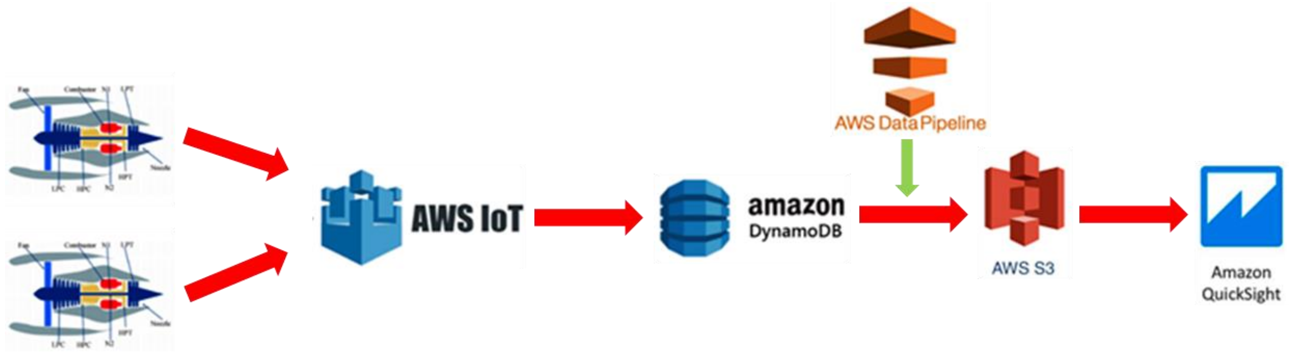
National University of Singapore

EE5111 Selected Topics in Industrial Control & Instrumentation

Project: IoT project on Amazon AWS



Name: CHOONG SEAN FEN
Matriculation Number: A0103783H

Name: YANG ZHAO
Matriculation Number: A0103729H

DATE: Sep 20, 2019

Report Link: https://github.com/A0103729/EE5111_IoT-project-on-Amazon-AWS_A0103783H-A0103729H/commit/b0a90267a4df7a569610f3bd159a2c755d256591

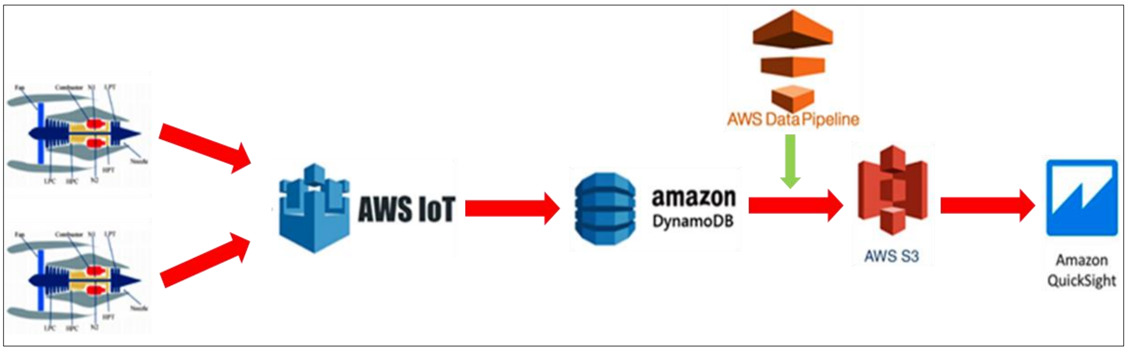
Code Link: https://github.com/A0103729/Q2-3-5-6-coding

Contents

1. Project Objective and Briefing

- **PART I:**

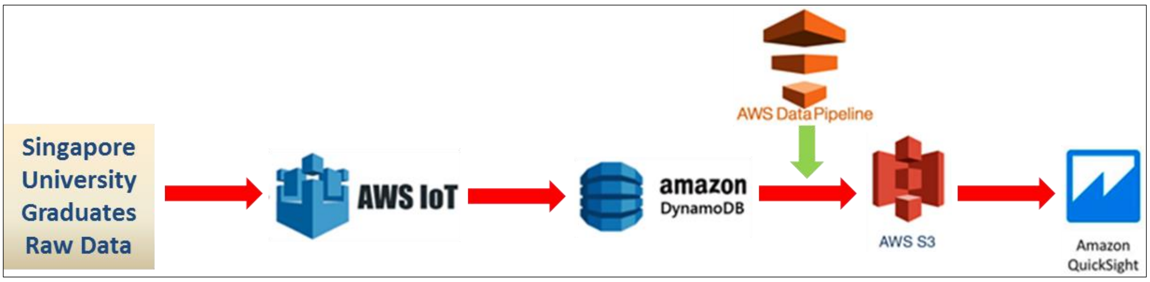
  We simulated one and two small IoT things record and push data from two jet engines and visualizing the data through Amazon quick sight.



- **PART II:**

  We simulated Singapore University Graduates Raw Data record and push from data.gov.sg website and visualizing the data through Amazon quick sight.

  https://data.gov.sg/dataset/graduates-from-university-first-degree-courses-by-type-of-course

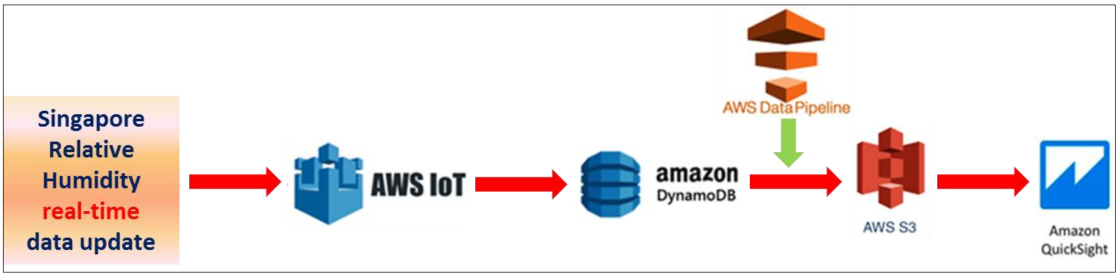


- **PART III:**

  We simulated real-time data from embedded systems with AWS Cloud platform and visualize the data through Amazon quick sight.

  Real-time data name: Singapore Relative Humidity - Monthly Absolute Extreme Minimum

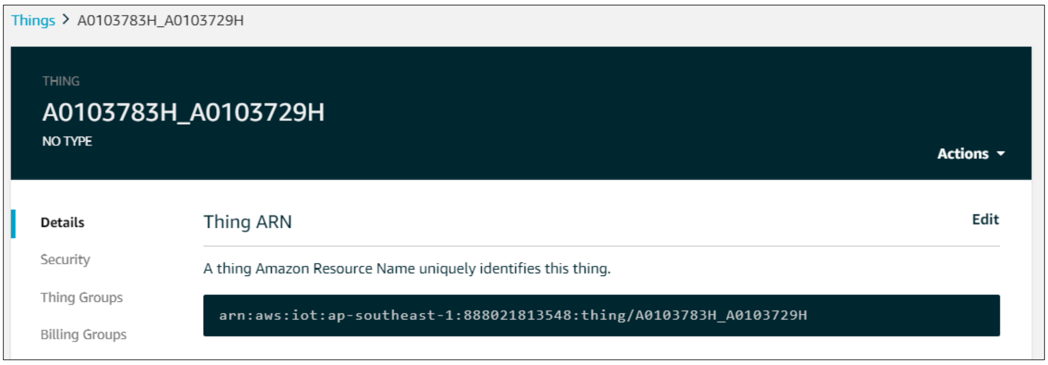
  Real-time data API:
  https://data.gov.sg/api/action/datastore_search?resource_id=585c24a5-76cd-4c48-9341-9223de5adc1d&q=99999



2. Simulation of Publish predefined engine data to AWS

1) Publish one “thing” pre-defined engine data to AWS.

i. Thing, Certificate, Policy, Rules and DynamoDB table Set up.



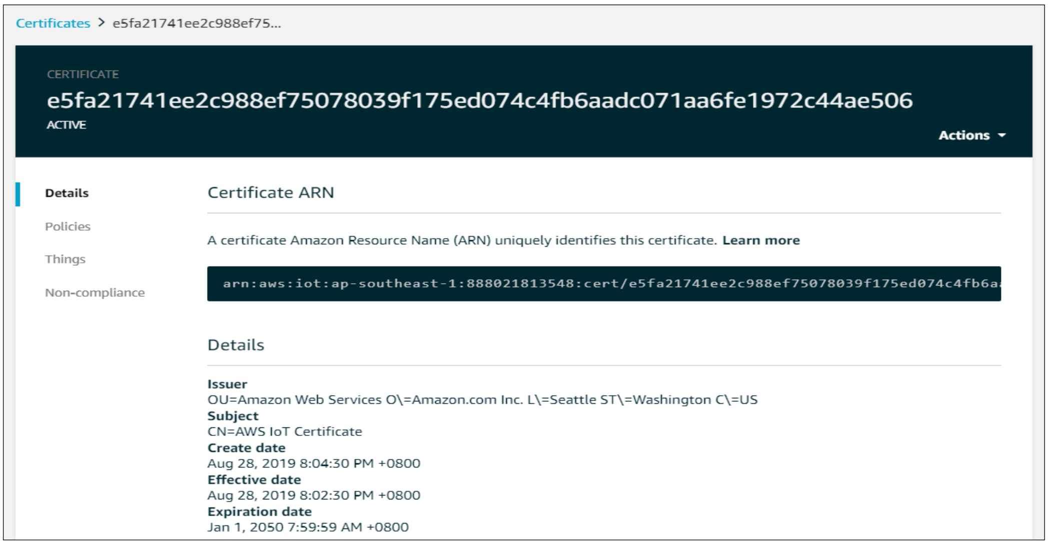
A0103783H_A0103729H

NO TYPE

Thing ARN

A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:ap-southeast-1:888021813548:thing/A0103783H_A0103729H
```



e5fa21741ee2c988ef75078039f175ed074c4fb6aadc071aa6fe1972c44ae506

ACTIVE

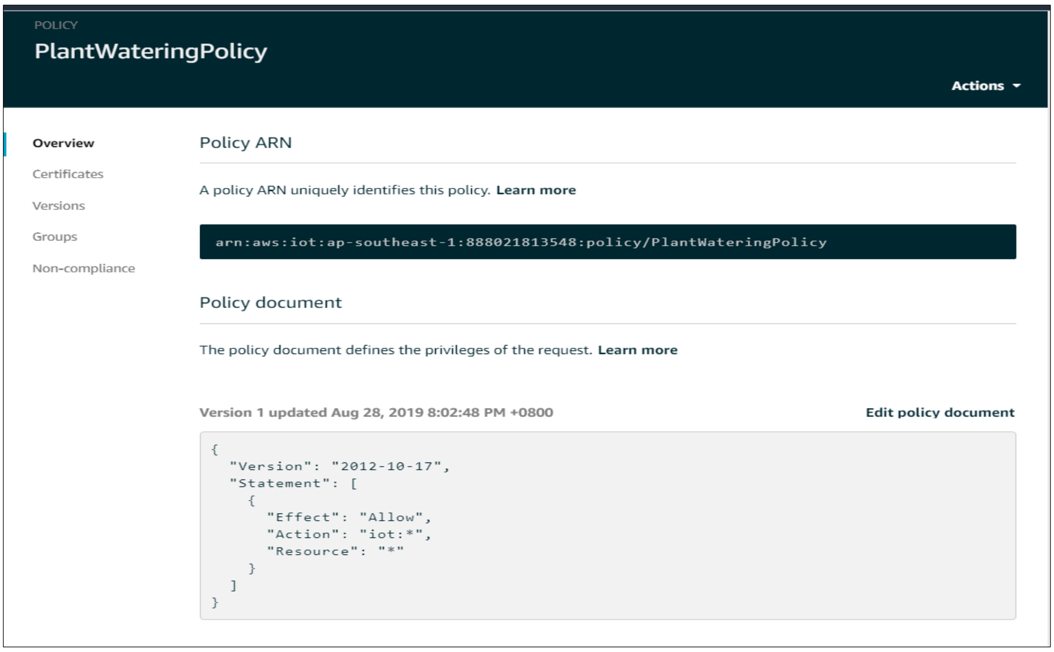
Certificate ARN

A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:ap-southeast-1:888021813548:cert/e5fa21741ee2c988ef75078039f175ed074c4fb6a
```

Details

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Aug 28, 2019 8:04:30 PM +0800
Effective date
Aug 28, 2019 8:02:30 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800



PlantWateringPolicy

Policy ARN

A policy ARN uniquely identifies this policy. **Learn more**

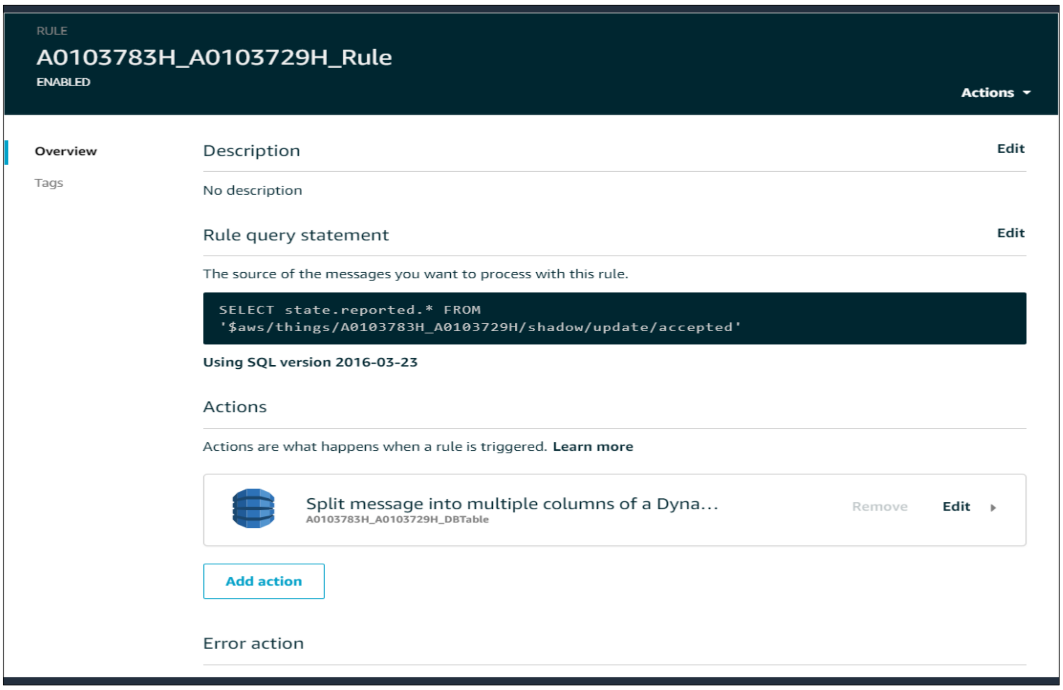
```
arn:aws:iot:ap-southeast-1:888021813548:policy/PlantWateringPolicy
```

Policy document

The policy document defines the privileges of the request. **Learn more**

Version 1 updated Aug 28, 2019 8:02:48 PM +0800

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```



RULE

A0103783H_A0103729H_Rule

ENABLED

Description

No description

Rule query statement

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

```
SELECT state.reported.* FROM
'$aws/things/A0103783H_A0103729H/shadow/update/accepted'
```

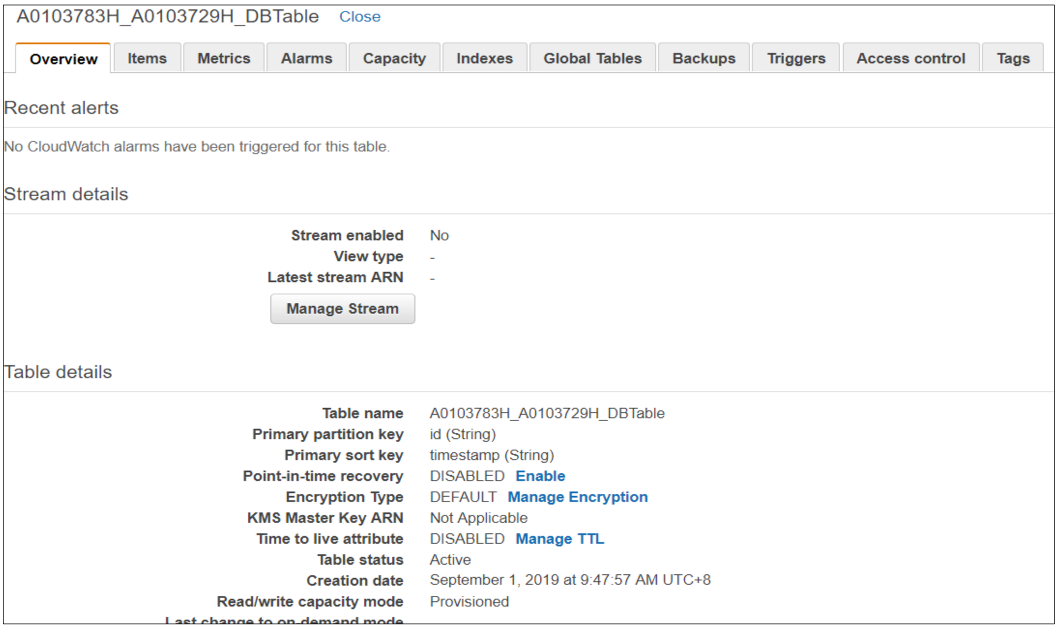
Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. **Learn more**

Split message into multiple columns of a Dyna...
A0103783H_A0103729H_DBTable

Error action



A0103783H_A0103729H_DBTable

Recent alerts

No CloudWatch alarms have been triggered for this table.

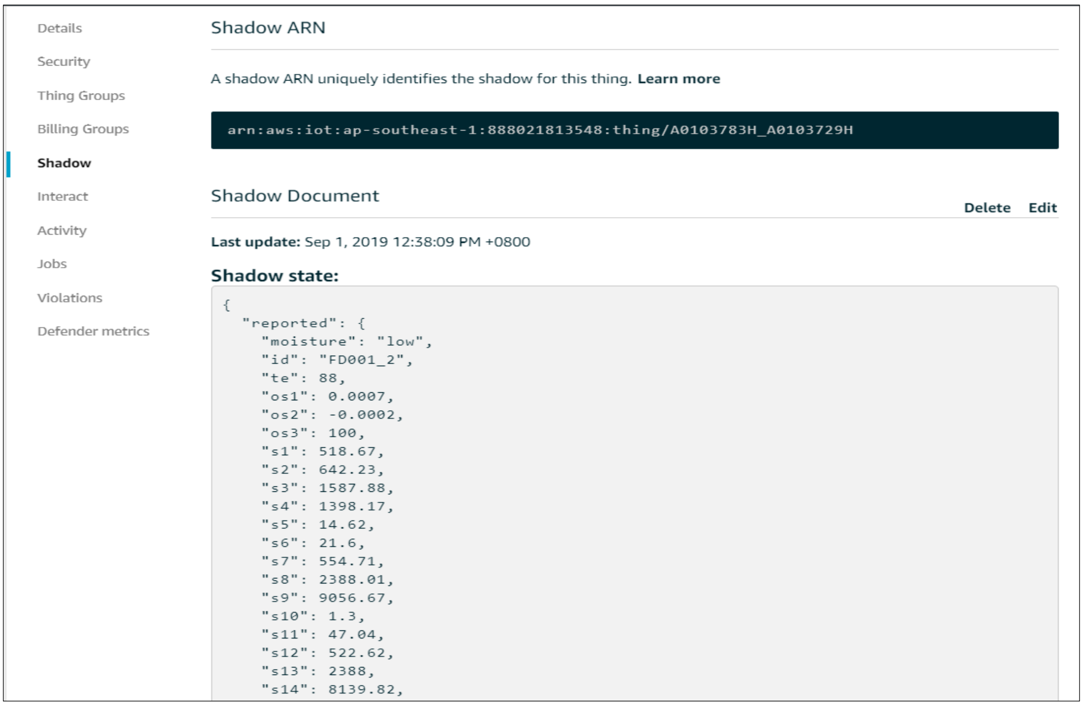
Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Table details

Table name	A0103783H_A0103729H_DBTable
Primary partition key	id (String)
Primary sort key	timestamp (String)
Point-in-time recovery	DISABLED **Enable**
Encryption Type	DEFAULT **Manage Encryption**
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED **Manage TTL**
Table status	Active
Creation date	September 1, 2019 at 9:47:57 AM UTC+8
Read/write capacity mode	Provisioned
Last change to on-demand mode	

ii. Output in AWS & DynamoDB table.



Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing. **Learn more**

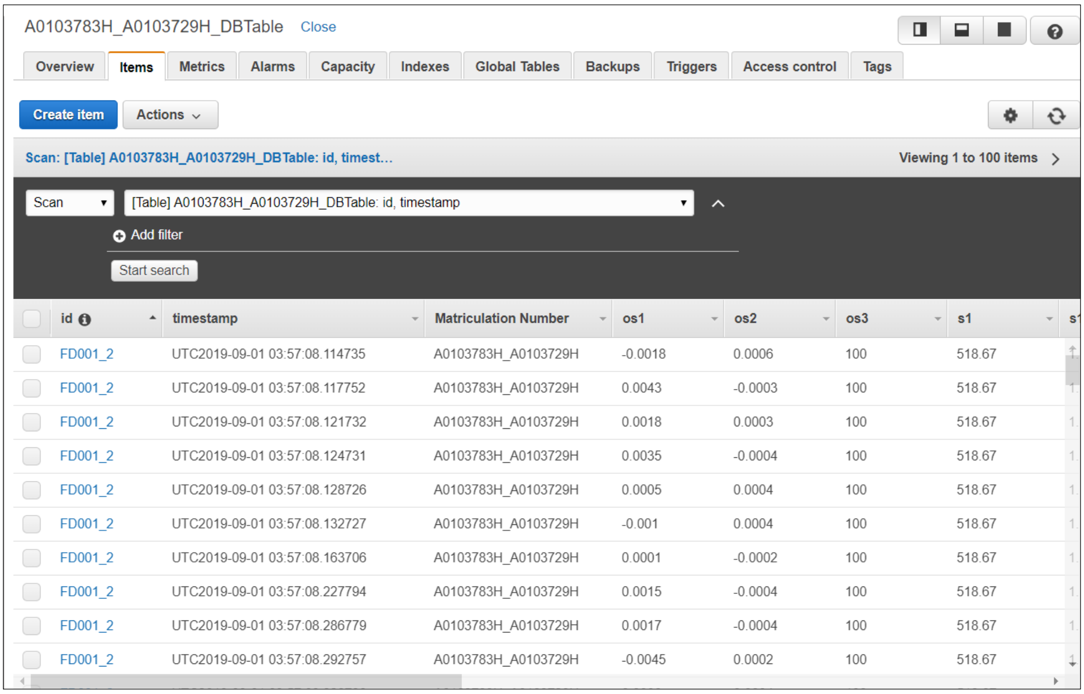
```
arn:aws:iot:ap-southeast-1:888021813548:thing/A0103783H_A0103729H
```

Shadow Document

Last update: Sep 1, 2019 12:38:09 PM +0800

Shadow state:

```
{
  "reported": {
    "moisture": "low",
    "id": "FD001_2",
    "te": 88,
    "os1": 0.0007,
    "os2": -0.0002,
    "os3": 100,
    "s1": 518.67,
    "s2": 642.23,
    "s3": 1587.88,
    "s4": 1398.17,
    "s5": 14.62,
    "s6": 21.6,
    "s7": 554.71,
    "s8": 2388.01,
    "s9": 9056.67,
    "s10": 1.3,
    "s11": 47.04,
    "s12": 522.62,
    "s13": 2388,
    "s14": 8139.82,
```



A0103783H_A0103729H_DBTable

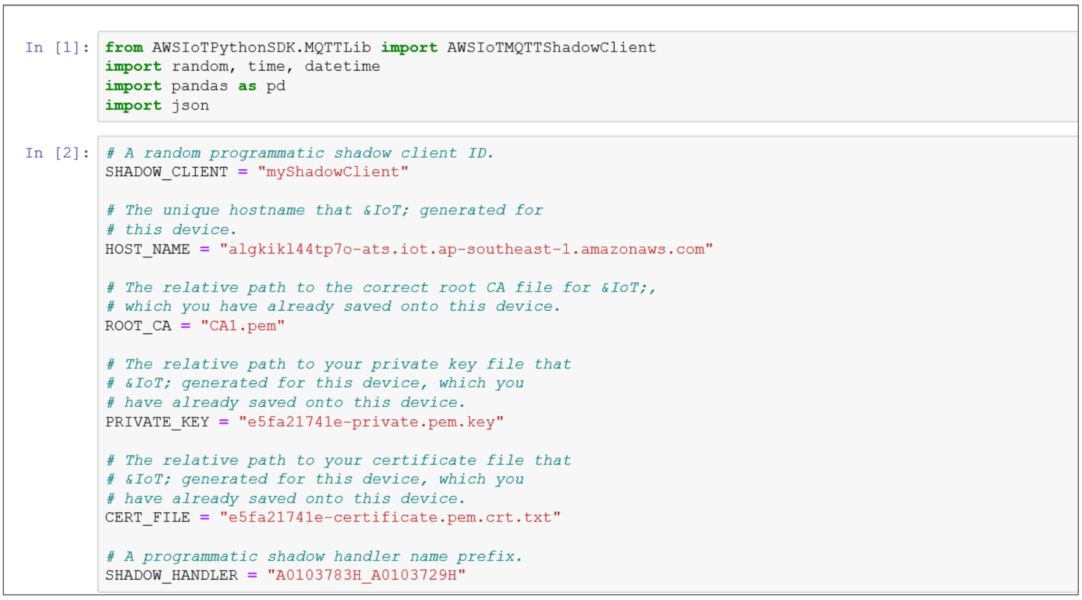
Scan: [Table] A0103783H_A0103729H_DBTable: id, timest...

Viewing 1 to 100 items

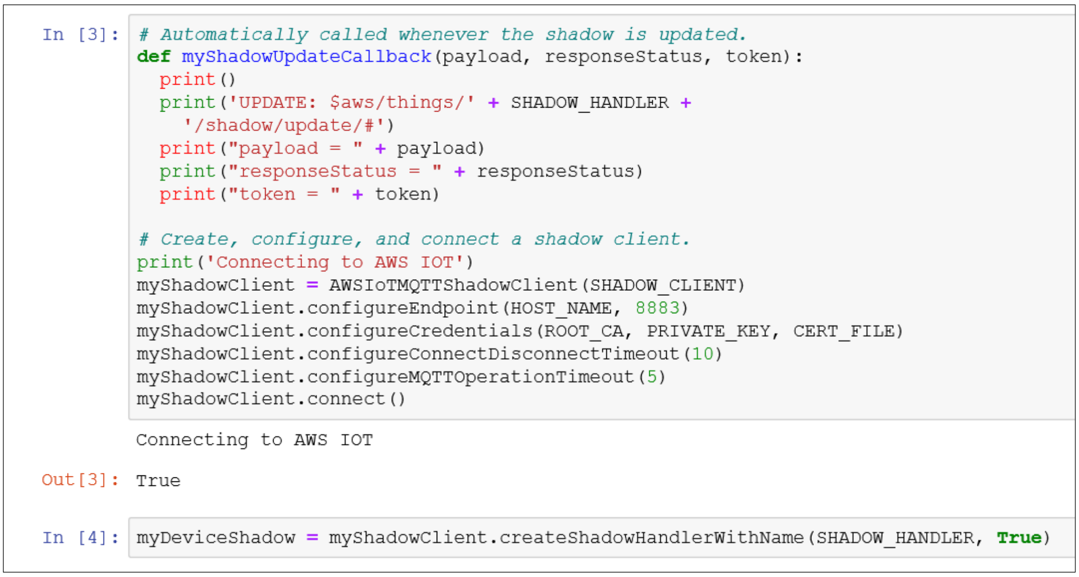
Scan: [Table] A0103783H_A0103729H_DBTable: id, timestamp

id	timestamp	Matriculation Number	os1	os2	os3	s1
FD001_2	UTC2019-09-01 03:57:08.114735	A0103783H_A0103729H	-0.0018	0.0006	100	518.67
FD001_2	UTC2019-09-01 03:57:08.117752	A0103783H_A0103729H	0.0043	-0.0003	100	518.67
FD001_2	UTC2019-09-01 03:57:08.121732	A0103783H_A0103729H	0.0018	0.0003	100	518.67
FD001_2	UTC2019-09-01 03:57:08.124731	A0103783H_A0103729H	0.0035	-0.0004	100	518.67
FD001_2	UTC2019-09-01 03:57:08.128726	A0103783H_A0103729H	0.0005	0.0004	100	518.67
FD001_2	UTC2019-09-01 03:57:08.132727	A0103783H_A0103729H	-0.001	0.0004	100	518.67
FD001_2	UTC2019-09-01 03:57:08.163706	A0103783H_A0103729H	0.0001	-0.0002	100	518.67
FD001_2	UTC2019-09-01 03:57:08.227794	A0103783H_A0103729H	0.0015	-0.0004	100	518.67
FD001_2	UTC2019-09-01 03:57:08.286779	A0103783H_A0103729H	0.0017	-0.0004	100	518.67
FD001_2	UTC2019-09-01 03:57:08.292757	A0103783H_A0103729H	-0.0045	0.0002	100	518.67

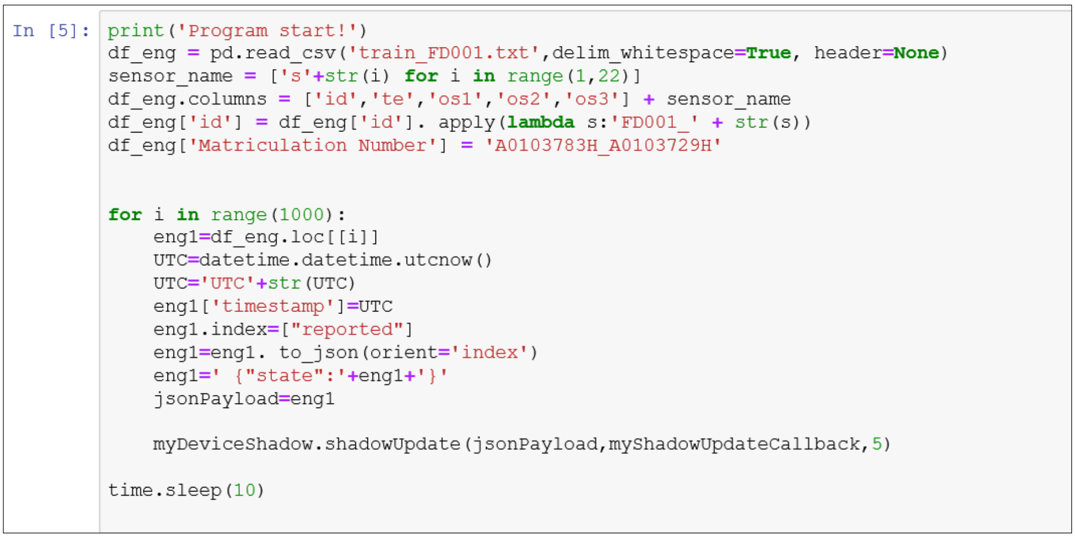
iii. Simulation code



```
In [1]: from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
        import random, time, datetime
        import pandas as pd
        import json

In [2]: # A random programmatic shadow client ID.
        SHADOW_CLIENT = "myShadowClient"

        # The unique hostname that &IoT; generated for
        # this device.
        HOST_NAME = "algkikl44tp7o-ats.iot.ap-southeast-1.amazonaws.com"

        # The relative path to the correct root CA file for &IoT;,
        # which you have already saved onto this device.
        ROOT_CA = "CA1.pem"

        # The relative path to your private key file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        PRIVATE_KEY = "e5fa21741e-private.pem.key"

        # The relative path to your certificate file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        CERT_FILE = "e5fa21741e-certificate.pem.crt.txt"

        # A programmatic shadow handler name prefix.
        SHADOW_HANDLER = "A0103783H_A0103729H"
```



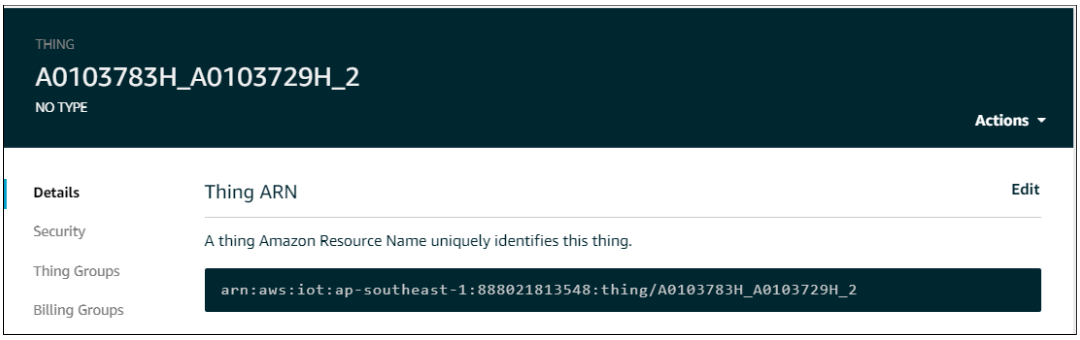
```
In [3]: # Automatically called whenever the shadow is updated.
        def myShadowUpdateCallback(payload, responseStatus, token):
          print()
          print('UPDATE: $aws/things/' + SHADOW_HANDLER +
            '/shadow/update/#')
          print("payload = " + payload)
          print("responseStatus = " + responseStatus)
          print("token = " + token)

        # Create, configure, and connect a shadow client.
        print('Connecting to AWS IOT')
        myShadowClient = AWSIoTMQTTShadowClient(SHADOW_CLIENT)
        myShadowClient.configureEndpoint(HOST_NAME, 8883)
        myShadowClient.configureCredentials(ROOT_CA, PRIVATE_KEY, CERT_FILE)
        myShadowClient.configureConnectDisconnectTimeout(10)
        myShadowClient.configureMQTTOperationTimeout(5)
        myShadowClient.connect()

        Connecting to AWS IOT

Out[3]: True

In [4]: myDeviceShadow = myShadowClient.createShadowHandlerWithName(SHADOW_HANDLER, True)
```



```
In [5]: print('Program start!')
        df_eng = pd.read_csv('train_FD001.txt',delim_whitespace=True, header=None)
        sensor_name = ['s'+str(i) for i in range(1,22)]
        df_eng.columns = ['id','te','os1','os2','os3'] + sensor_name
        df_eng['id'] = df_eng['id']. apply(lambda s:'FD001_' + str(s))
        df_eng['Matriculation Number'] = 'A0103783H_A0103729H'


        for i in range(1000):
            eng1=df_eng.loc[[i]]
            UTC=datetime.datetime.utcnow()
            UTC='UTC'+str(UTC)
            eng1['timestamp']=UTC
            eng1.index=["reported"]
            eng1=eng1. to_json(orient='index')
            eng1=' {"state":'+eng1+'}'
            jsonPayload=eng1

            myDeviceShadow.shadowUpdate(jsonPayload,myShadowUpdateCallback,5)

        time.sleep(10)
```

2) Simulating the two "things" to run in parallel to publish data.

i. Thing, Rules and DynamoDB table Set up.



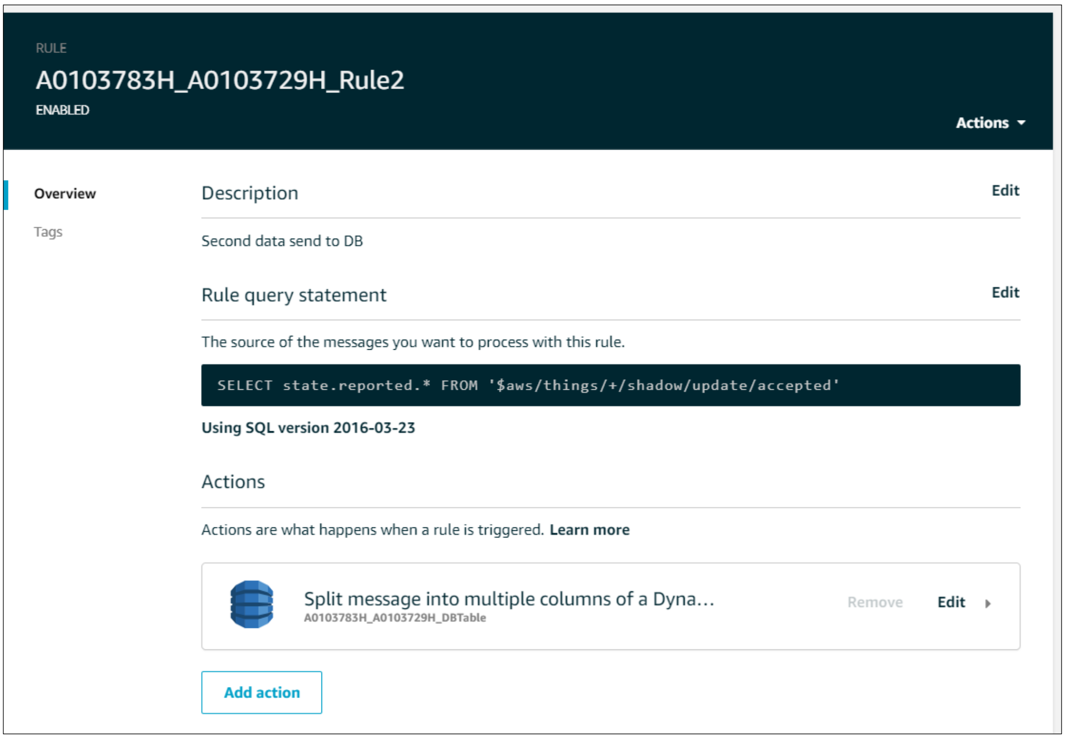
A0103783H_A0103729H_2

NO TYPE

Thing ARN

A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:ap-southeast-1:888021813548:thing/A0103783H_A0103729H_2
```



A0103783H_A0103729H_Rule2

ENABLED

Description

Second data send to DB

Rule query statement

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

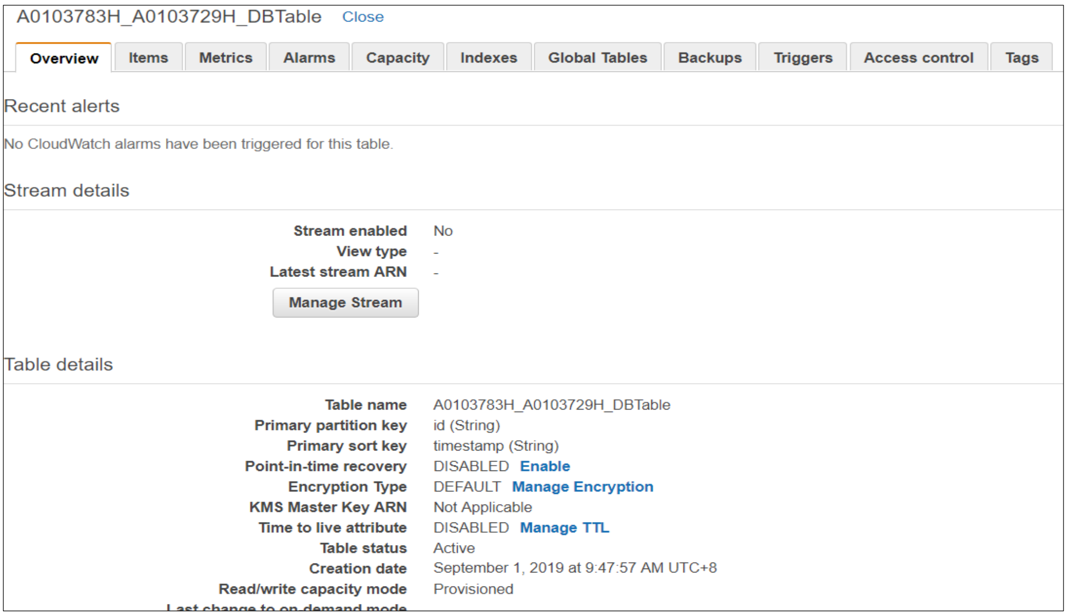
```
SELECT state.reported.* FROM '$aws/things/+/shadow/update/accepted'
```

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. Learn more

Split message into multiple columns of a Dyna...
A0103783H_A0103729H_DBTable



A0103783H_A0103729H_DBTable

Recent alerts

No CloudWatch alarms have been triggered for this table.

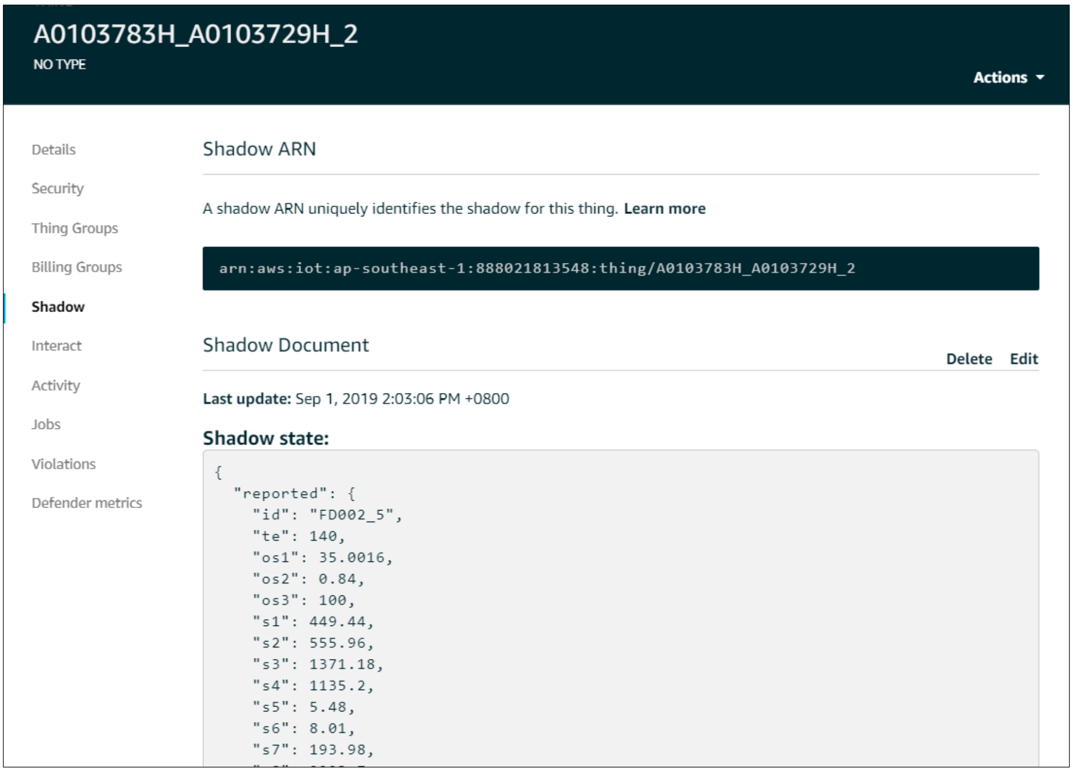
Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Table details

Table name	A0103783H_A0103729H_DBTable
Primary partition key	id (String)
Primary sort key	timestamp (String)
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	September 1, 2019 at 9:47:57 AM UTC+8
Read/write capacity mode	Provisioned
Last change to on-demand mode	

ii. Output in AWS & DynamoDB table.



A0103783H_A0103729H_2

NO TYPE

Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing. **Learn more**

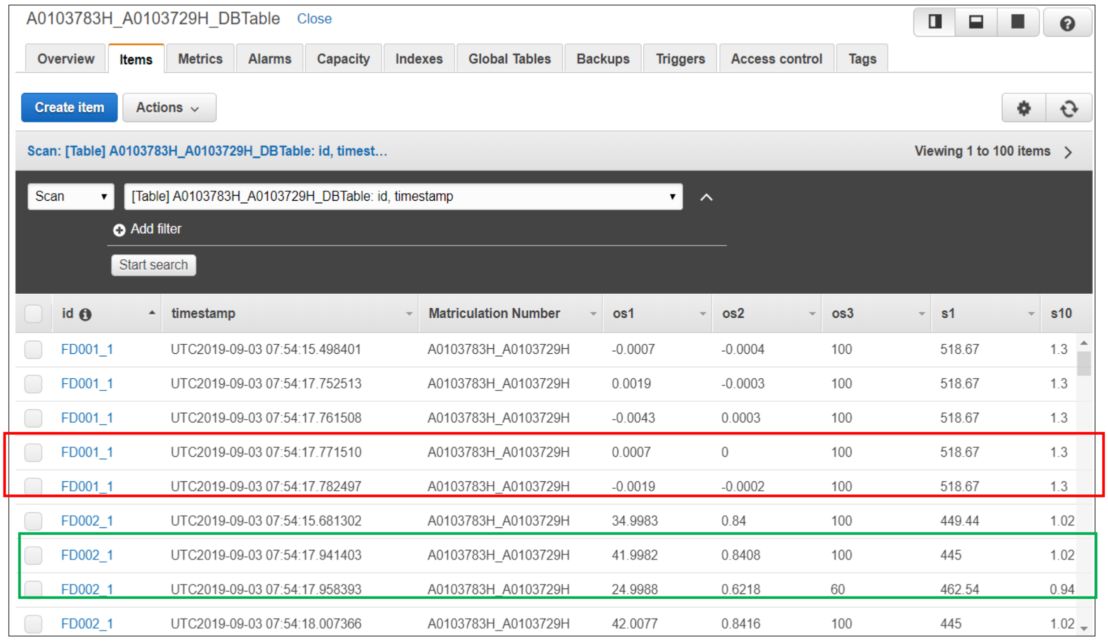
```
arn:aws:iot:ap-southeast-1:888021813548:thing/A0103783H_A0103729H_2
```

Shadow Document

Last update: Sep 1, 2019 2:03:06 PM +0800

Shadow state:

```
{
  "reported": {
    "id": "FD002_5",
    "te": 140,
    "os1": 35.0016,
    "os2": 0.84,
    "os3": 100,
    "s1": 449.44,
    "s2": 555.96,
    "s3": 1371.18,
    "s4": 1135.2,
    "s5": 5.48,
    "s6": 8.01,
    "s7": 193.98,
```



A0103783H_A0103729H_DBTable

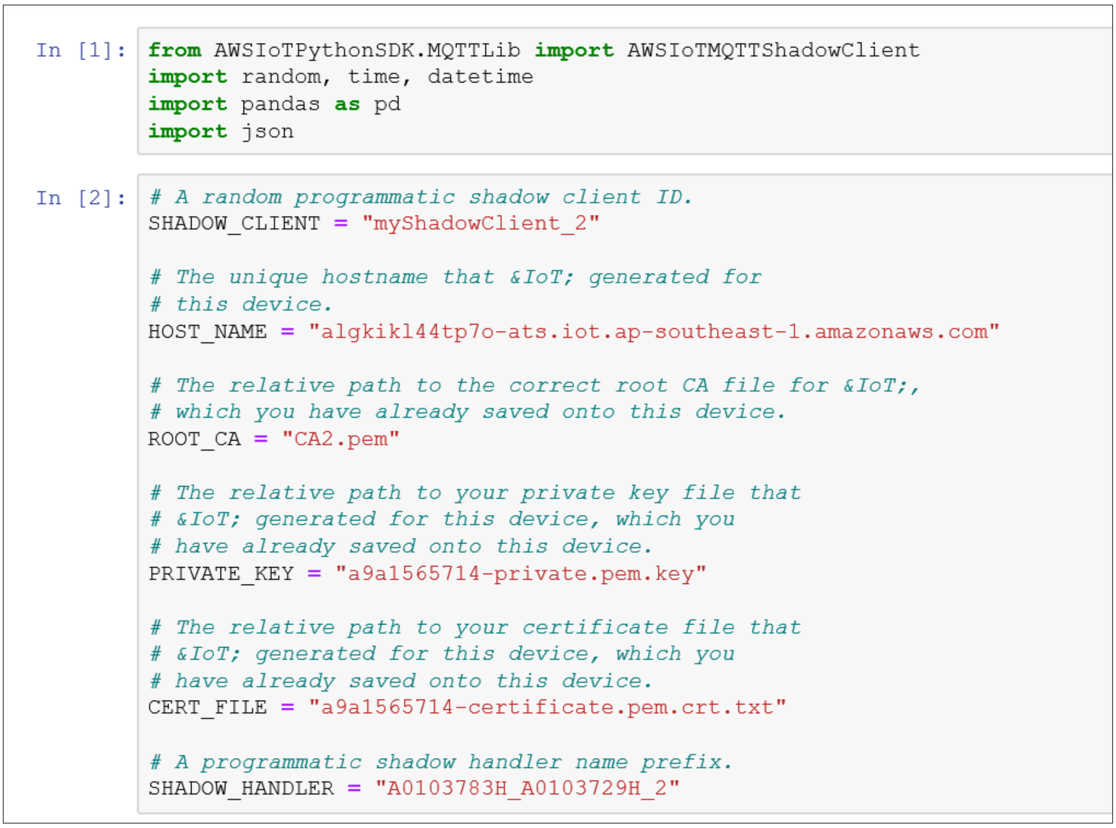
Scan: [Table] A0103783H_A0103729H_DBTable: id, timest...

Viewing 1 to 100 items

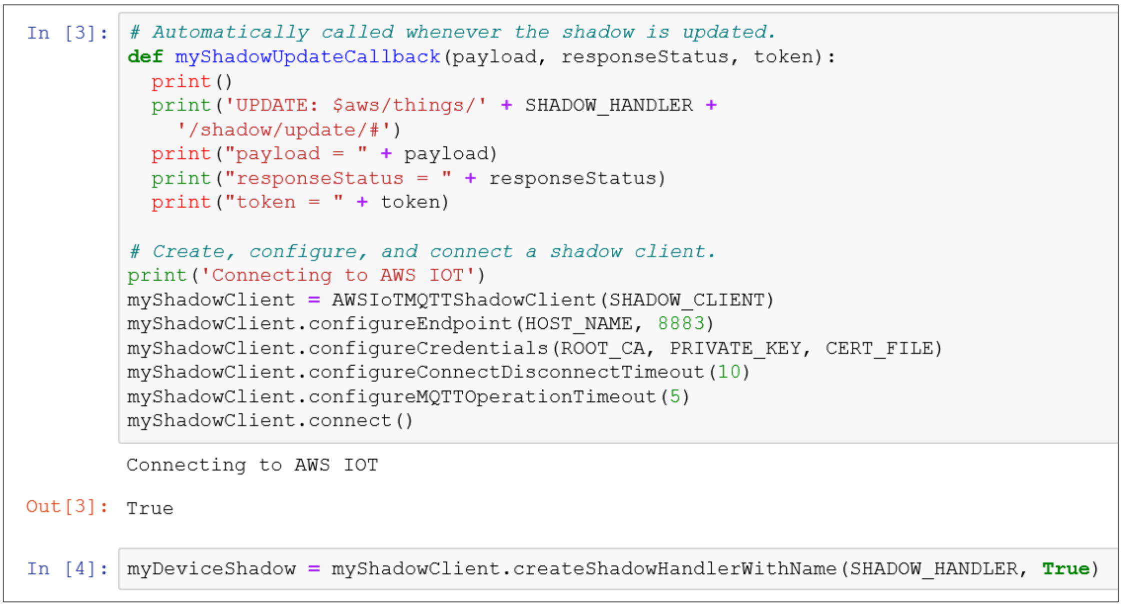
Scan [Table] A0103783H_A0103729H_DBTable: id, timestamp

id	timestamp	Matriculation Number	os1	os2	os3	s1	s10
FD001_1	UTC2019-09-03 07:54:15.498401	A0103783H_A0103729H	-0.0007	-0.0004	100	518.67	1.3
FD001_1	UTC2019-09-03 07:54:17.752513	A0103783H_A0103729H	0.0019	-0.0003	100	518.67	1.3
FD001_1	UTC2019-09-03 07:54:17.761508	A0103783H_A0103729H	-0.0043	0.0003	100	518.67	1.3
FD001_1	UTC2019-09-03 07:54:17.771510	A0103783H_A0103729H	0.0007	0	100	518.67	1.3
FD001_1	UTC2019-09-03 07:54:17.782497	A0103783H_A0103729H	-0.0019	-0.0002	100	518.67	1.3
FD002_1	UTC2019-09-03 07:54:15.681302	A0103783H_A0103729H	34.9983	0.84	100	449.44	1.02
FD002_1	UTC2019-09-03 07:54:17.941403	A0103783H_A0103729H	41.9982	0.8408	100	445	1.02
FD002_1	UTC2019-09-03 07:54:17.958393	A0103783H_A0103729H	24.9988	0.6218	60	462.54	0.94
FD002_1	UTC2019-09-03 07:54:18.007366	A0103783H_A0103729H	42.0077	0.8416	100	445	1.02

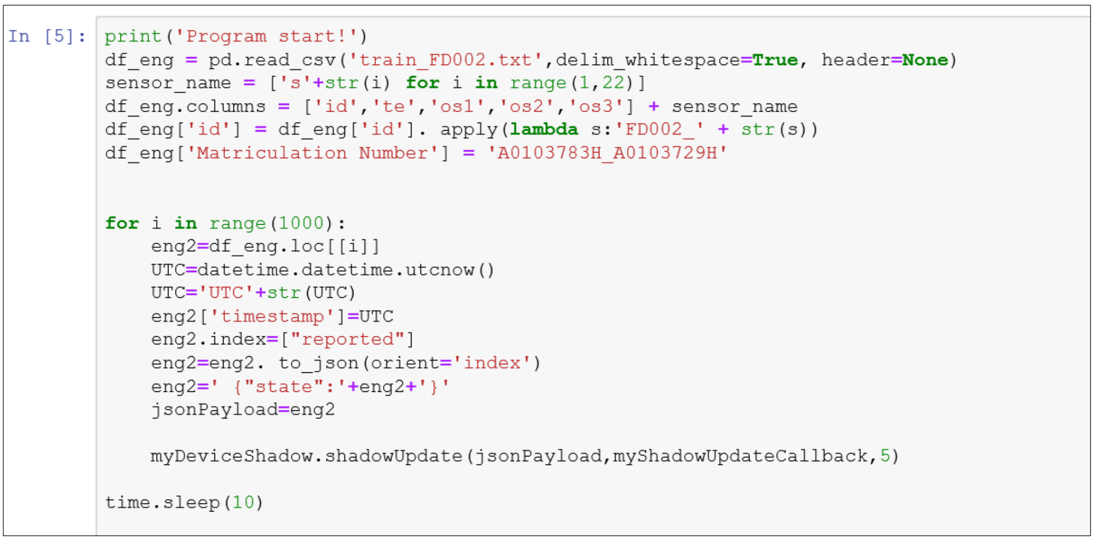
iii. Simulation code



```
In [1]: from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
        import random, time, datetime
        import pandas as pd
        import json

In [2]: # A random programmatic shadow client ID.
        SHADOW_CLIENT = "myShadowClient_2"

        # The unique hostname that &IoT; generated for
        # this device.
        HOST_NAME = "algkikl44tp7o-ats.iot.ap-southeast-1.amazonaws.com"

        # The relative path to the correct root CA file for &IoT;,
        # which you have already saved onto this device.
        ROOT_CA = "CA2.pem"

        # The relative path to your private key file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        PRIVATE_KEY = "a9a1565714-private.pem.key"

        # The relative path to your certificate file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        CERT_FILE = "a9a1565714-certificate.pem.crt.txt"

        # A programmatic shadow handler name prefix.
        SHADOW_HANDLER = "A0103783H_A0103729H_2"
```



```
In [3]: # Automatically called whenever the shadow is updated.
        def myShadowUpdateCallback(payload, responseStatus, token):
          print()
          print('UPDATE: $aws/things/' + SHADOW_HANDLER +
            '/shadow/update/#')
          print("payload = " + payload)
          print("responseStatus = " + responseStatus)
          print("token = " + token)

        # Create, configure, and connect a shadow client.
        print('Connecting to AWS IOT')
        myShadowClient = AWSIoTMQTTShadowClient(SHADOW_CLIENT)
        myShadowClient.configureEndpoint(HOST_NAME, 8883)
        myShadowClient.configureCredentials(ROOT_CA, PRIVATE_KEY, CERT_FILE)
        myShadowClient.configureConnectDisconnectTimeout(10)
        myShadowClient.configureMQTTOperationTimeout(5)
        myShadowClient.connect()

        Connecting to AWS IOT

Out[3]: True

In [4]: myDeviceShadow = myShadowClient.createShadowHandlerWithName(SHADOW_HANDLER, True)
```

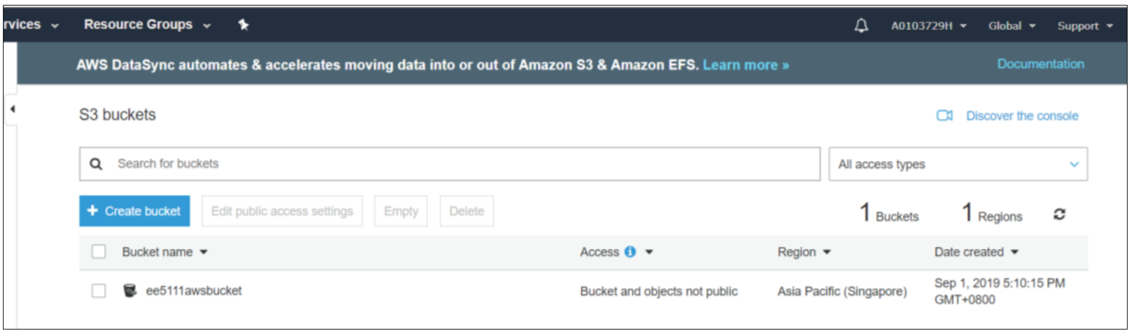


```
In [5]: print('Program start!')
        df_eng = pd.read_csv('train_FD002.txt',delim_whitespace=True, header=None)
        sensor_name = ['s'+str(i) for i in range(1,22)]
        df_eng.columns = ['id','te','os1','os2','os3'] + sensor_name
        df_eng['id'] = df_eng['id']. apply(lambda s:'FD002_' + str(s))
        df_eng['Matriculation Number'] = 'A0103783H_A0103729H'


        for i in range(1000):
            eng2=df_eng.loc[[i]]
            UTC=datetime.datetime.utcnow()
            UTC='UTC'+str(UTC)
            eng2['timestamp']=UTC
            eng2.index=["reported"]
            eng2=eng2. to_json(orient='index')
            eng2=' {"state":'+eng2+'}'
            jsonPayload=eng2

            myDeviceShadow.shadowUpdate(jsonPayload,myShadowUpdateCallback,5)

        time.sleep(10)
```

3) Visualize the two engines for all the sensors by querying the data from AWS DynamoDB

- ➢ Export data from DynamoDB to ‘S3’ through ‘Data Pipeline’
- ➢ Upload the two engines data from S3 and visualization the data by QuickSight.

i. Set Up S3 buckets. (S3 buckets name is ee5111awsbucket)



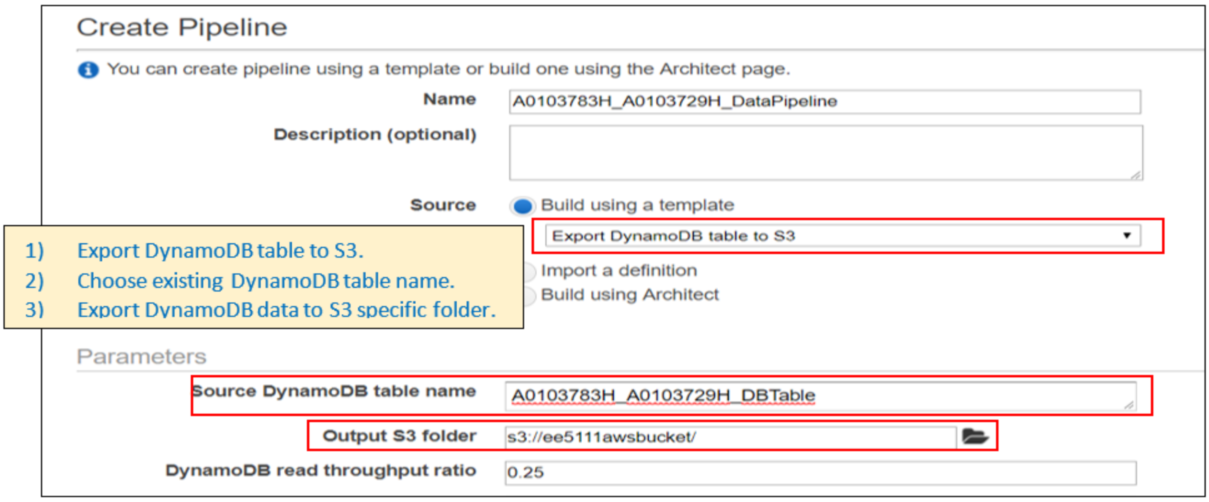
AWS DataSync automates & accelerates moving data into or out of Amazon S3 & Amazon EFS. Learn more »

S3 buckets

1 Buckets 1 Regions

□	Bucket name	Access	Region	Date created
□	ee5111awsbucket	Bucket and objects not public	Asia Pacific (Singapore)	Sep 1, 2019 5:10:15 PM GMT+0800

ii. Create Pipeline. (Pipeline name is ‘A0103783H_A0103729H_DataPepeline)



Create Pipeline

You can create pipeline using a template or build one using the Architect page.

Name: A0103783H_A0103729H_DataPipeline

Description (optional):

Source: ● Build using a template

Export DynamoDB table to S3

○ Import a definition

○ Build using Architect

1) Export DynamoDB table to S3.
2) Choose existing DynamoDB table name.
3) Export DynamoDB data to S3 specific folder.

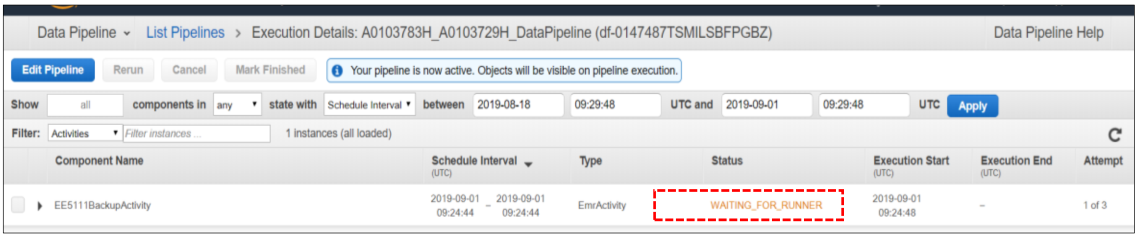
Parameters

Source DynamoDB table name: A0103783H_A0103729H_DBTable

Output S3 folder: s3://ee5111awsbucket/

DynamoDB read throughput ratio: 0.25

iii. Export data from DynamoDB to S3 through data pipeline



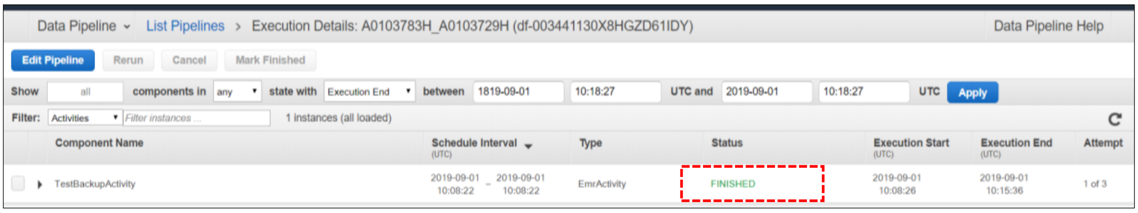
Execution Details: A0103783H_A0103729H_DataPipeline (df-0147487TSMILSBFPGBZ)

Your pipeline is now active. Objects will be visible on pipeline execution.

Show all components in any state with Schedule Interval between 2019-08-18 09:29:48 UTC and 2019-09-01 09:29:48 UTC

Filter: Activities Filter instances ... 1 instances (all loaded)

Component Name	Schedule Interval (UTC)	Type	Status	Execution Start (UTC)	Execution End (UTC)	Attempt
EE5111BackupActivity	2019-09-01 09:24:44 – 2019-09-01 09:24:44	EmrActivity	WAITING_FOR_RUNNER	2019-09-01 09:24:48	-	1 of 3



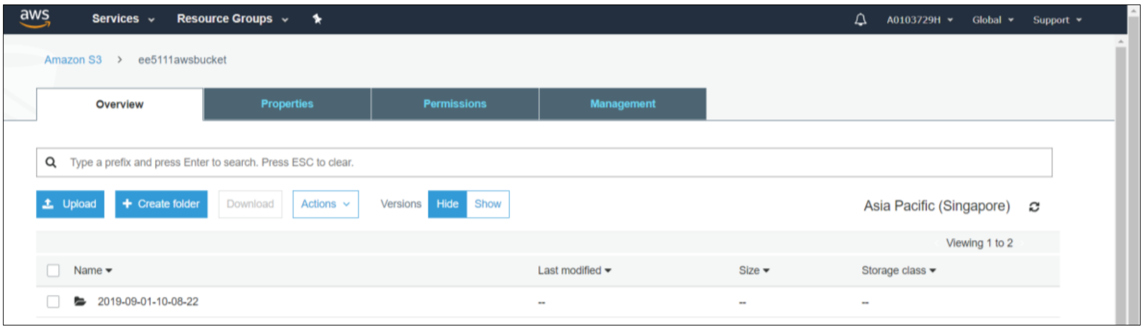
Execution Details: A0103783H_A0103729H (df-003441130X8HGZD61IDY)

Show all components in any state with Execution End between 1819-09-01 10:18:27 UTC and 2019-09-01 10:18:27 UTC

Filter: Activities Filter instances ... 1 instances (all loaded)

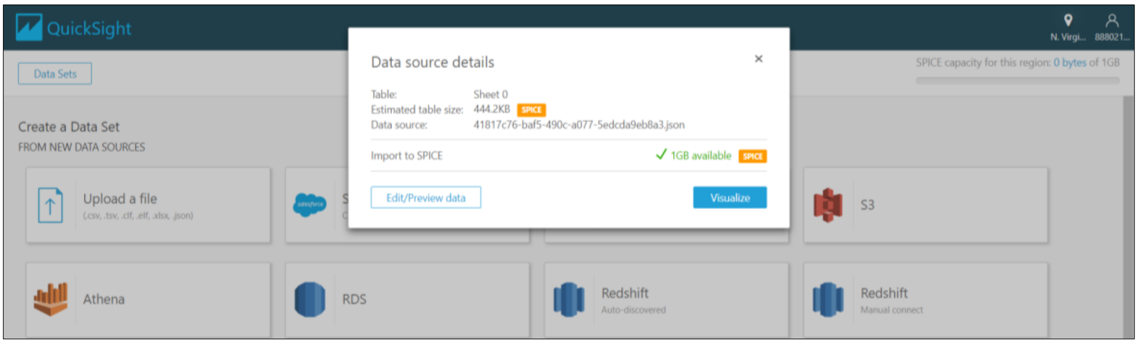
Component Name	Schedule Interval (UTC)	Type	Status	Execution Start (UTC)	Execution End (UTC)	Attempt
TestBackupActivity	2019-09-01 10:08:22 – 2019-09-01 10:08:22	EmrActivity	FINISHED	2019-09-01 10:08:26	2019-09-01 10:15:36	1 of 3

iv. Data successful export to S3 bucket and upload the data to Desktop.



Name	Last modified	Size	Storage class
2019-09-01-10-08-22	--	--	--

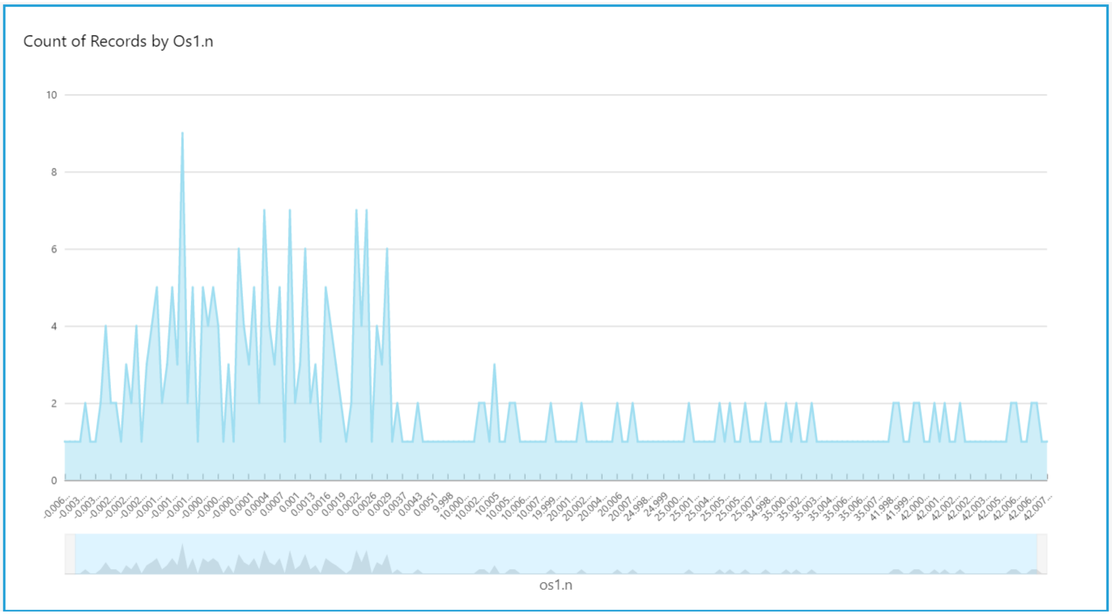
v. Using QucikSight to visualize the data from desktop.



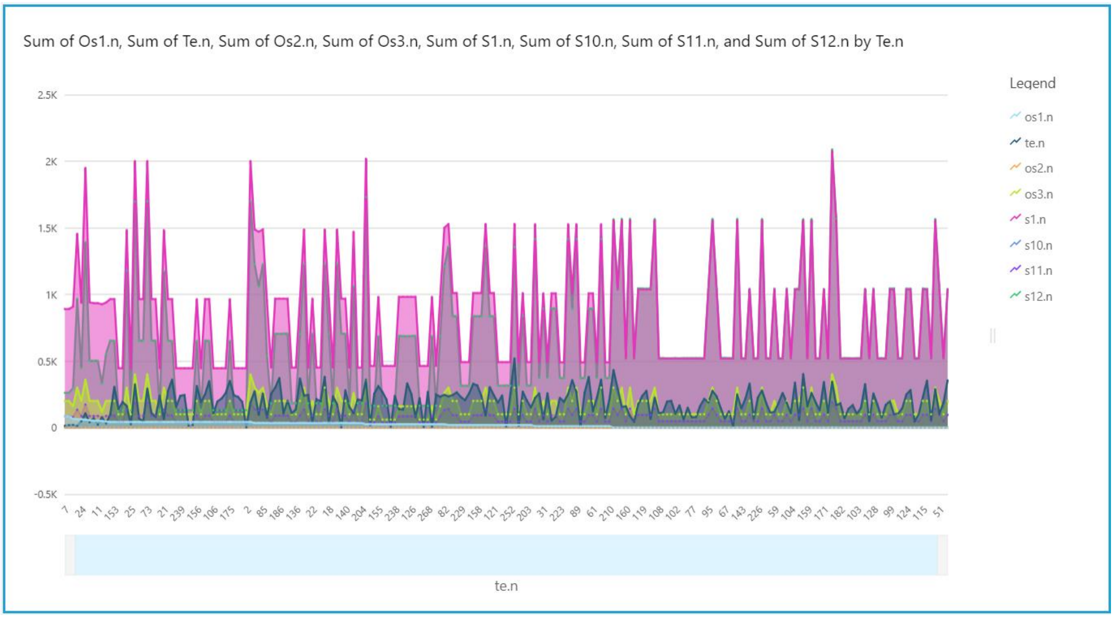
Data source details

Table: Sheet 0
Estimated table size: 444.2KB SPICE
Data source: 41817c76-baf5-490c-a077-5edcda9eb8a3.json

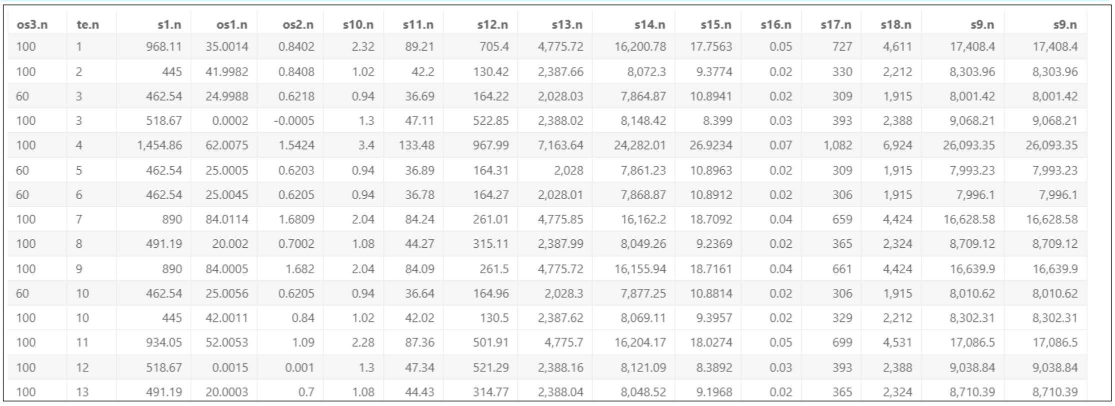
Import to SPICE ✓ 1GB available SPICE



Count of Records by Os1.n



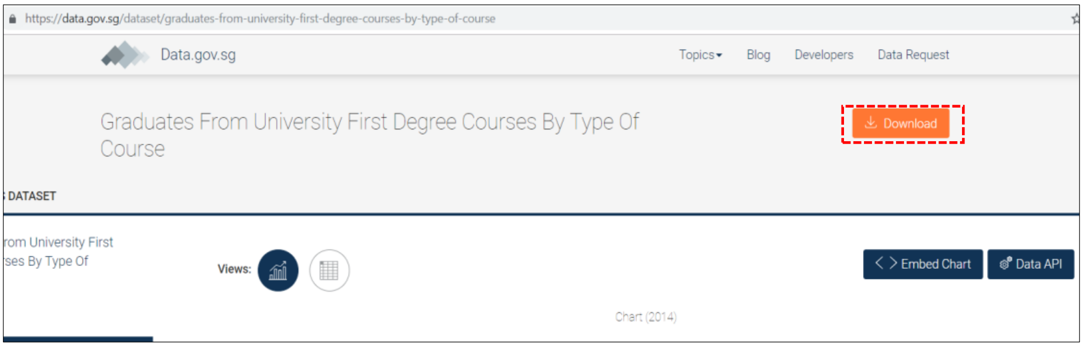
Sum of Os1.n, Sum of Te.n, Sum of Os2.n, Sum of Os3.n, Sum of S1.n, Sum of S10.n, Sum of S11.n, and Sum of S12.n by Te.n



os3.n	te.n	s1.n	os1.n	os2.n	s10.n	s11.n	s12.n	s13.n	s14.n	s15.n	s16.n	s17.n	s18.n	s9.n	s9.n
100	1	968.11	35.0014	0.8402	2.32	89.21	705.4	4,775.72	16,200.78	17.7563	0.05	727	4,611	17,408.4	17,408.4
100	2	445	41.9982	0.8408	1.02	42.2	130.42	2,387.66	8,072.3	9.3774	0.02	330	2,212	8,303.96	8,303.96
60	3	462.54	24.9988	0.6218	0.94	36.69	164.22	2,028.03	7,864.87	10.8941	0.02	309	1,915	8,001.42	8,001.42
100	3	518.67	0.0002	-0.0005	1.3	47.11	522.85	2,388.02	8,148.42	8.399	0.03	393	2,388	9,068.21	9,068.21
100	4	1,454.86	62.0075	1.5424	3.4	133.48	967.99	7,163.64	24,282.01	26.9234	0.07	1,082	6,924	26,093.35	26,093.35
60	5	462.54	25.0005	0.6203	0.94	36.89	164.31	2,028	7,861.23	10.8963	0.02	309	1,915	7,993.23	7,993.23
60	6	462.54	25.0045	0.6205	0.94	36.78	164.27	2,028.01	7,868.87	10.8912	0.02	306	1,915	7,996.1	7,996.1
100	7	890	84.0114	1.6809	2.04	84.24	261.01	4,775.85	16,162.2	18.7092	0.04	659	4,424	16,628.58	16,628.58
100	8	491.19	20.002	0.7002	1.08	44.27	315.11	2,387.99	8,049.26	9.2369	0.02	365	2,324	8,709.12	8,709.12
100	9	890	84.0005	1.682	2.04	84.09	261.5	4,775.72	16,155.94	18.7161	0.04	661	4,424	16,639.9	16,639.9
60	10	462.54	25.0056	0.6205	0.94	36.64	164.96	2,028.3	7,877.25	10.8814	0.02	306	1,915	8,010.62	8,010.62
100	10	445	42.0011	0.84	1.02	42.02	130.5	2,387.62	8,069.11	9.3957	0.02	329	2,212	8,302.31	8,302.31
100	11	934.05	52.0053	1.09	2.28	87.36	501.91	4,775.7	16,204.17	18.0274	0.05	699	4,531	17,086.5	17,086.5
100	12	518.67	0.0015	0.001	1.3	47.34	521.29	2,388.16	8,121.09	8.3892	0.03	393	2,388	9,038.84	9,038.84
100	13	491.19	20.0003	0.7	1.08	44.43	314.77	2,388.04	8,048.52	9.1968	0.02	365	2,324	8,710.39	8,710.39

3. Simulation of Singapore Graduates from University First Degree Courses By Type Of Course

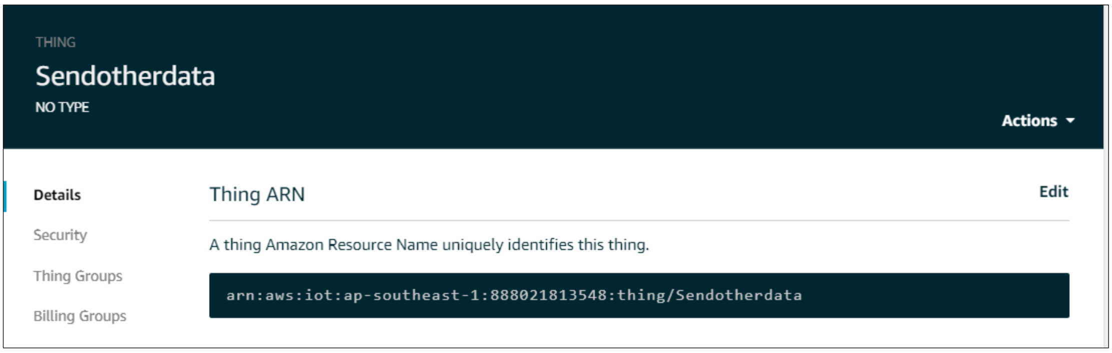
i. Download data from gov website
From https://data.gov.sg/dataset/graduates-from-university-first-degree-courses-by-type-of-course website.



https://data.gov.sg/dataset/graduates-from-university-first-degree-courses-by-type-of-course

Graduates From University First Degree Courses By Type Of Course

ii. Thing, Certificate, Policy, Rules and DynamoDB table Set up.



THING

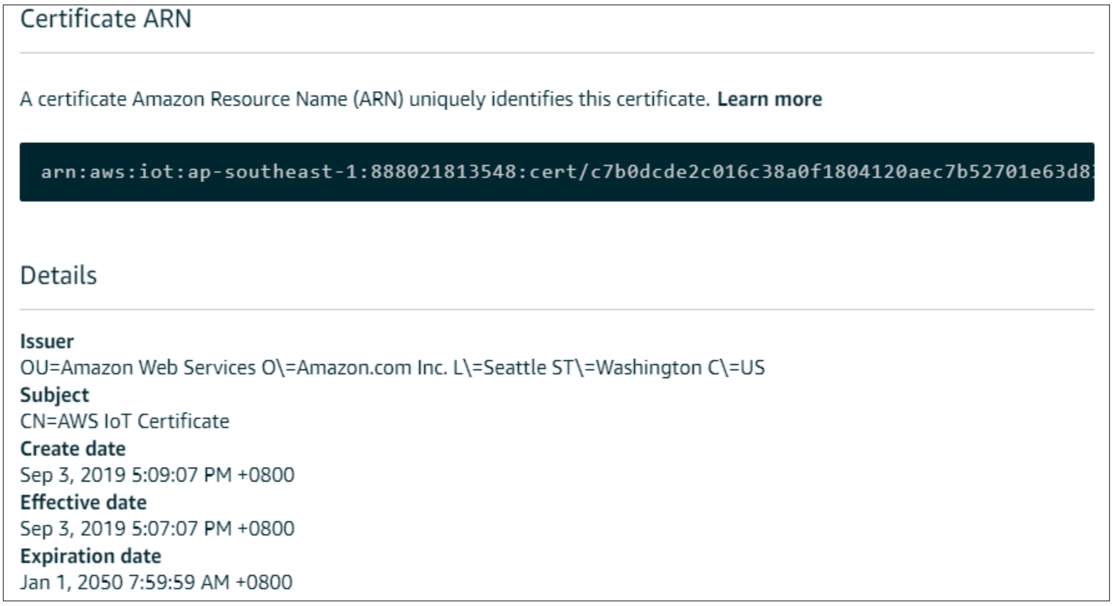
Sendotherdata

NO TYPE

Thing ARN

A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:ap-southeast-1:888021813548:thing/Sendotherdata
```



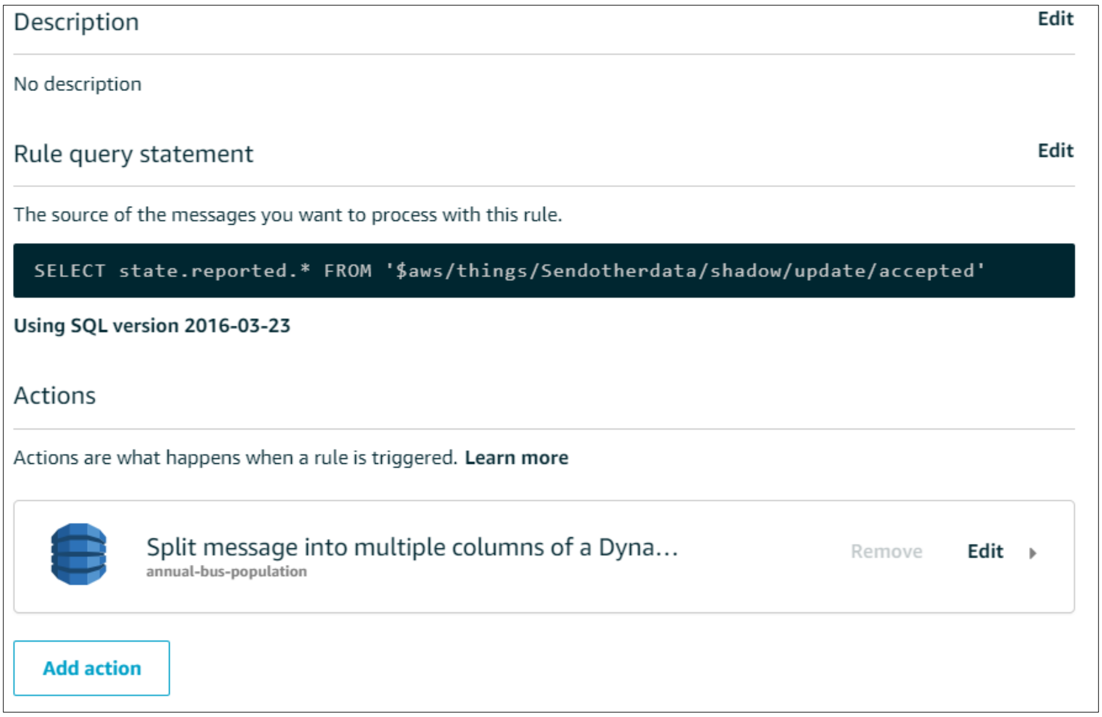
Certificate ARN

A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:ap-southeast-1:888021813548:cert/c7b0dcde2c016c38a0f1804120aec7b52701e63d8
```

Details

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Sep 3, 2019 5:09:07 PM +0800
Effective date
Sep 3, 2019 5:07:07 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800



Description

No description

Rule query statement

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

```
SELECT state.reported.* FROM '$aws/things/Sendotherdata/shadow/update/accepted'
```

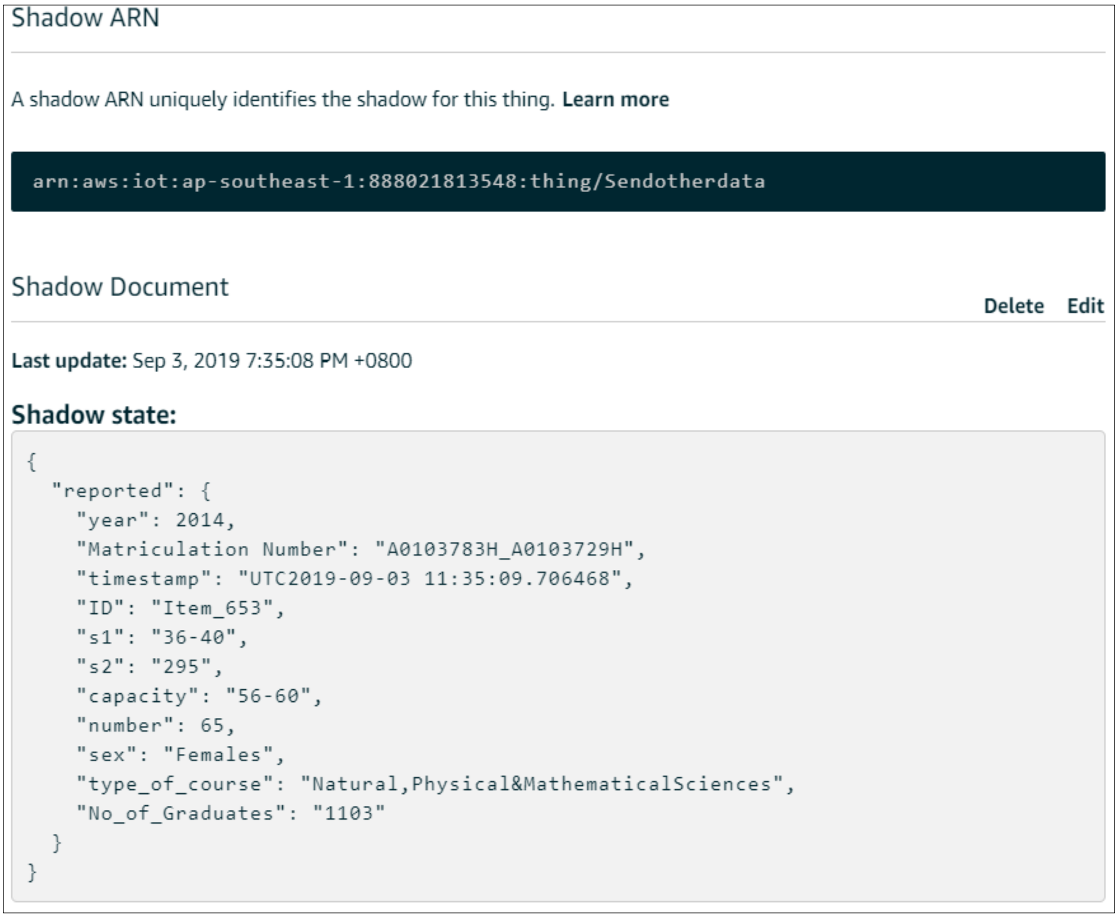
Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. **Learn more**

Split message into multiple columns of a Dyna…
annual-bus-population

iii. Output in AWS & DynamoDB table.



Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing. **Learn more**

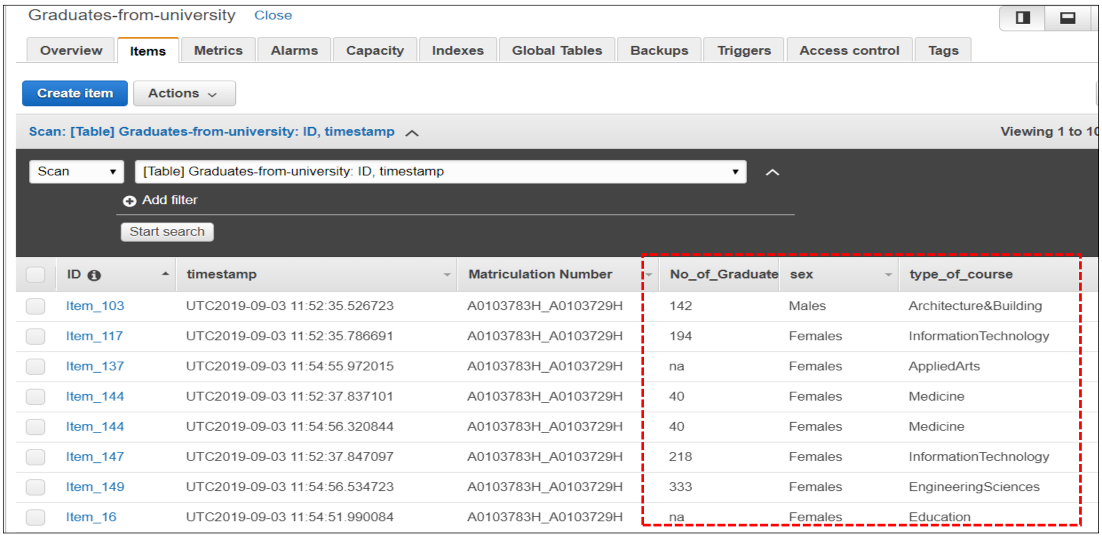
```
arn:aws:iot:ap-southeast-1:888021813548:thing/Sendotherdata
```

Shadow Document

Last update: Sep 3, 2019 7:35:08 PM +0800

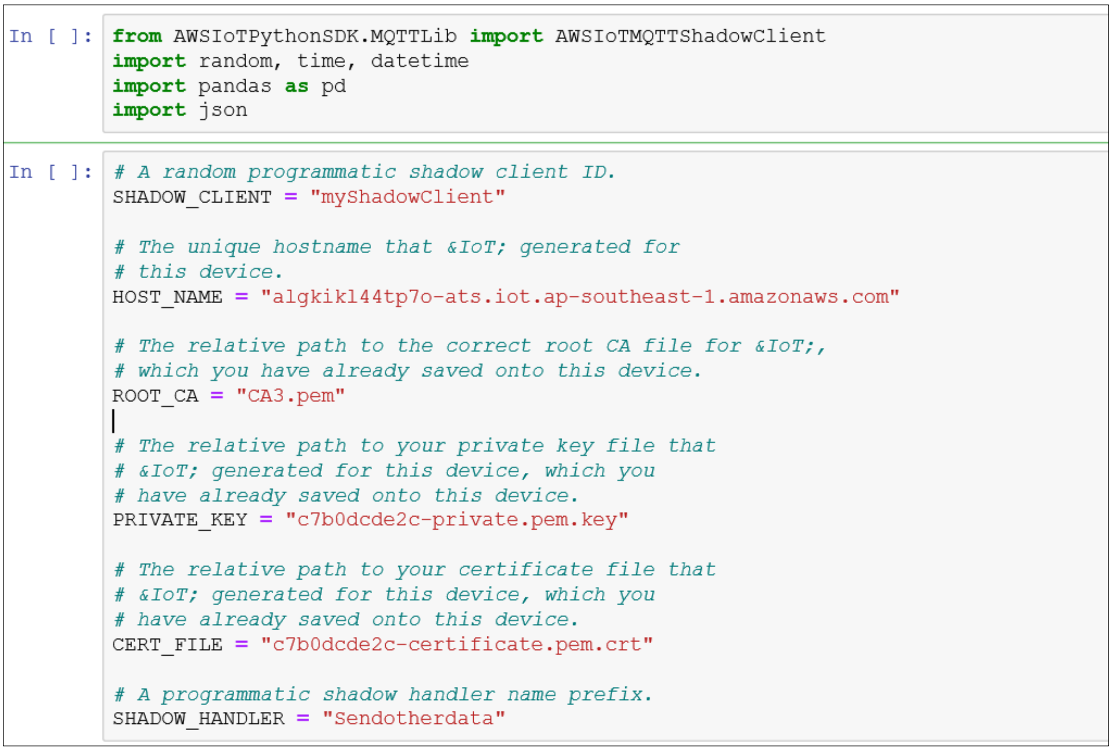
Shadow state:

```
{
  "reported": {
    "year": 2014,
    "Matriculation Number": "A0103783H_A0103729H",
    "timestamp": "UTC2019-09-03 11:35:09.706468",
    "ID": "Item_653",
    "s1": "36-40",
    "s2": "295",
    "capacity": "56-60",
    "number": 65,
    "sex": "Females",
    "type_of_course": "Natural,Physical&MathematicalSciences",
    "No_of_Graduates": "1103"
  }
}
```



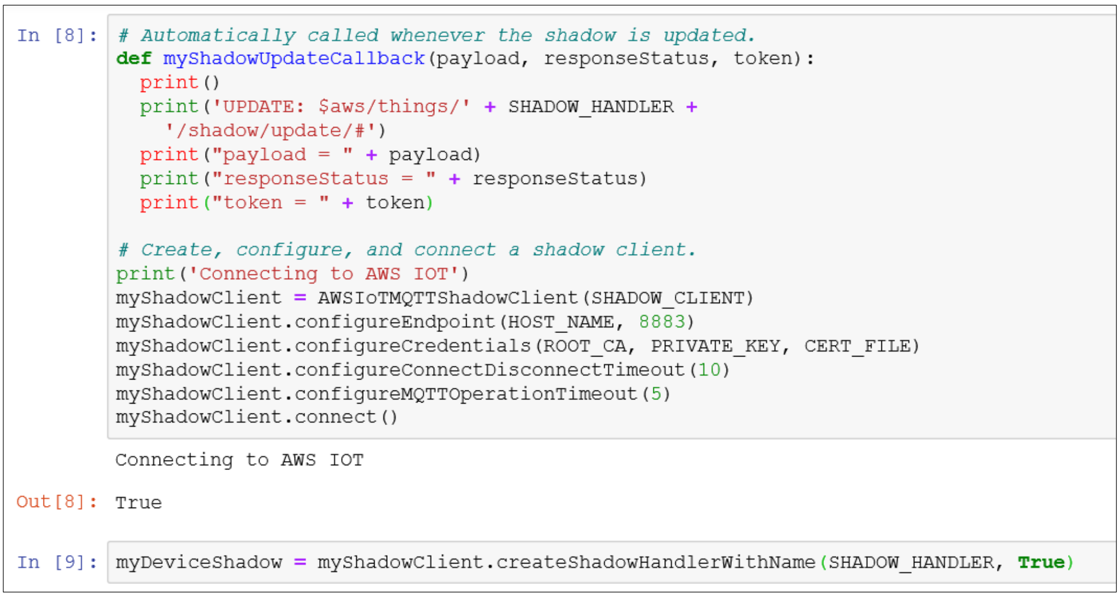
ID	timestamp	Matriculation Number	No_of_Graduate	sex	type_of_course
Item_103	UTC2019-09-03 11:52:35.526723	A0103783H_A0103729H	142	Males	Architecture&Building
Item_117	UTC2019-09-03 11:52:35.786691	A0103783H_A0103729H	194	Females	InformationTechnology
Item_137	UTC2019-09-03 11:54:55.972015	A0103783H_A0103729H	na	Females	AppliedArts
Item_144	UTC2019-09-03 11:52:37.837101	A0103783H_A0103729H	40	Females	Medicine
Item_144	UTC2019-09-03 11:54:56.320844	A0103783H_A0103729H	40	Females	Medicine
Item_147	UTC2019-09-03 11:52:37.847097	A0103783H_A0103729H	218	Females	InformationTechnology
Item_149	UTC2019-09-03 11:54:56.534723	A0103783H_A0103729H	333	Females	EngineeringSciences
Item_16	UTC2019-09-03 11:54:51.990084	A0103783H_A0103729H	na	Females	Education

iv. Simulation code

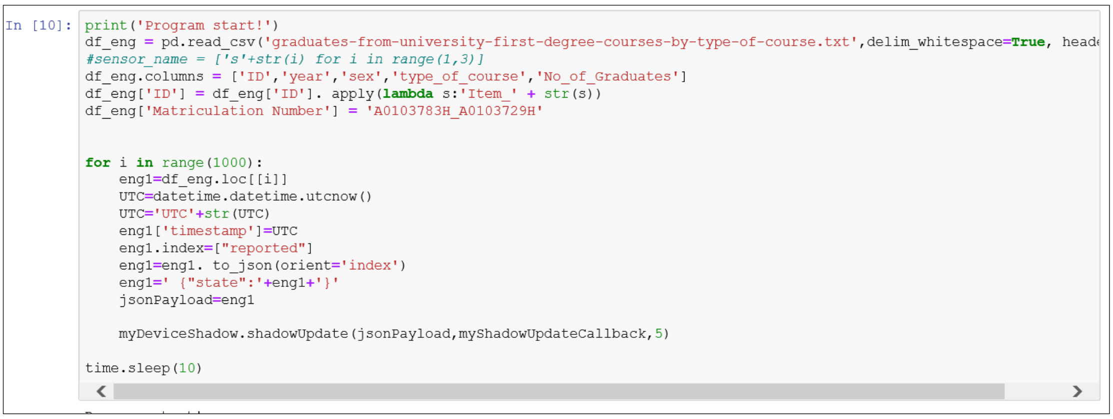


```
In [ ]: from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
        import random, time, datetime
        import pandas as pd
        import json
```

```
In [ ]: # A random programmatic shadow client ID.
        SHADOW_CLIENT = "myShadowClient"

        # The unique hostname that &IoT; generated for
        # this device.
        HOST_NAME = "algkikl44tp7o-ats.iot.ap-southeast-1.amazonaws.com"

        # The relative path to the correct root CA file for &IoT;,
        # which you have already saved onto this device.
        ROOT_CA = "CA3.pem"

        # The relative path to your private key file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        PRIVATE_KEY = "c7b0dcde2c-private.pem.key"

        # The relative path to your certificate file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        CERT_FILE = "c7b0dcde2c-certificate.pem.crt"

        # A programmatic shadow handler name prefix.
        SHADOW_HANDLER = "Sendotherdata"
```

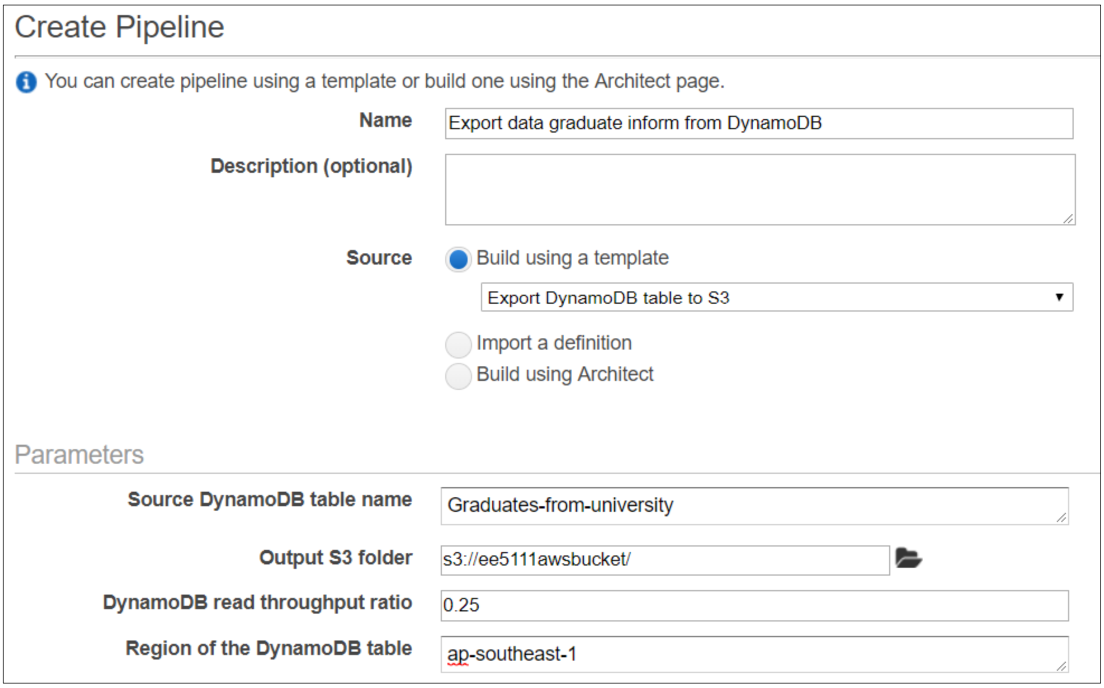


```
In [8]: # Automatically called whenever the shadow is updated.
        def myShadowUpdateCallback(payload, responseStatus, token):
          print()
          print('UPDATE: $aws/things/' + SHADOW_HANDLER +
            '/shadow/update/#')
          print("payload = " + payload)
          print("responseStatus = " + responseStatus)
          print("token = " + token)

        # Create, configure, and connect a shadow client.
        print('Connecting to AWS IOT')
        myShadowClient = AWSIoTMQTTShadowClient(SHADOW_CLIENT)
        myShadowClient.configureEndpoint(HOST_NAME, 8883)
        myShadowClient.configureCredentials(ROOT_CA, PRIVATE_KEY, CERT_FILE)
        myShadowClient.configureConnectDisconnectTimeout(10)
        myShadowClient.configureMQTTOperationTimeout(5)
        myShadowClient.connect()

        Connecting to AWS IOT

Out[8]: True

In [9]: myDeviceShadow = myShadowClient.createShadowHandlerWithName(SHADOW_HANDLER, True)
```



```
In [10]: print('Program start!')
         df_eng = pd.read_csv('graduates-from-university-first-degree-courses-by-type-of-course.txt',delim_whitespace=True, heade
         #sensor_name = ['s'+str(i) for i in range(1,3)]
         df_eng.columns = ['ID','year','sex','type_of_course','No_of_Graduates']
         df_eng['ID'] = df_eng['ID']. apply(lambda s:'Item_' + str(s))
         df_eng['Matriculation Number'] = 'A0103783H_A0103729H'


         for i in range(1000):
             eng1=df_eng.loc[[i]]
             UTC=datetime.datetime.utcnow()
             UTC='UTC'+str(UTC)
             eng1['timestamp']=UTC
             eng1.index=["reported"]
             eng1=eng1. to_json(orient='index')
             eng1=' {"state":'+eng1+'}'
             jsonPayload=eng1

             myDeviceShadow.shadowUpdate(jsonPayload,myShadowUpdateCallback,5)

         time.sleep(10)
```

v. Create Pipeline & Set Up



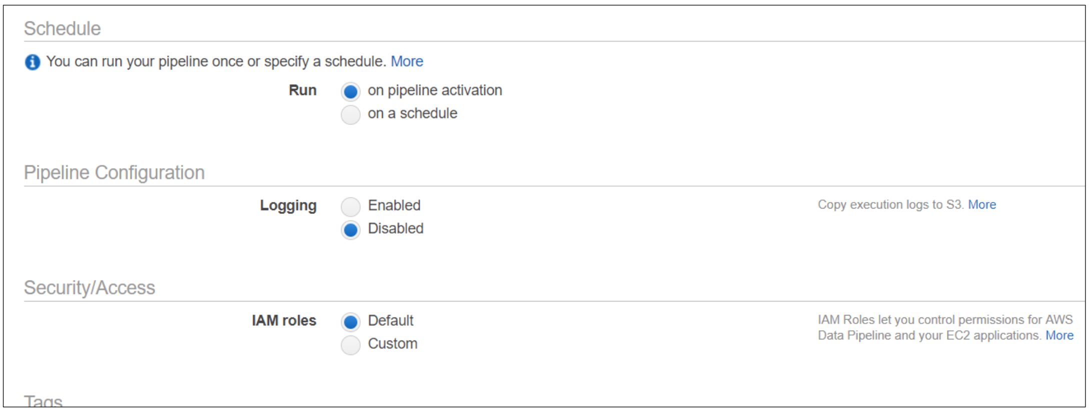
Create Pipeline

You can create pipeline using a template or build one using the Architect page.

Name	Export data graduate inform from DynamoDB
Description (optional)	
Source	● Build using a template Export DynamoDB table to S3 ▾ ○ Import a definition ○ Build using Architect

Parameters

Source DynamoDB table name	Graduates-from-university
Output S3 folder	s3://ee5111awsbucket/
DynamoDB read throughput ratio	0.25
Region of the DynamoDB table	ap-southeast-1



Schedule

You can run your pipeline once or specify a schedule. More

Run ● on pipeline activation
○ on a schedule

Pipeline Configuration

Logging ○ Enabled
● Disabled

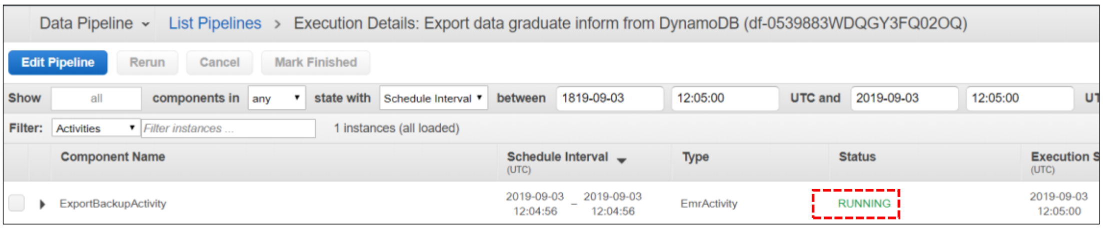
Copy execution logs to S3. More

Security/Access

IAM roles ● Default
○ Custom

IAM Roles let you control permissions for AWS Data Pipeline and your EC2 applications. More

Tags

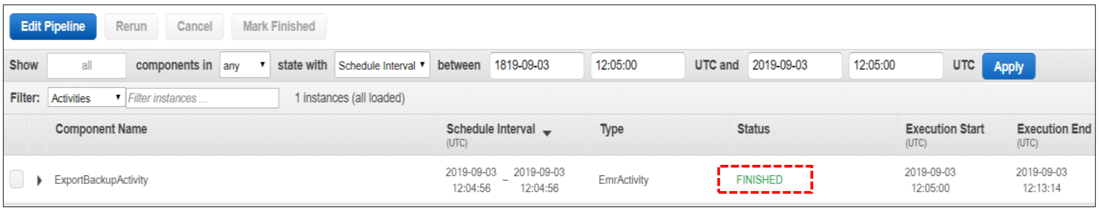


Data Pipeline > List Pipelines > Execution Details: Export data graduate inform from DynamoDB (df-0539883WDQGY3FQ02OQ)

Show all components in any state with Schedule Interval between 1819-09-03 12:05:00 UTC and 2019-09-03 12:05:00 UT

Filter: Activities Filter instances ... 1 instances (all loaded)

Component Name	Schedule Interval (UTC)	Type	Status	Execution S (UTC)
ExportBackupActivity	2019-09-03 12:04:56 – 2019-09-03 12:04:56	EmrActivity	RUNNING	2019-09-03 12:05:00

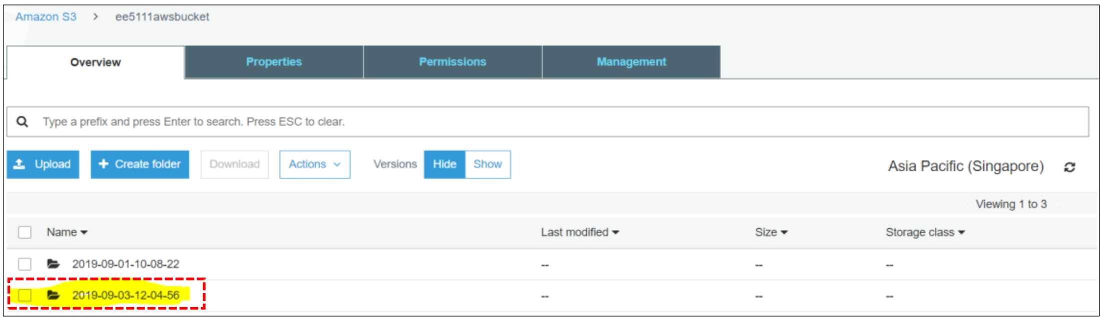


Show all components in any state with Schedule Interval between 1819-09-03 12:05:00 UTC and 2019-09-03 12:05:00 UTC

Filter: Activities Filter instances ... 1 instances (all loaded)

Component Name	Schedule Interval (UTC)	Type	Status	Execution Start (UTC)	Execution End (UTC)
ExportBackupActivity	2019-09-03 12:04:56 – 2019-09-03 12:04:56	EmrActivity	FINISHED	2019-09-03 12:05:00	2019-09-03 12:13:14

vi. Download data from S3 Bucket to Desktop

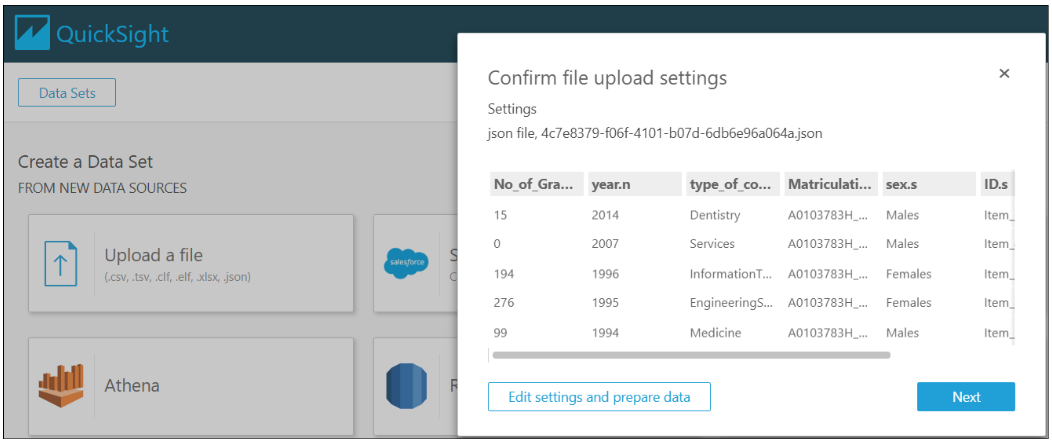


Amazon S3 > ee5111awsbucket

Viewing 1 to 3

Name	Last modified	Size	Storage class
2019-09-01-10-08-22	--	--	--
2019-09-03-12-04-56	--	--	--

vii. Using QucikSight to visualize the data from desktop.

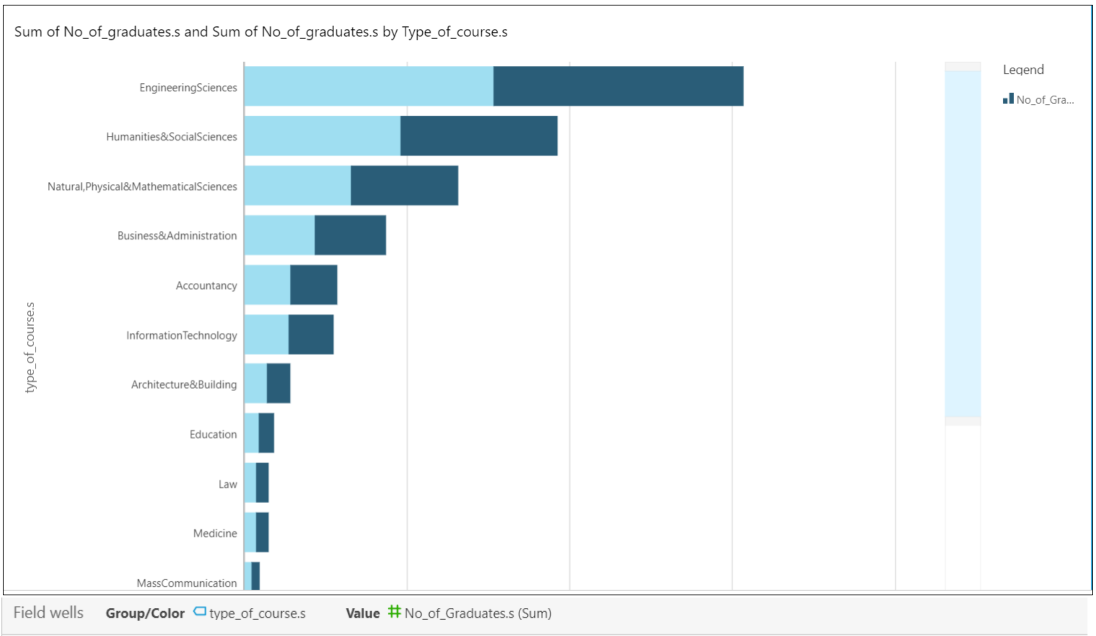


Confirm file upload settings

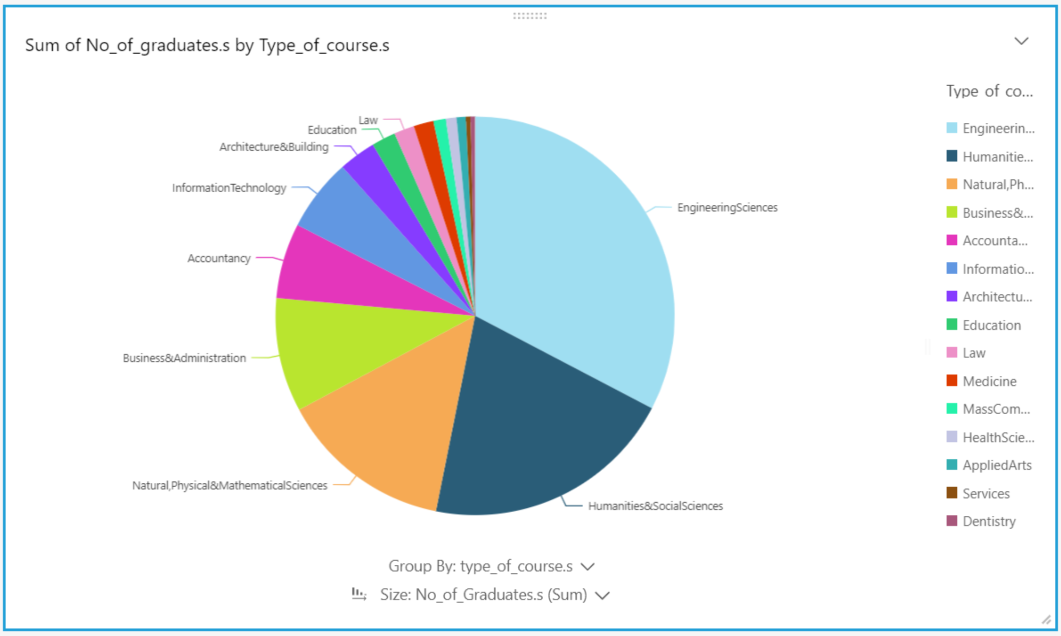
Settings

json file, 4c7e8379-f06f-4101-b07d-6db6e96a064a.json

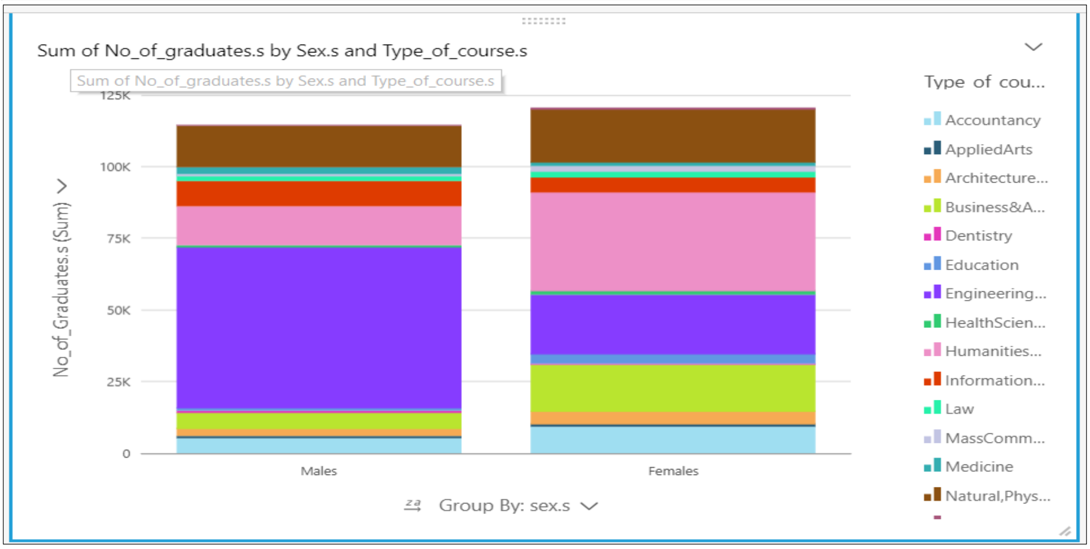
No_of_Gra...	year.n	type_of_co...	Matriculati...	sex.s	ID.s
15	2014	Dentistry	A0103783H_...	Males	Item_
0	2007	Services	A0103783H_...	Males	Item_
194	1996	InformationT...	A0103783H_...	Females	Item_
276	1995	EngineeringS...	A0103783H_...	Females	Item_
99	1994	Medicine	A0103783H_...	Males	Item_



Field wells Group/Color type_of_course.s Value No_of_Graduates.s (Sum)



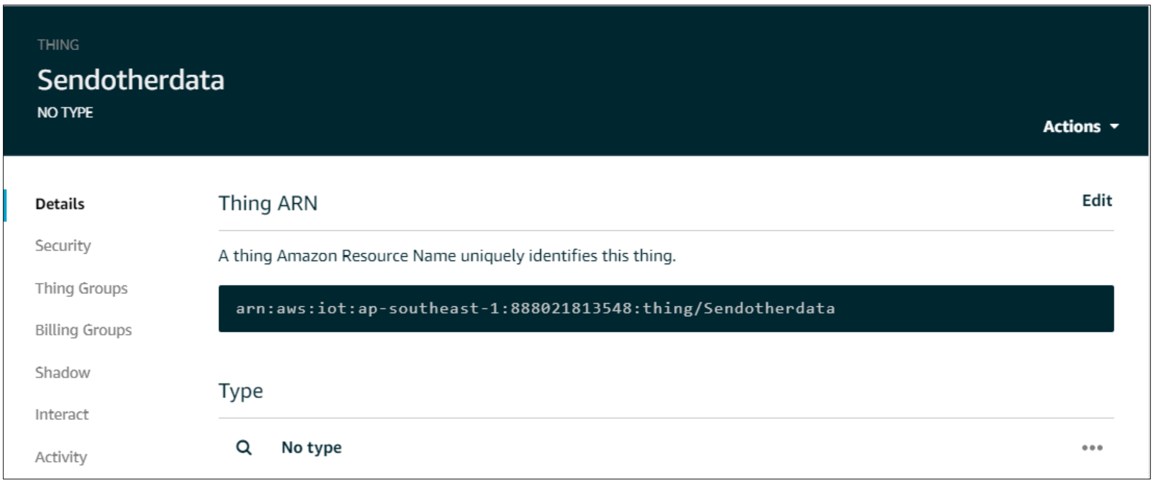
Group By: type_of_course.s
Size: No_of_Graduates.s (Sum)



Group By: sex.s

4. Simulation of real-time data of Singapore Relative Humidity - Monthly Absolute Extreme Minimum

i. Thing, Certificate, Policy, Rules and DynamoDB table Set up.



Sendotherdata

NO TYPE

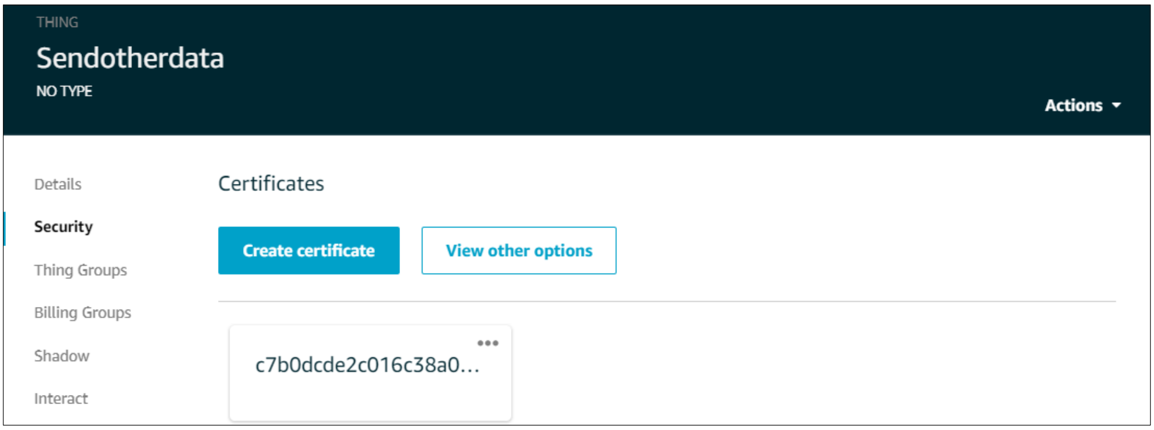
Thing ARN

A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:ap-southeast-1:888021813548:thing/Sendotherdata
```

Type

No type

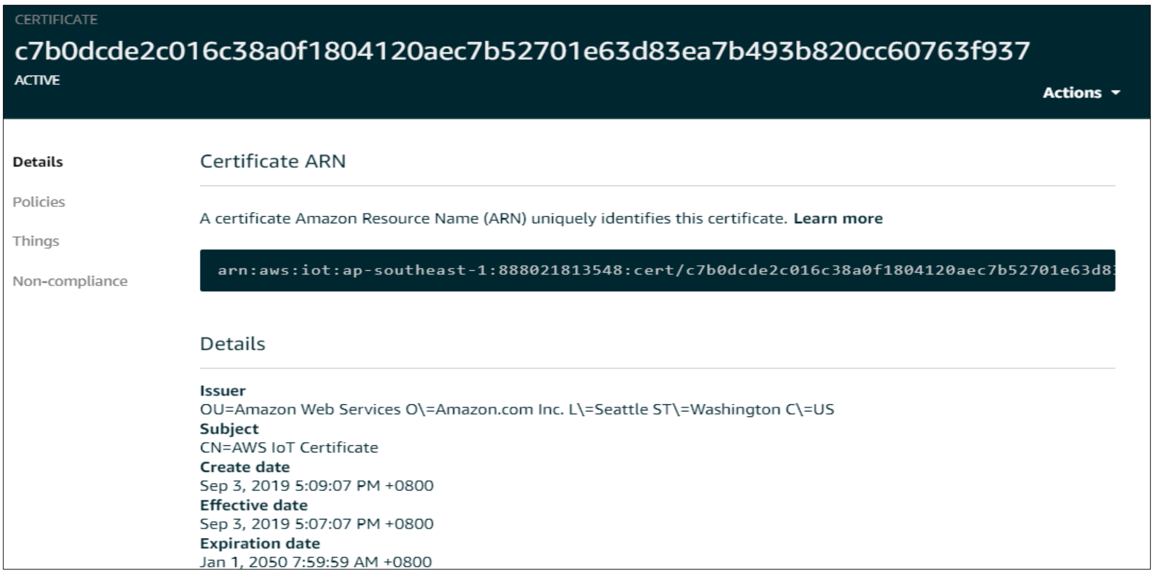


Sendotherdata

NO TYPE

Certificates

c7b0dcde2c016c38a0...



c7b0dcde2c016c38a0f1804120aec7b52701e63d83ea7b493b820cc60763f937

ACTIVE

Certificate ARN

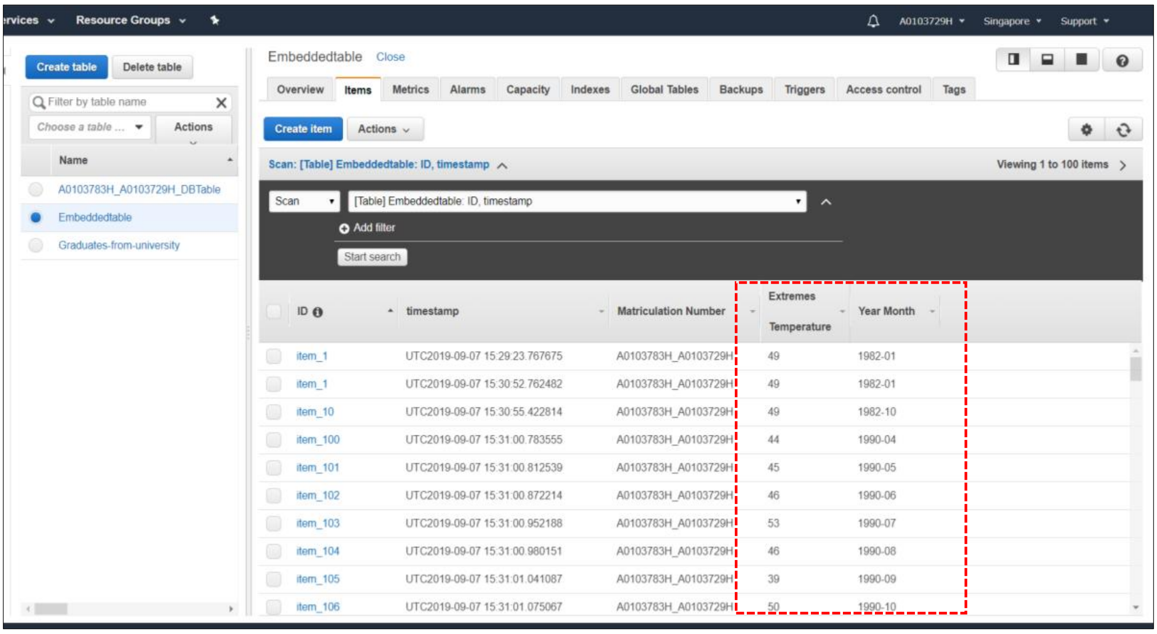
A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:ap-southeast-1:888021813548:cert/c7b0dcde2c016c38a0f1804120aec7b52701e63d8
```

Details

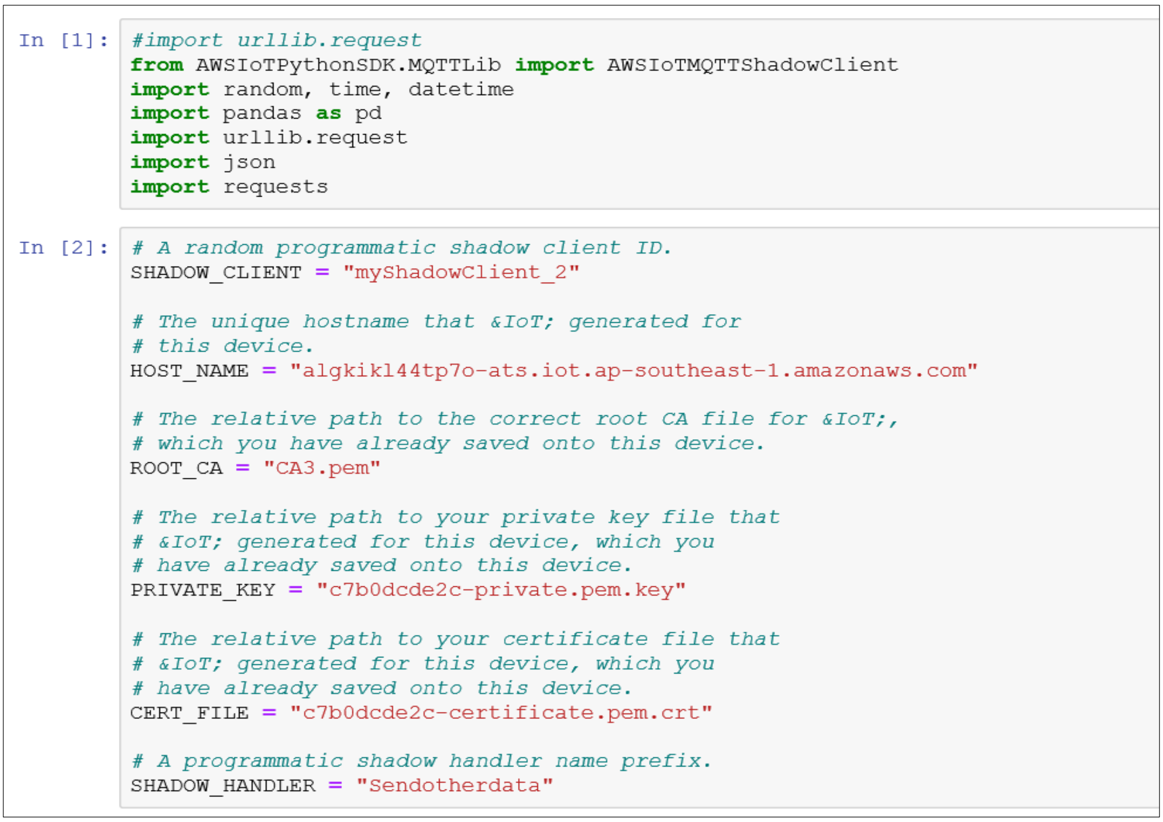
Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Sep 3, 2019 5:09:07 PM +0800
Effective date
Sep 3, 2019 5:07:07 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800

ii. Output in AWS & DynamoDB table.

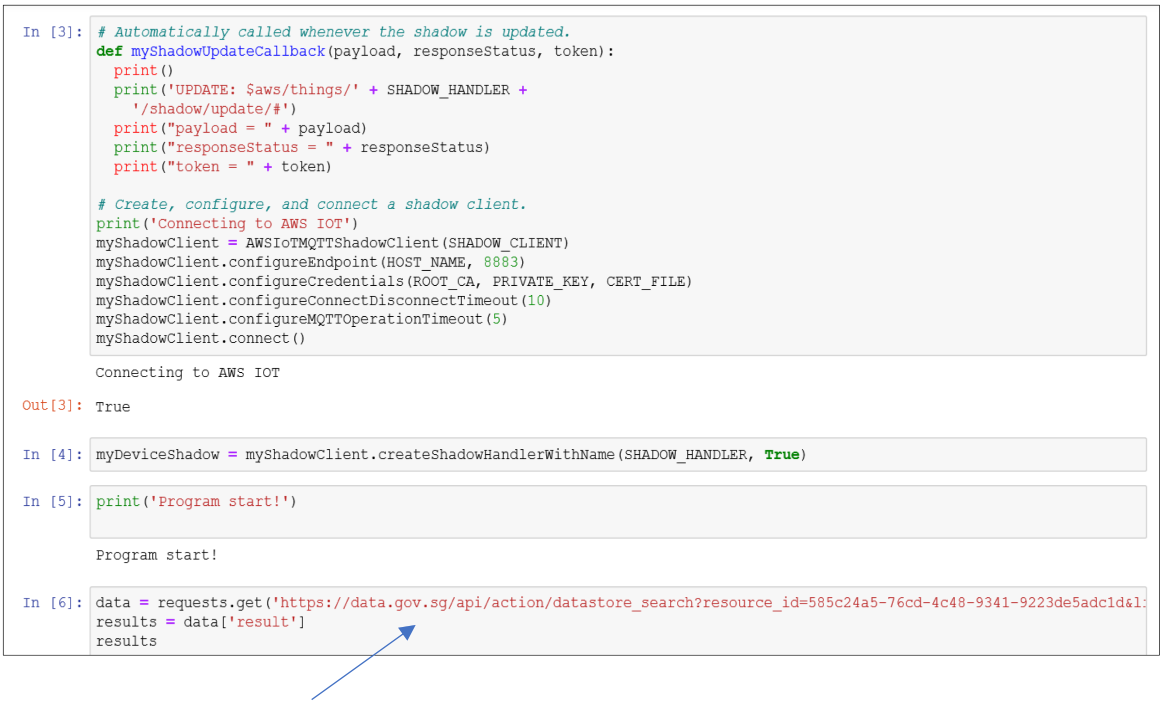


ID	timestamp	Matriculation Number	Extremes Temperature	Year Month
item_1	UTC2019-09-07 15:29:23.767675	A0103783H_A0103729H	49	1982-01
item_1	UTC2019-09-07 15:30:52.762482	A0103783H_A0103729H	49	1982-01
item_10	UTC2019-09-07 15:30:55.422814	A0103783H_A0103729H	49	1982-10
item_100	UTC2019-09-07 15:31:00.783555	A0103783H_A0103729H	44	1990-04
item_101	UTC2019-09-07 15:31:00.812539	A0103783H_A0103729H	45	1990-05
item_102	UTC2019-09-07 15:31:00.872214	A0103783H_A0103729H	46	1990-06
item_103	UTC2019-09-07 15:31:00.952188	A0103783H_A0103729H	53	1990-07
item_104	UTC2019-09-07 15:31:00.980151	A0103783H_A0103729H	46	1990-08
item_105	UTC2019-09-07 15:31:01.041087	A0103783H_A0103729H	39	1990-09
item_106	UTC2019-09-07 15:31:01.075067	A0103783H_A0103729H	50	1990-10

iii. Simulation code

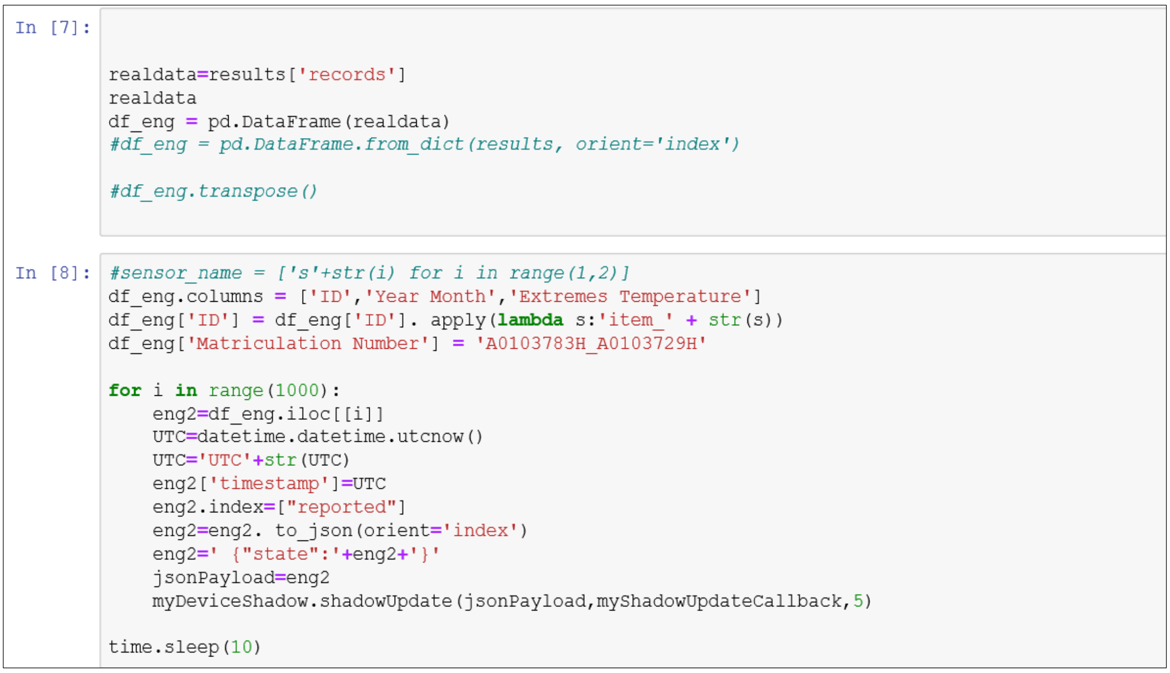


```
In [1]: #import urllib.request
        from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
        import random, time, datetime
        import pandas as pd
        import urllib.request
        import json
        import requests

In [2]: # A random programmatic shadow client ID.
        SHADOW_CLIENT = "myShadowClient_2"

        # The unique hostname that &IoT; generated for
        # this device.
        HOST_NAME = "algkikl44tp7o-ats.iot.ap-southeast-1.amazonaws.com"

        # The relative path to the correct root CA file for &IoT;,
        # which you have already saved onto this device.
        ROOT_CA = "CA3.pem"

        # The relative path to your private key file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        PRIVATE_KEY = "c7b0dcde2c-private.pem.key"

        # The relative path to your certificate file that
        # &IoT; generated for this device, which you
        # have already saved onto this device.
        CERT_FILE = "c7b0dcde2c-certificate.pem.crt"

        # A programmatic shadow handler name prefix.
        SHADOW_HANDLER = "Sendotherdata"
```



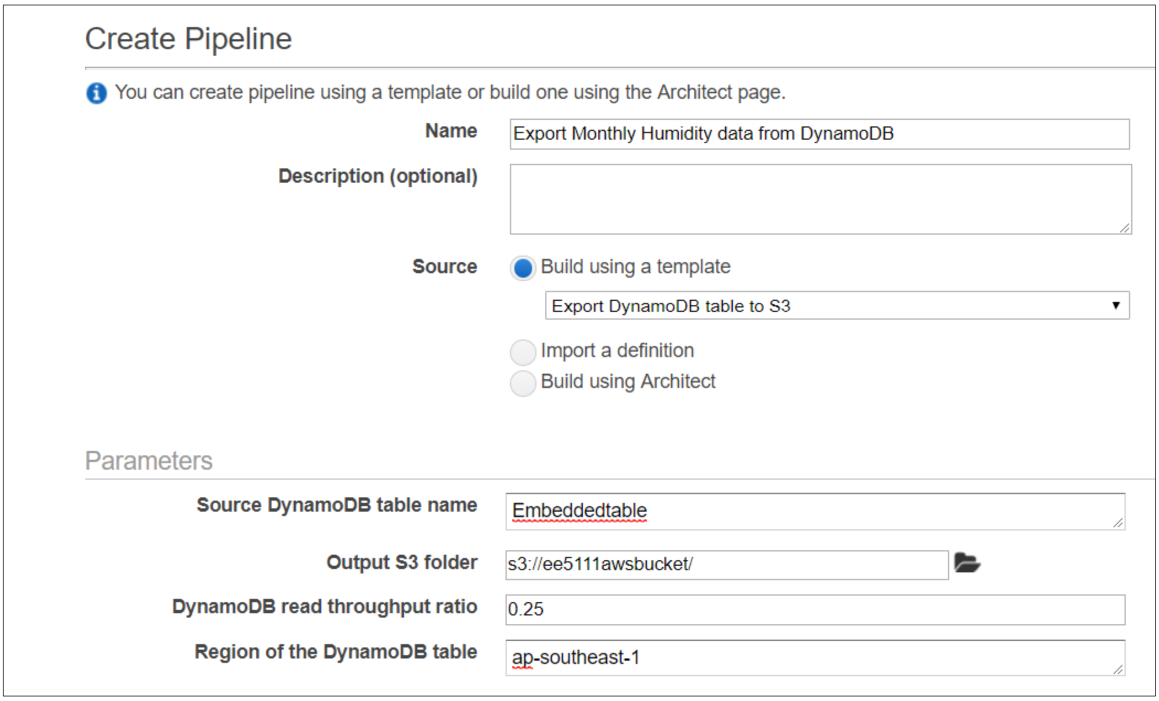
```
In [3]: # Automatically called whenever the shadow is updated.
        def myShadowUpdateCallback(payload, responseStatus, token):
          print()
          print('UPDATE: $aws/things/' + SHADOW_HANDLER +
            '/shadow/update/#')
          print("payload = " + payload)
          print("responseStatus = " + responseStatus)
          print("token = " + token)

        # Create, configure, and connect a shadow client.
        print('Connecting to AWS IOT')
        myShadowClient = AWSIoTMQTTShadowClient(SHADOW_CLIENT)
        myShadowClient.configureEndpoint(HOST_NAME, 8883)
        myShadowClient.configureCredentials(ROOT_CA, PRIVATE_KEY, CERT_FILE)
        myShadowClient.configureConnectDisconnectTimeout(10)
        myShadowClient.configureMQTTOperationTimeout(5)
        myShadowClient.connect()

        Connecting to AWS IOT

Out[3]: True

In [4]: myDeviceShadow = myShadowClient.createShadowHandlerWithName(SHADOW_HANDLER, True)

In [5]: print('Program start!')

        Program start!

In [6]: data = requests.get('https://data.gov.sg/api/action/datastore_search?resource_id=585c24a5-76cd-4c48-9341-9223de5adc1d&l:
        results = data['result']
        results
```

Get real time data from gov website



```
In [7]:

        realdata=results['records']
        realdata
        df_eng = pd.DataFrame(realdata)
        #df_eng = pd.DataFrame.from_dict(results, orient='index')

        #df_eng.transpose()

In [8]: #sensor_name = ['s'+str(i) for i in range(1,2)]
        df_eng.columns = ['ID','Year Month','Extremes Temperature']
        df_eng['ID'] = df_eng['ID']. apply(lambda s:'item_' + str(s))
        df_eng['Matriculation Number'] = 'A0103783H_A0103729H'

        for i in range(1000):
            eng2=df_eng.iloc[[i]]
            UTC=datetime.datetime.utcnow()
            UTC='UTC'+str(UTC)
            eng2['timestamp']=UTC
            eng2.index=["reported"]
            eng2=eng2. to_json(orient='index')
            eng2=' {"state":'+eng2+'}'
            jsonPayload=eng2
            myDeviceShadow.shadowUpdate(jsonPayload,myShadowUpdateCallback,5)

        time.sleep(10)
```

iv. Create Pipeline & Set Up



Create Pipeline

You can create pipeline using a template or build one using the Architect page.

Name: Export Monthly Humidity data from DynamoDB

Description (optional):

Source:
● Build using a template — Export DynamoDB table to S3
○ Import a definition
○ Build using Architect

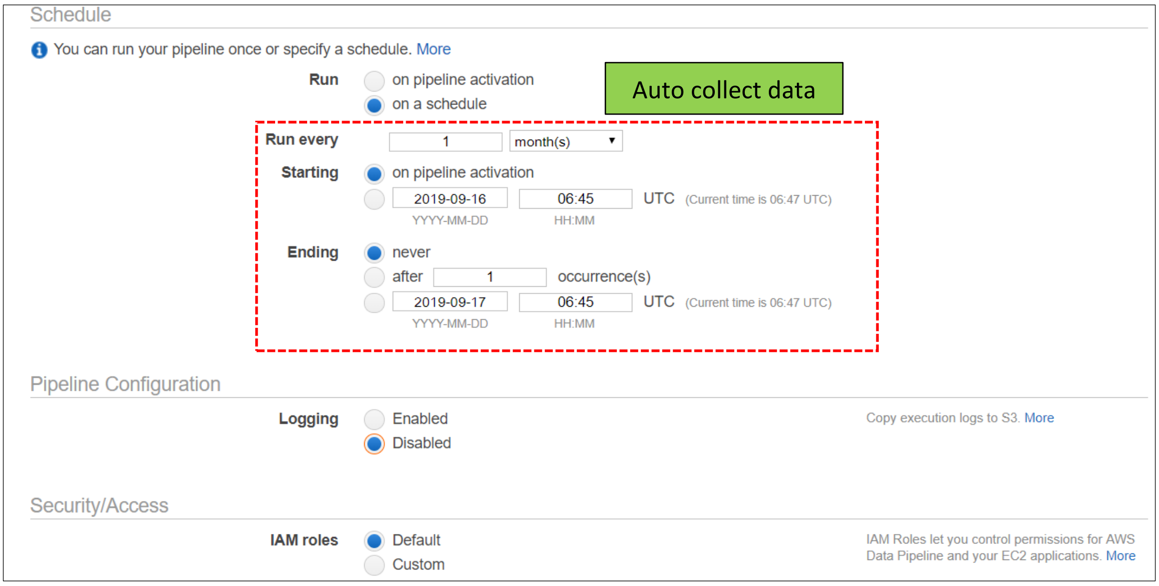
Parameters

Source DynamoDB table name: Embeddedtable

Output S3 folder: s3://ee5111awsbucket/

DynamoDB read throughput ratio: 0.25

Region of the DynamoDB table: ap-southeast-1



Schedule

You can run your pipeline once or specify a schedule. More

Run:
○ on pipeline activation
● on a schedule

Auto collect data

Run every: 1 month(s)

Starting:
● on pipeline activation
○ 2019-09-16 (YYYY-MM-DD) 06:45 (HH:MM) UTC (Current time is 06:47 UTC)

Ending:
● never
○ after 1 occurrence(s)
○ 2019-09-17 (YYYY-MM-DD) 06:45 (HH:MM) UTC (Current time is 06:47 UTC)

Pipeline Configuration

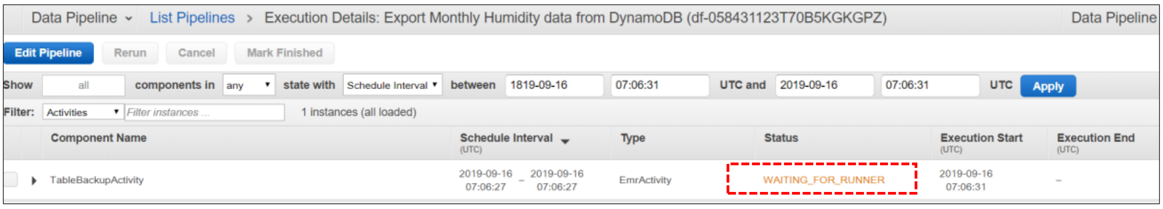
Logging:
○ Enabled
● Disabled

Copy execution logs to S3. More

Security/Access

IAM roles:
● Default
○ Custom

IAM Roles let you control permissions for AWS Data Pipeline and your EC2 applications. More

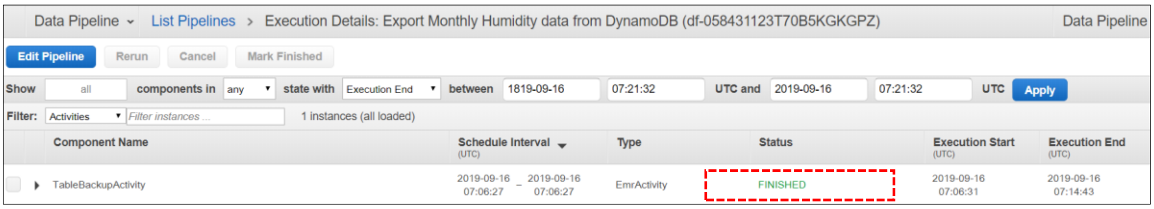


Data Pipeline > List Pipelines > Execution Details: Export Monthly Humidity data from DynamoDB (df-058431123T70B5KGKGPZ)

Show: all components in any state with Schedule Interval between 1819-09-16 07:06:31 UTC and 2019-09-16 07:06:31 UTC

Filter: Activities — Filter instances ... 1 instances (all loaded)

Component Name	Schedule Interval (UTC)	Type	Status	Execution Start (UTC)	Execution End (UTC)
TableBackupActivity	2019-09-16 07:06:27 – 2019-09-16 07:06:27	EmrActivity	WAITING_FOR_RUNNER	2019-09-16 07:06:31	–



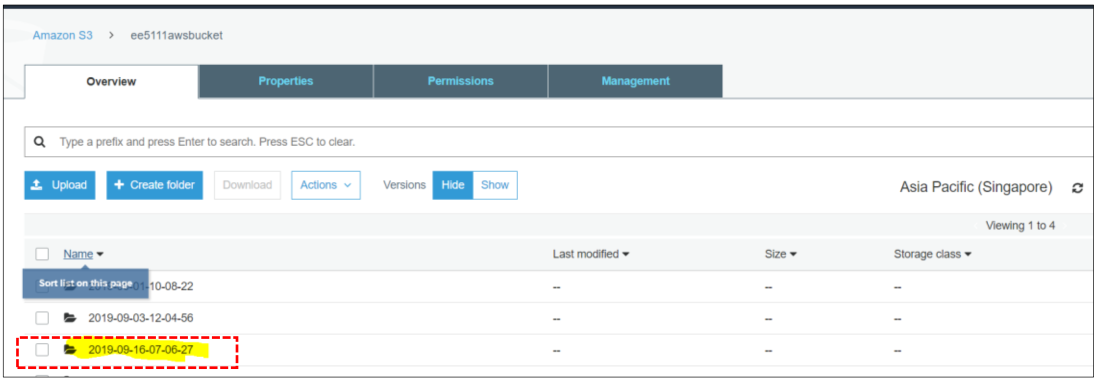
Data Pipeline > List Pipelines > Execution Details: Export Monthly Humidity data from DynamoDB (df-058431123T70B5KGKGPZ)

Show: all components in any state with Execution End between 1819-09-16 07:21:32 UTC and 2019-09-16 07:21:32 UTC

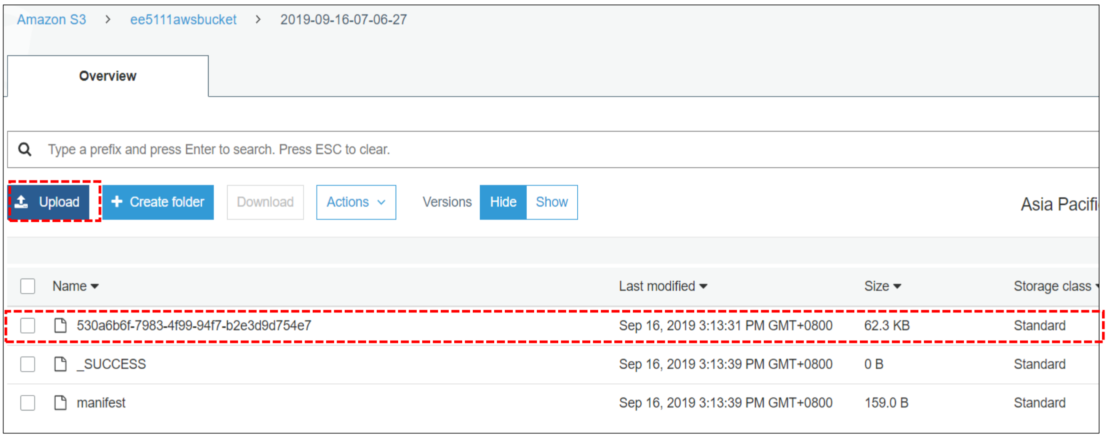
Filter: Activities — Filter instances ... 1 instances (all loaded)

Component Name	Schedule Interval (UTC)	Type	Status	Execution Start (UTC)	Execution End (UTC)
TableBackupActivity	2019-09-16 07:06:27 – 2019-09-16 07:06:27	EmrActivity	FINISHED	2019-09-16 07:06:31	2019-09-16 07:14:43

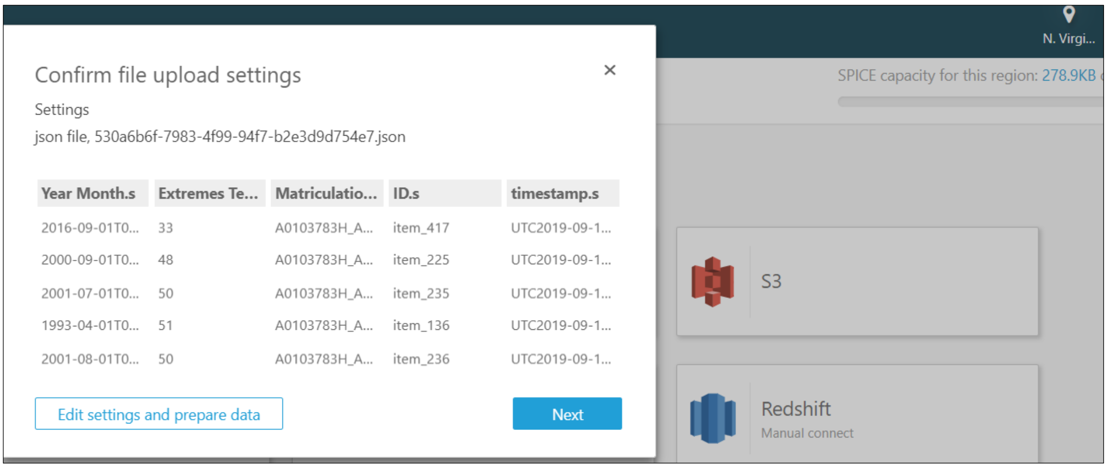
v. Download data from S3 Bucket



□	Name	Last modified	Size	Storage class
□	01-10-08-22	--	--	--
□	2019-09-03-12-04-56	--	--	--
□	2019-09-16-07-06-27	--	--	--


Amazon S3 > ee5111awsbucket > 2019-09-16-07-06-27
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions
Versions
Hide
Show
Asia Pacifi

□	Name	Last modified	Size	Storage class
□	530a6b6f-7983-4f99-94f7-b2e3d9d754e7	Sep 16, 2019 3:13:31 PM GMT+0800	62.3 KB	Standard
□	_SUCCESS	Sep 16, 2019 3:13:39 PM GMT+0800	0 B	Standard
□	manifest	Sep 16, 2019 3:13:39 PM GMT+0800	159.0 B	Standard

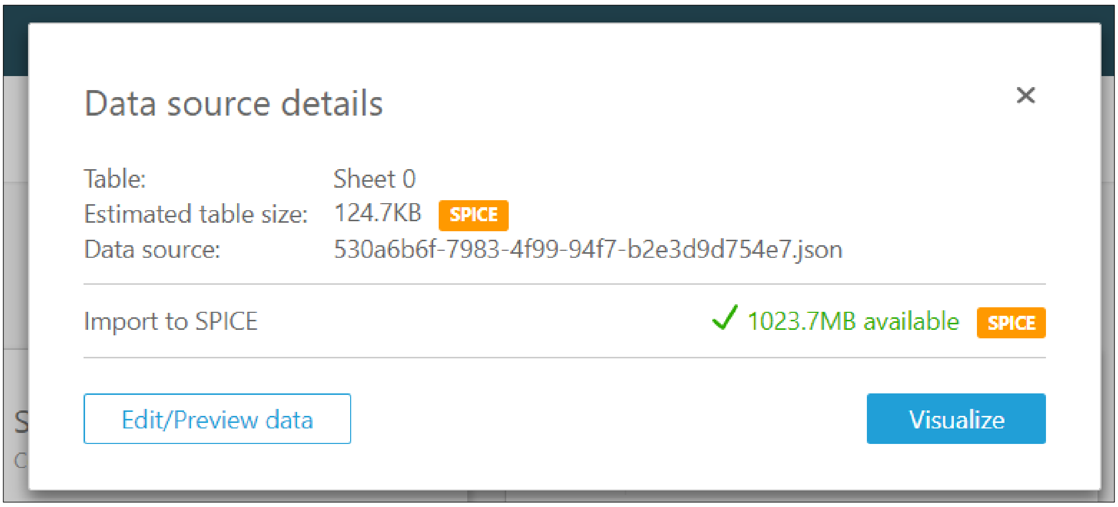


Confirm file upload settings

Settings

json file, 530a6b6f-7983-4f99-94f7-b2e3d9d754e7.json

Year Month.s	Extremes Te...	Matriculatio...	ID.s	timestamp.s
2016-09-01T0...	33	A0103783H_A...	item_417	UTC2019-09-1...
2000-09-01T0...	48	A0103783H_A...	item_225	UTC2019-09-1...
2001-07-01T0...	50	A0103783H_A...	item_235	UTC2019-09-1...
1993-04-01T0...	51	A0103783H_A...	item_136	UTC2019-09-1...
2001-08-01T0...	50	A0103783H_A...	item_236	UTC2019-09-1...



Data source details

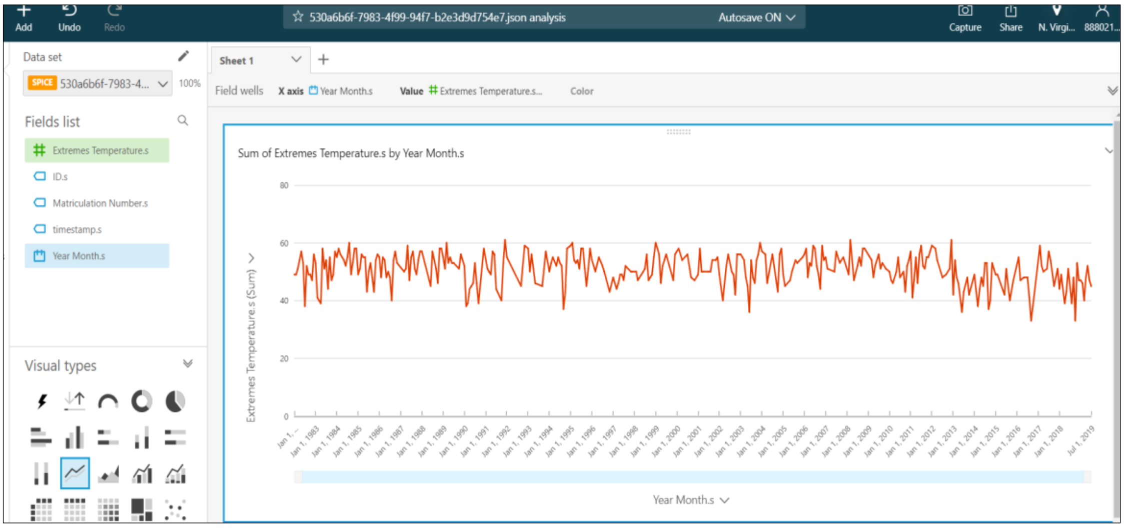
Table: Sheet 0

Estimated table size: 124.7KB SPICE

Data source: 530a6b6f-7983-4f99-94f7-b2e3d9d754e7.json

Import to SPICE ✓ 1023.7MB available SPICE

vi. Using QucikSight to visualize the data from desktop.



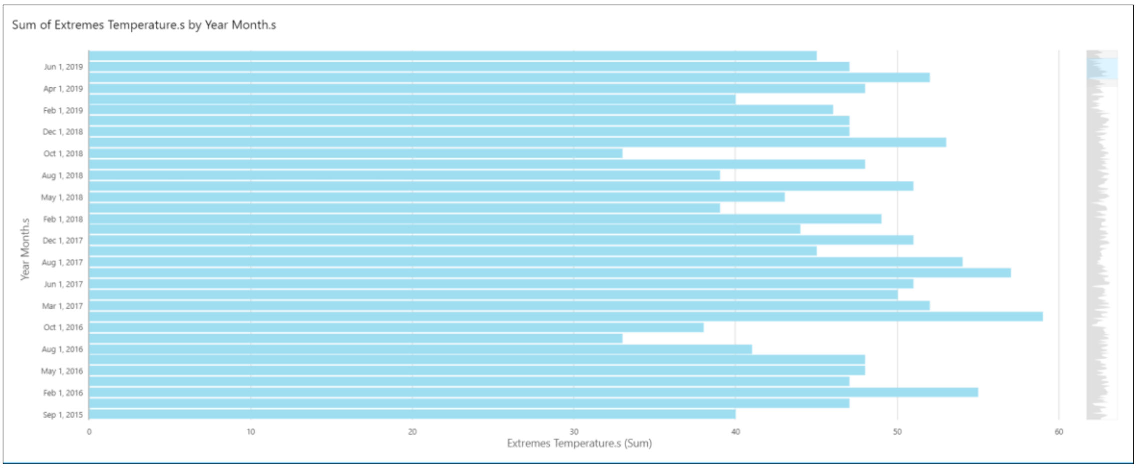
☆ 530a6b6f-7983-4f99-94f7-b2e3d9d754e7.json analysis

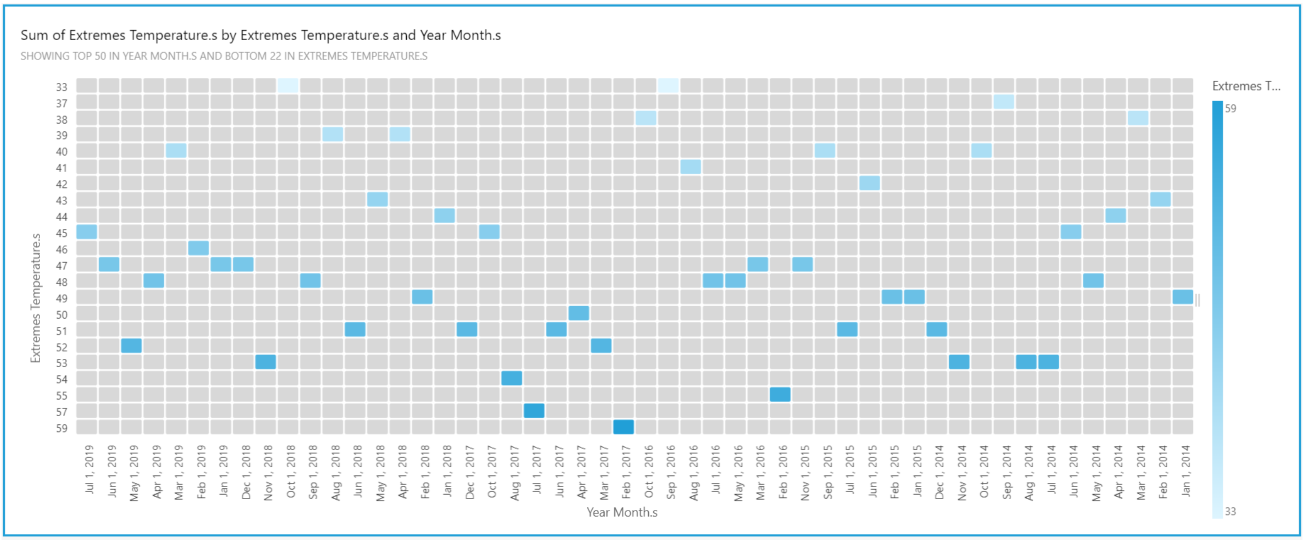
Autosave ON

Data set: 530a6b6f-7983-4... 100%

Field wells X axis Year Month.s Value Extremes Temperature.s... Color

Fields list: Extremes Temperature.s, ID.s, Matriculation Number.s, timestamp.s, Year Month.s





Sum of Extremes Temperature.s by Extremes Temperature.s and Year Month.s

SHOWING TOP 50 IN YEAR MONTH.S AND BOTTOM 22 IN EXTREMES TEMPERATURE.S