LABORATORY OF INTERNET OF THINGS

PREPARED FOR

Roberto Di Prisco

Professor

Department of Computer Science

PREPARED BY

Muhammad Tanveer Jan

0522500631

SUMMARY

This application is a simple implementation of connecting your Arduino with cloud for storing of data, although this is only the connection between these two things and many further improvements can be made.

I use Arduino MKR 1010 version because of its specifications and large memory than other Arduino boards.

The Arduino is connected to a wireless connection and then try to connect to “the Internet of Things Core” of Amazon Web Service. Authentication of the client is made by providing several parameters, which will be shown later, to AWS-IoT. The data is received in the AWS-IoT core and can be monitored on it. The next step is saving the data into storage, for which we use the NoSQL database called “DynamoDB”.

1. Project Overview

Simple project based on the Arduino board to show the connection to AWS-IoT Core with a wireless connection.

2. Obstacles

Creating the CertificateSigningRequest (CSR) is the main problem as few as it can be generated and applied to Arduino 4 times. Now this isn't a problem is the device is static but with mobile devices, this can be a problem.

3. Hardware

The following pieces of equipment are used to implement this project.

1. Arduino MKR 1010 WiFI
2. Jump Wires
3. DHT22 - Temperature & humidity Sensor
4. BH1750 - Ambient Light Sensor
5. Breadboard

Caution: Please be careful with power as this project was connected to a USB port of Laptop, if you want to connect this directly to a power source then suitable resistors should be used

7. Software

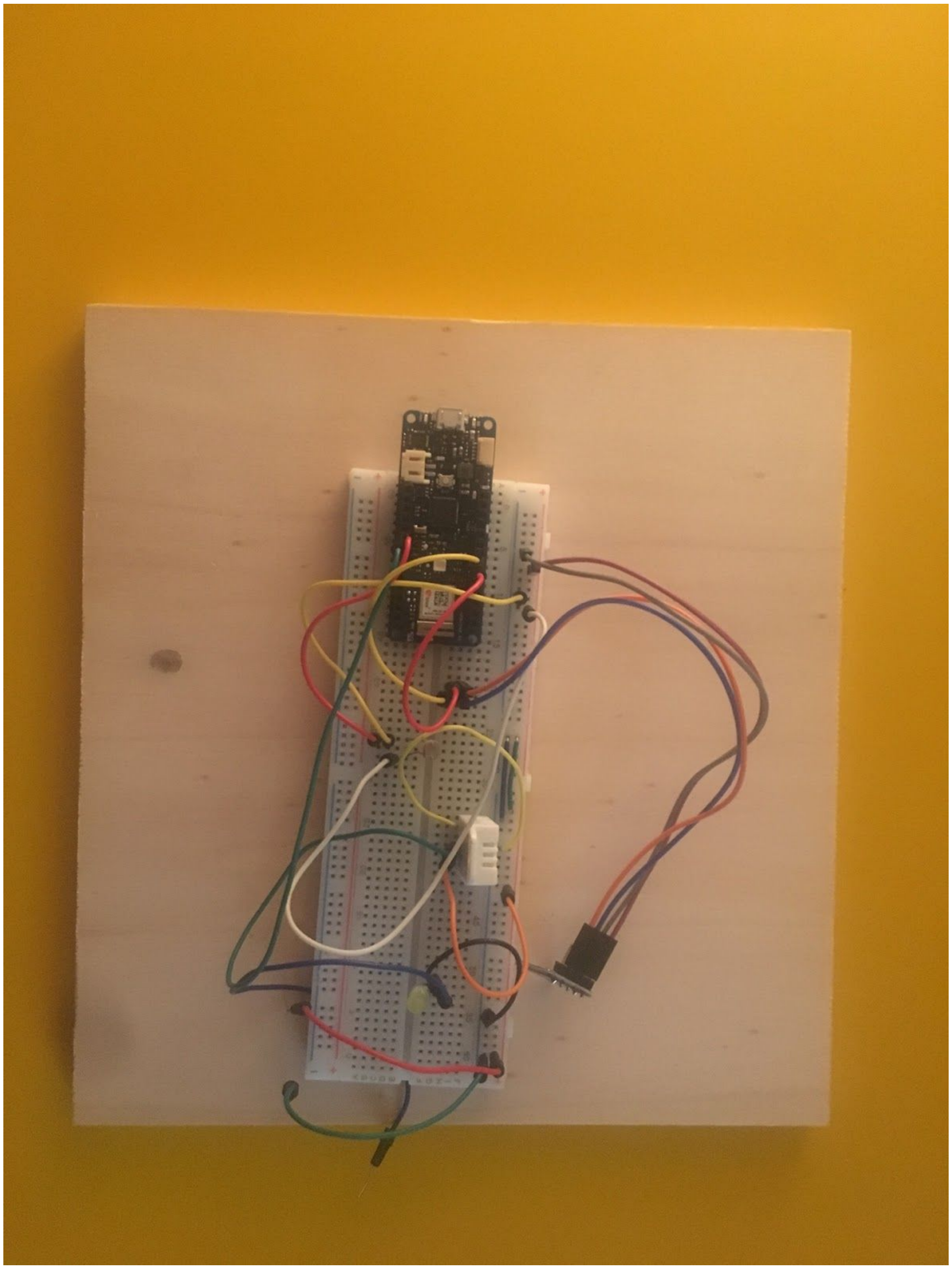
Arduino IDE

AWS-IoT Core- (You need to have amazon web services account which is paid and can't get free, but lucky you amazon provide a few working hours to students, so you should create amazon student account here)

8. Walkthrough of the Process

I will guide you through the process of creating this application step-by-step.

First lets wire all the equipment with Arduino. The following image shows all the connections



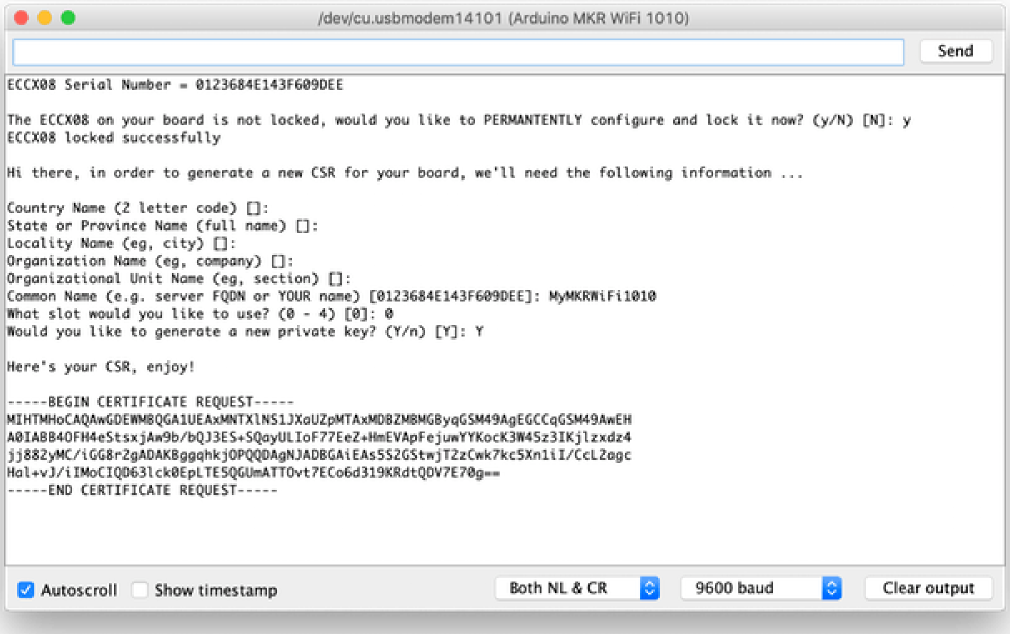
That’s all with the hardware, now let’s jump into the

The first thing is generating the CSR which will help us authorize the device into AWS-IoT Core.

1. Generating CSR

You need to install the ArduinoECCX08 library in your ArduinoIDE, which can be done by going to menu bar of ArduinoIDE > Sketch > Include Library > Manage Libraries.

Then search for the ArduinoECCX08 in the search bar. Install the library and restart the ArduinoIDE.



```
ECCX08 Serial Number = 0123684E143F609DEE

The ECCX08 on your board is not locked, would you like to PERMANTENTLY configure and lock it now? (y/N) [N]: y
ECCX08 locked successfully

Hi there, in order to generate a new CSR for your board, we'll need the following information ...

Country Name (2 letter code) []:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) [0123684E143F609DEE]: MyMKRWiFi1010
What slot would you like to use? (0 - 4) [0]: 0
Would you like to generate a new private key? (Y/n) [Y]: Y

Here's your CSR, enjoy!

-----BEGIN CERTIFICATE REQUEST-----
MIHTMHoCAQAwGDEWMBQGA1UEAxMNTXlNS1JXaUZpMTAxMDBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABB4OFH4eStsxjAw9b/bQJ3ES+SQayULIoF77EeZ+HmEVApFejuwYYKocK3W45z3IKjlzxdz4
jj88ZyMC/iGG8r2gADAKBggqhkjOPQQDAgNJADBGAiEAs5S2GStwjTZzCwk7kc5Xn1iI/CcL2agc
Hal+vJ/iIMoCIQD63lck0EpLTE5QGUmATTOvt7ECo6d319KRdtQDV7E70g==
-----END CERTIFICATE REQUEST-----
```

After restarting go File > Examples > ArduinoECCX08 > Tools > ECCX08CSR

Upload the sketch to your board and then open the serial monitor. It will ask for some fields, all of them should be left empty except name of the device and number.

Copy the generated CSR and save it in .txt file, which will be uploaded to AWS-IoT.

2. Code the Projects

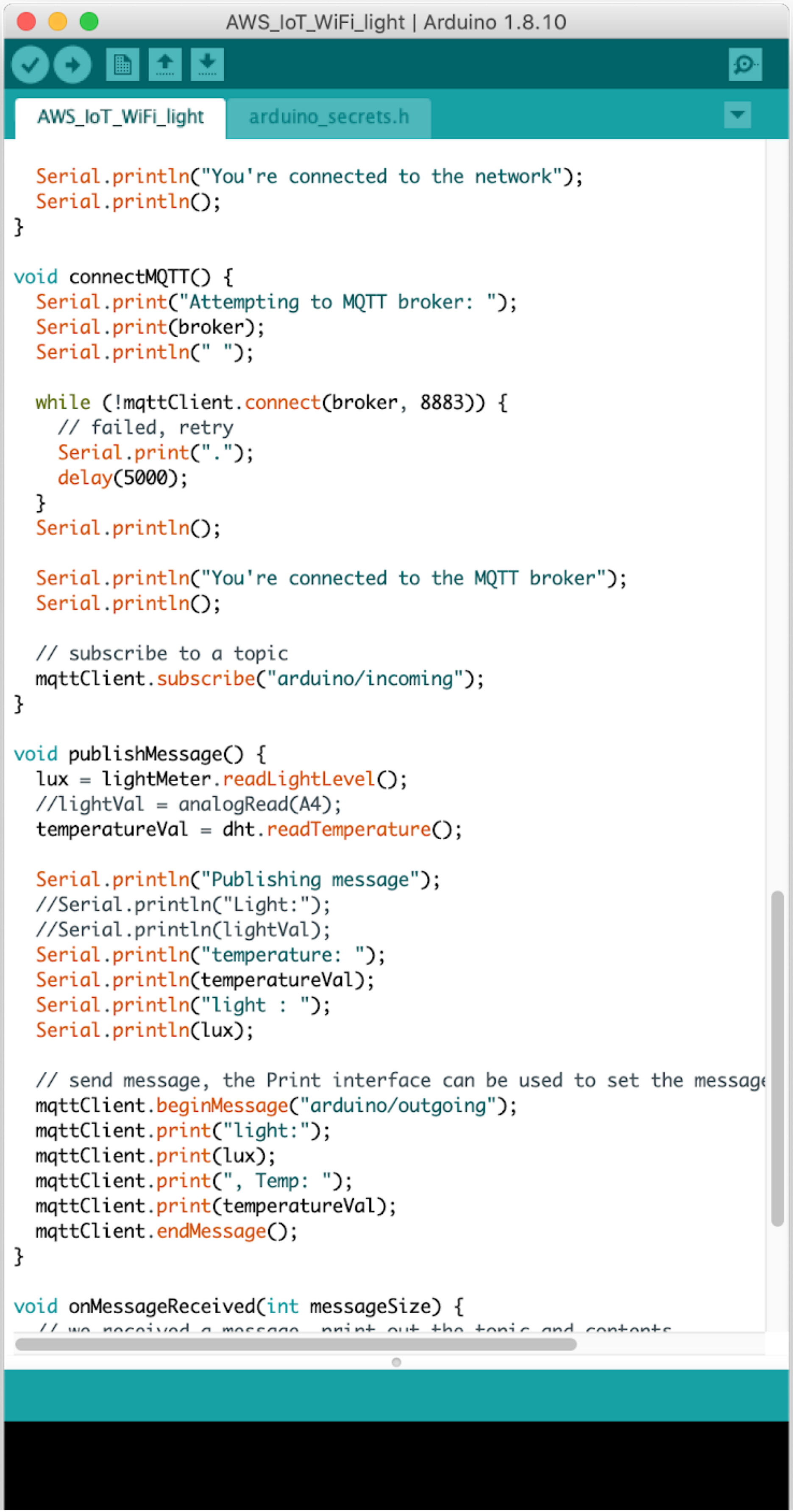
The following is some screenshot of the code but it can be found on the repository

Here is the link to the repository if you don't have already

https://github.com/tjan90/ArduinoMKR1010-AWSIoT.git

Also, remember to change the wireless connection credential in the arduino_secrets.h

You also need to install libraries that are mentioned in the heading of the code.

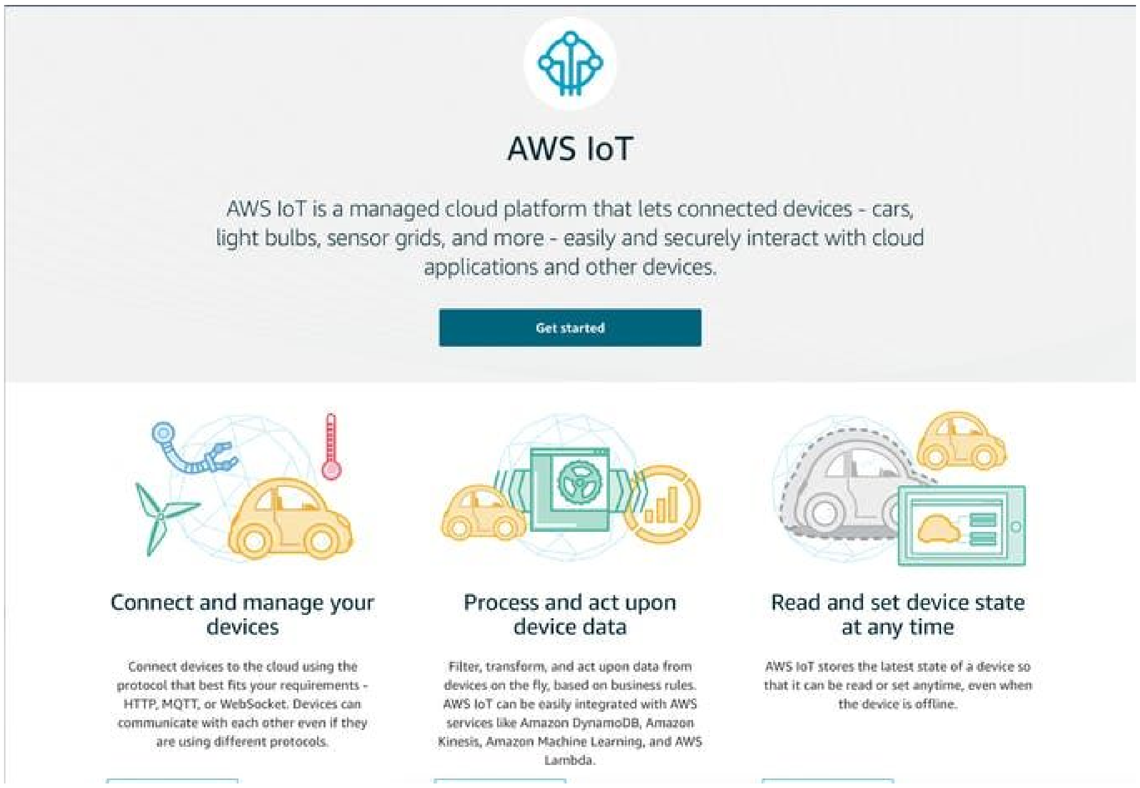


```
  Serial.println("You're connected to the network");
  Serial.println();
}

void connectMQTT() {
  Serial.print("Attempting to MQTT broker: ");
  Serial.print(broker);
  Serial.println(" ");

  while (!mqttClient.connect(broker, 8883)) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }
  Serial.println();

  Serial.println("You're connected to the MQTT broker");
  Serial.println();

  // subscribe to a topic
  mqttClient.subscribe("arduino/incoming");
}

void publishMessage() {
  lux = lightMeter.readLightLevel();
  //lightVal = analogRead(A4);
  temperatureVal = dht.readTemperature();

  Serial.println("Publishing message");
  //Serial.println("Light:");
  //Serial.println(lightVal);
  Serial.println("temperature: ");
  Serial.println(temperatureVal);
  Serial.println("light : ");
  Serial.println(lux);

  // send message, the Print interface can be used to set the message
  mqttClient.beginMessage("arduino/outgoing");
  mqttClient.print("light:");
  mqttClient.print(lux);
  mqttClient.print(", Temp: ");
  mqttClient.print(temperatureVal);
  mqttClient.endMessage();
}

void onMessageReceived(int messageSize) {
```

3. **Configuring AWS-IoT**

Sign in to your aws account go to aws console. Search for IoT Core and open the service.

You will then be presented with a welcome page if this is your first time using IoT Core. Click the "Get started" button to continue



AWS IoT

AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices.

Connect and manage your devices

Connect devices to the cloud using the protocol that best fits your requirements - HTTP, MQTT, or WebSocket. Devices can communicate with each other even if they are using different protocols.

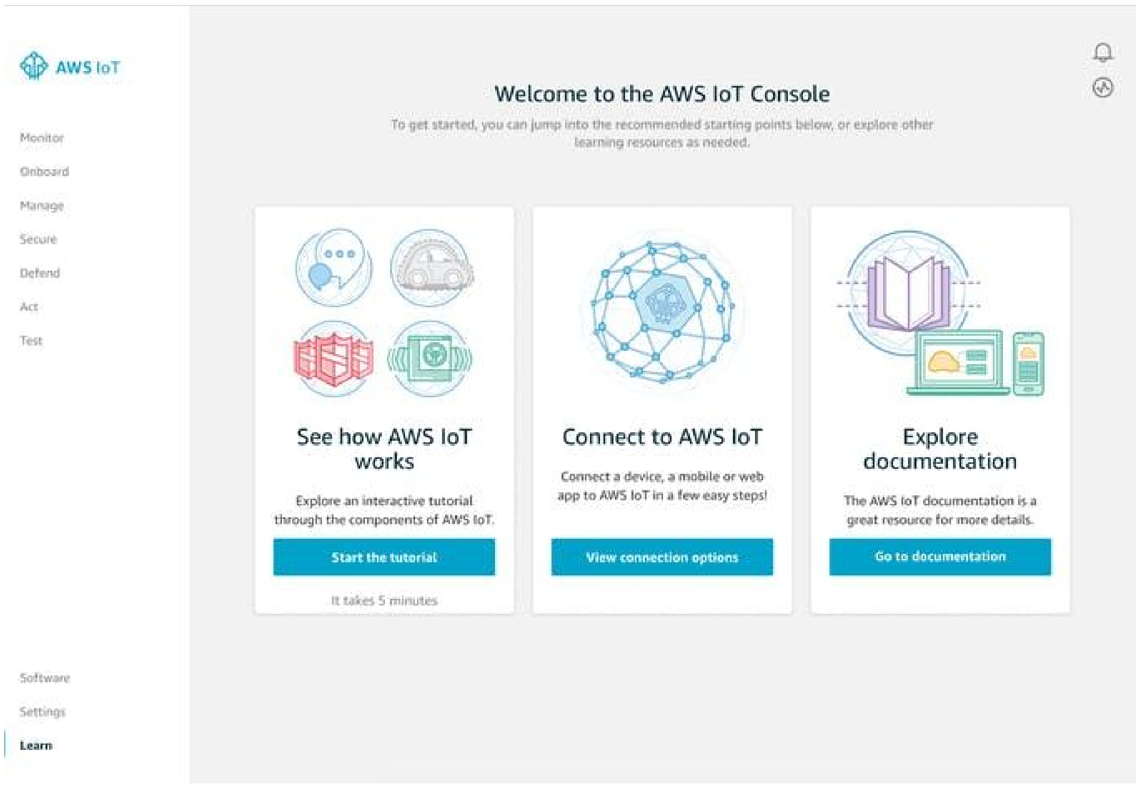
Process and act upon device data

Filter, transform, and act upon data from devices on the fly, based on business rules. AWS IoT can be easily integrated with AWS services like Amazon DynamoDB, Amazon Kinesis, Amazon Machine Learning, and AWS Lambda.

Read and set device state at any time

AWS IoT stores the latest state of a device so that it can be read or set anytime, even when the device is offline.

Now the main AWS IoT page will be visible.



Welcome to the AWS IoT Console

To get started, you can jump into the recommended starting points below, or explore other learning resources as needed.

See how AWS IoT works

Explore an interactive tutorial through the components of AWS IoT.

It takes 5 minutes

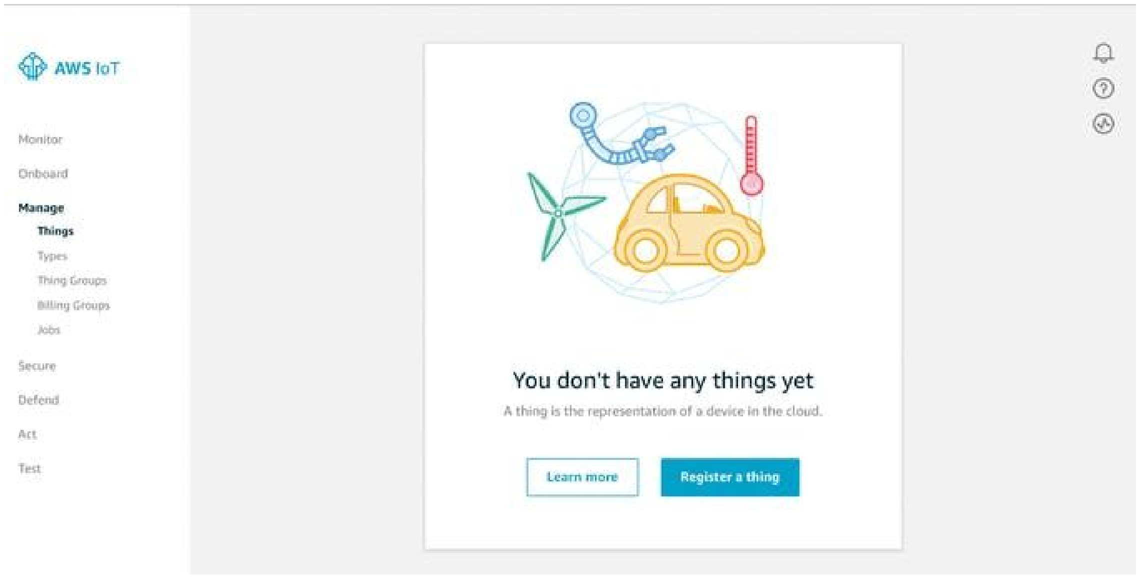
Connect to AWS IoT

Connect a device, a mobile or web app to AWS IoT in a few easy steps!

Explore documentation

The AWS IoT documentation is a great resource for more details.

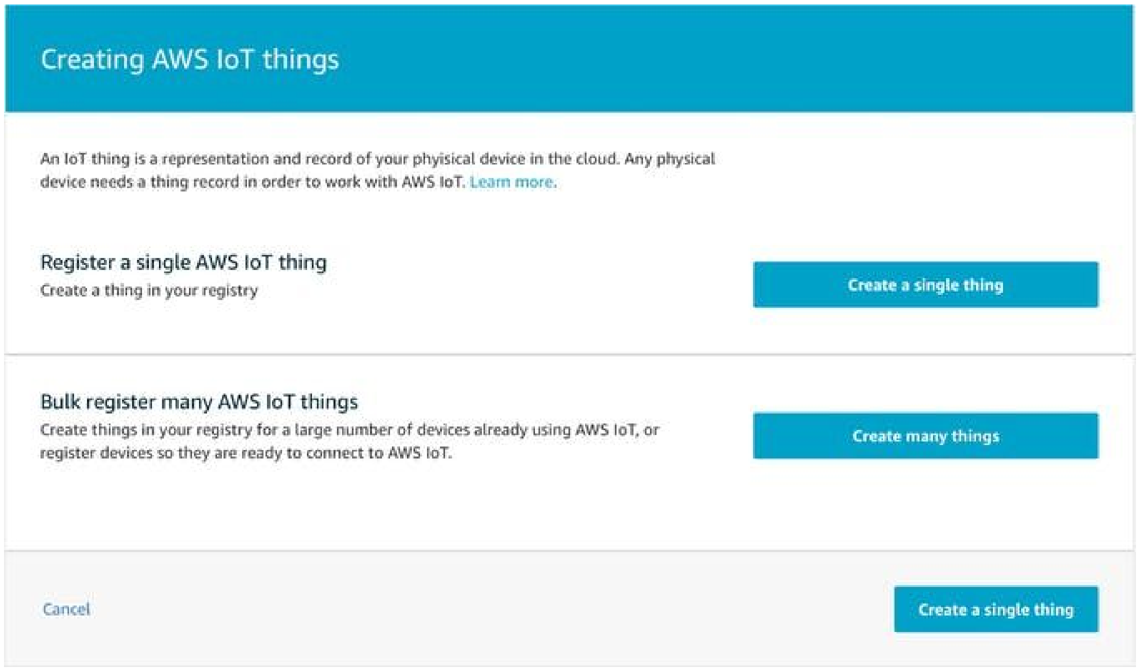
Click the "Manage" on the left, then "Things," and then click the "Register a thing" button.



You don't have any things yet

A thing is the representation of a device in the cloud.

On the next page, click "Create a single thing."



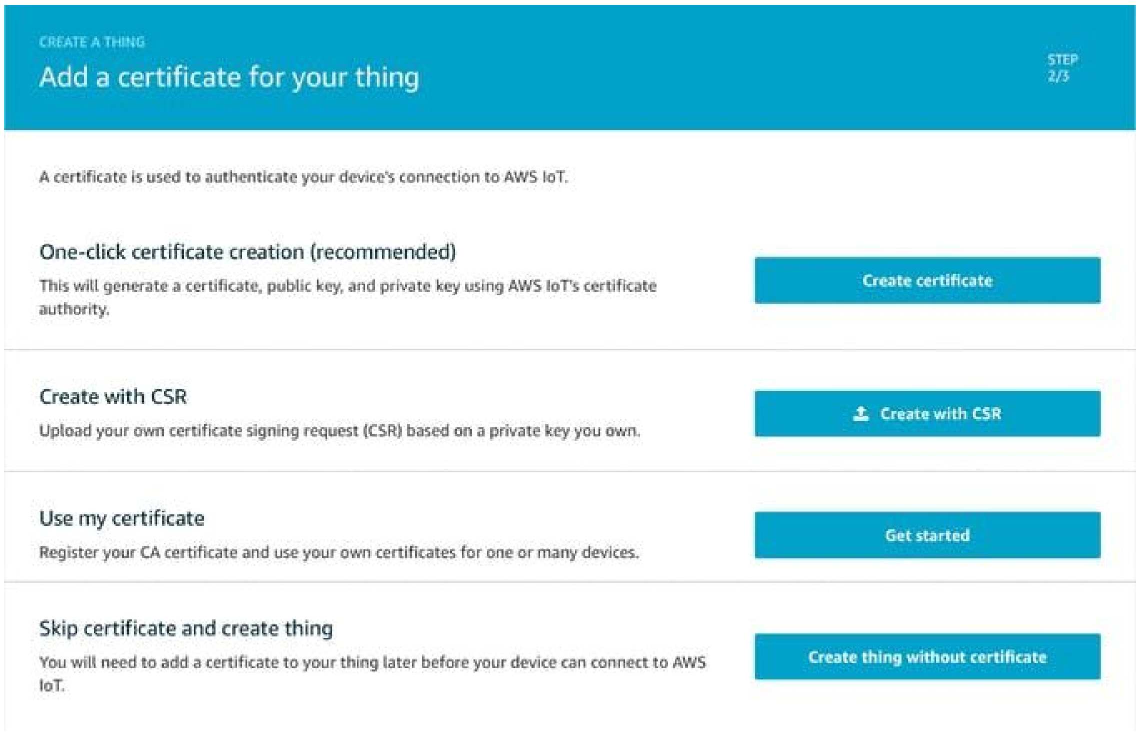
Creating AWS IoT things

An IoT thing is a representation and record of your phyisical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

Register a single AWS IoT thing
Create a thing in your registry

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Give the thing a name like MyMKRWiFi1010, and click "Next." The other entries on the form can be left empty.



CREATE A THING

Add a certificate for your thing

STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

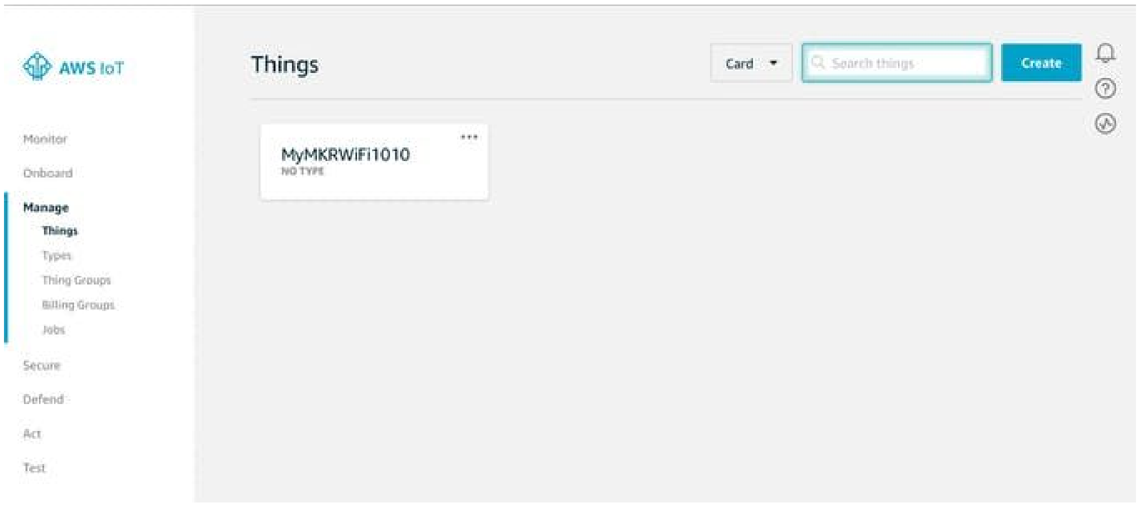
Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT.

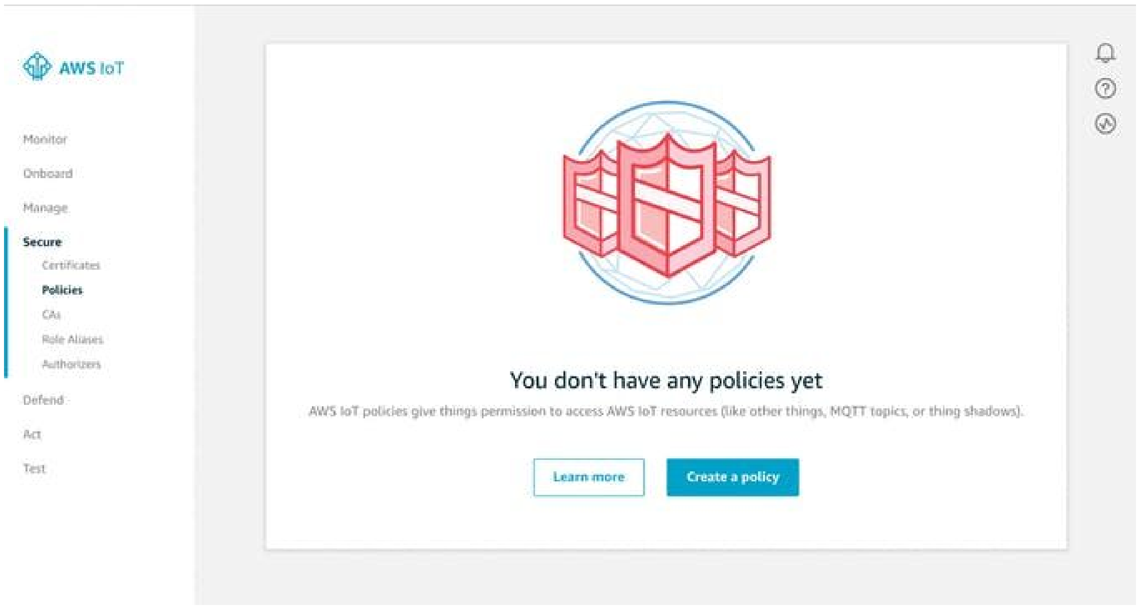
Since we have already generated a CSR on the board, click the "Create with CSR" button, and select the CSR that was generated and saved to the text file earlier. Then click "Upload."

On the next screen, click "Register Thing." We don't have any policies setup yet, and will set one up in a later step.



You should now see a new entry on the Things page.

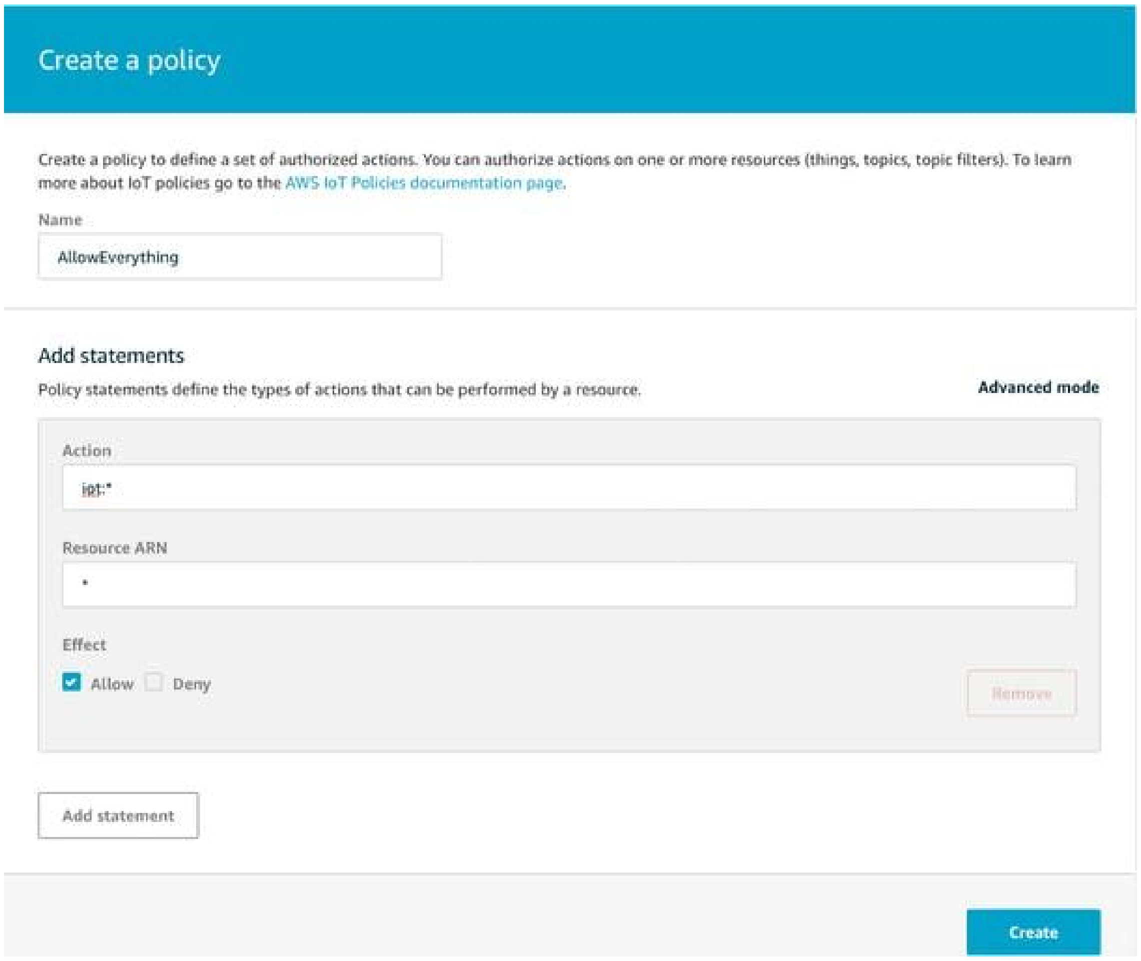
Now we need to create and attach a policy to the thing's certificate. Click on the "Secure" link on the left, then "Policies."



You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows).

Click "Create a policy." We'll be creating a very open policy for testing, later on we suggest you create a stricter policy. We'll call this policy "AllowEverything," fill in "iot:*" for the Action and "*" for the Resource ARN, then check the "Allow" box, then click "Create."


Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: AllowEverything

Add statements

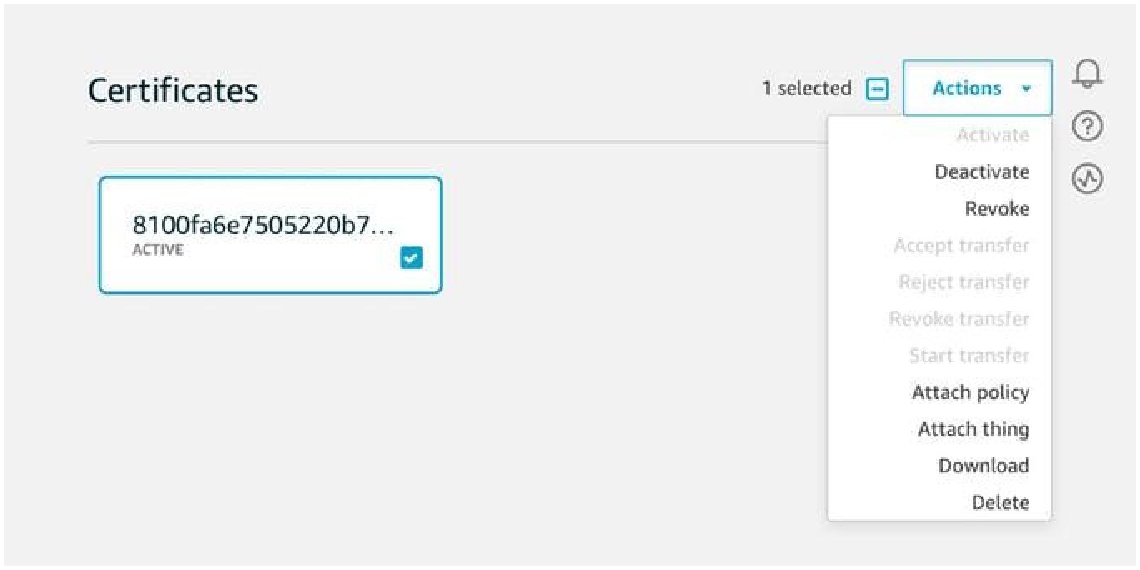
Policy statements define the types of actions that can be performed by a resource.

Action: iot:*

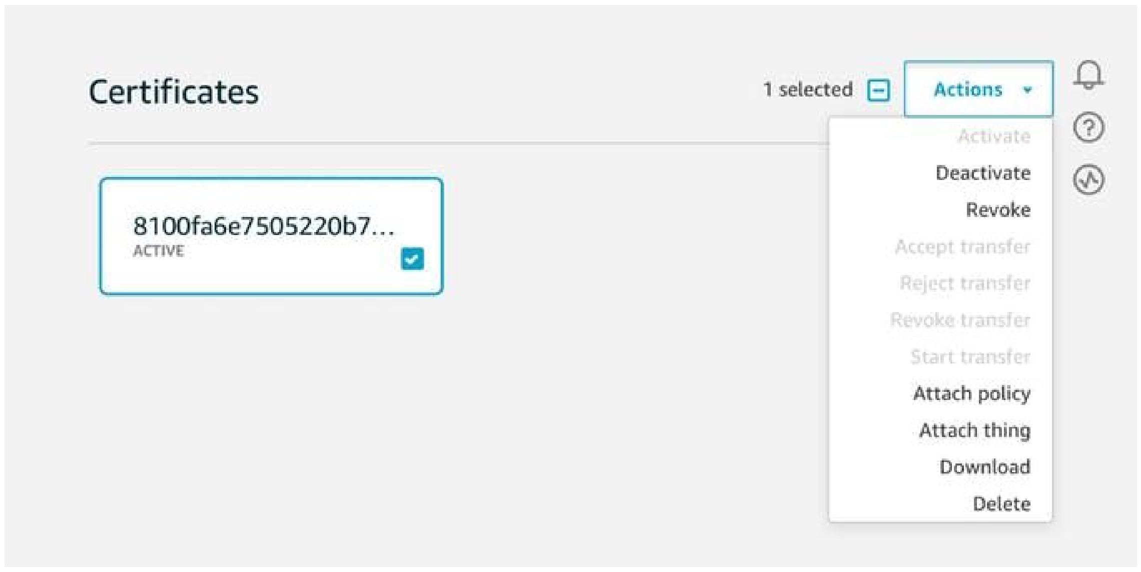
Resource ARN: *

Effect: [✓ Allow] Deny

Now that the policy has been created, navigate to the "Certificates" page using the link on the left. You'll see a certificate entry for the certificate we created earlier.



Check the certificate, then in the "Actions" drop down click "Download" to save the certificate to your computer.



Now let's attach the policy we created to this certificate. in the "Actions" drop-down click "Attach Policy." Then check the "AllowEverything" policy and click the "Attach" button.

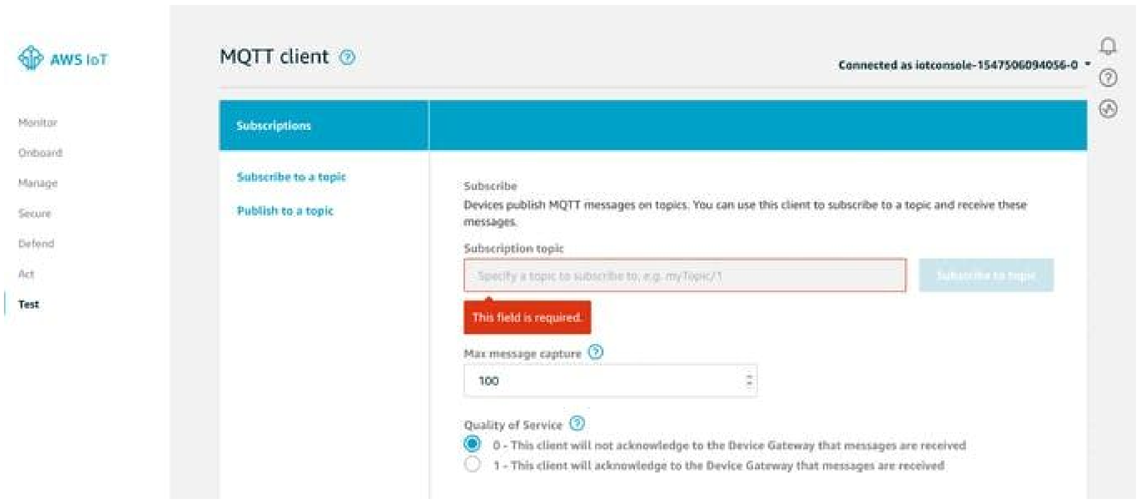
Now click the "Settings" link on the left to get the MQTT endpoint to use for connections. Copy the endpoint and save it to a text file for reference later on.

AWS IoT is now configured for our board.

Interacting with the Arduino

Now that your board has successfully connected to AWS IoT, we can use the AWS console to interact with it. The sketch sends a message to the arduino/outgoing topic every 5 seconds and listens for messages on the arduino/incoming topic.

In the AWS IoT Core console, click the "Test" link on the left.



Connected as iotconsole-1547506094056-0

Subscribe

Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic: Specify a topic to subscribe to, e.g. myTopic/1 — Subscribe to topic

This field is required.

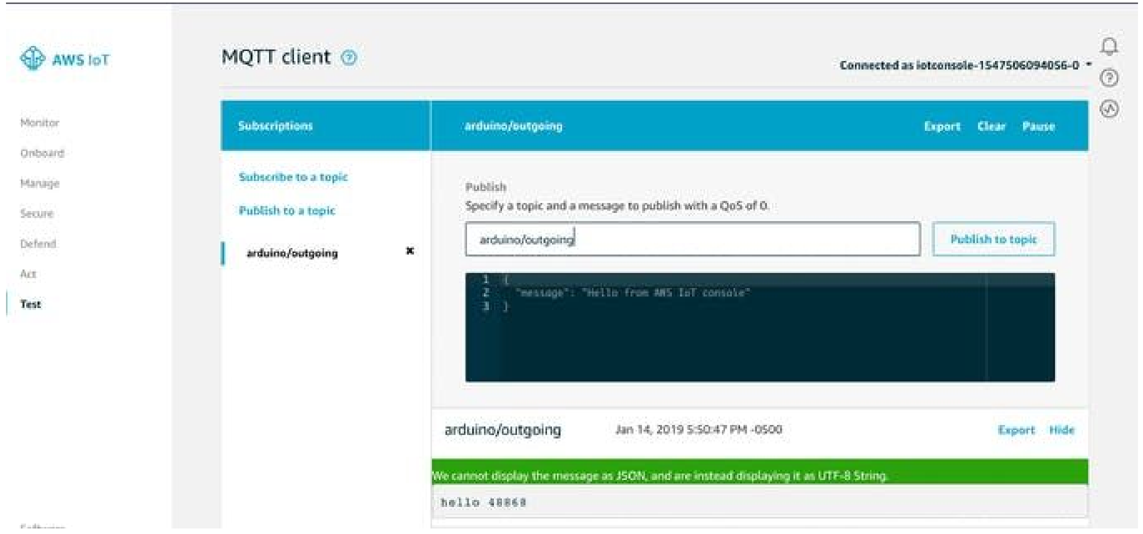
Max message capture: 100

Quality of Service

[✓ 0 - This client will not acknowledge to the Device Gateway that messages are received]

1 - This client will acknowledge to the Device Gateway that messages are received

In the "Subscribe topic" text box enter arduino/outgoing then click "Subscribe to topic." Every five seconds the board sends a hello message with the current millis() value



Connected as iotconsole-1547506094056-0

Publish

Specify a topic and a message to publish with a QoS of 0.

arduino/outgoing

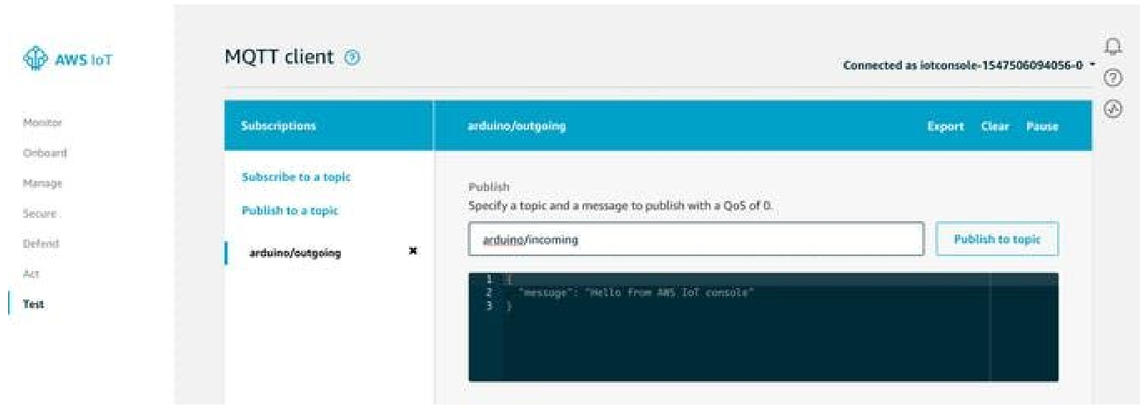
```
{
  "message": "Hello from AWS IoT console"
}
```

arduino/outgoing   Jan 14, 2019 5:50:47 PM -0500

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

```
hello 48868
```

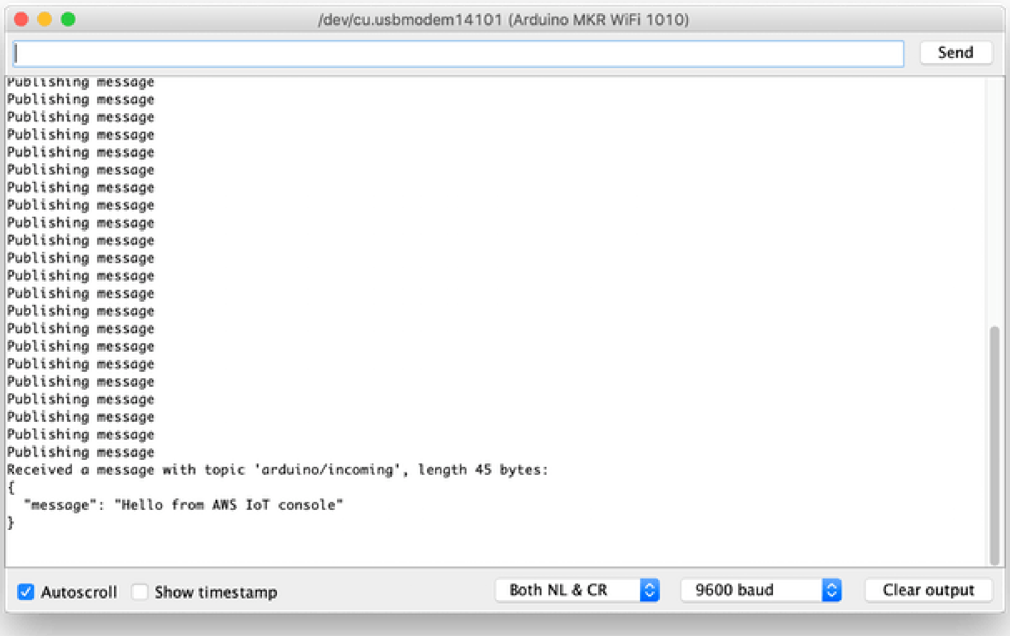
Now let's send a message to the board. In the Publish section, change the topic to arduino/incoming and click the "Publish to topic" button.



Specify a topic and a message to publish with a QoS of 0.

arduino/incoming

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

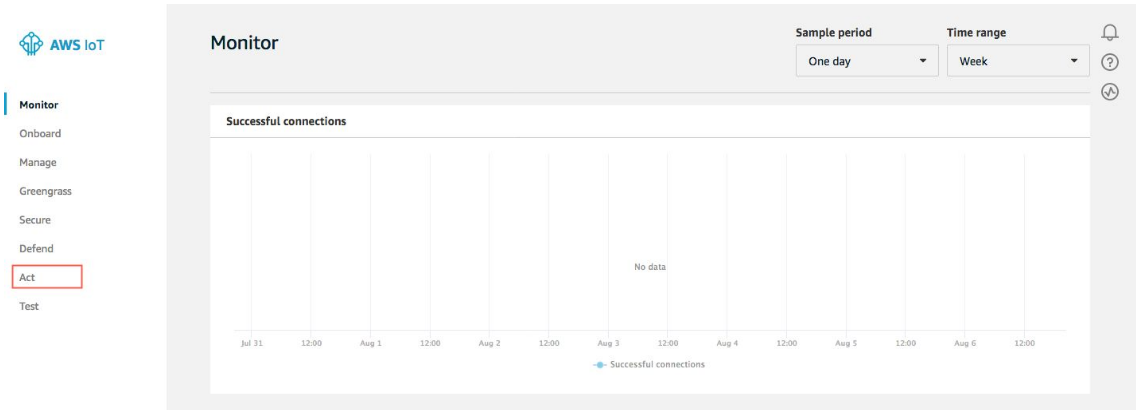


```
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Received a message with topic 'arduino/incoming', length 45 bytes:
{
  "message": "Hello from AWS IoT console"
}
```

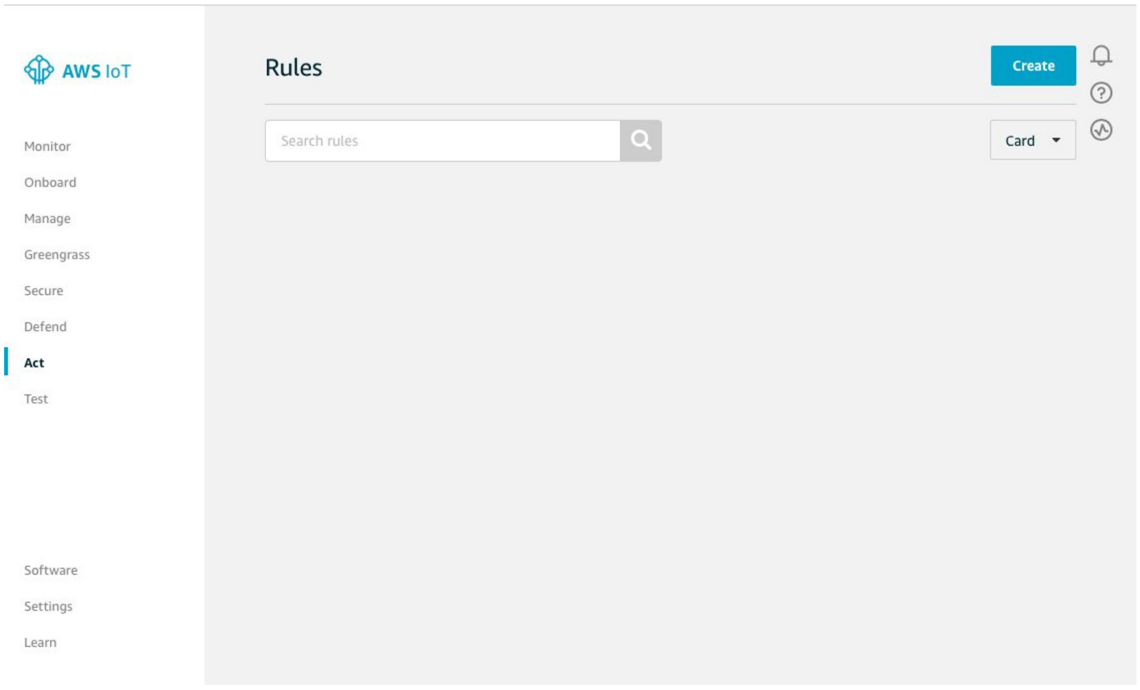
And voila here it is, communication between your Arduino and AWS-IoT.

Storing your data into DynamoDB

1. In the [AWS IoT console](), in the navigation pane, choose Act.



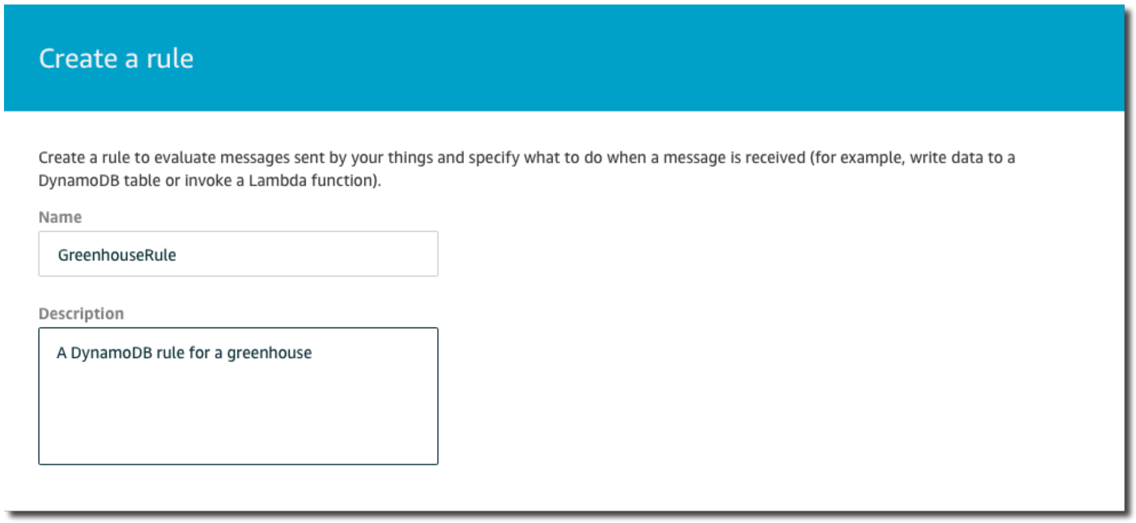
2. On the Rules page, choose Create.



3. On the Create a rule page, enter a name and description for your rule.

   Note

   We do not recommend the use of personally identifiable information in rule names or descriptions.


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

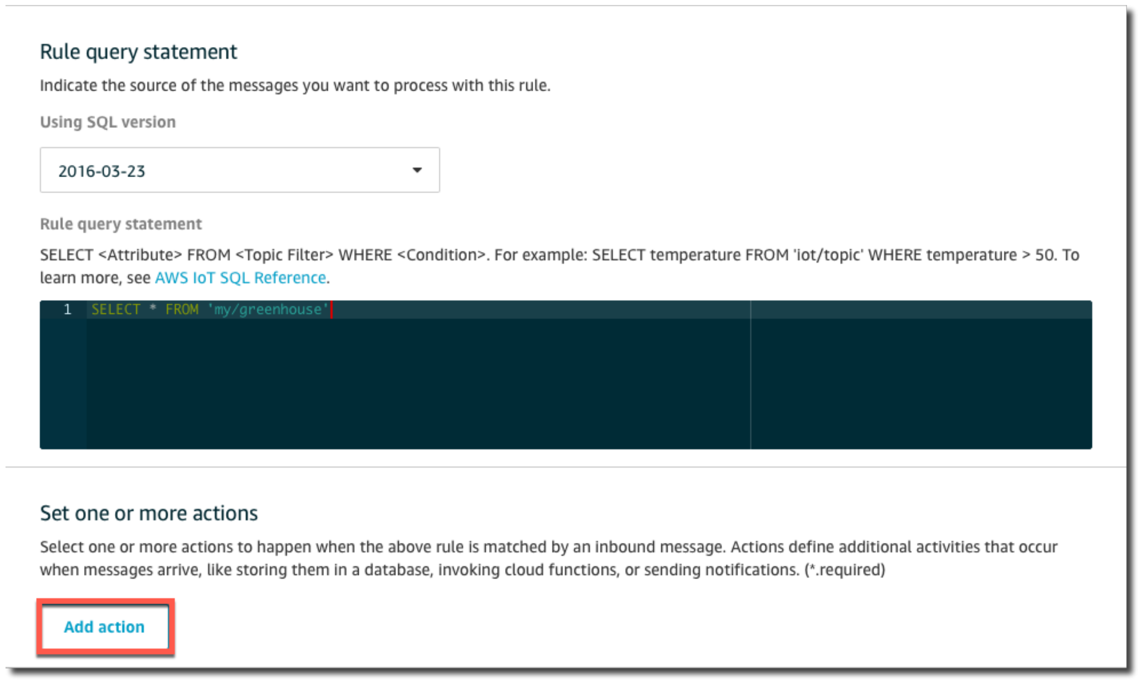
GreenhouseRule

Description

A DynamoDB rule for a greenhouse

4. Under Rule query statement, choose the latest version from the Using SQL version list. For Rule query statement, enter:
5. SELECT * FROM 'arduino/outgoing' (this is you topic name, ignore the one in picture)
6. ("SELECT *" specifies that you want to send the entire MQTT message that triggered the rule. "FROM 'my/greenhouse'" tells the rules engine to trigger this rule when an MQTT message whose topic matches this topic filter is received.

Choose Add action.



Rule query statement

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

Using SQL version

2016-03-23

Rule query statement

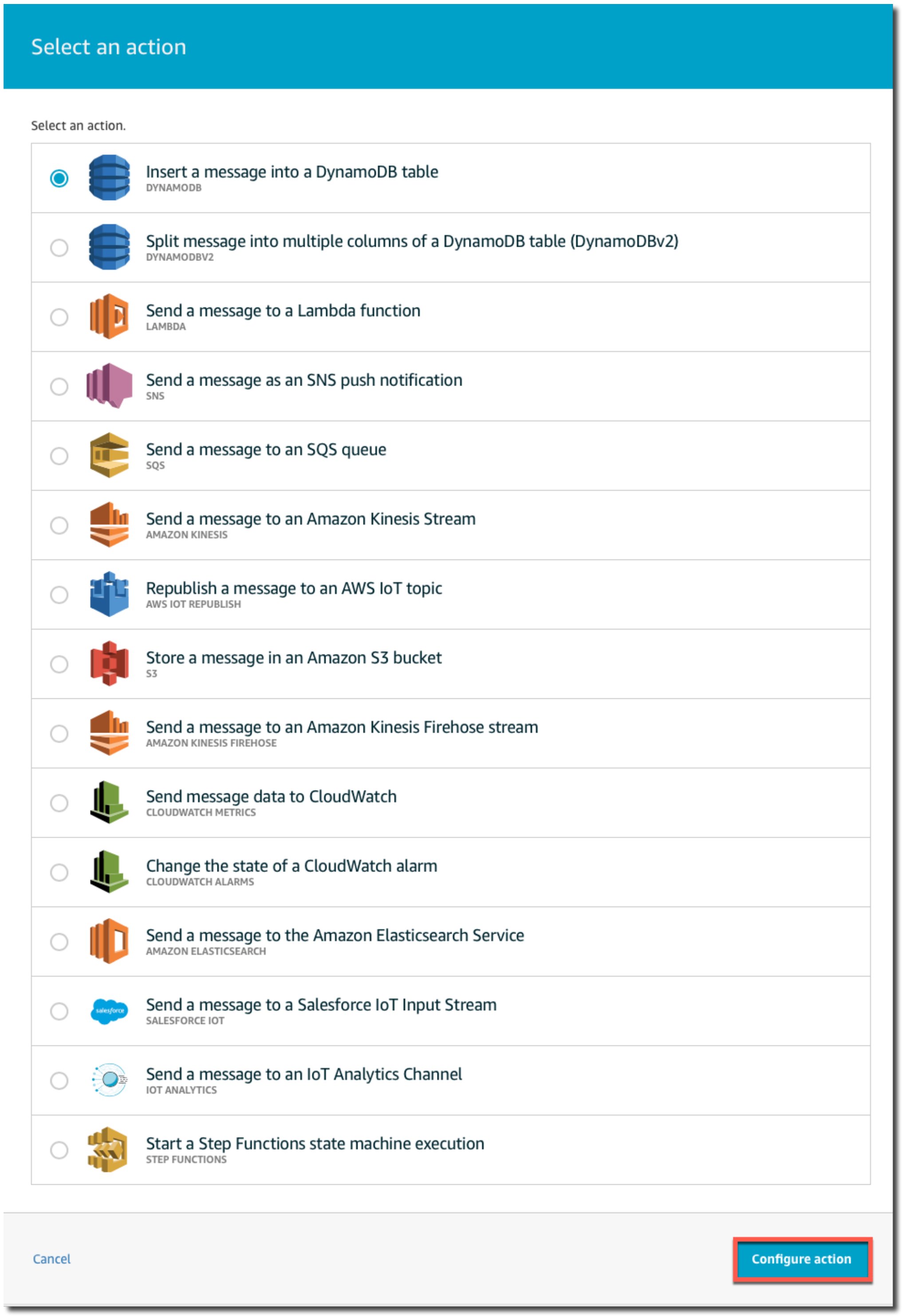
SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM 'my/greenhouse'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

7. Select an action, choose to Insert a message into a DynamoDB table, and then choose Configure action.

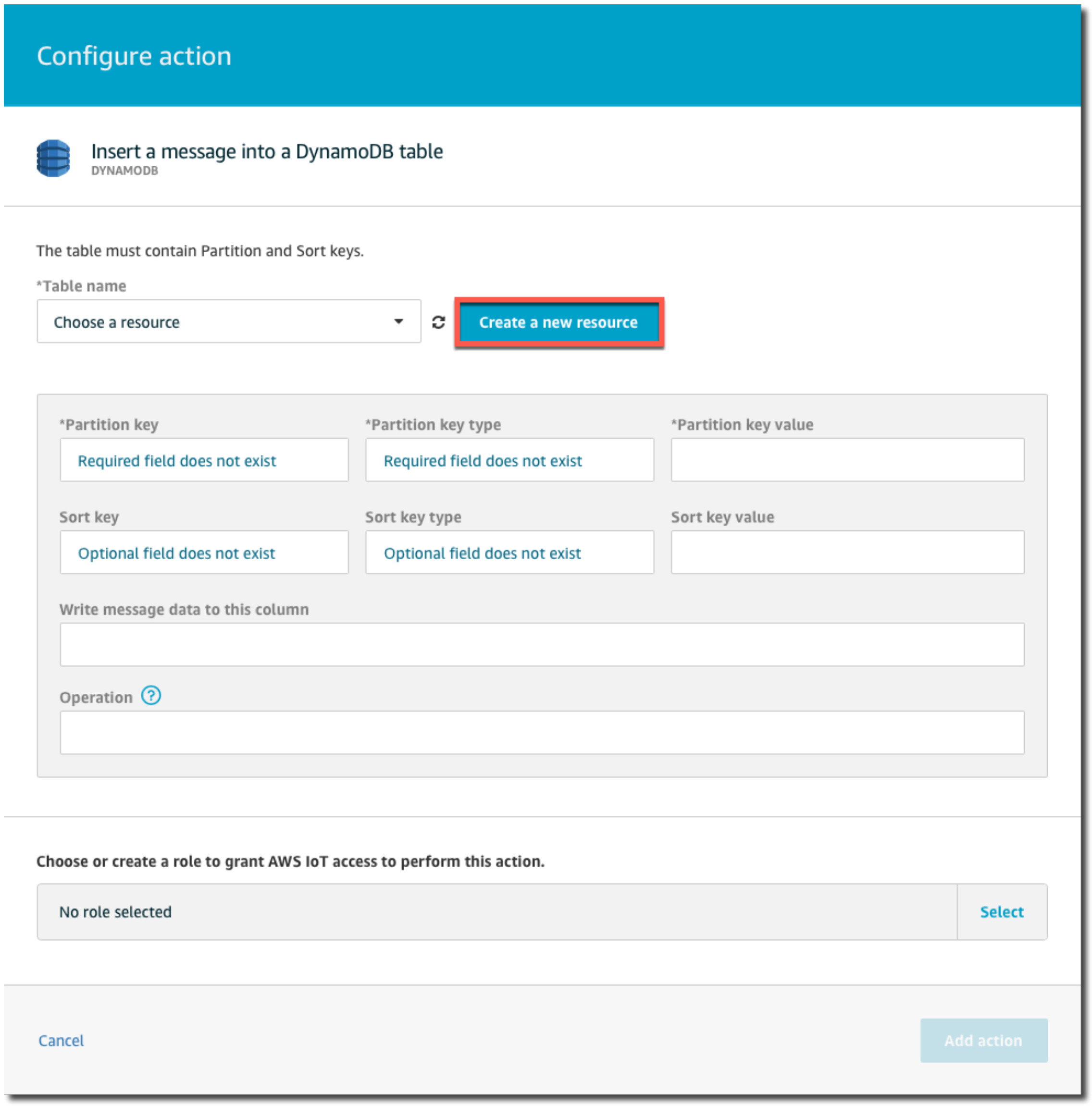


Select an action

Select an action.

- [✓ Insert a message into a DynamoDB table] DYNAMODB
- Split message into multiple columns of a DynamoDB table (DynamoDBv2) DYNAMODBV2
- Send a message to a Lambda function LAMBDA
- Send a message as an SNS push notification SNS
- Send a message to an SQS queue SQS
- Send a message to an Amazon Kinesis Stream AMAZON KINESIS
- Republish a message to an AWS IoT topic AWS IOT REPUBLISH
- Store a message in an Amazon S3 bucket S3
- Send a message to an Amazon Kinesis Firehose stream AMAZON KINESIS FIREHOSE
- Send message data to CloudWatch CLOUDWATCH METRICS
- Change the state of a CloudWatch alarm CLOUDWATCH ALARMS
- Send a message to the Amazon Elasticsearch Service AMAZON ELASTICSEARCH
- Send a message to a Salesforce IoT Input Stream SALESFORCE IOT
- Send a message to an IoT Analytics Channel IOT ANALYTICS
- Start a Step Functions state machine execution STEP FUNCTIONS

8. On Configure action, choose to Create a new resource.



Configure action

Insert a message into a DynamoDB table
DYNAMODB

The table must contain Partition and Sort keys.

*Table name

Choose a resource

Create a new resource

*Partition key	*Partition key type	*Partition key value
Required field does not exist	Required field does not exist	
Sort key	Sort key type	Sort key value
Optional field does not exist	Optional field does not exist	

Write message data to this column

Operation

Choose or create a role to grant AWS IoT access to perform this action.

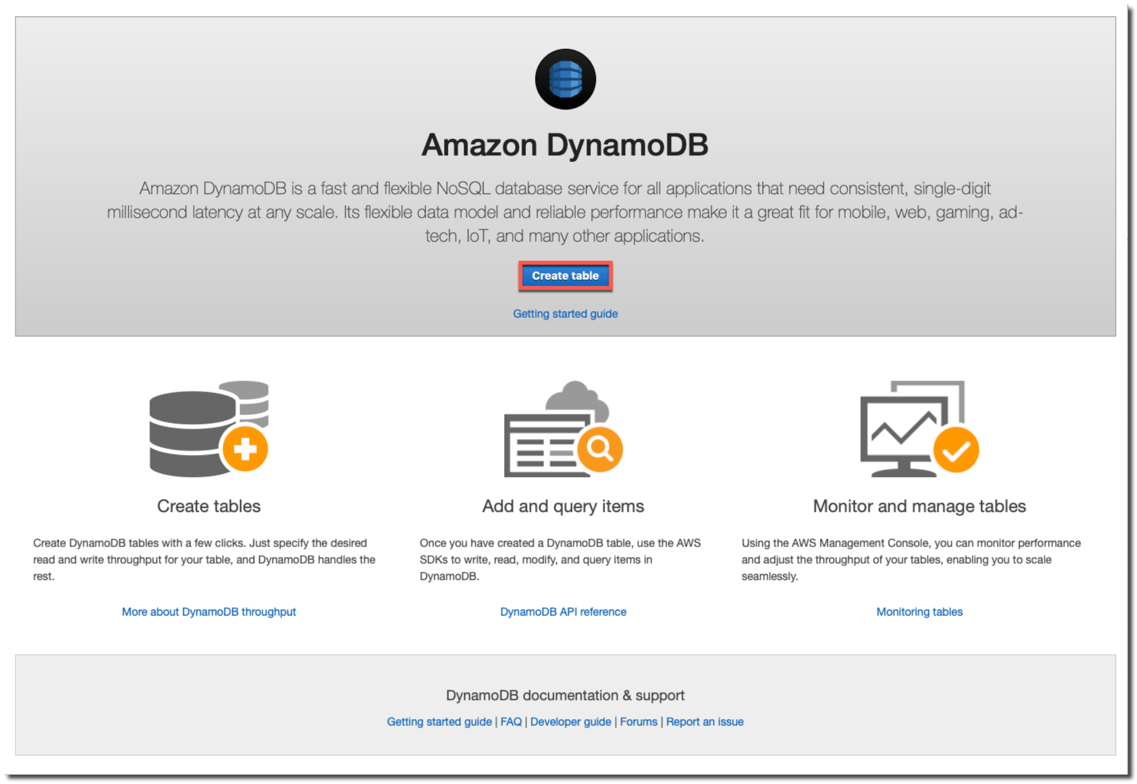
No role selected

Select

Cancel

Add action

9. On the Amazon DynamoDB page, choose Create table.



Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Create tables

Create DynamoDB tables with a few clicks. Just specify the desired read and write throughput for your table, and DynamoDB handles the rest.

Add and query items

Once you have created a DynamoDB table, use the AWS SDKs to write, read, modify, and query items in DynamoDB.

Monitor and manage tables

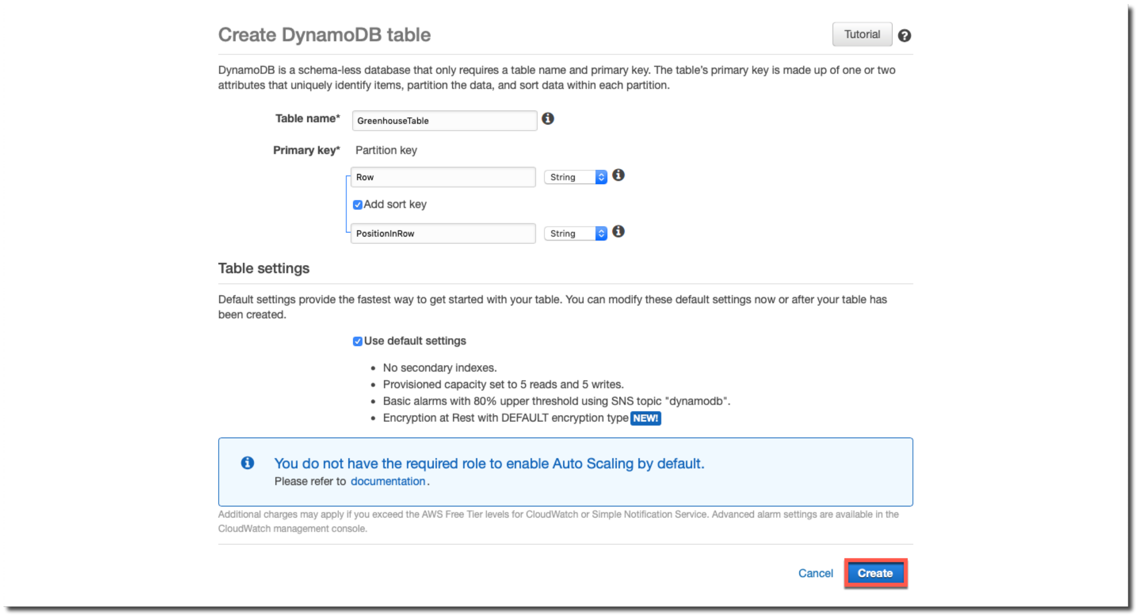
Using the AWS Management Console, you can monitor performance and adjust the throughput of your tables, enabling you to scale seamlessly.

DynamoDB documentation & support

Getting started guide | FAQ | Developer guide | Forums | Report an issue

10. On Create DynamoDB table, enter a name. In Partition key, enter Row. Select Add sort key, and then enter PositionInRow in the Sort key field. A row represents a row of plants in a greenhouse. PositionInRow represents the position of a plant in the row. Choose String for both the partition and sort keys, and then choose to Create. It takes a few seconds to create your DynamoDB table. Close the browser tab where the Amazon DynamoDB console is open. If you don't close the tab, your DynamoDB table is not displayed in the Table name list on the Configure action page of the AWS IoT

console.



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* GreenhouseTable

Primary key* Partition key

Row String

Add sort key

PositionInRow String

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

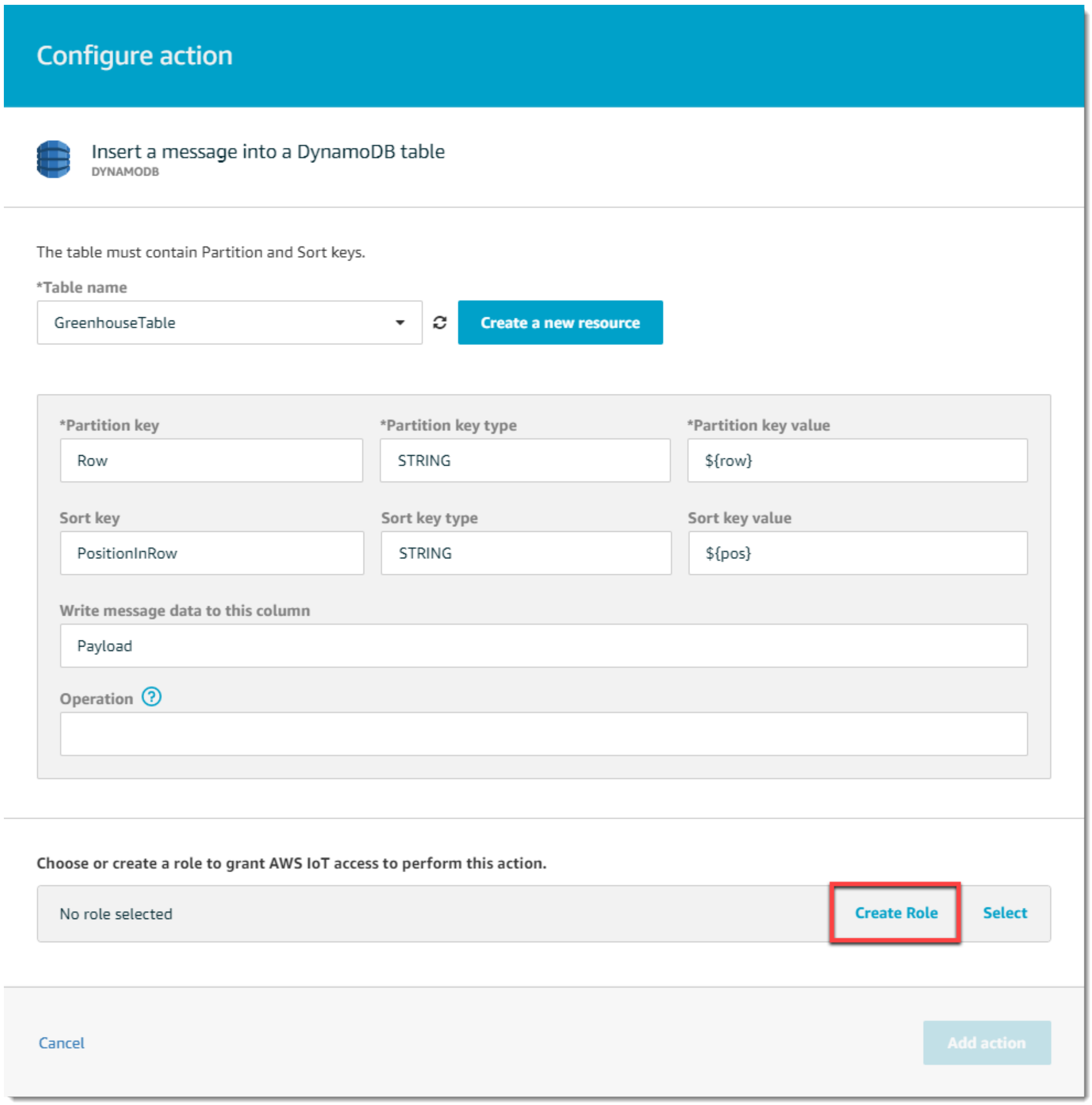
- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type NEW!

You do not have the required role to enable Auto Scaling by default.
Please refer to documentation.

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

11. On Configure action, choose your new table from the Table name list. In Partition key-value, enter ${row}. This instructs the rule to take the value of the row attribute from the MQTT message and write it into the Row column in the DynamoDB table. In Sort key-value, enter ${pos}. This writes the value of the pos attribute into the PositionInRow column. Write message data to this column, enter Payload. This inserts the message payload into the Payload column. Leave Operation blank. This field allows you to specify which operation (INSERT, UPDATE, or DELETE) you want to perform when the action is

triggered. Choose Create a new role.


Configure action

Insert a message into a DynamoDB table
DYNAMODB

The table must contain Partition and Sort keys.

*Table name: GreenhouseTable

*Partition key	*Partition key type	*Partition key value
Row	STRING	${row}
Sort key	Sort key type	Sort key value
PositionInRow	STRING	${pos}

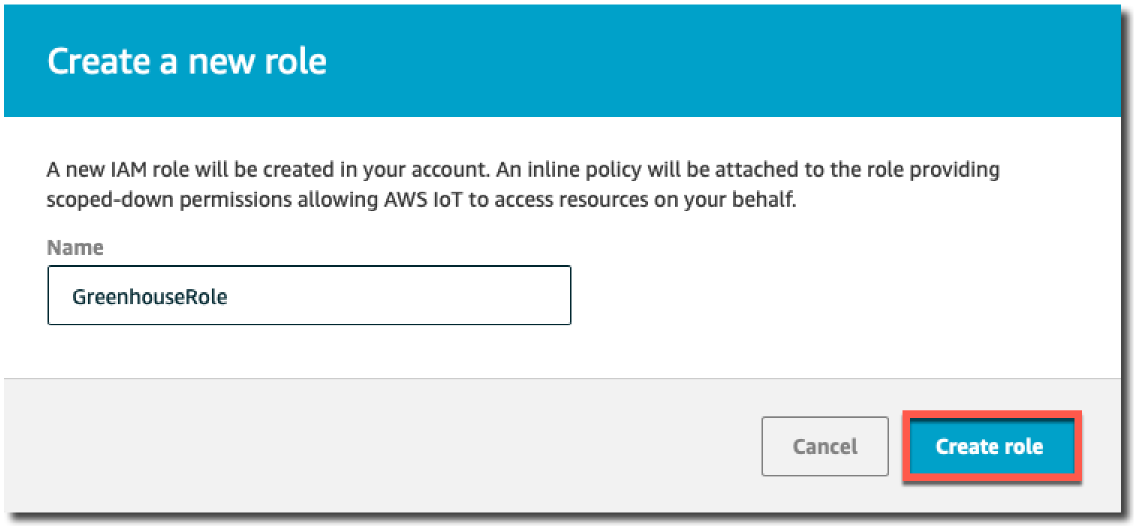
Write message data to this column: Payload

Operation:

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

12. In Create a new role, enter a unique name, and then choose Create role.

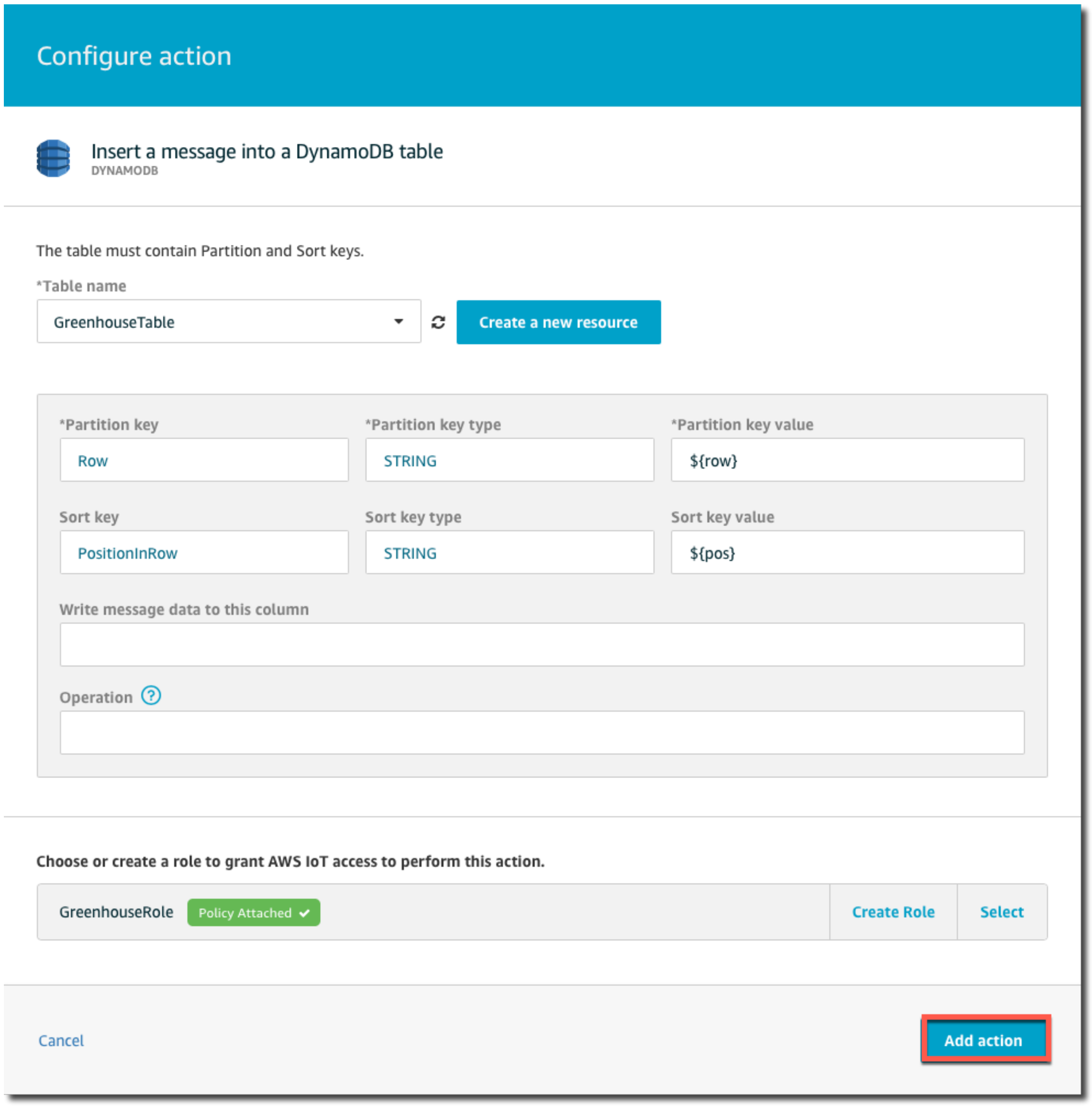

Create a new role

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name

GreenhouseRole

13. Choose Add action.



Configure action

Insert a message into a DynamoDB table
DYNAMODB

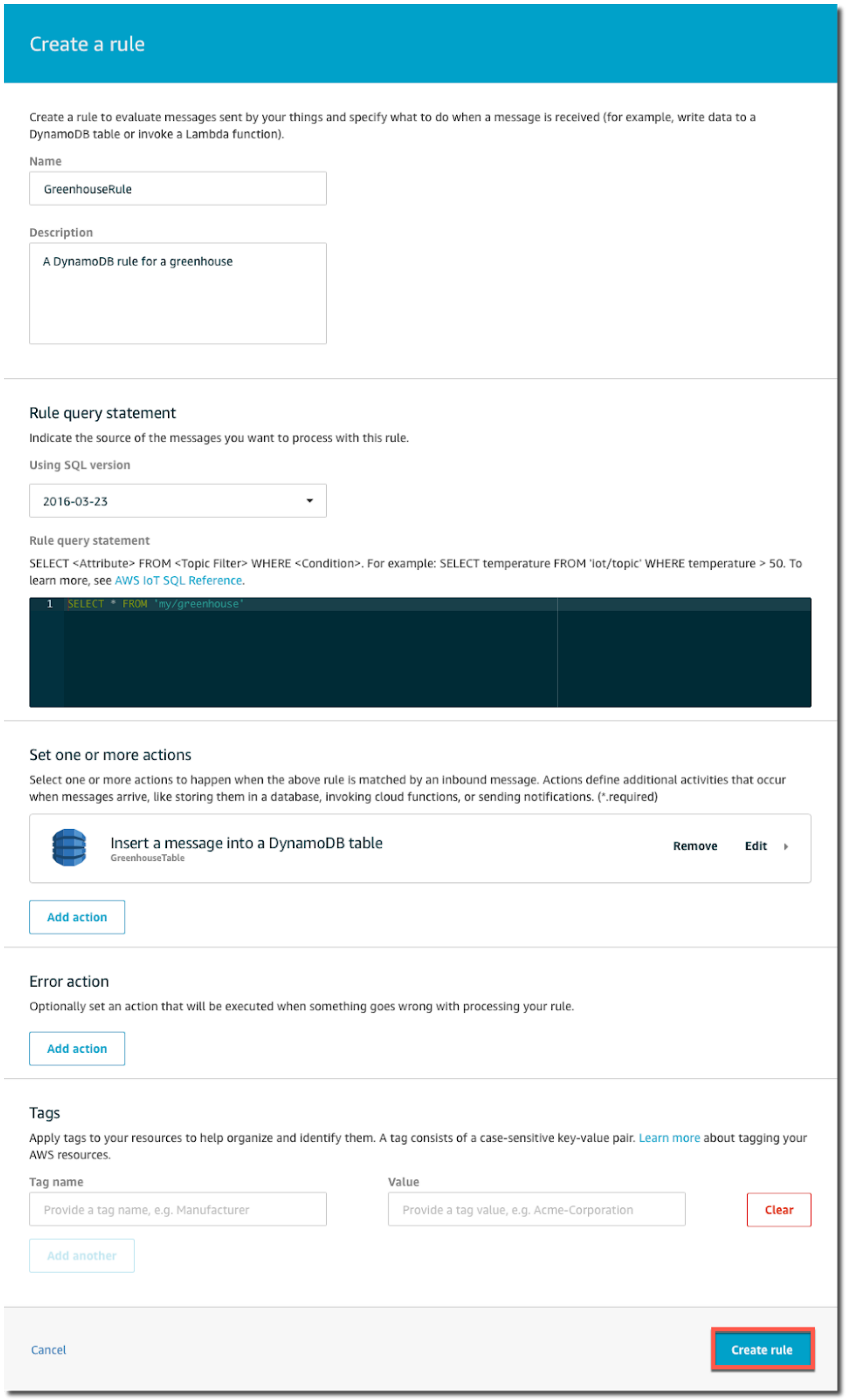
The table must contain Partition and Sort keys.

*Table name	GreenhouseTable	
*Partition key: Row	*Partition key type: STRING	*Partition key value: ${row}
Sort key: PositionInRow	Sort key type: STRING	Sort key value: ${pos}
Write message data to this column		
Operation		

Choose or create a role to grant AWS IoT access to perform this action.

GreenhouseRole Policy Attached ✓

14. Choose Create rule.



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

GreenhouseRule

Description

A DynamoDB rule for a greenhouse

Rule query statement

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

Using SQL version

2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM 'my/greenhouse'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Insert a message into a DynamoDB table
GreenhouseTable

Remove Edit

Add action

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. Learn more about tagging your AWS resources.

Tag name

Provide a tag name, e.g. Manufacturer

Value

Provide a tag value, e.g. Acme-Corporation

Clear

Add another

Cancel

Create rule

You can check the values that are being transferred from aws-iot to DynamoDB by going to DynamoDB and records in the table you created.

Feel free to ask if you have any questions on m.jan@studenti.unisa.it