**EE5111**

Selected topics in Industrial Control & Instrumentation

Project1

Report

A0206691A

Chen Yuxiang

2019/9/15

contents

[1. Introduction 3](#_Toc19520645)

[**1.1Background** 3](#_Toc19520646)

[1.2 Goals 5](#_Toc19520647)

[1.3 Tools 5](#_Toc19520648)

[2. Publish engine data to AWS 5](#_Toc19520649)

[2.1 Step 1: AWS planting watering sample 5](#_Toc19520650)

[2.2 step 2: Simulate jet engine data 6](#_Toc19520651)

[2.3 step 3: simulate two IoT things 14](#_Toc19520652)

[3. Data visualize 14](#_Toc19520653)

[3.1 load data 14](#_Toc19520654)

[3.2 visualize the data 15](#_Toc19520655)

[4. Use other data sources 21](#_Toc19520656)

[5. Conclusion 22](#_Toc19520657)

[5.1 potential improvements 22](#_Toc19520658)

[5.2 summary 23](#_Toc19520659)

1. **Introduction**
   1. **Background**

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The IoT involves extending Internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of traditionally dumb or non-internet-enabled physical devices and everyday objects. Embedded with technology, these devices can communicate and interact over the Internet, and they can be remotely monitored and controlled.

This project requires an implementation of a simple IoT pipeline with AWS Cloud platform and visualise the data. We will simulate two small IoT setups that record and push data from two jet engines. And the experimental scenario for th two jet engines is as follows.

Each dataset consists of multiple multivariate time series. Each data set is further divided into training and test subsets. Each time series is from a different engine ,i.e., the data can be considered to be from a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition. There are three operational settings that have a substantial effect on engine performance. These settings are also included in the data. The data is contaminated with sensor noise.

The engine is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends some time prior to system failure. The objective of the competition is to predict the number of remaining operational cycles before failure in the test set, i.e., the number of operational cycles after the last cycle that the engine will continue to operate. Also provided a vector of true Remaining Useful Life (RUL) values for the test data.

The data are provided as a zip-compressed text file with 26 columns of numbers, separated by spaces. Each row is a snapshot of data taken during a single operational cycle, each column is a different variable. The columns correspond to:

1) unit number（id）

2) time, in cycles（circle）

3) operational setting 1 (os1）

4) operational setting 2 (os2)

5) operational setting 3 (os3)

6) sensor measurement 1 (sensor1)

7) sensor measurement 2 (sensor2)

...

21) sensor measurement 21 (snesor21)

## 1.2 Goals

1.Follow the AWS instruction to understand how to populate, send data from a computer to AWS through MQTT protocol.

2.Publish pre-defined engine data to AWS.

3.Simulate two IoT things

4.Visualise the data for the two engines by querying the data from AWS DynamoDB.

5.Use other data sources.

## 1.3 Tools

1.AWS free tier account;

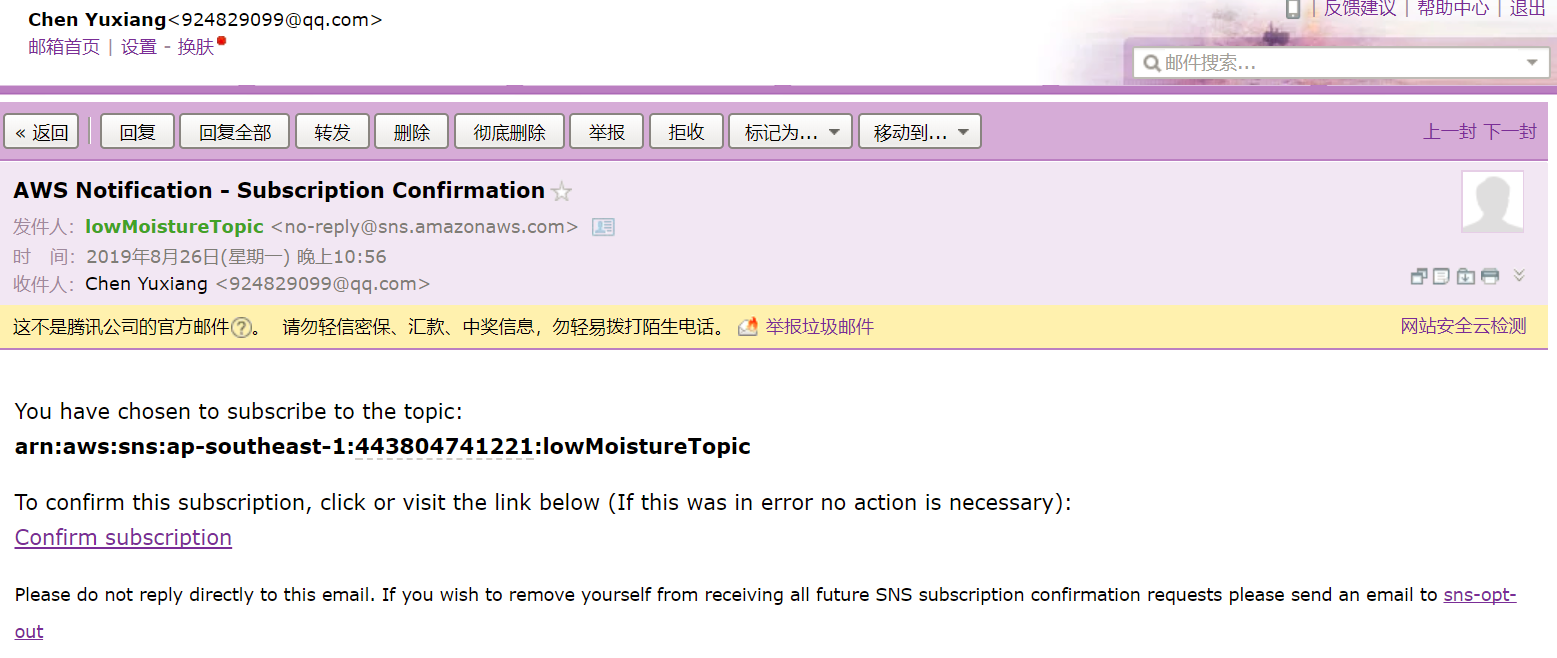
2.laptop with Anaconda 3 installed(Python >=3.5)

3.boto3

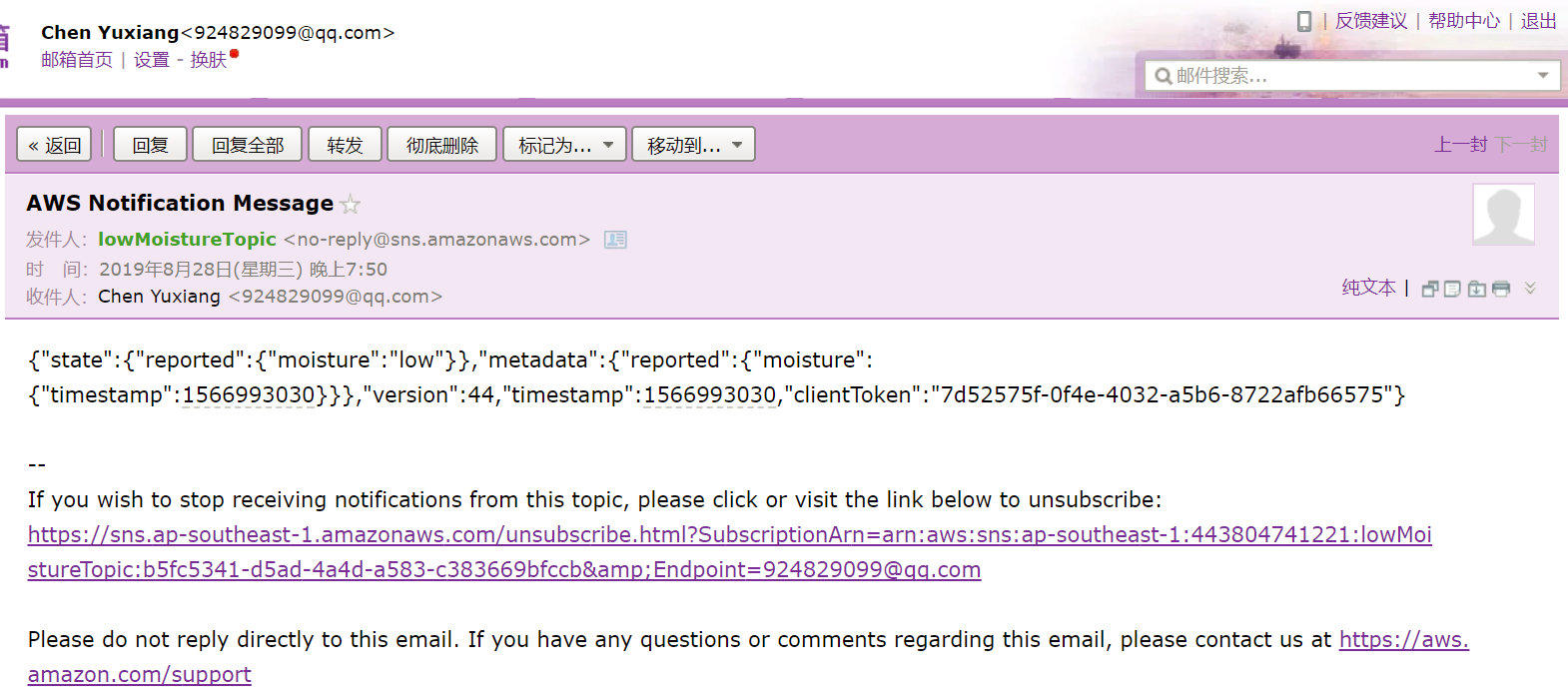
# 2. Publish engine data to AWS

## 2.1 Step 1: AWS planting watering sample

In this step, I follow the instructions from <https://docs.aws.amazon.com/iot/latest/developerguide/iot-plant-step1.html> to set up AWS IoT to begin receiving and storing soil moisture readings coming from my development computer as a device simulator, or from a Raspberry Pi. Then I set up AWS IoT to send email alerts through Amazon Simple Notification Service (Amazon SNS) that are based on those readings.

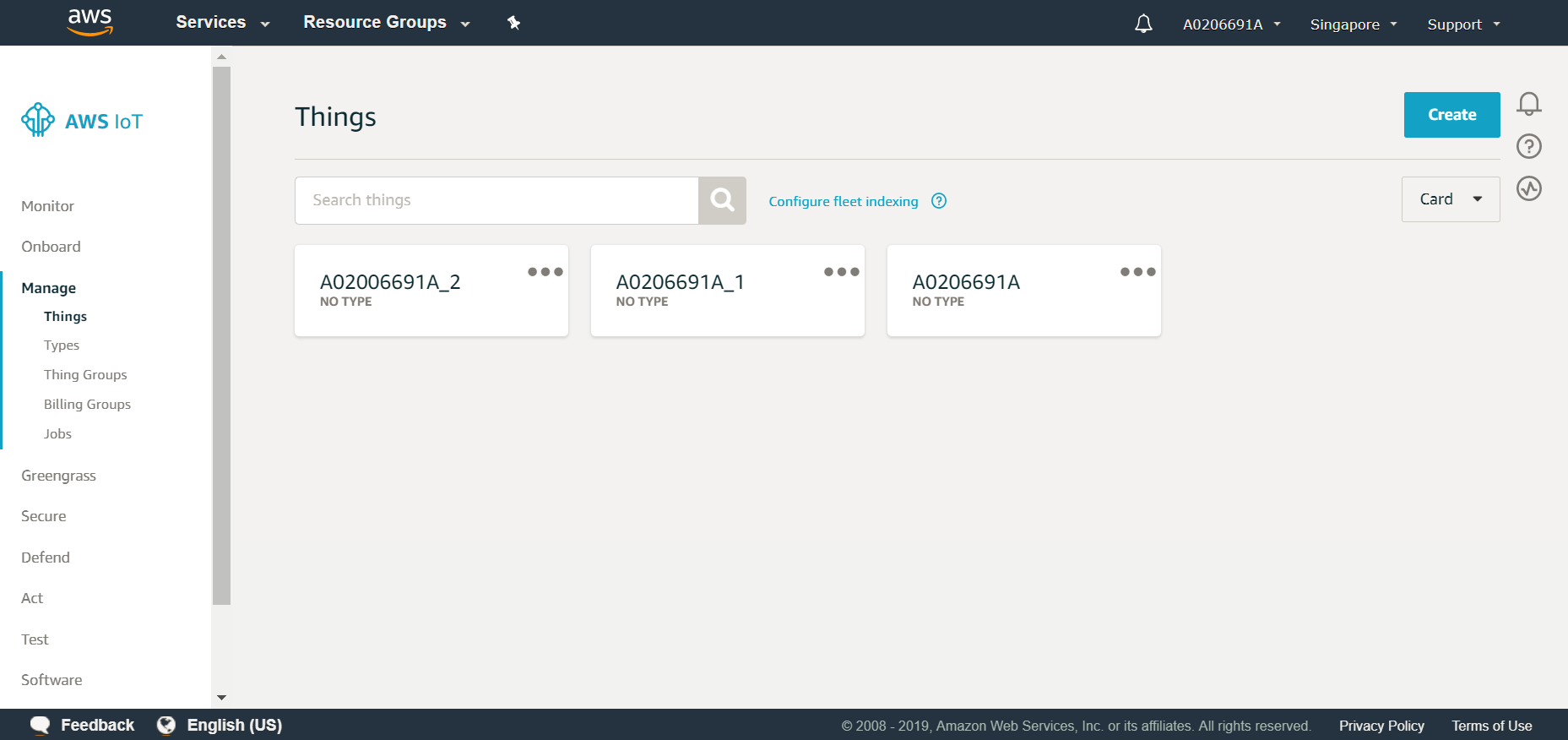


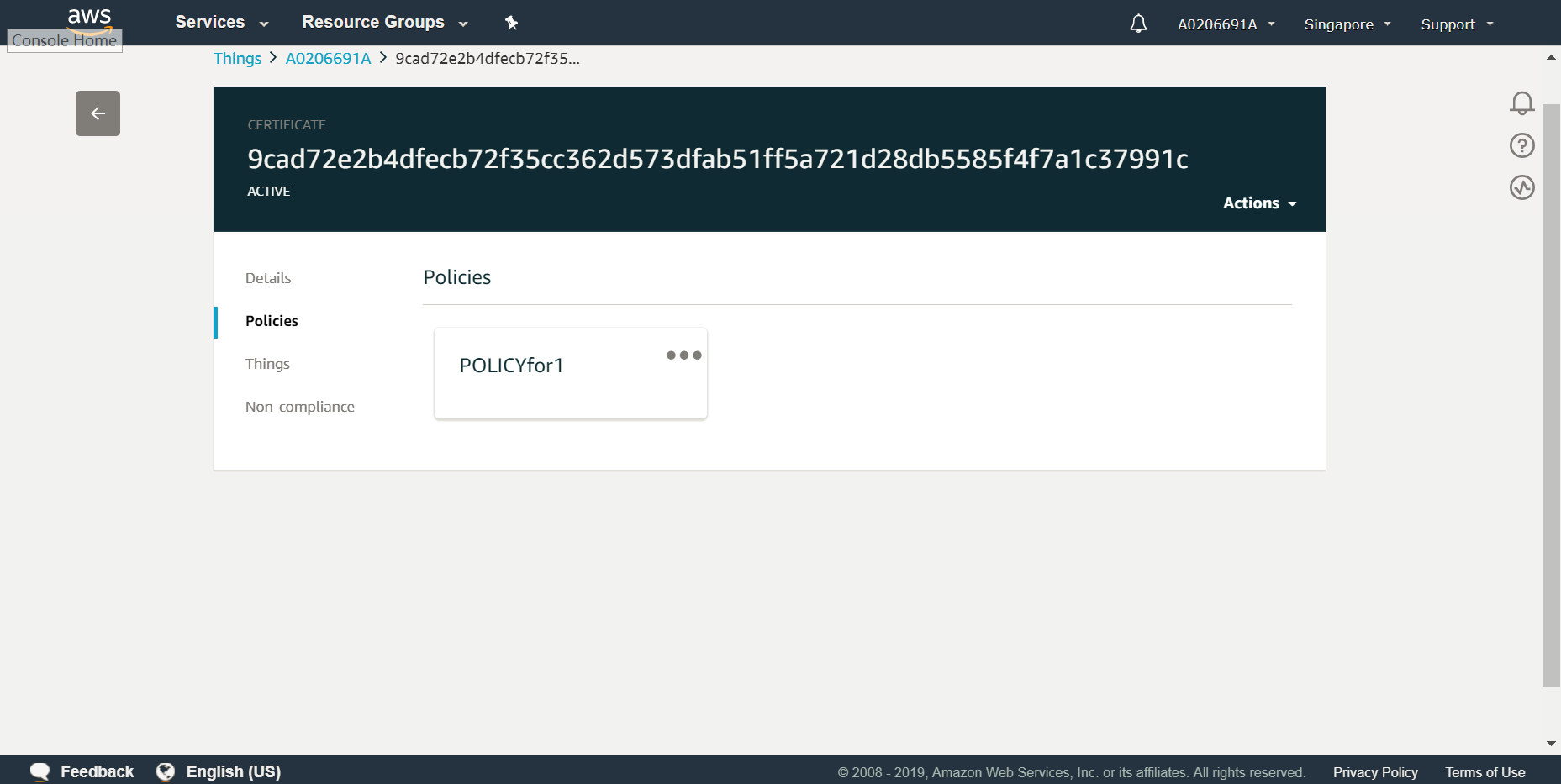
Finally, I use my development computer to simulate soil moisture readings by generating random data; then push those readings into AWS IoT . When the readings get too low, Amazon SNS automatically sends an email alert.

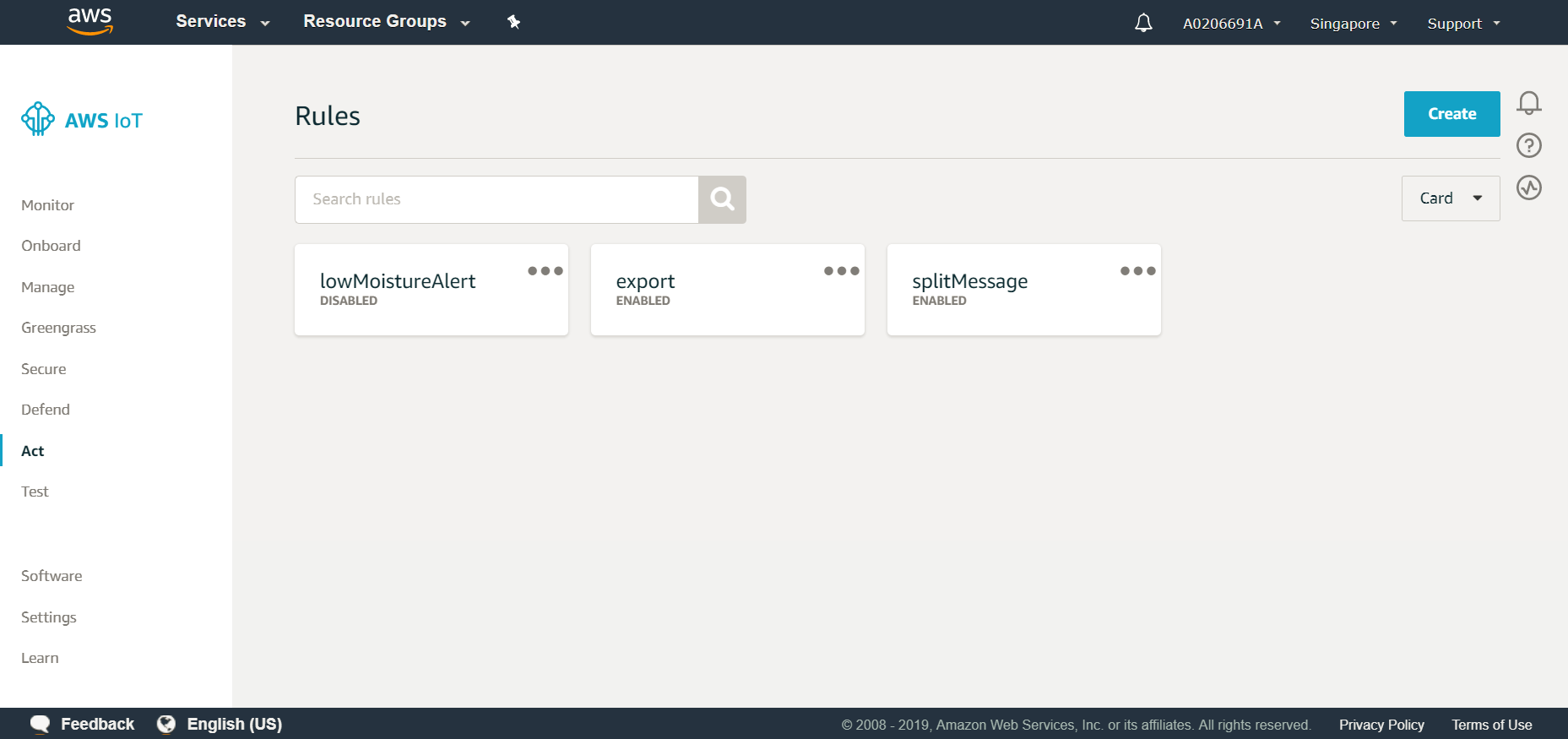


## 2.2 step 2: Simulate jet engine data

1. First, I create things(with cetificate), policy, rules and tables, the results are shown below.







**Create AWS IoT policy:**

In this step, to allow my development computer as a substitute simulator, to perform AWS IoT operations, you create an AWS IoT policy. X.509 certificates are used to authenticate devices with AWS IoT . AWS IoT policies are used to authorize devices to perform AWS IoT operations, such as subscribing or publishing to MQTT topics. Later on, you attach the policy to a device certificate.

1. Use operating system’s web browser to sign in to the AWS Management Console, at https://aws.amazon.com.
2. On the AWS navigation bar, choose the AWS Region where you want to create the AWS IoT resources in your AWS account. This sample was tested with the **Asia Pacific(Singapore)** Region.
3. Open the AWS IoT console. To do this, on the AWS navigation bar, choose Services. In the Find a service by name or feature box, enter IoT Core, and then press Enter.
4. In the service navigation pane, expand Secure, and then choose Policies.
5. Choose Create a policy.
6. Provide a Name that represents this policy, i.e, **POLICYfor1**.
7. For Action, enter iot:\*.
8. For Resource ARN, replace the suggested value with an asterisk (\*).
9. For Effect, choose Allow.
10. Choose Create.

**Create the Thing:**

In this step, I create a thing in AWS IoT to represent my development computer as a device simulator.

Devices connected to AWS IoT are represented by things in the AWS IoT registry. The registry enables me to keep a record of all of the devices that are connected to my AWS account in AWS IoT .

(1) With the AWS IoT console open, in the service navigation pane, choose Manage.

(2) In the service navigation pane, with Manage expanded, choose Things.

(3) Since I don’t have any things yet dialog box is displayed, choose Register a thing. Otherwise, chooseCreate.

(4) On the Creating AWS IoT things page, for Register a single AWS IoT thing, choose Create a single thing.

(5) On the Add your device to the device registry page, provide a Name that represents the development computer as a device simulator, i.e, **A0206691A**.

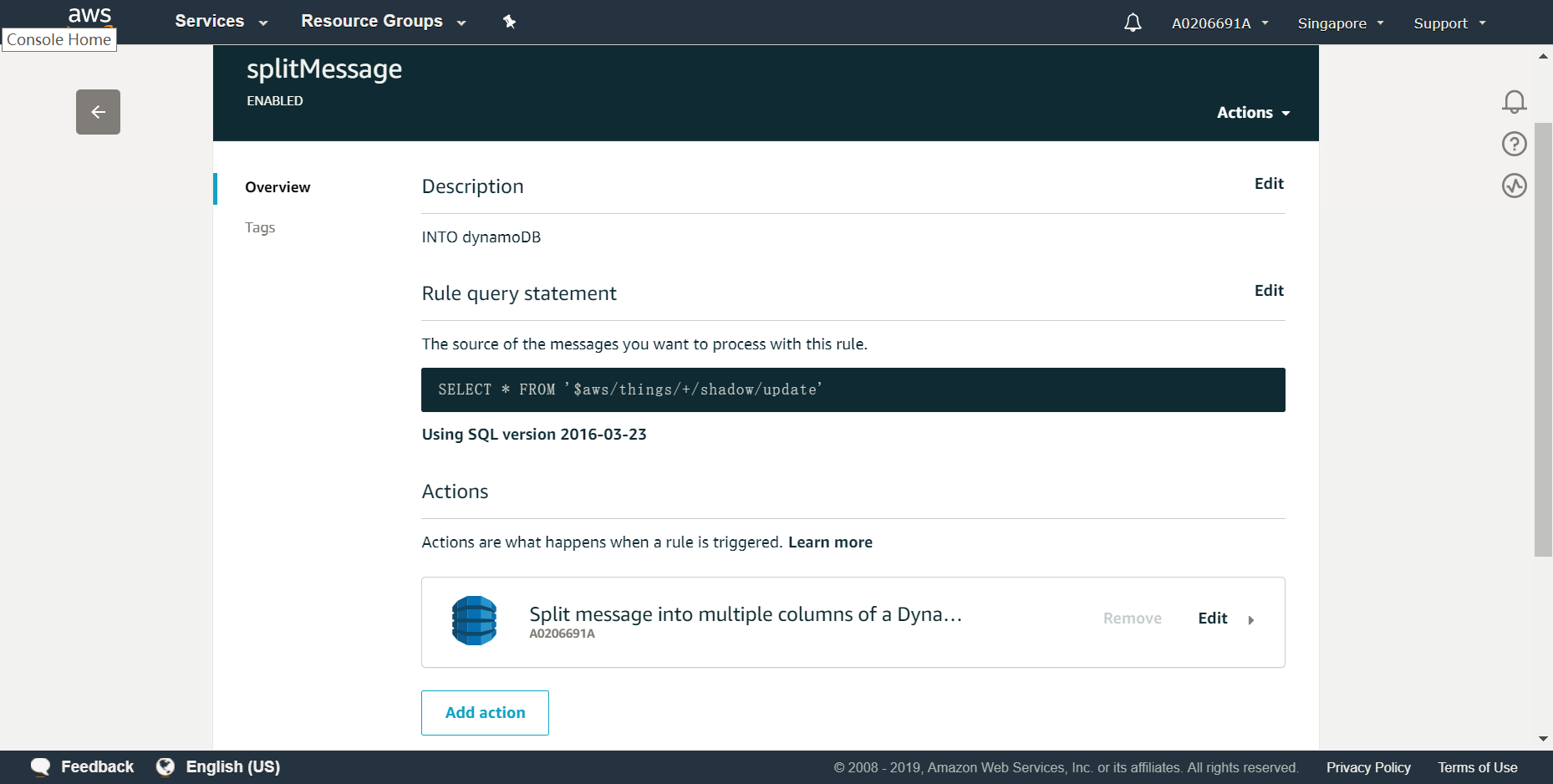
1. Leave the rest of this page unchanged, and then choose Next.
2. On the Add a certificate for the thing page, choose Create certificate.
3. For A certificate for this thing, choose Download. (Although the dialog box shows a file ending in cert.pem, the file download ends in certificate.pem.crt.txt.)
4. Repeat the previous step in this section for A public key, A private key, and A root CA for AWS IoT. Save the files ending in public.pem.key, private.pem.key, and .pem, respectively.
5. Choose Activate.
6. Choose Attach a policy.
7. For Add a policy for my thing, select **POLICYfor1** (0 policies selected changes to 1 policy selected). Then choose Register Thing.

**Create rules and dynamoDB tables:**

DynamoDB rules allow you to take information from an incoming MQTT message and write it to a DynamoDB table.

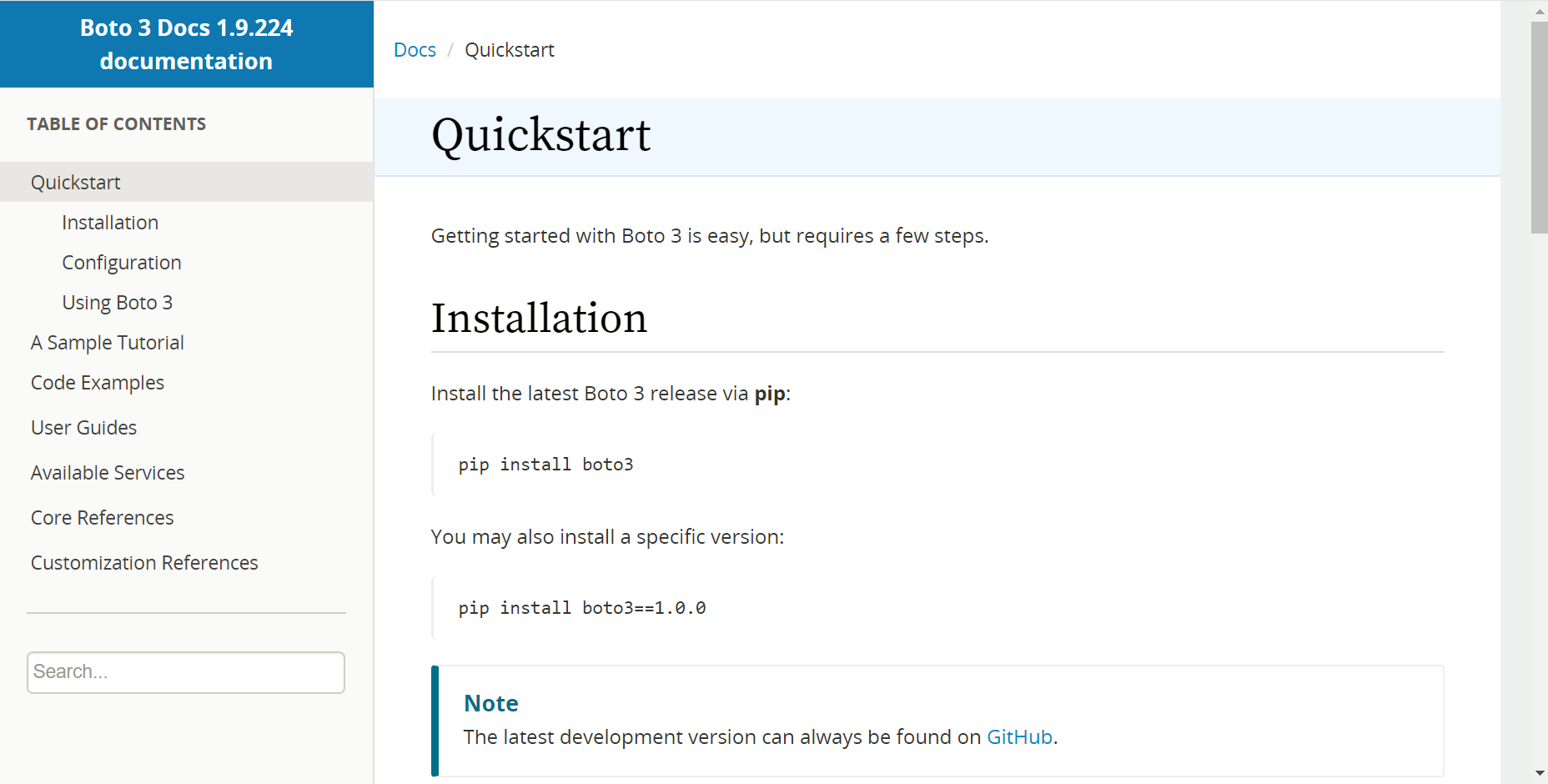
1. In the AWS IoT console, in the navigation pane, choose Act.
2. On the Rules page, choose Create.
3. On the Create a rule page, enter a name for my rule, i.e, **splitMessage**.
4. Under Rule query statement, choose the latest version from the Using SQL version list. For Rule query statement, enter:

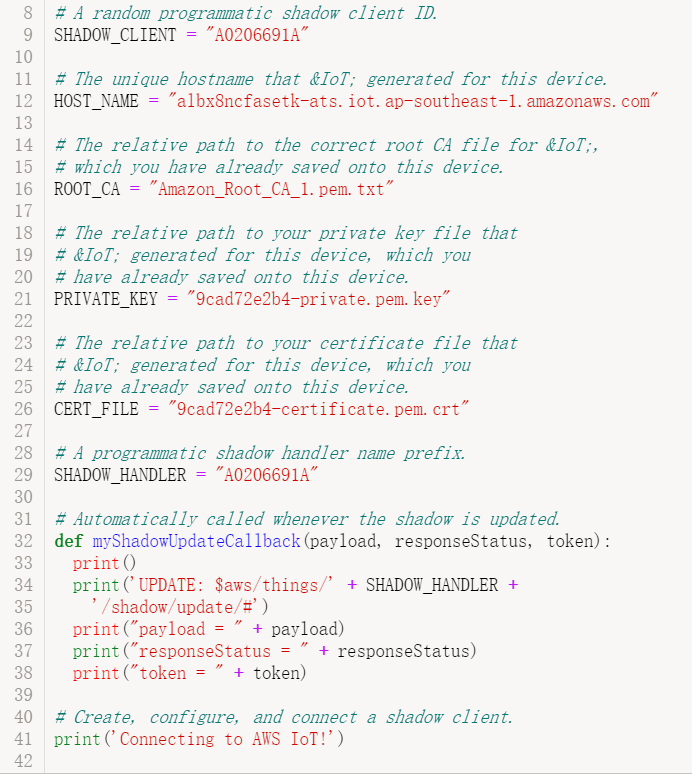
**SELECT \* FROM '$aws/things/+/shadow/update'**

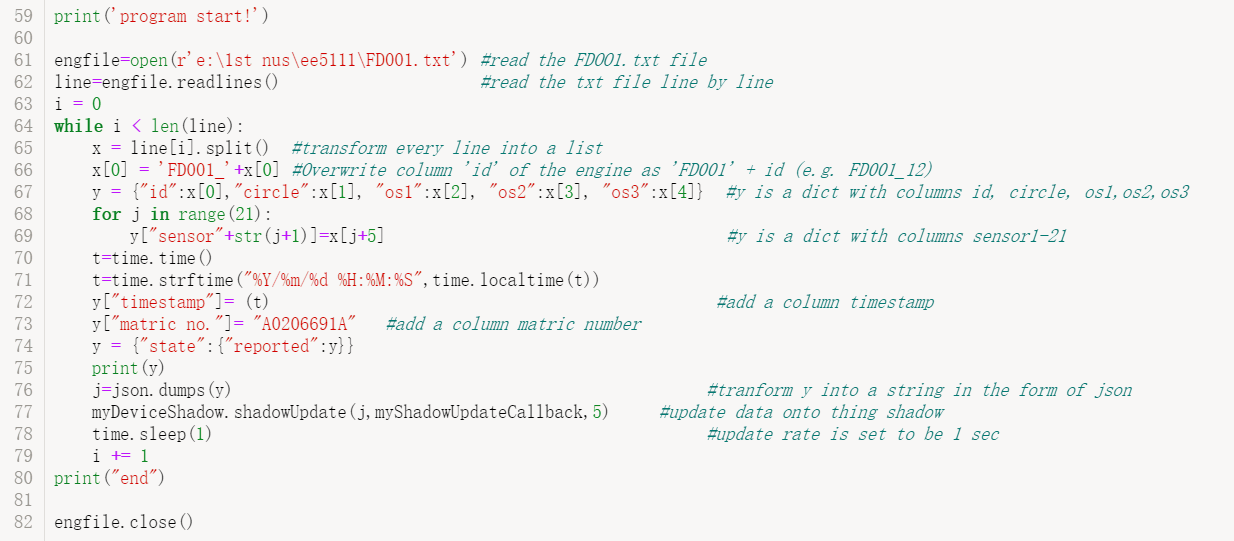
1. Choose Add action.
2. On the Select an action page, choose **Split message into multiple columns of a DynamoDB table**, and then choose Configure action.
3. On the Configure action page, choose Create a new resource.
4. On the Amazon DynamoDB page, choose Create table.
5. On the Create DynamoDB table page, enter a name in Table name,ie, **A0206691A**. In Partition key, enter **id**. Select Add sort key, and then enter **timestamp** in the Sort key field. Choose String for both the partition and sort keys, and then choose Create.
6. On the Configure action page, choose the new table from the Table name list. In Partition key value, enter ${id}. This instructs the rule to take the value of the id attribute from the MQTT message and write it into the id column in the DynamoDB table. In Sort key value, enter ${timestamp}. This writes the value of the timestamp attribute into the timestamp column. Choose Create a new role, put in a unique name,i.e **A0206691A**.
7. Then choose Create role , Add Action, Create rule, successively. 

**Install boto3:**

I follow the instructions from <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html> to complete installation, configuration and then boto3 can be utilized to make requests and process responses from the service.



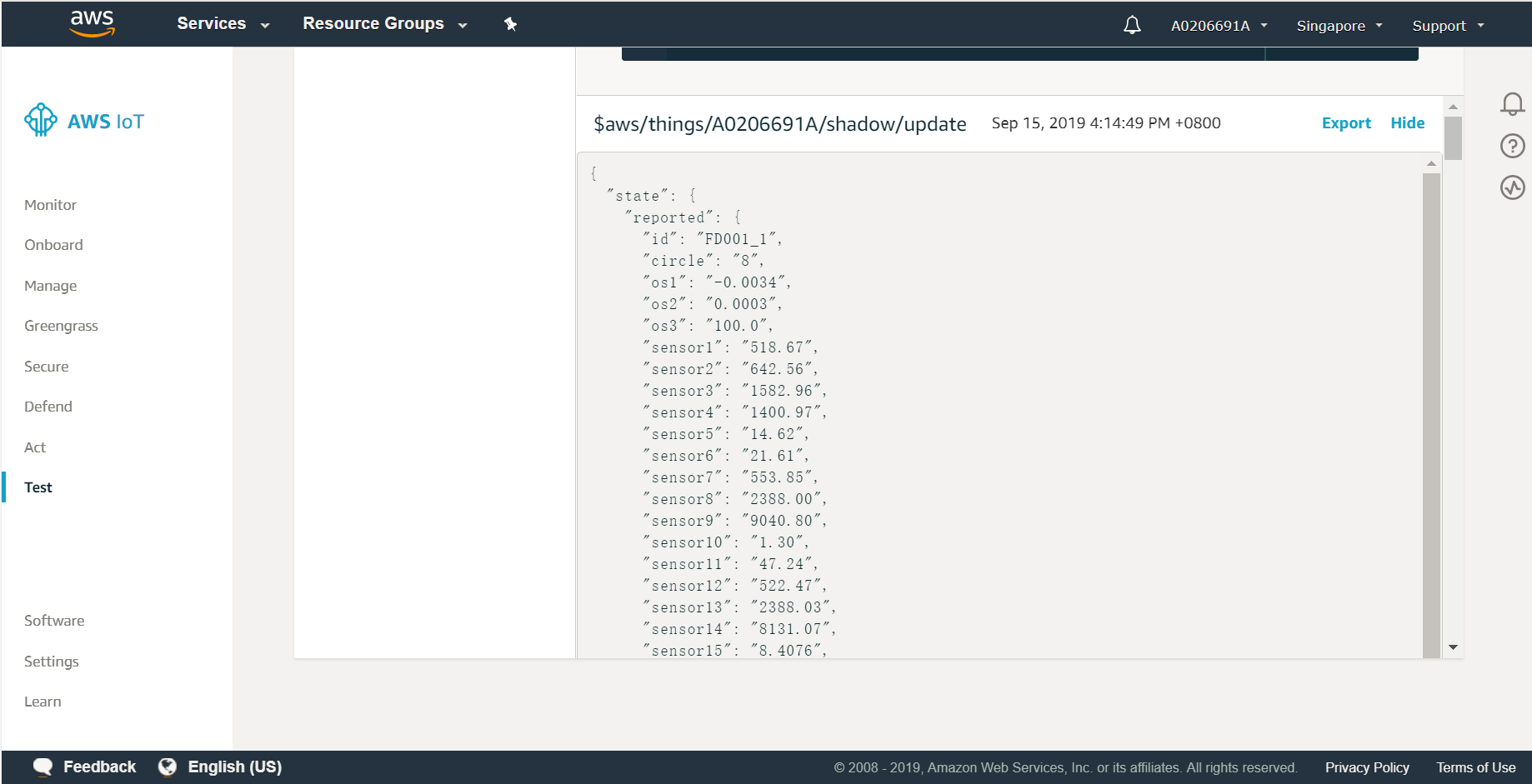
1. Read and publish data of FD001.txt
2. Modify the Jupyter notebook for **planting watering sample** to read and publish data from FD001.txt to the thing under AWS IoT platform(i.e, A0206691A). 
3. Publish at the rate of 10 seconds per row; Overwrite column 'id' of the engine as 'FD001' + id (e.g. FD001\_12); Add one more column 'timestamp' as timestamp in UTC (e.g. UTC 2019-01-28 14:41:15.237); Also add one more column that contains my Matric number(A0206691A).



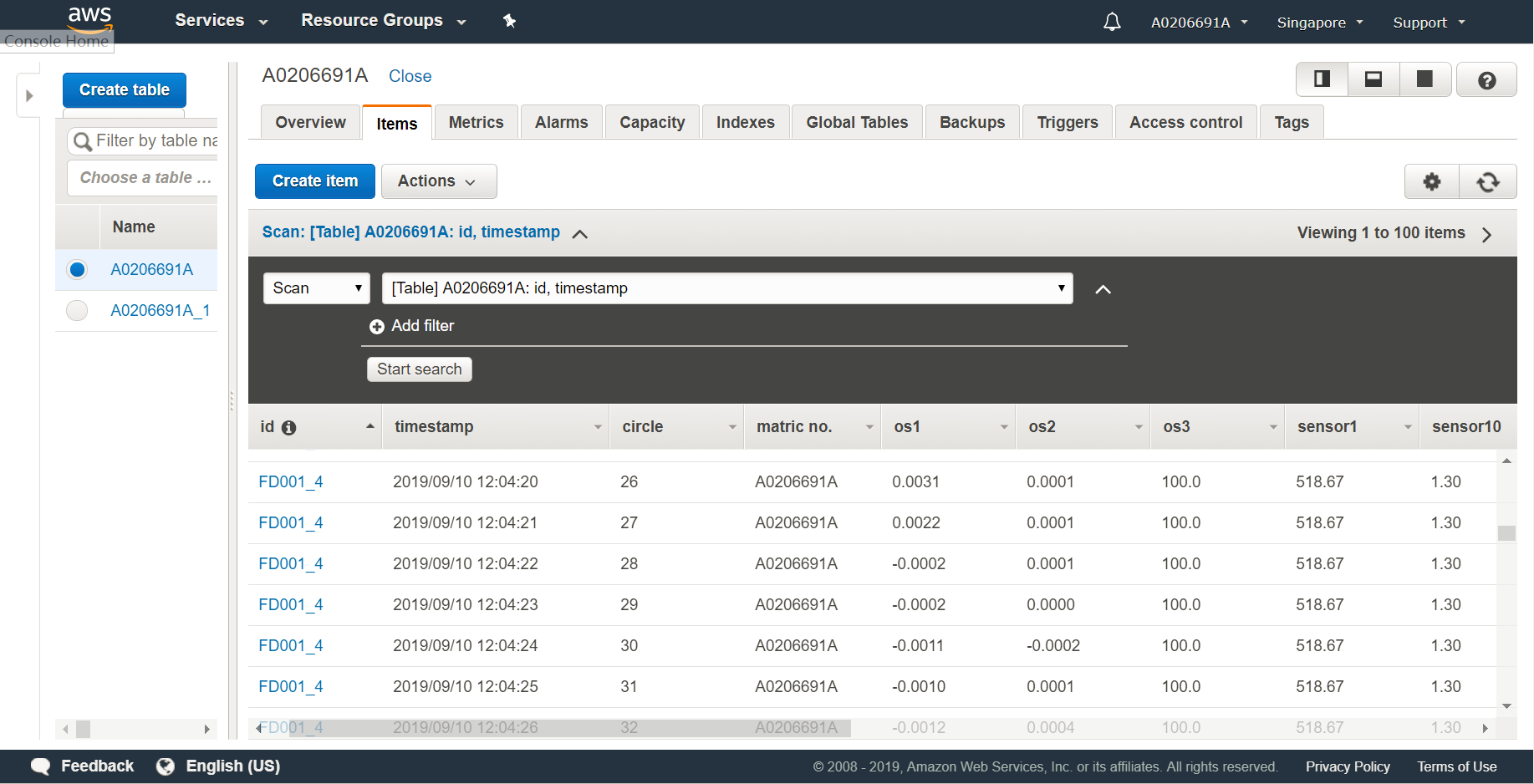
1. Run the jupyter notebook and check the dynamoDB table.



Subscribe to a topic and see the shadow update:



Check the dynamoDB table:

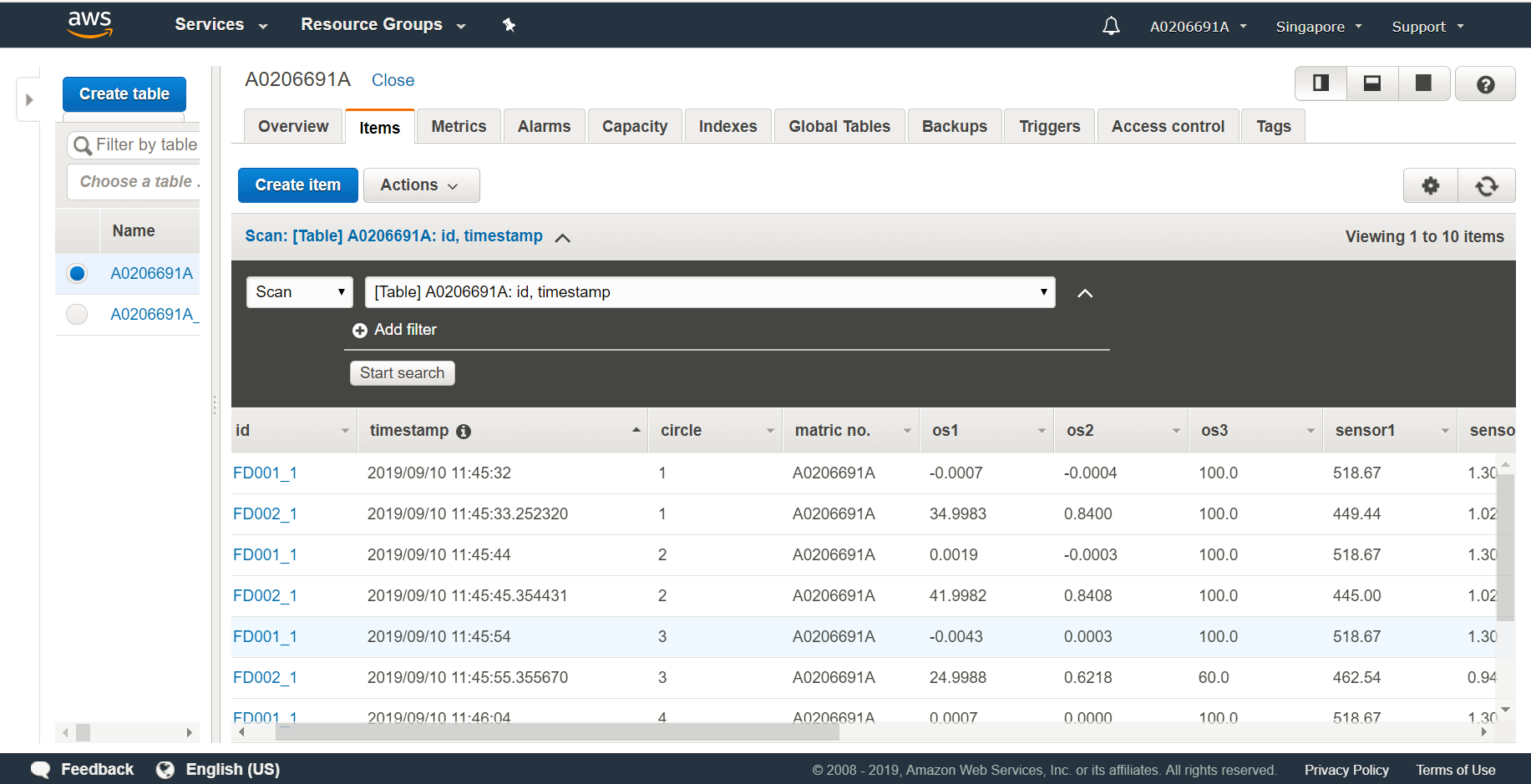


## 2.3 step 3: simulate two IoT things

(1) Add one more thing under AWS IoT platform and one more certificate. Create a copy of the Jupyter notebook above, renaming the client name, certificate, data source FD002.txt.

(2) Let the two Jupyter notebooks run at the same time, simulating the two "things" to run in parallel to publish data with a sampling rate of one record per 1 seconds in each thing.

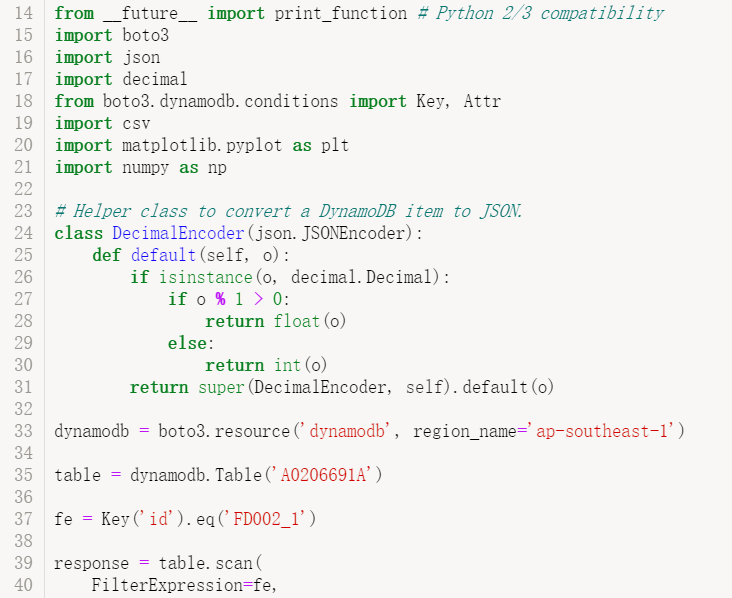
And check the dynamoDB table.



# 3. Data visualize

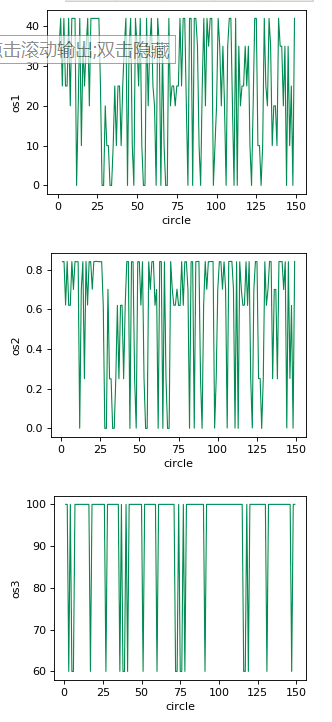
## 3.1 load data

First, connect to my AWS DynamoDB table with a code containing the key and export the data to the jupyter notebook. (for simplicity, I upload around a few hundreds of rows and then stop the jupyter notebook)

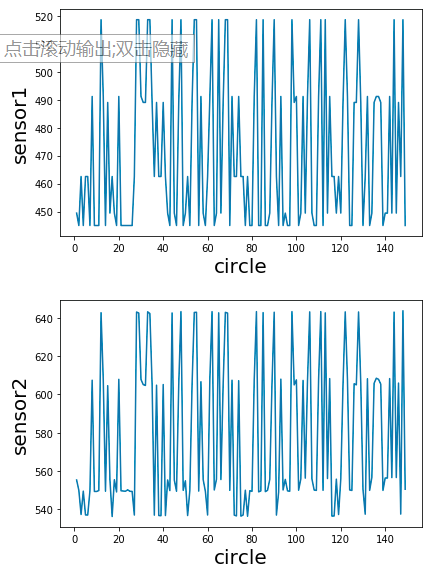


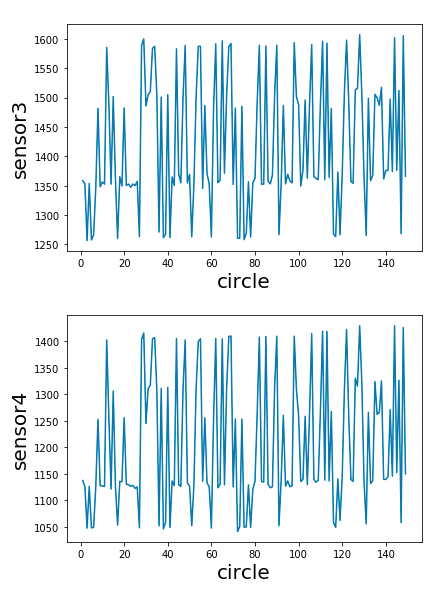
## 3.2 visualize the data

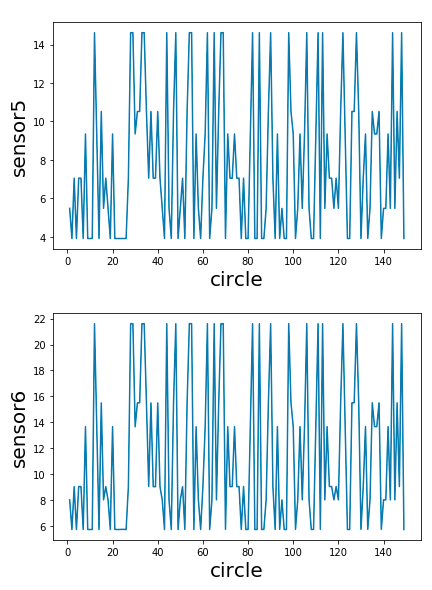
I take the FD002\_1 data as an example. First, I visualize the three os data as the follows:

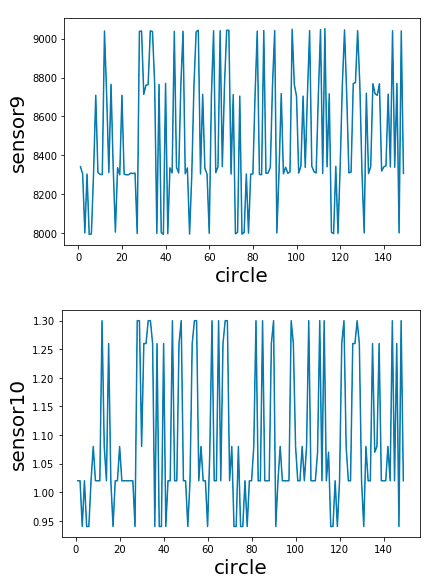
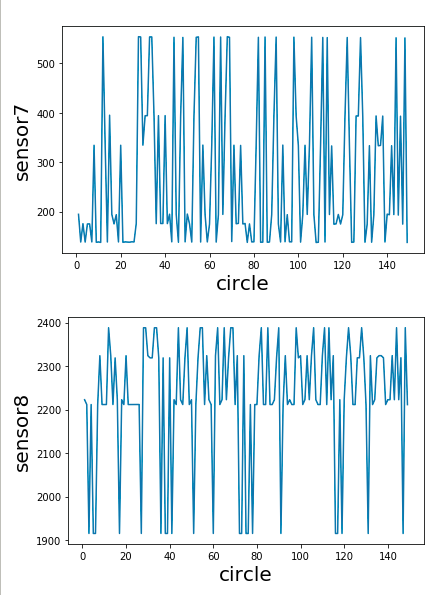


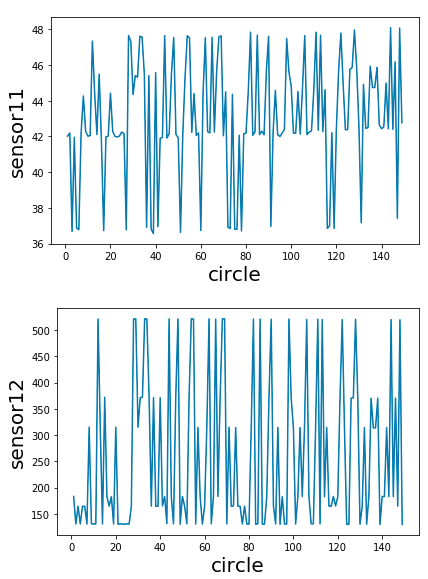
Then, every sensor is visualized through a line chart as the follows:

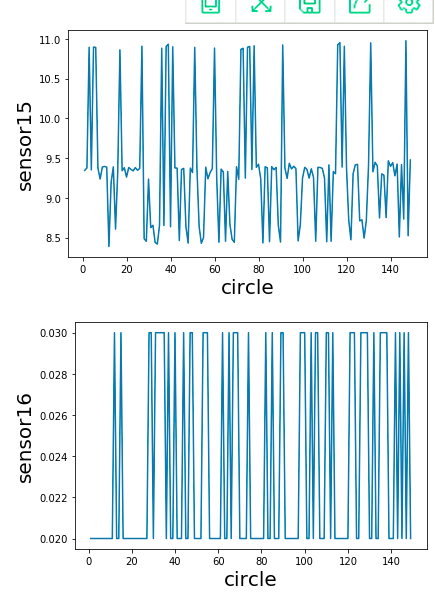
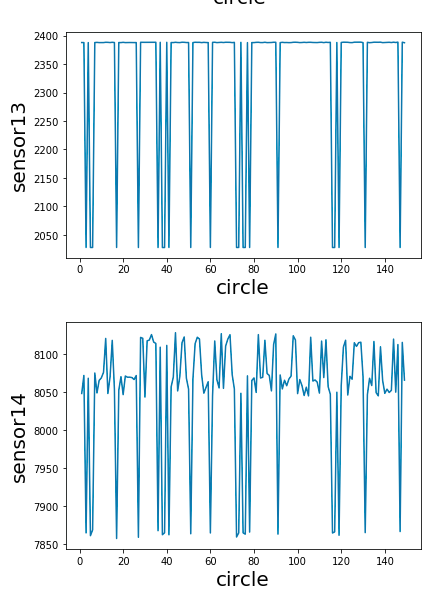


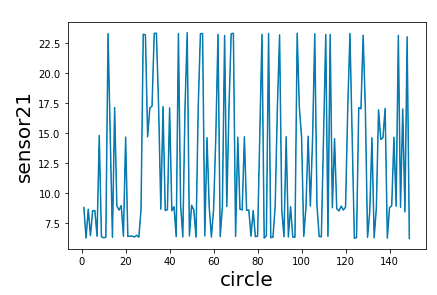
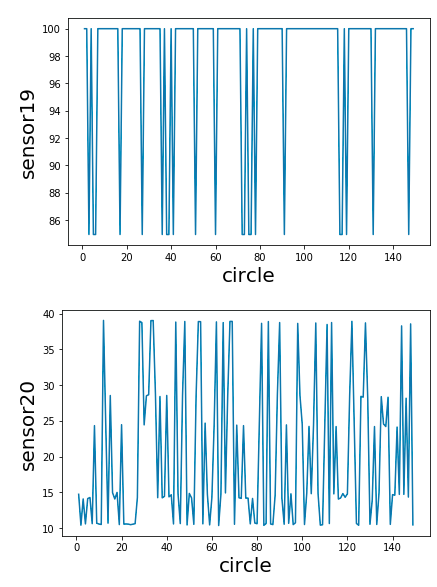
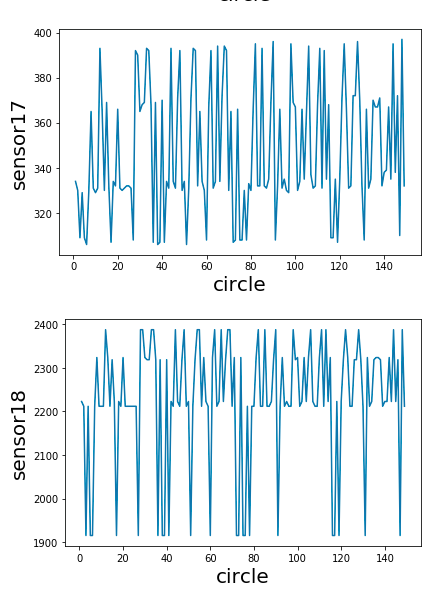






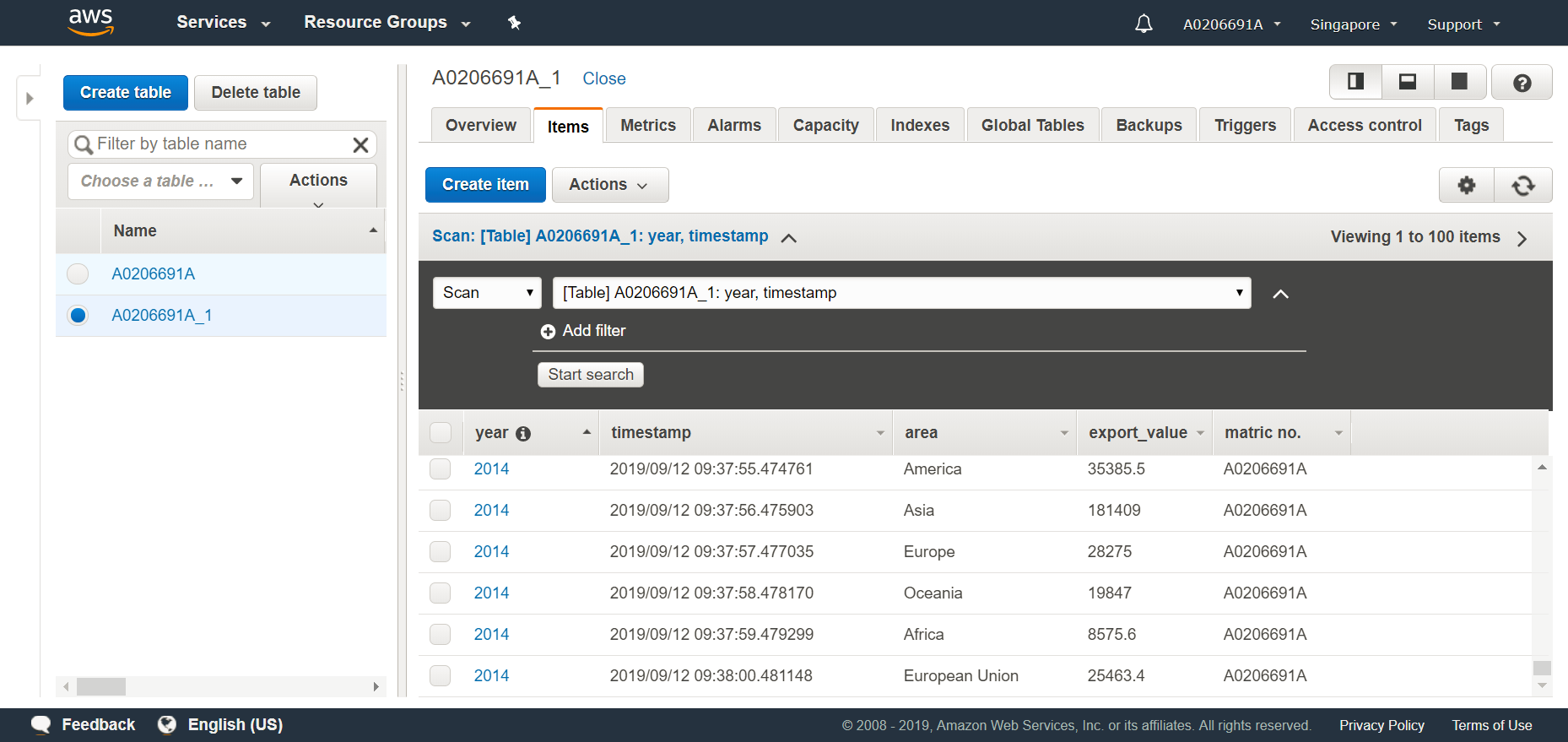




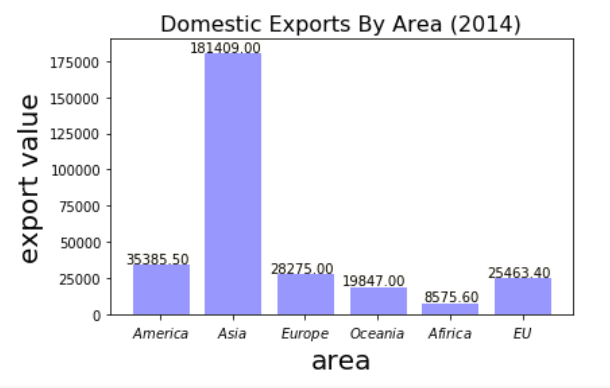


# 4. Use other data sources

I downloaded another data source of domestic exports by area from data.gov.sg. The data gives informatin about the domestic export from 1980 to 2014 with America, Aisa, Europe, Oceania, Afirica and EU. And the following is the dynamoDB for the export data.



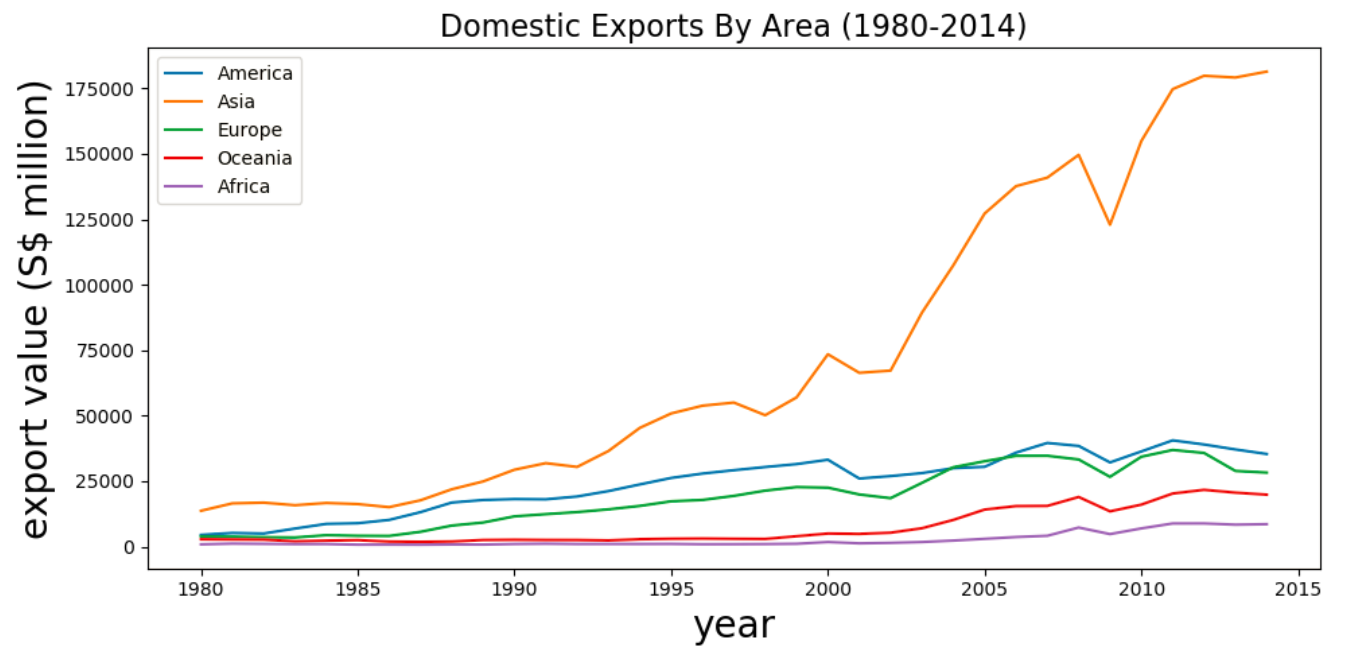
I extract the data in 2014 and plot a histogram as pucture below :



Then, I extract the data in 2013 and plot a pie chart as follows (with corresponding codes):



At last, all data value are used to show the export data from 1980 to 2014 with each area dipicted in different color.



1. **Conclusion** 
   1. **potential improvements**

With more time and space, some improvements in data analysis may be added, such as data prediction with regression models like Lasso or decision tree.

* 1. **summary**

This project is a very intresting and practical one, because I learned the basic knowledge of IoT , learned what is AWS and how to use it, learned how to use python with jupyter all for the first time.

Although I faced many problems in the project, the experience teaches me a lot, which is valuable and significant for my future study and life.

Medium link: <https://medium.com/p/2aad59a74d57/edit>

Code source: <https://github.com/Lesleychen/chenlesley001/tree/master>