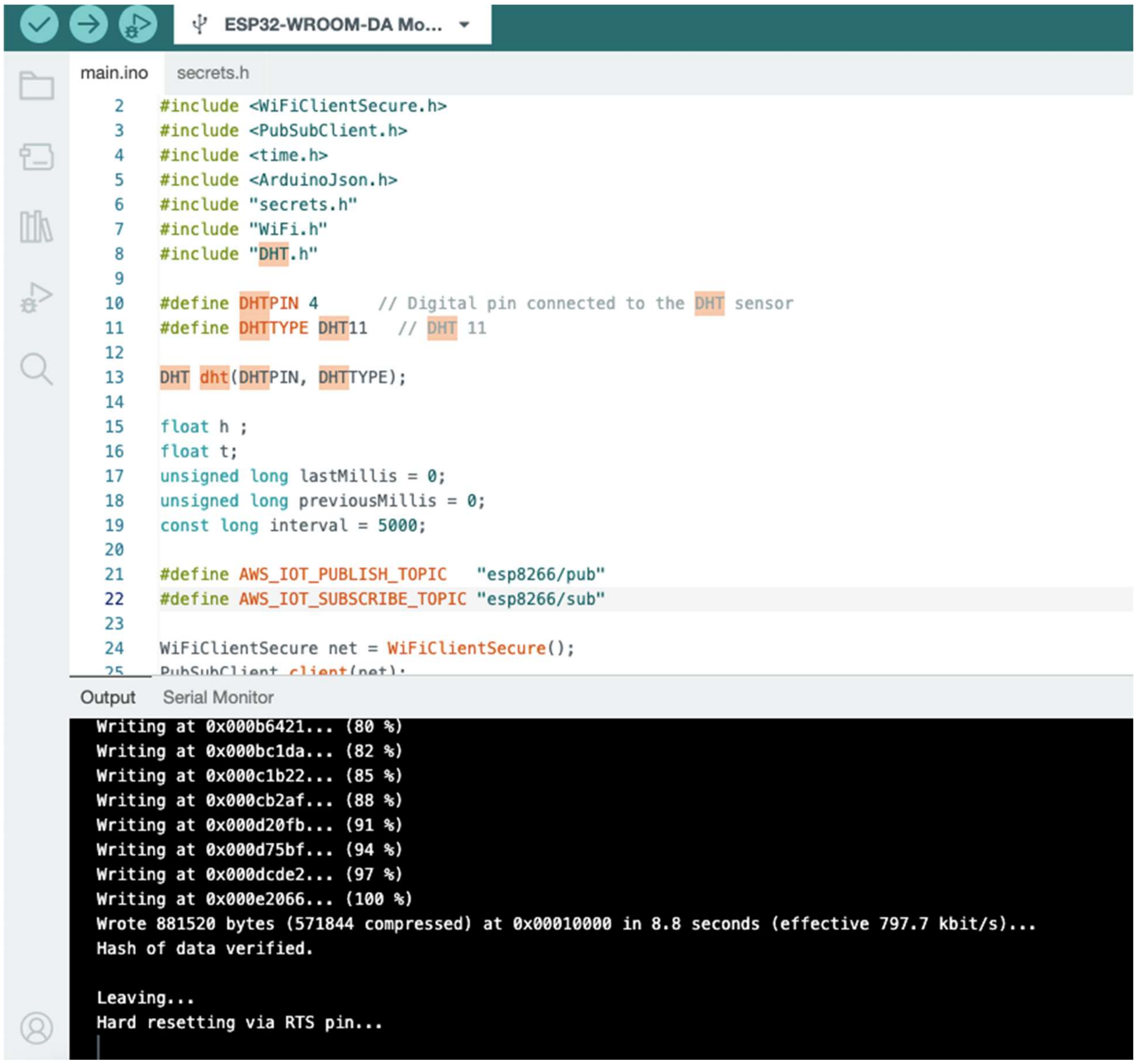
Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more

Publish the code to ESP32.

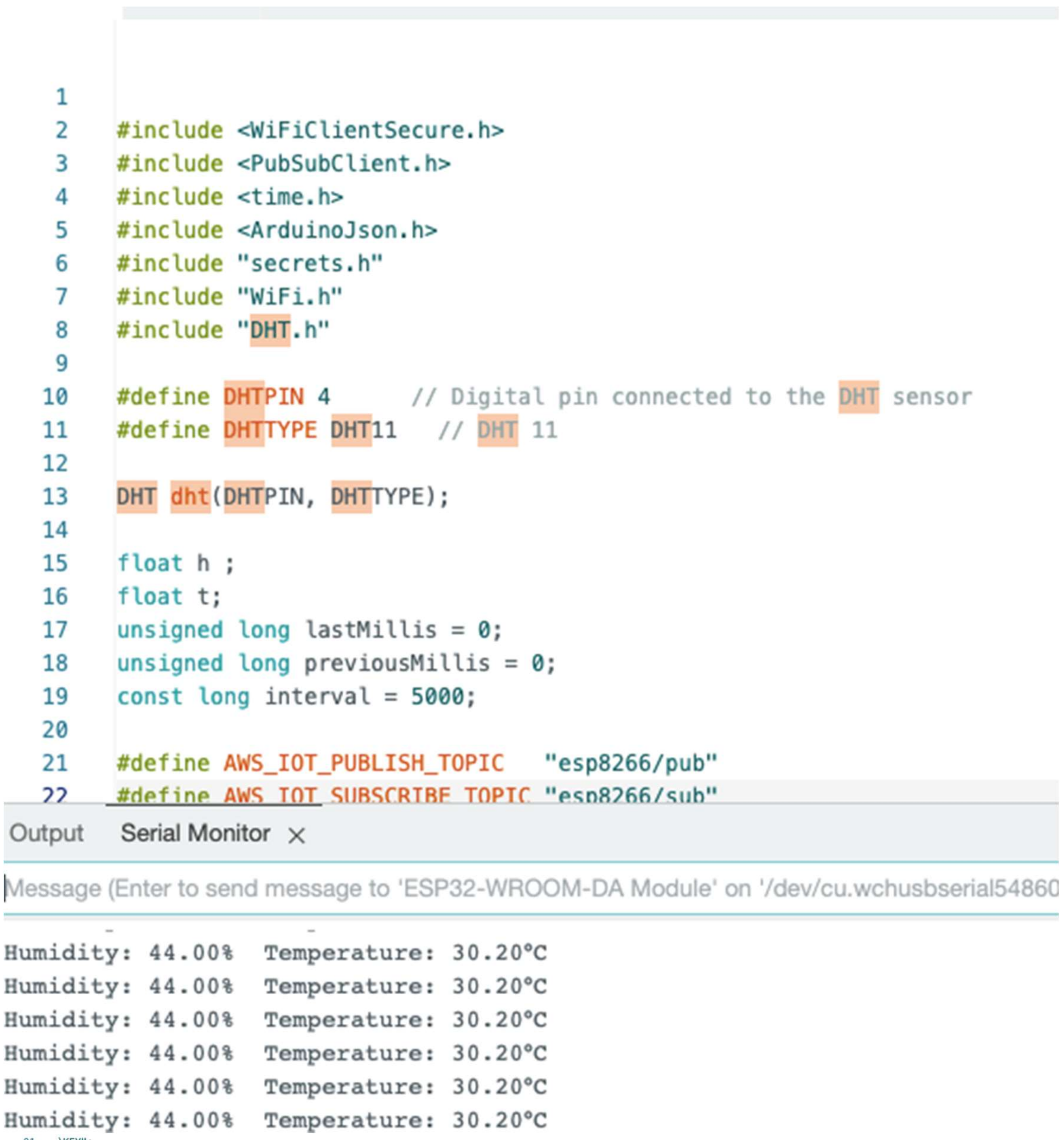
Look, I preferred to publish to the data to “**esp8266/pub**”. You can mention any name.



```
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <time.h>
#include <ArduinoJson.h>
#include "secrets.h"
#include "WiFi.h"
#include "DHT.h"

#define DHTPIN 4      // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

float h ;
float t;
unsigned long lastMillis = 0;
unsigned long previousMillis = 0;
const long interval = 5000;

#define AWS_IOT_PUBLISH_TOPIC   "esp8266/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"

WiFiClientSecure net = WiFiClientSecure();
PubSubClient client(net);
```

```
Writing at 0x000b6421... (80 %)
Writing at 0x000bc1da... (82 %)
Writing at 0x000c1b22... (85 %)
Writing at 0x000cb2af... (88 %)
Writing at 0x000d20fb... (91 %)
Writing at 0x000d75bf... (94 %)
Writing at 0x000dcde2... (97 %)
Writing at 0x000e2066... (100 %)
Wrote 881520 bytes (571844 compressed) at 0x00010000 in 8.8 seconds (effective 797.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

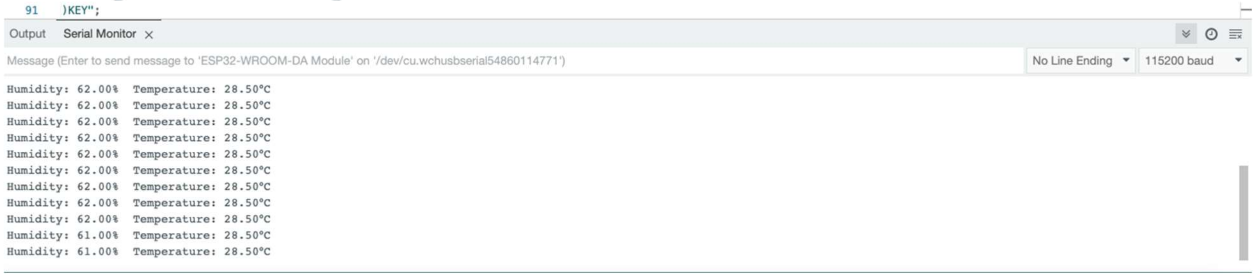
Once Code is pushed to the device. Device start sending the Temperature and Humidity Data.



```
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <time.h>
#include <ArduinoJson.h>
#include "secrets.h"
#include "WiFi.h"
#include "DHT.h"

#define DHTPIN 4       // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

float h ;
float t;
unsigned long lastMillis = 0;
unsigned long previousMillis = 0;
const long interval = 5000;

#define AWS_IOT_PUBLISH_TOPIC   "esp8266/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp8266/sub"
```

Output
Serial Monitor ×
Message (Enter to send message to 'ESP32-WROOM-DA Module' on '/dev/cu.wchusbserial54860

```
Humidity: 44.00%  Temperature: 30.20°C
Humidity: 44.00%  Temperature: 30.20°C
Humidity: 44.00%  Temperature: 30.20°C
Humidity: 44.00%  Temperature: 30.20°C
Humidity: 44.00%  Temperature: 30.20°C
Humidity: 44.00%  Temperature: 30.20°C
```



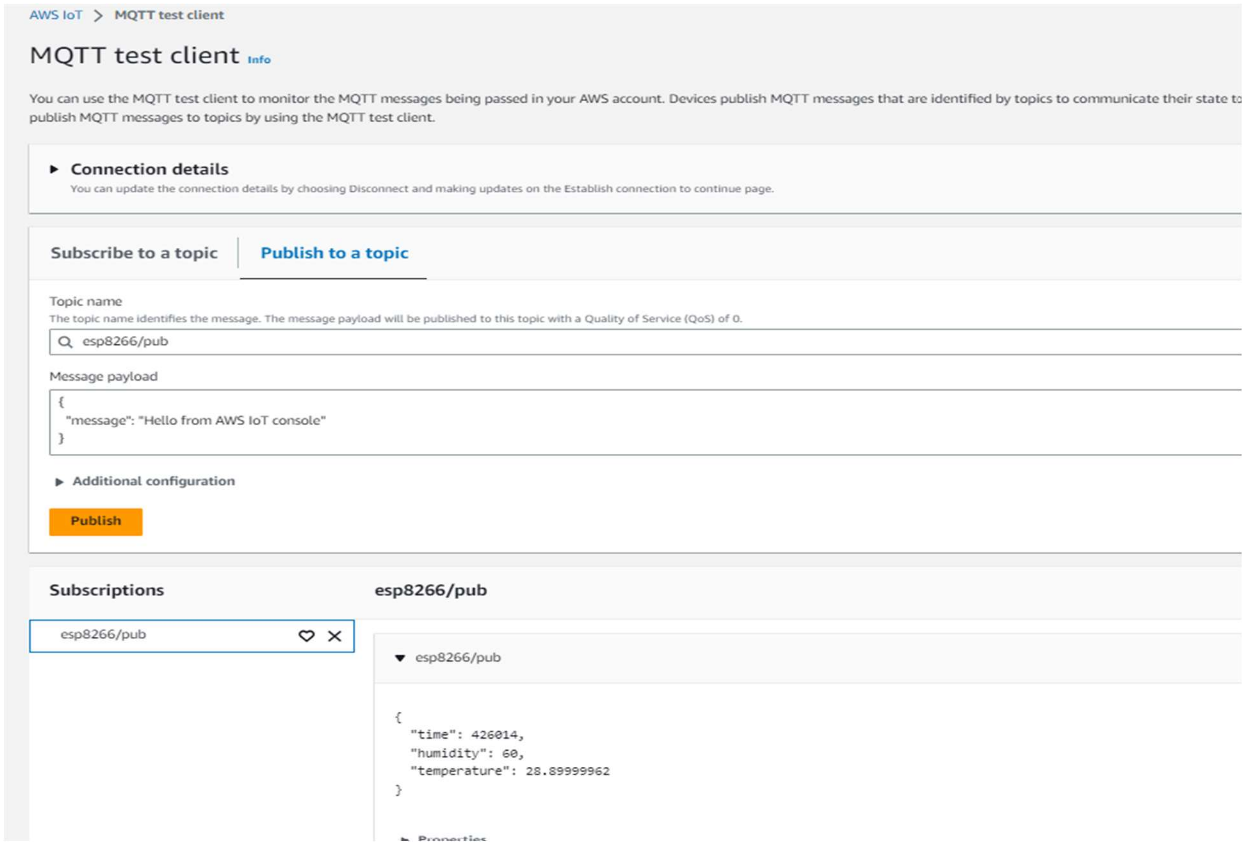
```
)KEY";
```

Output
Serial Monitor ×
Message (Enter to send message to 'ESP32-WROOM-DA Module' on '/dev/cu.wchusbserial54860114771')
No Line Ending
115200 baud

```
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 62.00%  Temperature: 28.50°C
Humidity: 61.00%  Temperature: 28.50°C
Humidity: 61.00%  Temperature: 28.50°C
```

Source code is available here. https://github.com/ThulasiKandhati/ESP32-DHT11-AWS

You can also find the data in aws . Query with the topic.



MQTT test client Info

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to
publish MQTT messages to topics by using the MQTT test client.

▶ Connection details
You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.
esp8266/pub

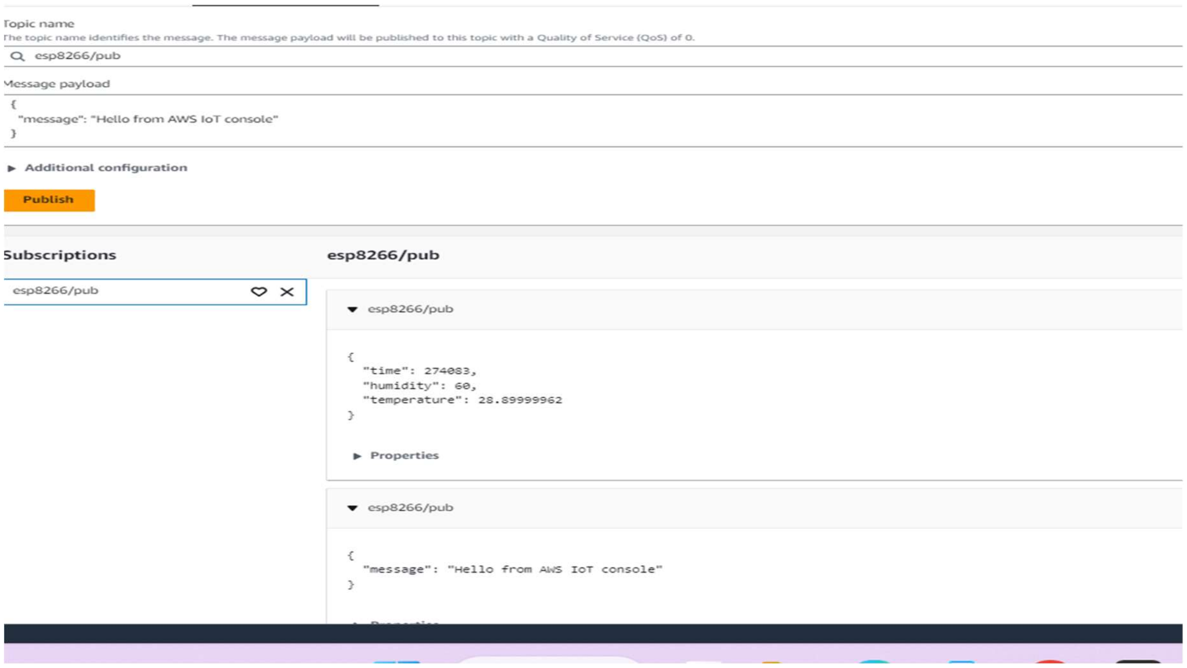
Message payload
```
{
 "message": "Hello from AWS IoT console"
}
```

esp8266/pub

esp8266/pub

▼ esp8266/pub
```
{
  "time": 426014,
  "humidity": 60,
  "temperature": 28.89999962
}
```

You can also send messages to the device.


Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.
esp8266/pub

Message payload
```
{
 "message": "Hello from AWS IoT console"
}
```

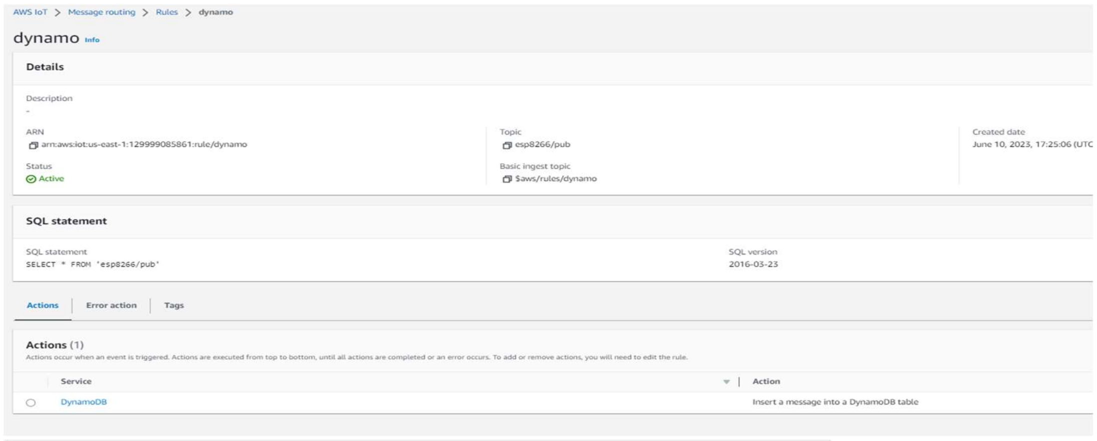
esp8266/pub

esp8266/pub

▼ esp8266/pub
```
{
  "time": 274083,
  "humidity": 60,
  "temperature": 28.89999962
}
```

▼ esp8266/pub
```
{
  "message": "Hello from AWS IoT console"
}
```

Create a rule to publish data to DynamoDB table.



AWS IoT > Message routing > Rules > dynamo

dynamo Info

Details

Description -		
ARN arn:aws:iot:us-east-1:129999085861:rule/dynamo	Topic esp8266/pub	Created date June 10, 2023, 17:25:06 (UTC
Status Active	Basic ingest topic $aws/rules/dynamo	

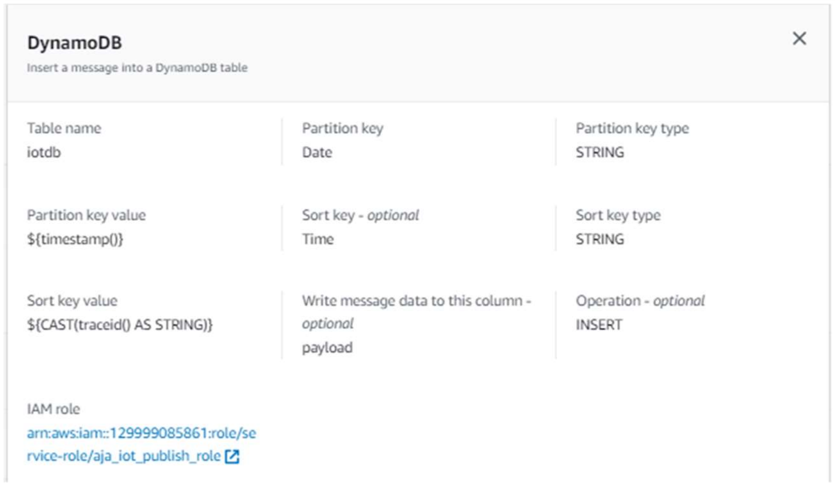
SQL statement

SQL statement
```
SELECT * FROM 'esp8266/pub'
```
SQL version
2016-03-23

Actions | Error action | Tags

Actions (1)
Actions occur when an event is triggered. Actions are executed from top to bottom, until all actions are completed or an error occurs. To add or remove actions, you will need to edit the rule.

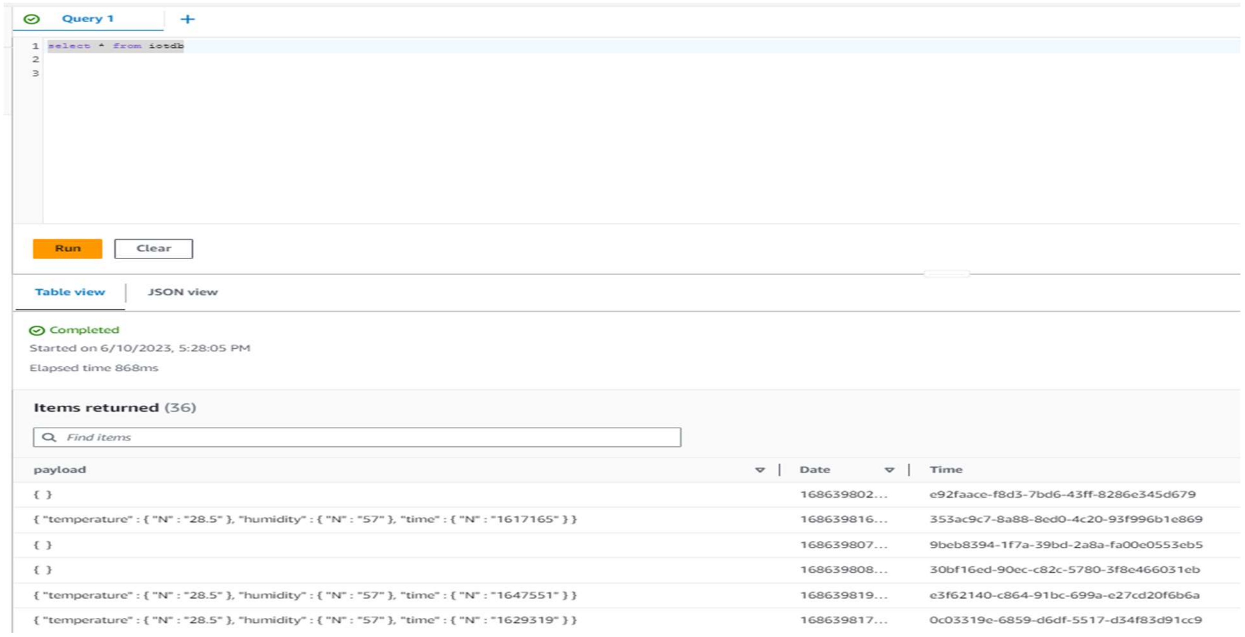
Service	Action
DynamoDB	Insert a message into a DynamoDB table



DynamoDB
Insert a message into a DynamoDB table

Table name iotdb	Partition key Date	Partition key type STRING
Partition key value ${timestamp()}	Sort key - *optional* Time	Sort key type STRING
Sort key value ${CAST(traceid() AS STRING)}	Write message data to this column - *optional* payload	Operation - *optional* INSERT
IAM role arn:aws:iam::129999085861:role/service-role/aja_iot_publish_role		

Query data from Dynamodb.



Query 1

```
select * from iotdb
```

Run Clear

Table view | JSON view

Completed
Started on 6/10/2023, 5:28:05 PM
Elapsed time 868ms

Items returned (36)

Find items

payload	Date	Time
{ }	168639802...	e92faace-f8d3-7bd6-43ff-8286e345d679
{ "temperature" : { "N" : "28.5" }, "humidity" : { "N" : "57" }, "time" : { "N" : "1617165" } }	168639816...	353ac9c7-8a88-8ed0-4c20-93f996b1e869
{ }	168639807...	9beb8394-1f7a-39bd-2a8a-fa00e0553eb5
{ }	168639808...	30bf16ed-90ec-c82c-5780-3f8e466031eb
{ "temperature" : { "N" : "28.5" }, "humidity" : { "N" : "57" }, "time" : { "N" : "1647551" } }	168639819...	e3f62140-c864-91bc-699a-e27cd20f6b6a
{ "temperature" : { "N" : "28.5" }, "humidity" : { "N" : "57" }, "time" : { "N" : "1629319" } }	168639817...	0c03319e-6859-d6df-5517-d34f83d91cc9