SCHOOL OF DIGITAL MEDIA AND INFOCOMM TECHNOLOGY (DMIT)

# IOT CA2
# Step-by-step Tutorial

**DIPLOMA IN BUSINESS INFORMATION TECHNOLOGY**
**DIPLOMA IN INFORMATION TECHNOLOGY**
**DIPLOMA IN INFOCOMM SECURITY MANAGEMENT**

## ST0324 Internet of Things (IOT)

**Date of Submission:** 20 February 2018

**Prepared for:** Ms Lin Zhao

**Class:** DISM/FT/3A/33

**Submitted by:**


| Student ID | Name |
|---|---|
| 1529377 | Law Si Ying, Joy |
| 1529393 | Sean Phang |

# Table of Contents

# Section 1
# Overview of Project

## A. Where We Have Uploaded Our Tutorial

https://github.com/joylaw/IOT_CA2

## B. Why Have We Chosen to Upload to This Site

The main reason why our group has decided to upload our project onto GitHub is that it makes the process of sharing and editing the code extremely easy for all of us. Additionally, we are very familiar with GitHub as compared to the rest of the sites since we have made use of it in the past.

## C. What Have We Uploaded

iot_ca2
- static (bower_components, build, dist, plugins, card.jpg, greenled.jpg, home.jpg, redled.jpg, stylesheet.css, yellowled.jpg)
- templates (image.html, index.html, newUser.html)
- iot-policy.json
- iot-role-trust.json
- MFRC522.py
- pub.py
- server.py

resources
- iot_file.zip

DISMFT3A33_CA2-TutorialDocument_JoyLaw_SeanPhang.pdf

Note: bower_components, build, dist & plugins contain other files needed for the web interface

# D. What is the Application About?

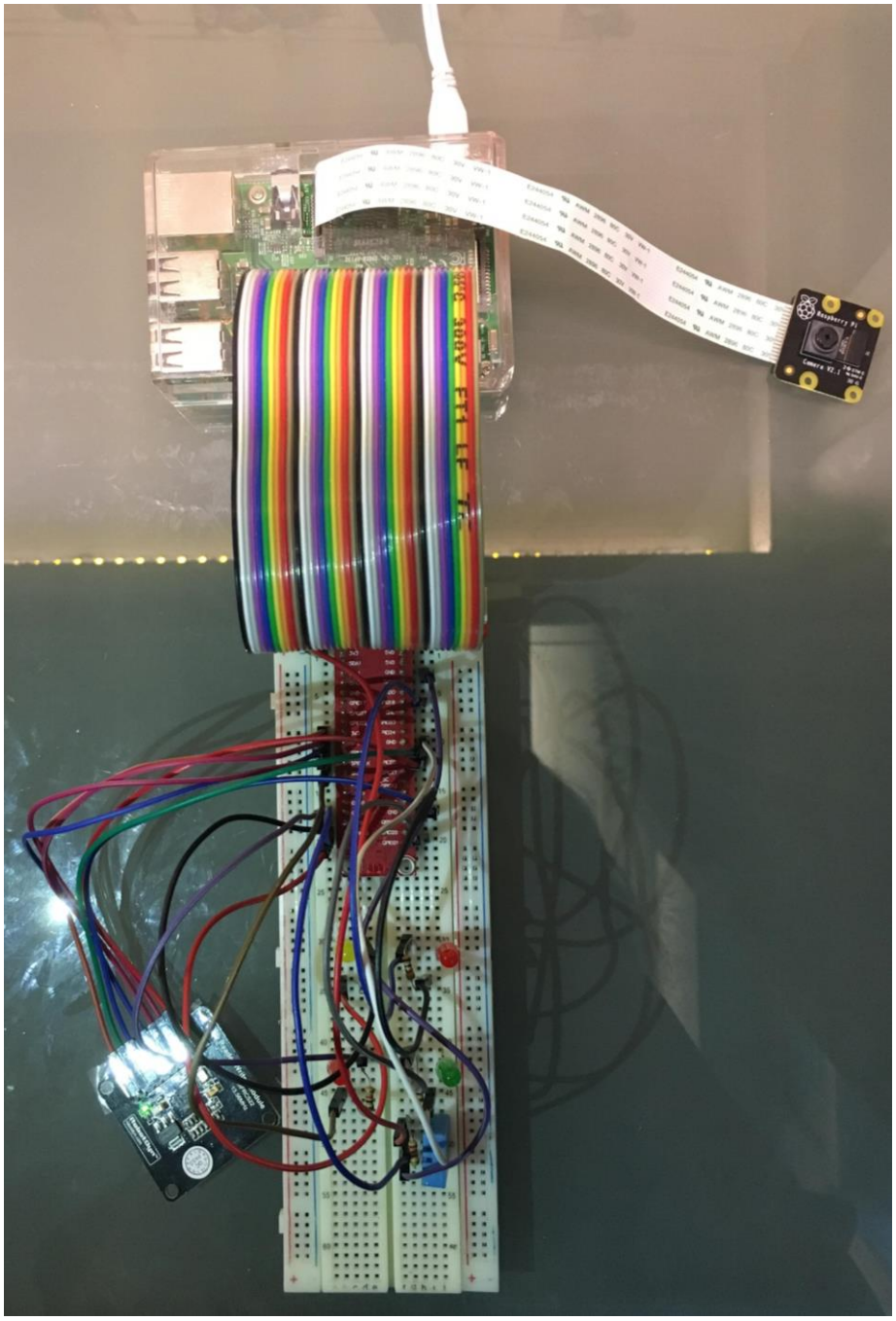
The concept of the application is a smart home security and environment monitoring system, where historical view of people who has entered the home via an image, date & time stamp, as well as live data of the current temperature and humidity of the home can be seen. Additionally, users are also able to control their lights in the house remotely.

# E. Summary of the Steps That will be Described

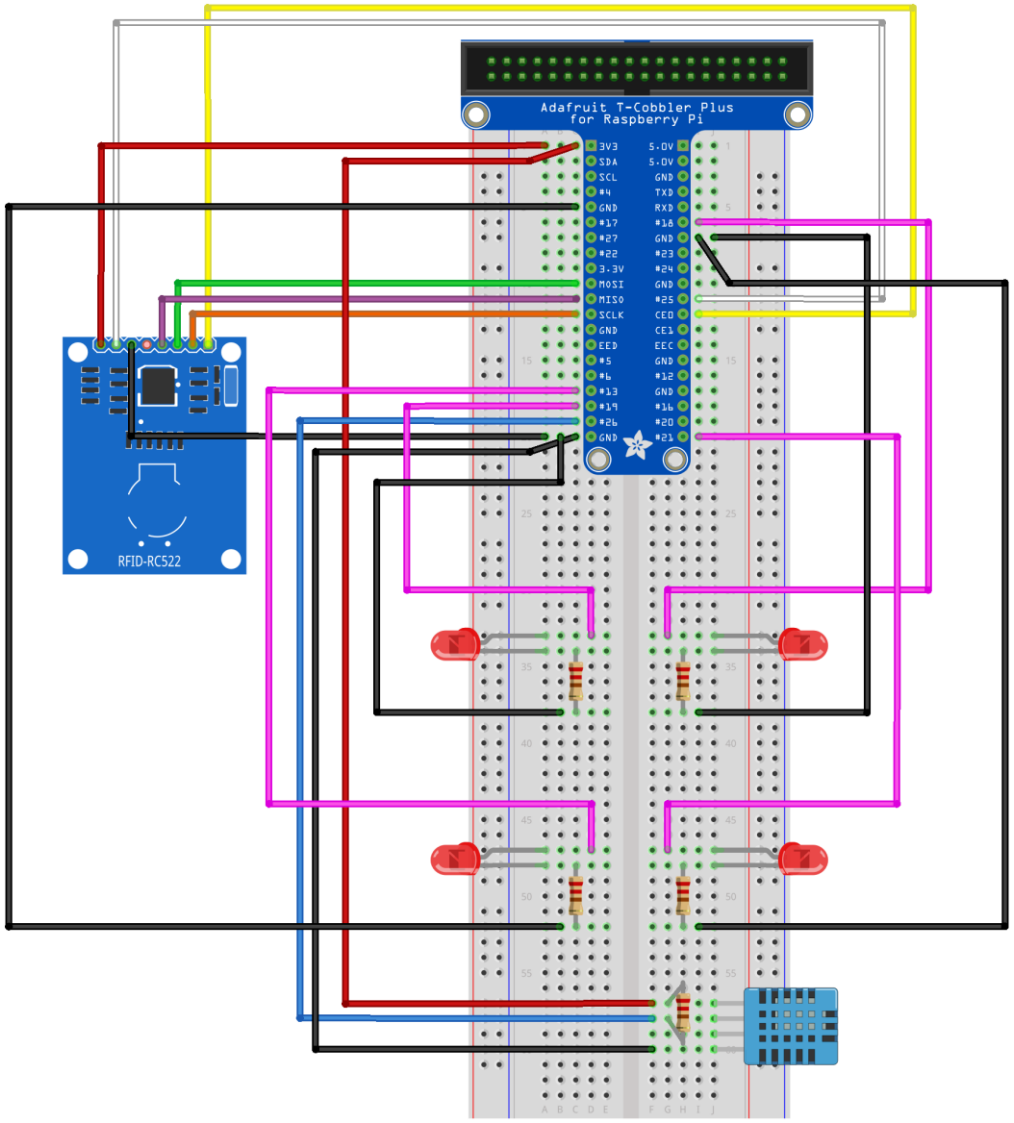
Provide a bullet list of the steps that will be covered in the other parts of this tutorial

| | Section | Description |
|---|---|---|
| 1) | Overview | Overview of this assignment tutorial |
| | | |
| **Sections 2 to 11 provides the step-by-step instructions to set up the application** | | |
| 2) | Hardware requirements | Provides a hardware checklist |
| 3) | Install Libraries | Shows the steps needed to installed the required libraries |
| 4) | Enable Configurations | Provides the steps needed to enable certain configurations |
| 5) | Create AWS Account | Steps needed to create an AWS Educate Account |
| 6) | Login to AWS | Logging in to AWS Educate |
| 7) | Configure AWS Services | Provides a step-by-step guide to enable the following AWS Services: AWS IoT, RDS, S3, Lambda |
| 8) | Configure AWS IoT Rules | Creating and assigning role and policies |
| 9) | The Subscriber | The code needed to run the server/subscriber |
| 10) | The Publisher | The code needed to run the publisher |
| 11) | The Web Interface Design | The code needed to display onto a dashboard |

# F. How does the Final RPI Set-up Looks Like?



# G. Completed Fritzing Diagram




Note: This fritzing diagram excludes the Raspberry Pi, Rainbow Ribbon, piCamera as well as the power source.

# H. How does the Web Application Look Like?

index.html

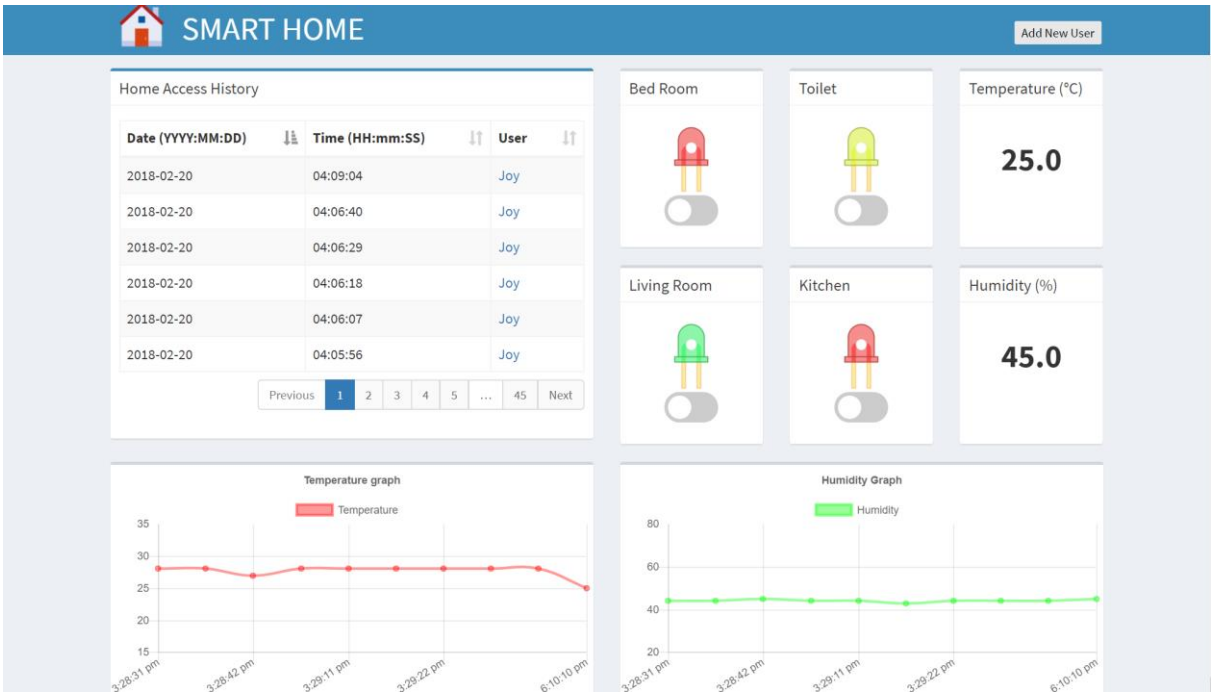
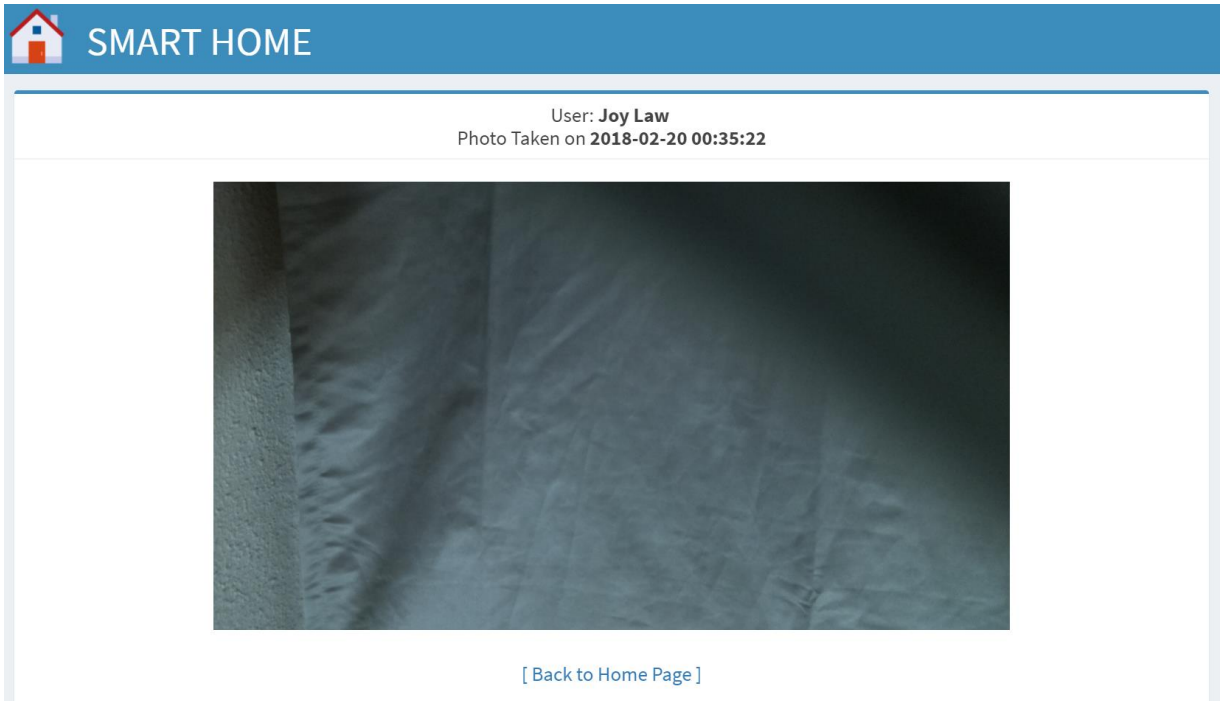
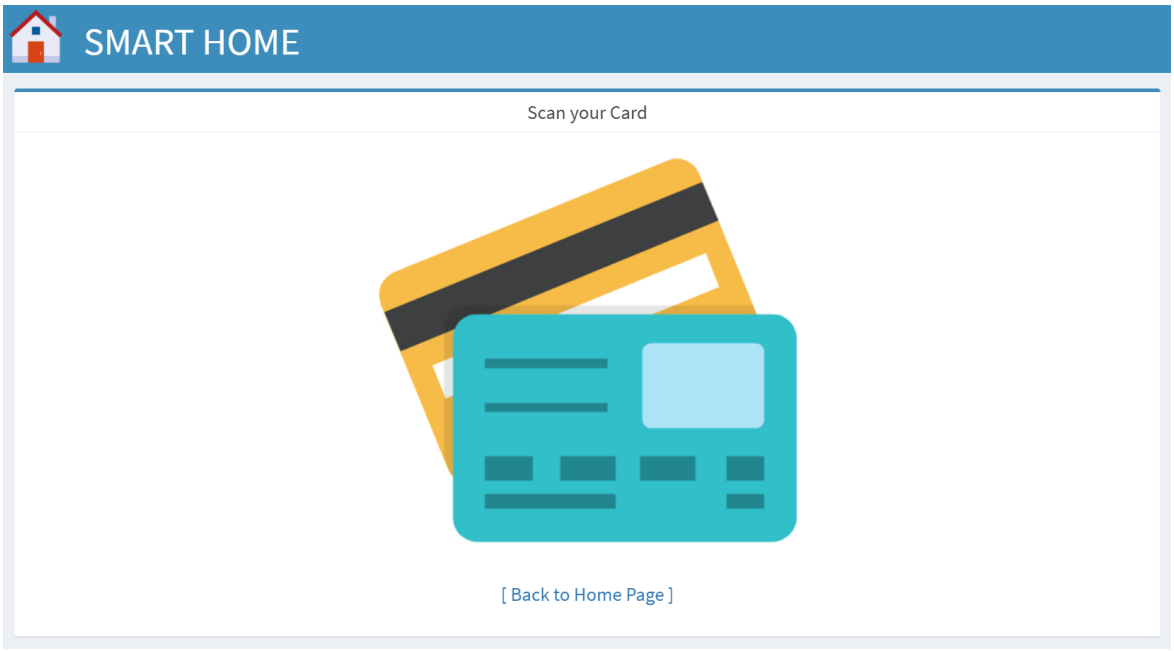



image.html



newUser.html



# Section 2
# Hardware Requirements

## Hardware Checklist

A review of the hardware components needed for this assignment.

| | Hardware | Quantity |
|---|---|---|
| 1) | Raspberry Pi 3 Model B | 1 |
| 2) | Breadboard | 1 |
| 3) | T-Cobbler Kit | 1 |
| 4) | Ribbon Cable | 1 |
| 5) | DHT11 Temperature & Humidity Sensor | 1 |
| 6) | 220Ω Resistor for DHT11 Sensor | 1 |
| 7) | LED | 4 |
| 8) | 1.6kΩ Resistor for LED | 4 |
| 9) | RFID/NFC MFRC522 Card Reader/Writer Module | 1 |
| 10) | Raspberry Pi Camera (piCam) | 1 |
| 11) | Jumper Wire | 18 |

# Section 3
# Install the Necessary Libraries

## A. Install Flask

a) To install flask:

```
sudo pip install flask
```

## B. Install gevent

a) To install gevent:

```
sudo pip install gevent
```

## C. Install Python-dev

a) To install the Python development libraries:

```
sudo apt-get install python-dev
```

## D. Install SPI-Py

a) Set up the SPI Python libraries as the card reader uses the SPI interface:

```
cd /home/pi
git clone https://github.com/lthiery/SPI-Py.git
cd /home/pi/SPI-Py
sudo python setup.py install
```

## E. Install Boto

a) To install Boto, the Python library for AWS:

```
sudo pip install boto3
```

# F. Install AWS CLI

a) To install the AWS Command-Line Interface Client:

```
sudo pip install awscli
```

# G. Install AWS Python Library

a) To install the AWS Python Library:

```
sudo pip install AWSIoTPythonSDK
```

# H. Install Paho MQTT

a) As the AWS Python SDK has dependency on paho-mqtt, install Paho MQTT:

```
sudo pip install paho-mqtt
```

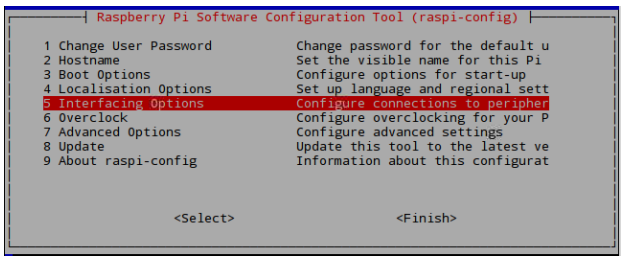
# Section 4
# Enable the Necessary Configurations

## A. Enable SPI via raspi-config

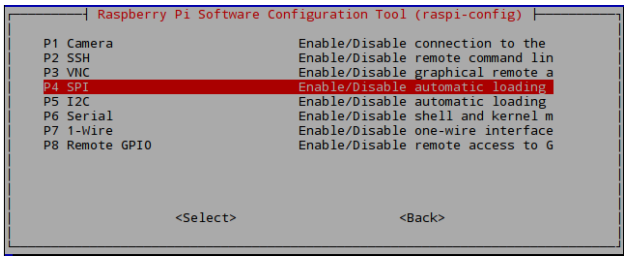
a) In your command prompt, enter the following:

```
sudo rasp-config
```

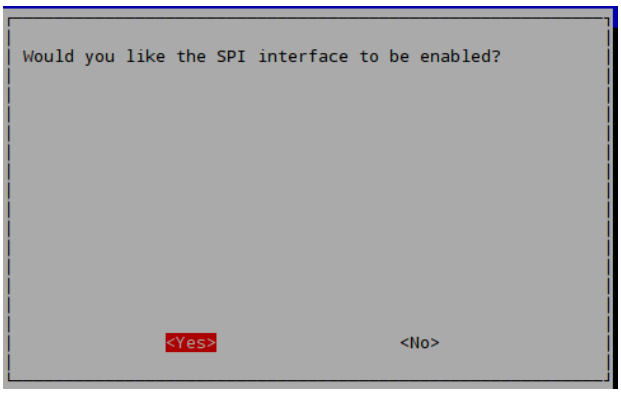
b) Using your up-down arrow on your keyboard, move the red highlight over to the following, 5 Interfacing Options, and press “Enter” on your keyboard.



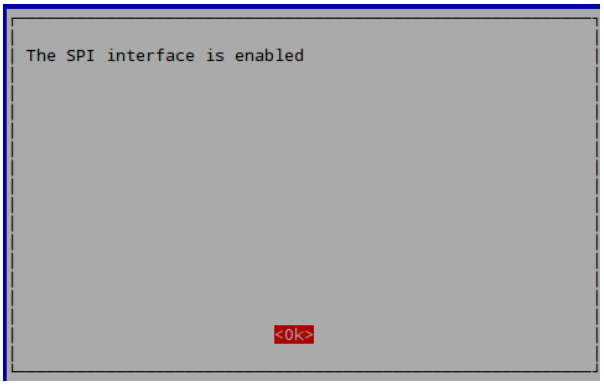

c) Move the red highlight over to P4 SPI and press “Enter” on your keyboard.




d) You will then be prompted as to whether you want to enable SPI interface, press “Enter” on your keyboard again for Yes.




e) You will then see the following. Press “Enter” again to select Ok.



f) You will then be directed back to the original raspi-config location, press 'Tab' on your keyboard twice, then 'Enter' to select Finish.

# B. Enable Device Tree in boot.txt

a) View /boot/config.txt such that SPI is enabled

```
sudo nano /boot/config.txt
```

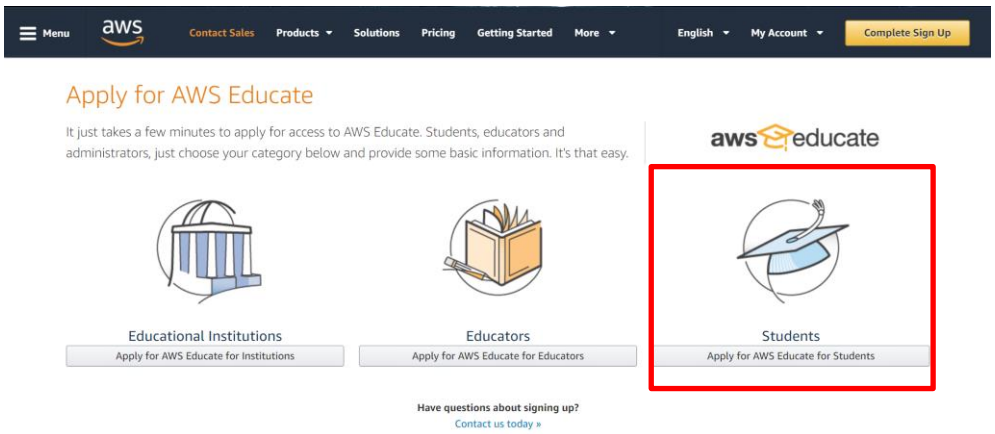
b) Ensure that the following are in /boot/config.txt, if not, add them in.

```
device_tree_param=spi=on
dtoverlay=spi-bcm2835
```

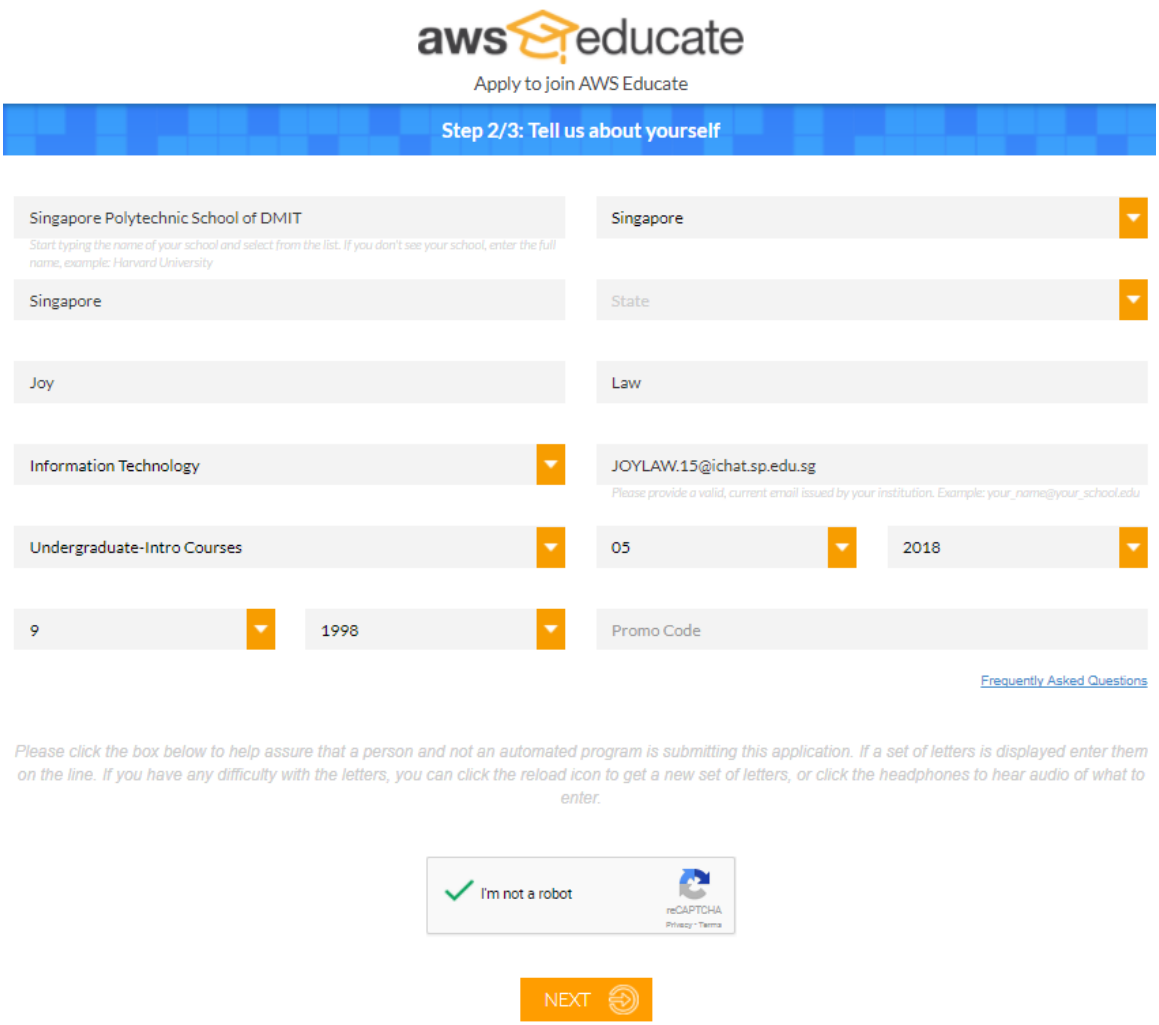
# Section 5
# Sign Up for an AWS Educate Account

## A. Create a New AWS Educate Account

a) Go to AWS Educate website: https://aws.amazon.com/education/awseducate/apply/
b) Select 'Students' by click on 'Apply for AWS Educate for Students'

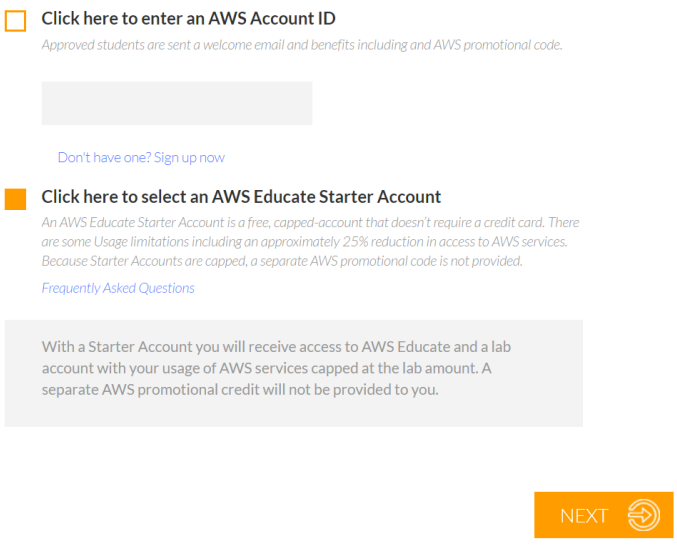


c) You will then be redirect to the following page to fill in your personal information.

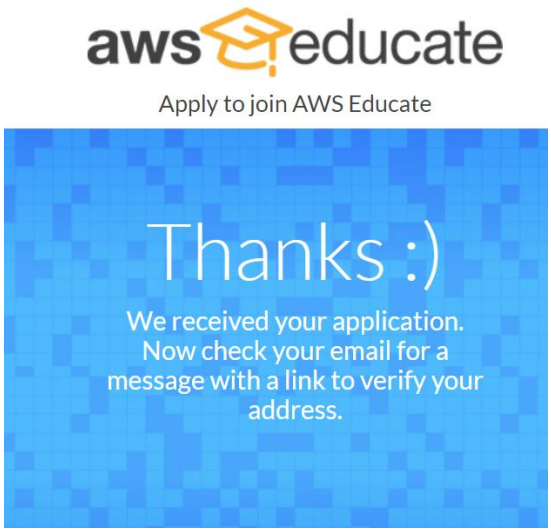



d) Click 'Next' to proceed.

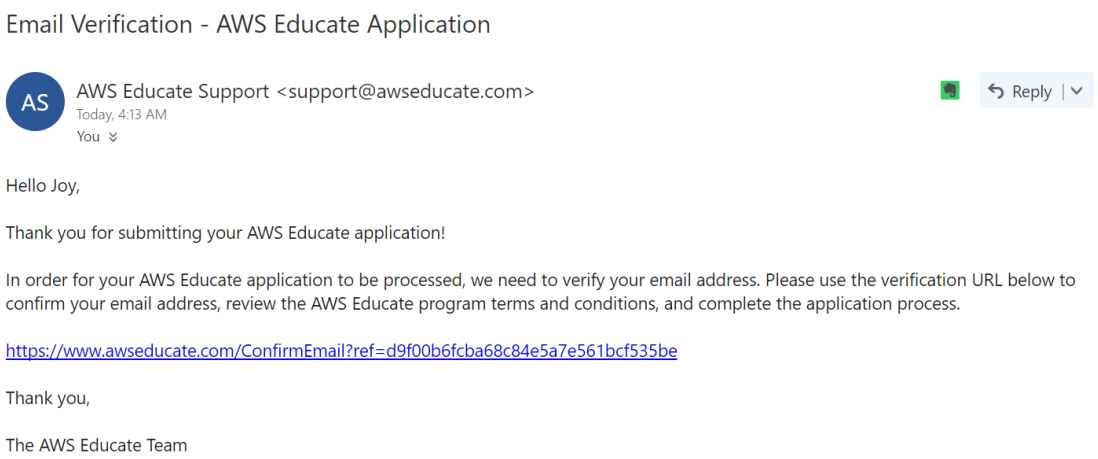
e) Select 'Click here to select an AWS Educate Starter Account' and click 'Next'



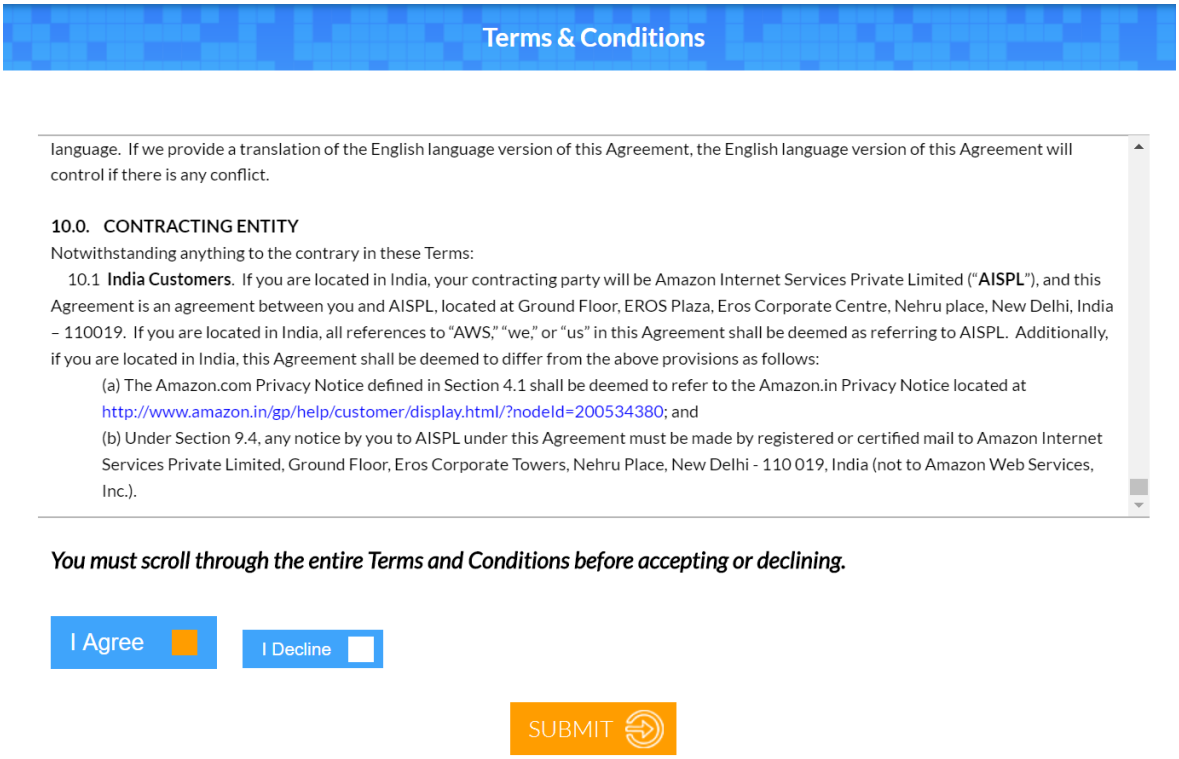
f) If the previous steps have been done properly, you should see the following web page.




g) The following step may take awhile, but once your AWS Educate account has been set up, you will receive an email that you used to signed up an account for. It will look similar to the following:

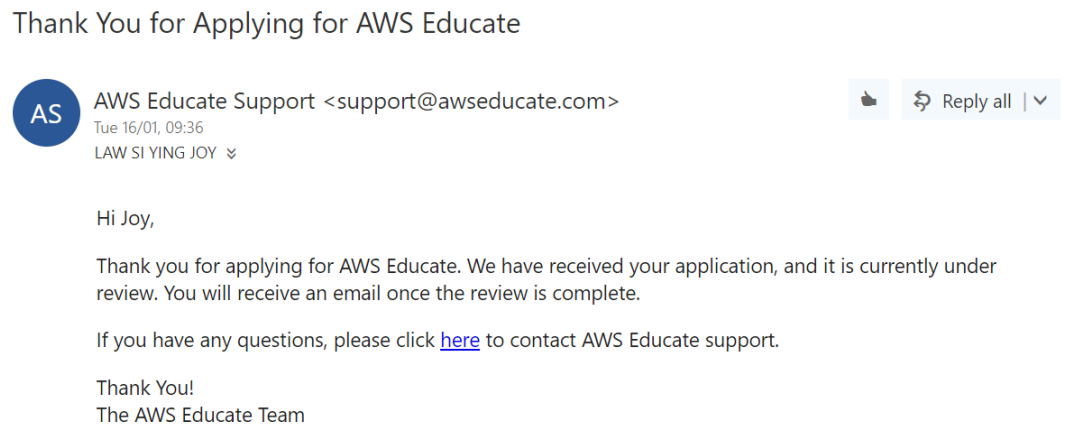


h) Click on the link that is attached to the email to verify your email address and you will be redirected to the following web page for terms and condition. Scroll down to the bottom of the terms & condition, then select 'I Agree' and click 'Submit'




i) If the verification is successful, you should see the following web page.

j) Additionally, you will receive an email similar to the one below.



Thank You for Applying for AWS Educate

AS AWS Educate Support <support@awseducate.com>
Tue 16/01, 09:36
LAW SI YING JOY

Reply all

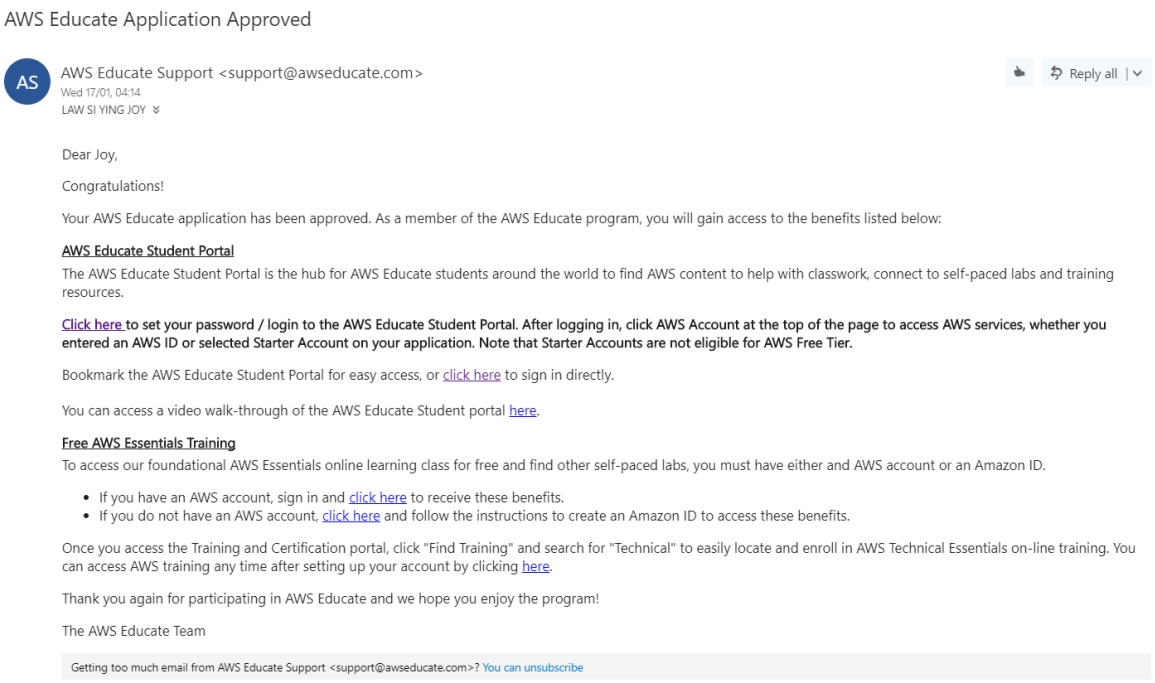
Hi Joy,

Thank you for applying for AWS Educate. We have received your application, and it is currently under review. You will receive an email once the review is complete.

If you have any questions, please click here to contact AWS Educate support.

Thank You!
The AWS Educate Team

k) The review process may take a while, but once it is successful, you will receive an email notifying you that your AWS Educate account is ready for use.



AWS Educate Application Approved

AS AWS Educate Support <support@awseducate.com>
Wed 17/01, 04:14
LAW SI YING JOY

Reply all

Dear Joy,

Congratulations!

Your AWS Educate application has been approved. As a member of the AWS Educate program, you will gain access to the benefits listed below:

**AWS Educate Student Portal**

The AWS Educate Student Portal is the hub for AWS Educate students around the world to find AWS content to help with classwork, connect to self-paced labs and training resources.

Click here to set your password / login to the AWS Educate Student Portal. After logging in, click AWS Account at the top of the page to access AWS services, whether you entered an AWS ID or selected Starter Account on your application. Note that Starter Accounts are not eligible for AWS Free Tier.

Bookmark the AWS Educate Student Portal for easy access, or click here to sign in directly.

You can access a video walk-through of the AWS Educate Student portal here.

**Free AWS Essentials Training**

To access our foundational AWS Essentials online learning class for free and find other self-paced labs, you must have either and AWS account or an Amazon ID.

- If you have an AWS account, sign in and click here to receive these benefits.
- If you do not have an AWS account, click here and follow the instructions to create an Amazon ID to access these benefits.

Once you access the Training and Certification portal, click "Find Training" and search for "Technical" to easily locate and enroll in AWS Technical Essentials on-line training. You can access AWS training any time after setting up your account by clicking here.

Thank you again for participating in AWS Educate and we hope you enjoy the program!

The AWS Educate Team

Getting too much email from AWS Educate Support <support@awseducate.com>? You can unsubscribe

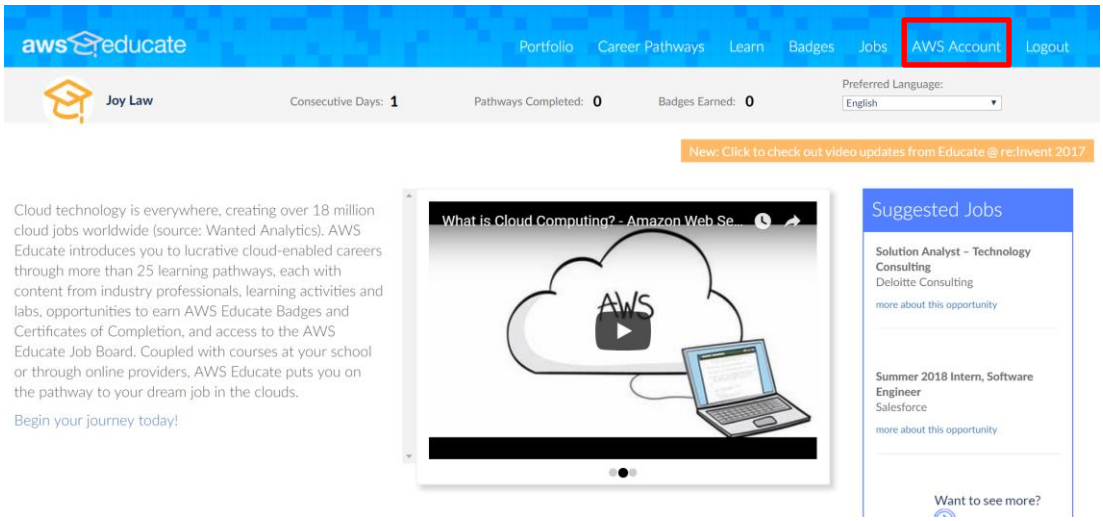
l) Following the instructions given in the email, set your password. Afterwards, click on the second ‘click here’ to access the login page. You can bookmark the page for easy access later on.

# Section 6
# Sign in to AWS

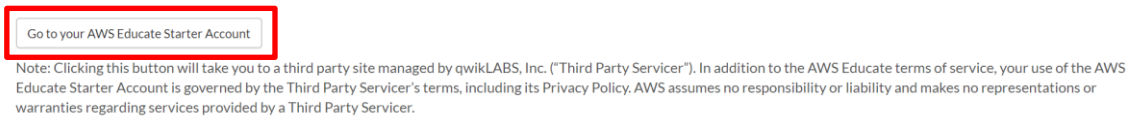
## A. Sign in to AWS

a) Login with the email and password that you have previously set. And click 'SIGN IN'.

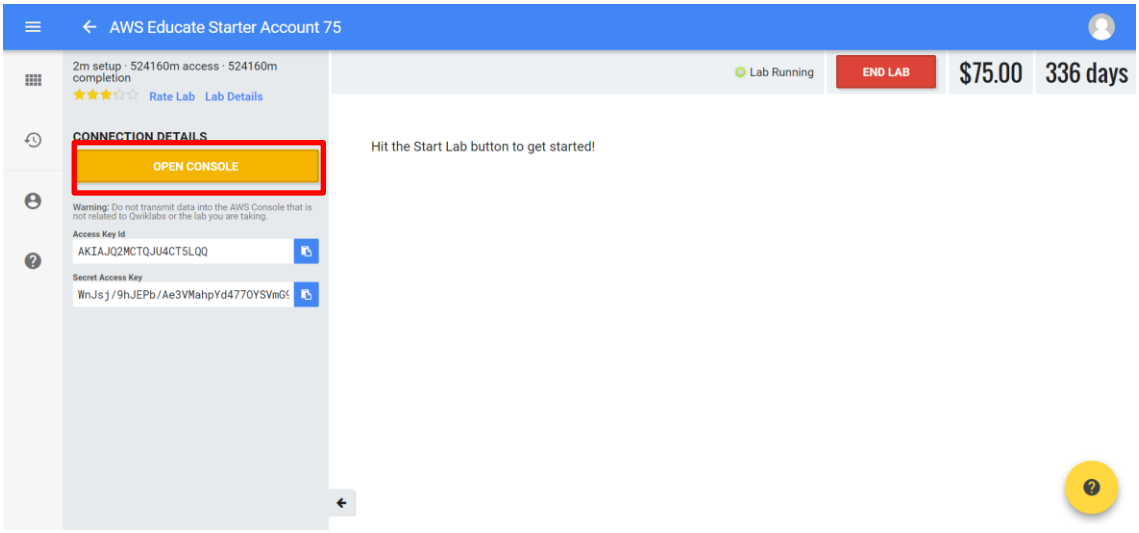

b) Upon logging in, you should see the following web page. Click on 'AWS Account' on the top right-hand corner.




c) Click on 'Go to your AWS Educate Starter Account'



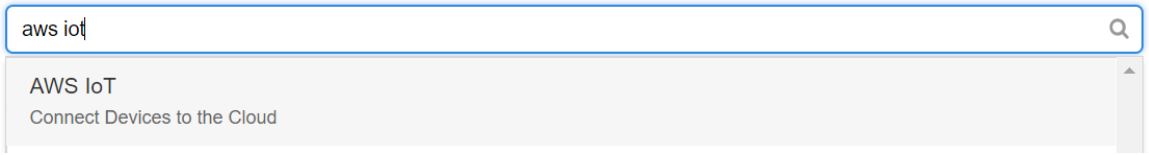
d) You will then be redirected to the following web page. Click on 'OPEN CONSOLE' on the left-hand menu.



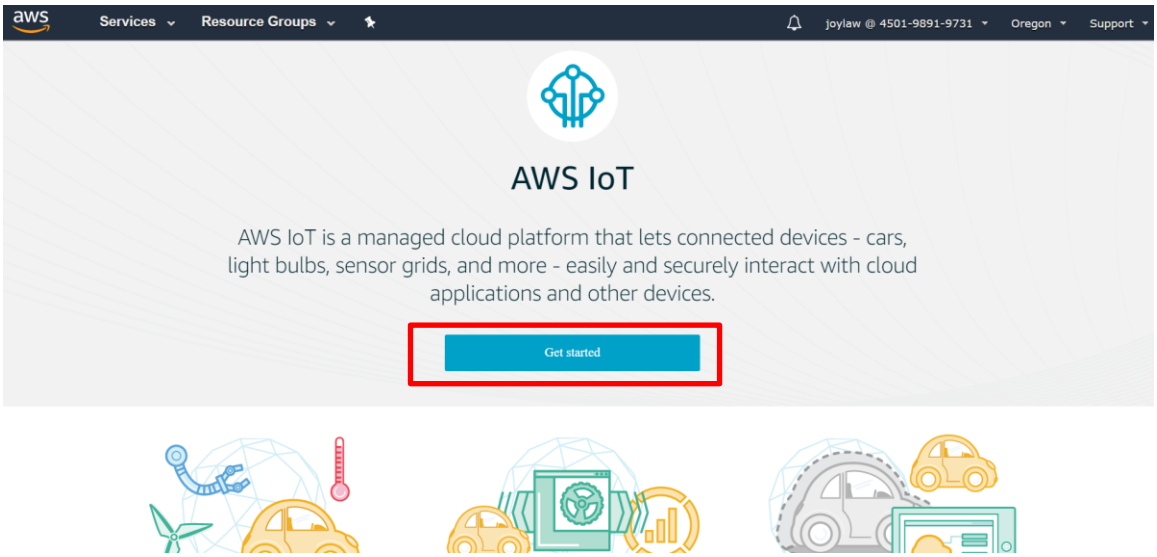
# Section 7
# Configure AWS Services

## A. Configure AWS IoT

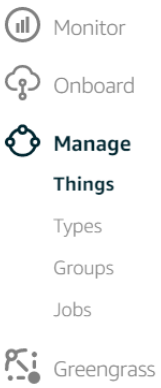
a) Go back to the home page of the console and search for the service: AWS IoT.



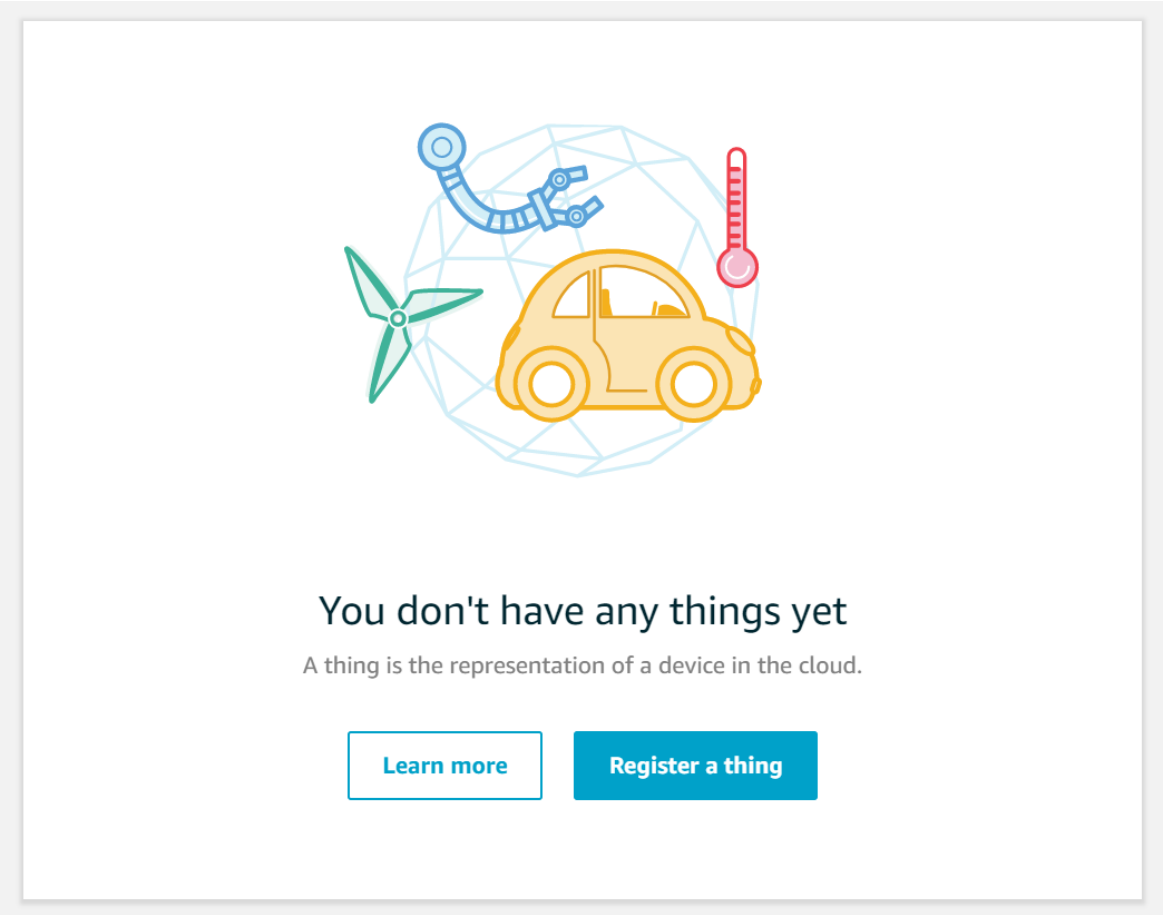

b) You will then be redirect to the following page and click on 'Get started'



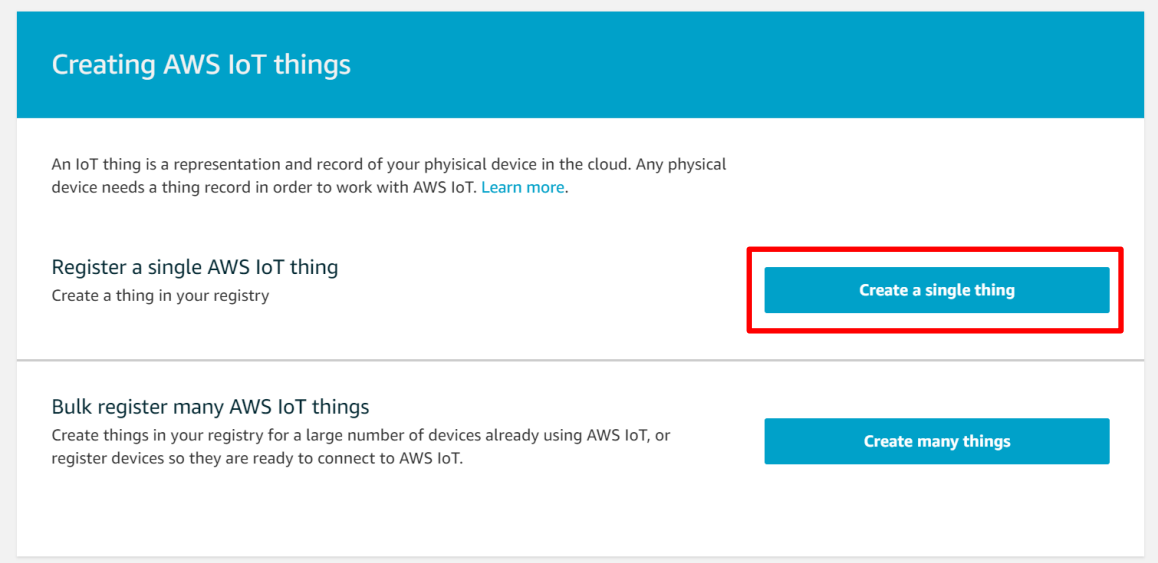

c) On the left-hand panel, click on 'Manage'



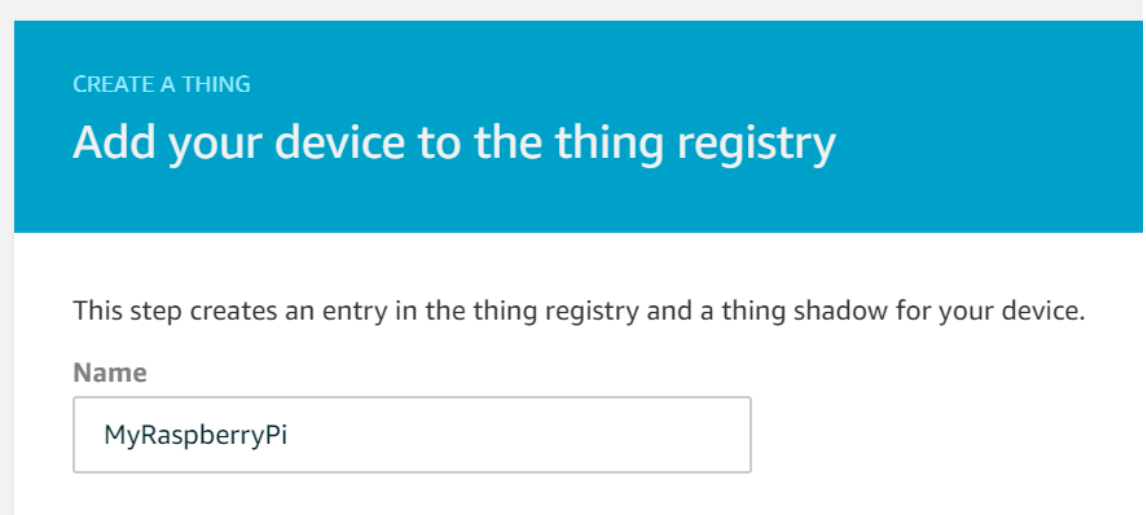
d) Then click on ‘Register a thing’.



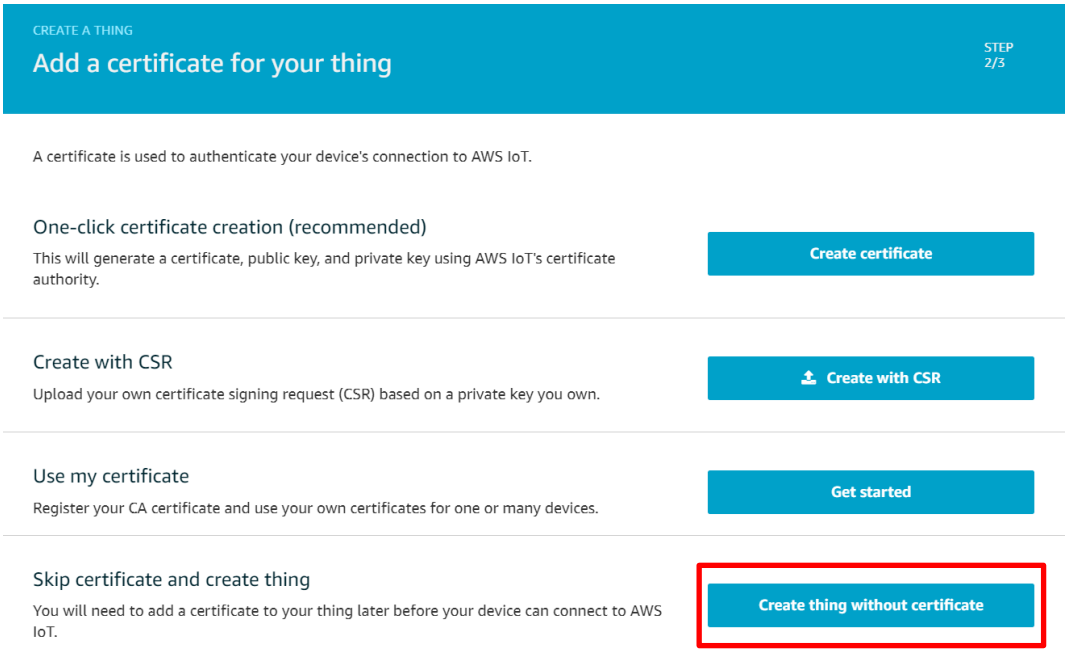

e) You will then be asked if you want to ‘Create a single thing’ or ‘Create many things’. Select the first one – ‘Create a single thing’.



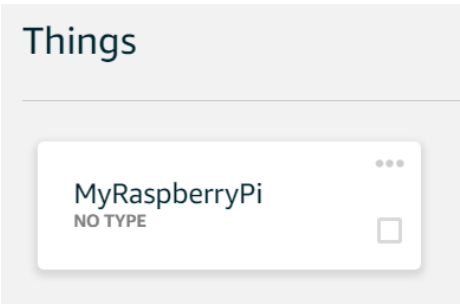
f) Add in a name for your Raspberry Pi, leave the rest as default and click 'Next'



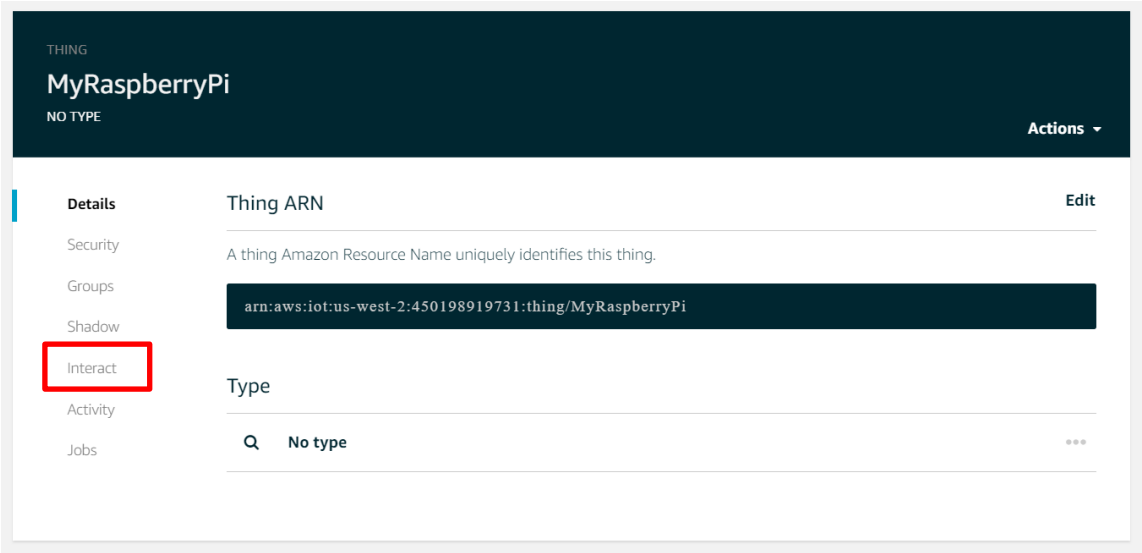

g) When asked if you want to create a certificate for your thing, select 'Create thing without certificate'.



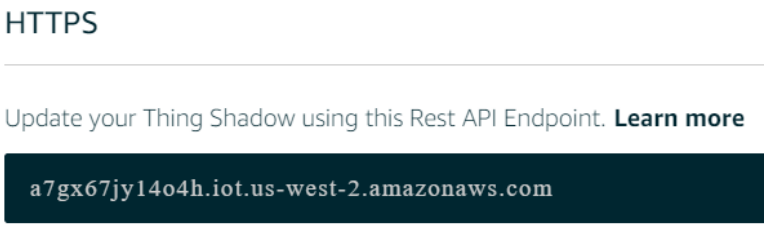

h) Once successful, you will be redirected to the following web page and you should see the following. Click on 'MyRaspberryPi'.



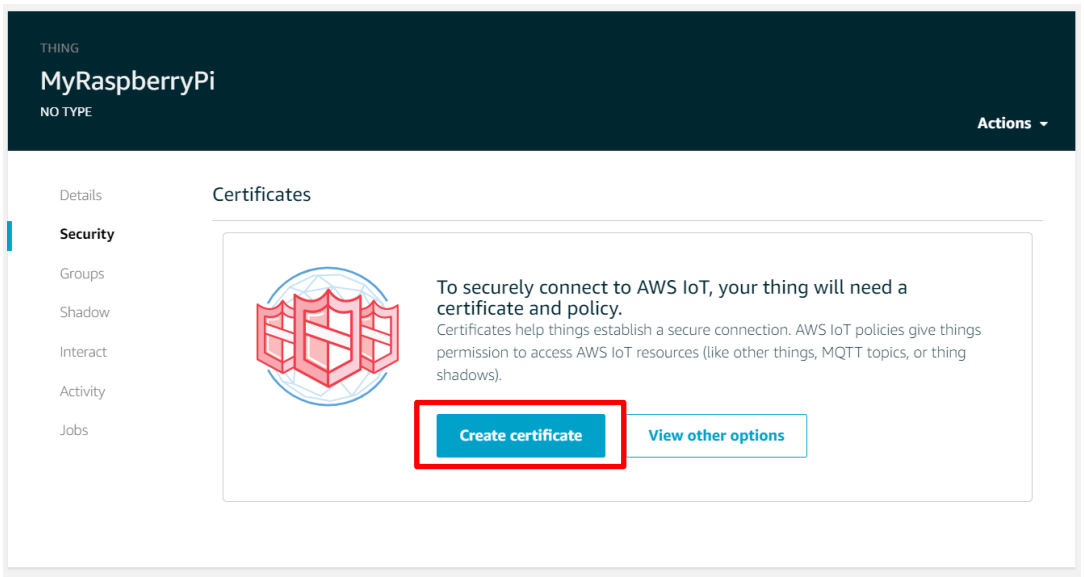
i) You should see the following. Click on 'Interact'.



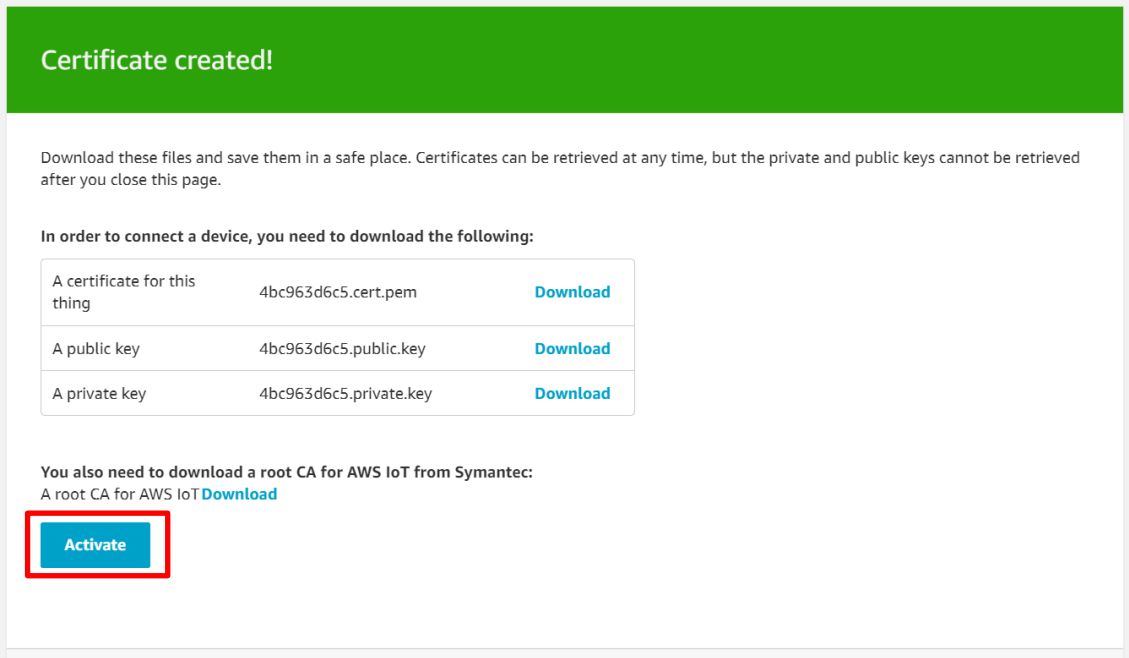

j) Copy the following onto a notepad as it will be used later on. Next, click 'Security' on the left menu



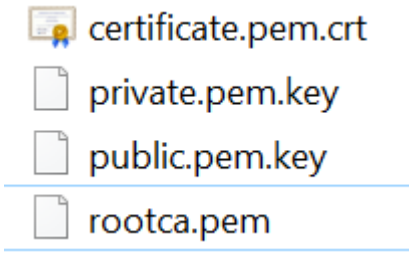

k) Click on 'Create certificate' to generate a X.509 certificate as well as key pair.



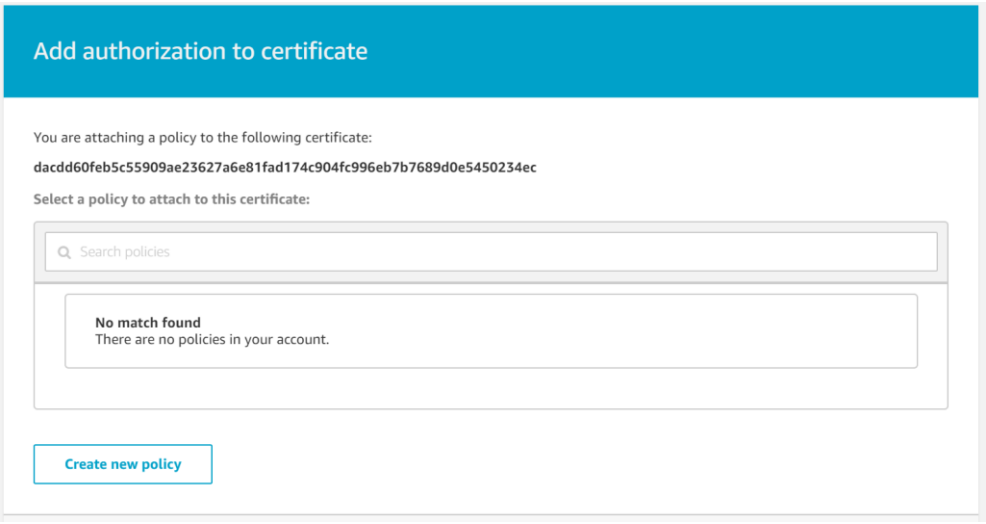
l) Once successful, you should see the following web page. Click on 'Activate'



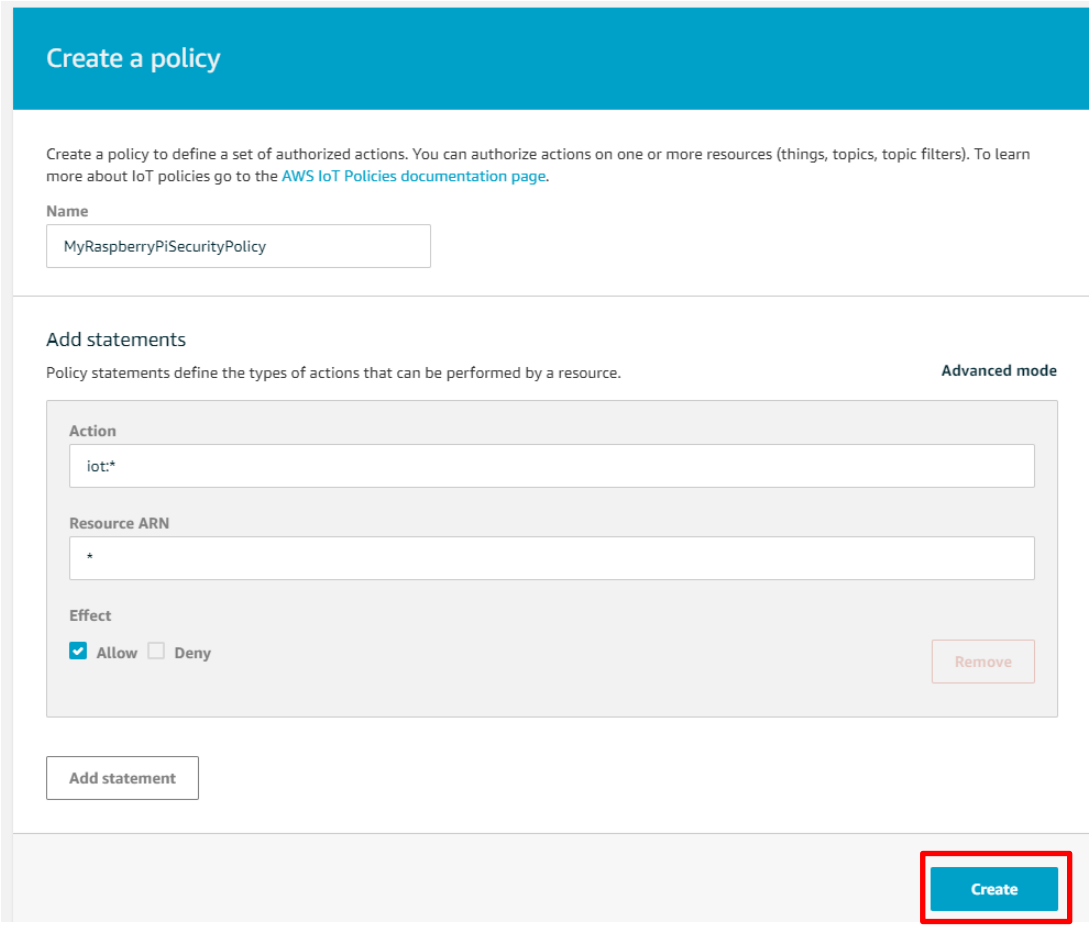

m) Click on the 4 download links to download the respective files and rename them accordingly.



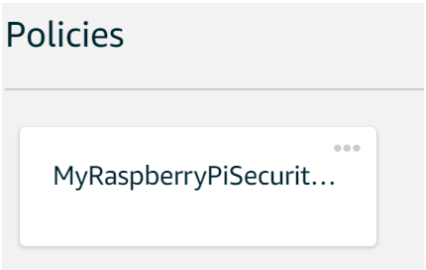

n) Afterwards, click on 'Attach a policy' at the bottom and you will be redirected to the following page.



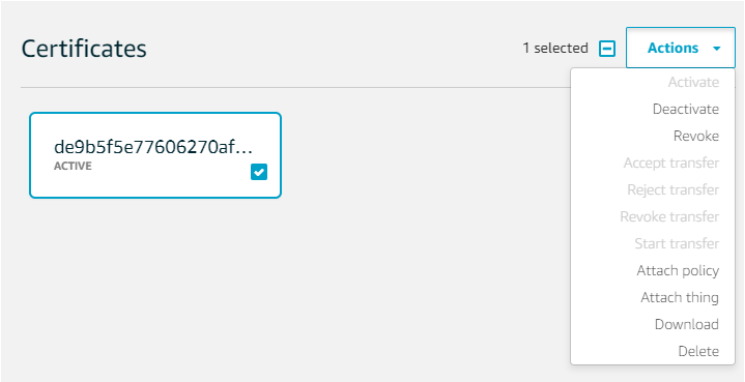
o) Afterwards, click on 'Create new policy' and enter the following details accordingly then click 'Create'.



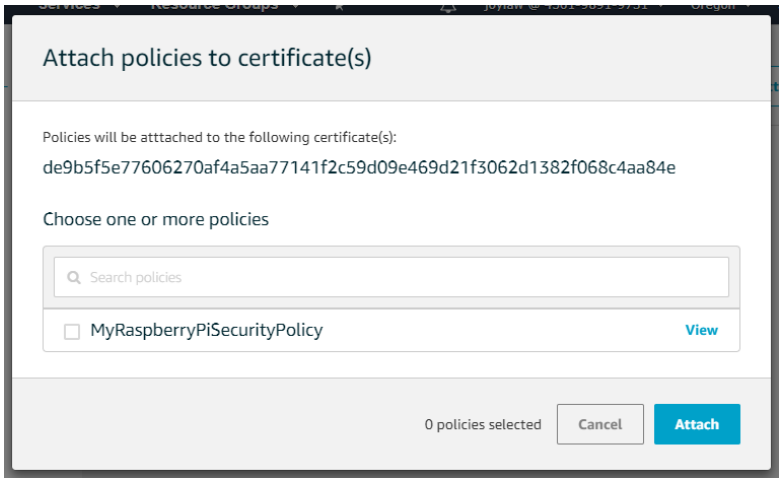

p) Once successful, you should see the following policy.



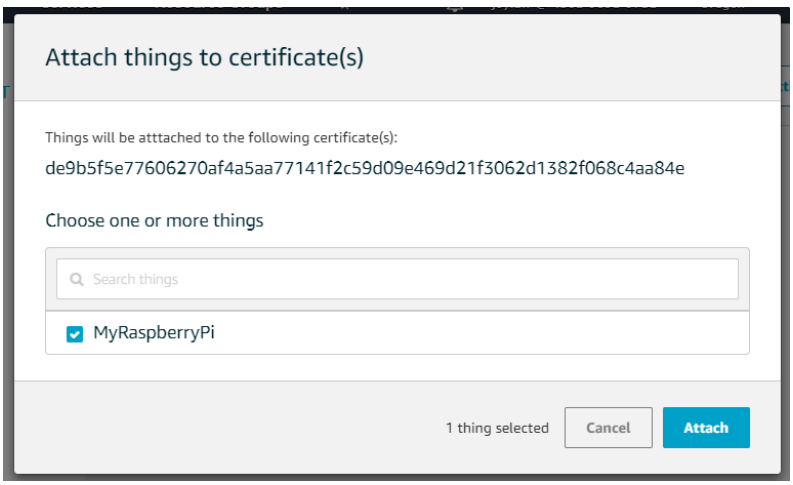
q) On the left-hand menu, select 'Certificate'. You should see the following X.509 certificate that you have craeted previously. Click on the checkbox beside the certificate and from the 'Action' drop down and select 'Attach policy'.




r) You will then be redirect the following screen and select 'MyRaspberryPiSecurityPolicy' then 'Attach'. Once successful, you will be notified.

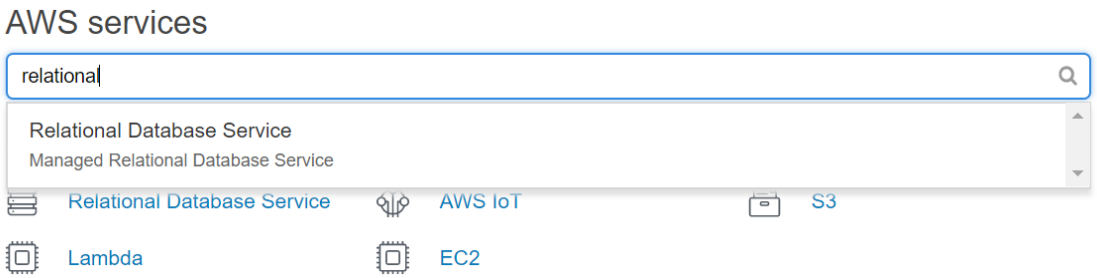



s) Click on the checkbox beside the certificate again and from the 'Action' drop down and select 'Attach thing', you will then see the following screen. Select 'MyRaspberryPi' and click 'Attach'

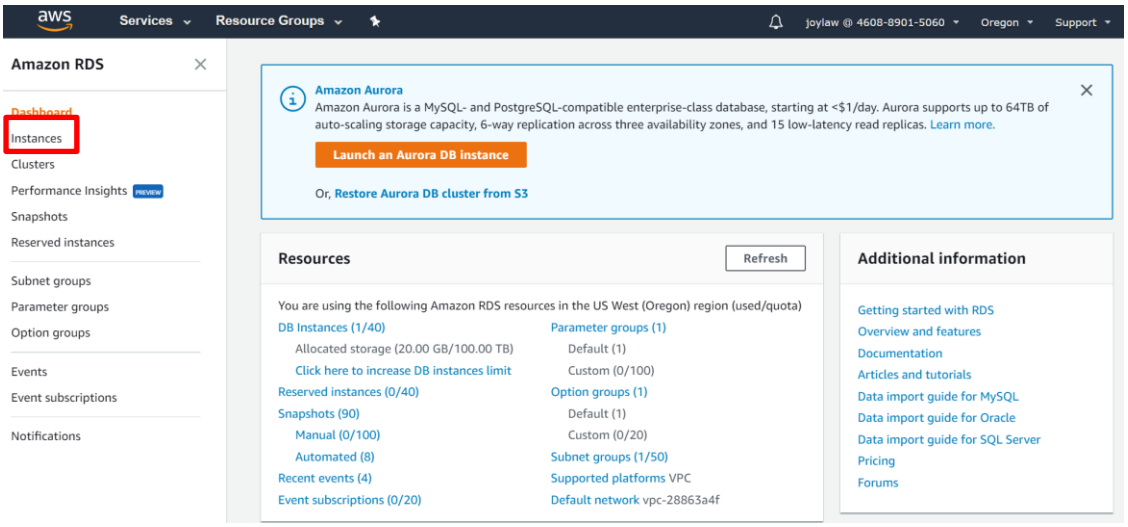


# B. Configure AWS Relational Database Service

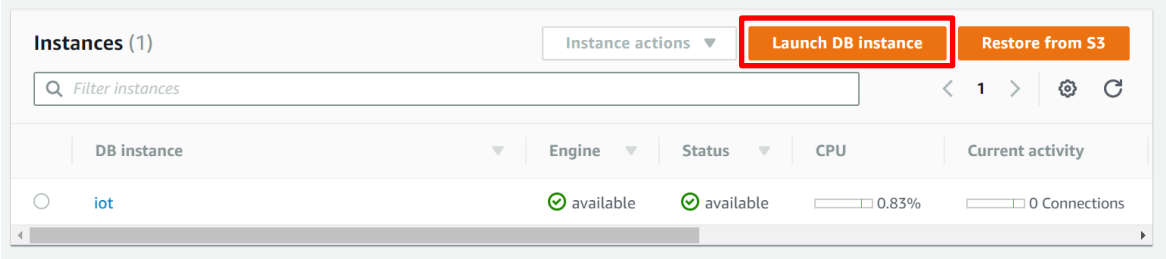
a) Continuing from the previous step, search for 'Relational Database Service' and click on it to begin configuring a MySQL database.



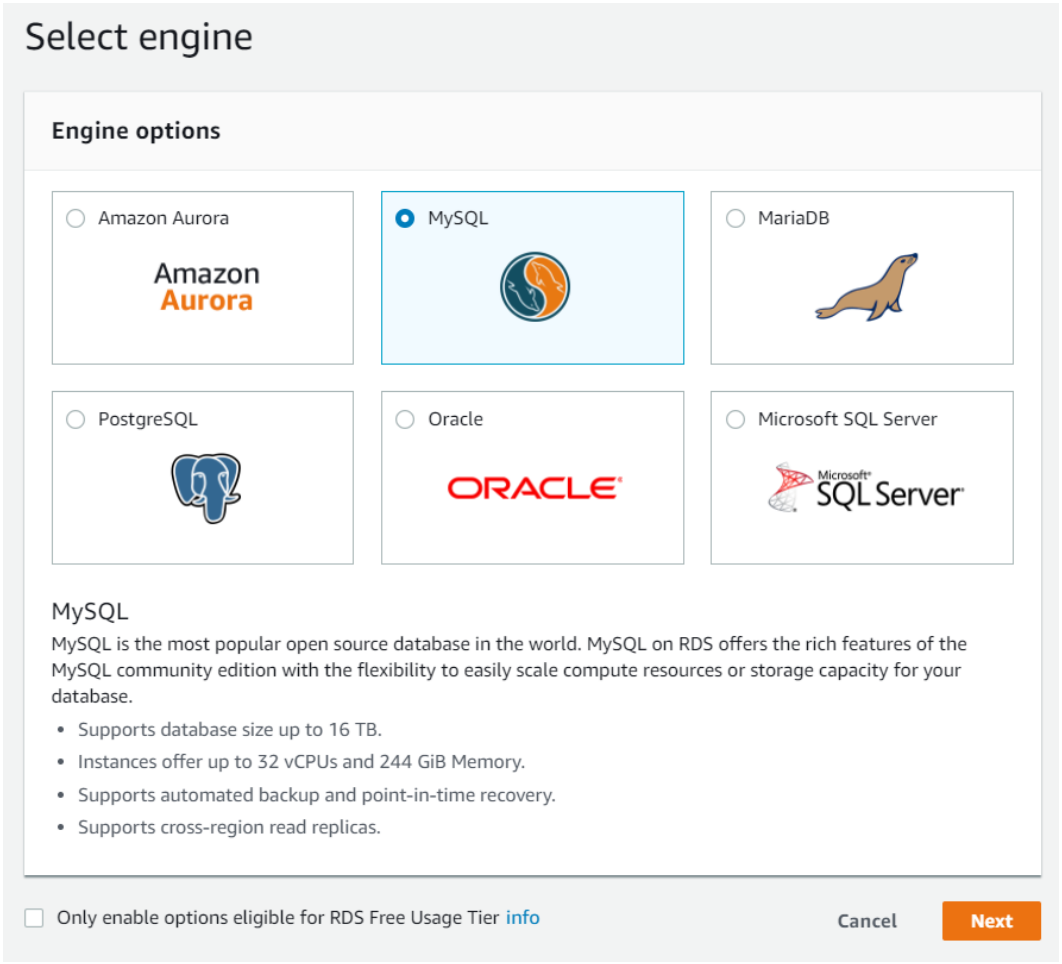

b) You will see the following page. Click on 'Instances' on the left-hand menu.



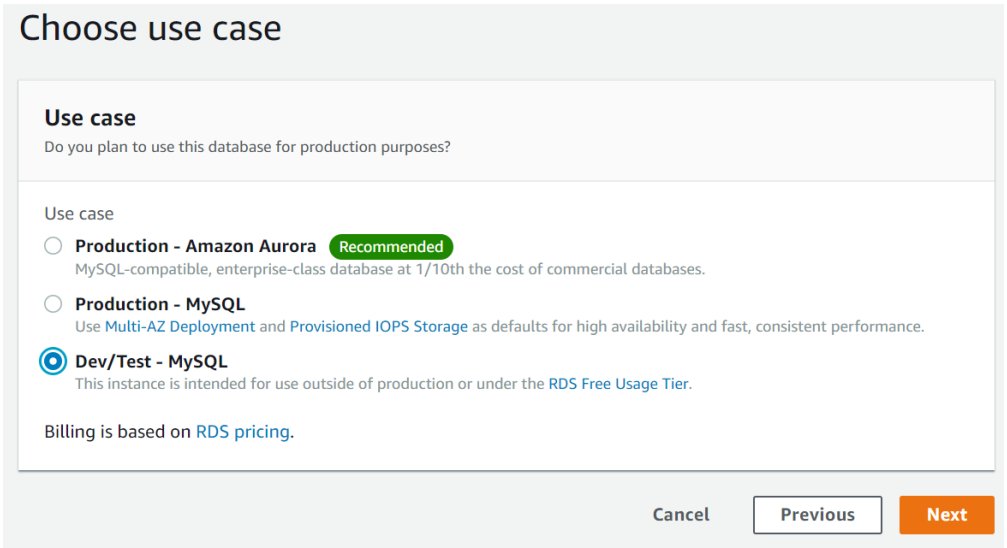

c) Click on 'Launch DB instance'. Note that you should not have an exising 'DB Instance' prior to this step.



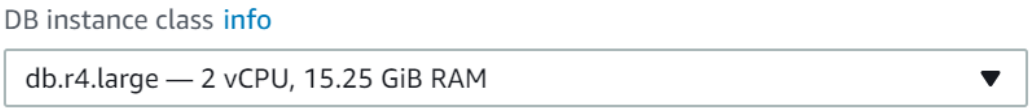
d) When selecting an engine, pick MySQL and click 'Next'.



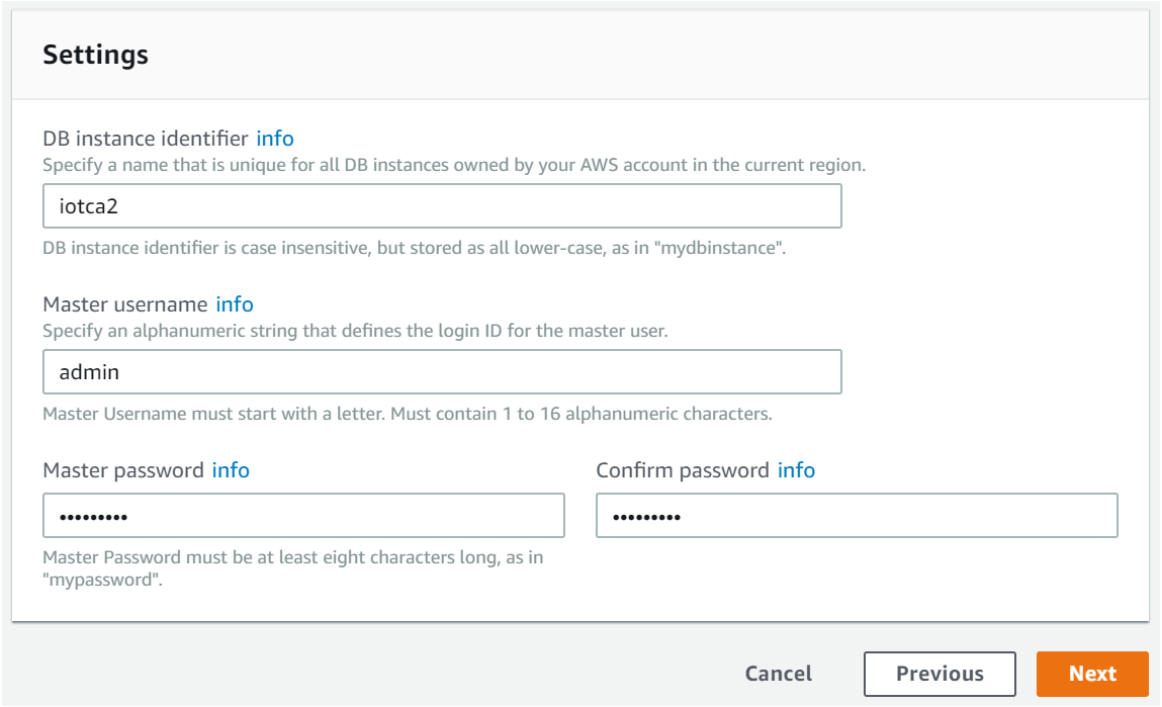

e) When prompted for use case, pick 'Dev/Test – MySQL' and click 'Next'.



f) Leave everything as default except for the following:

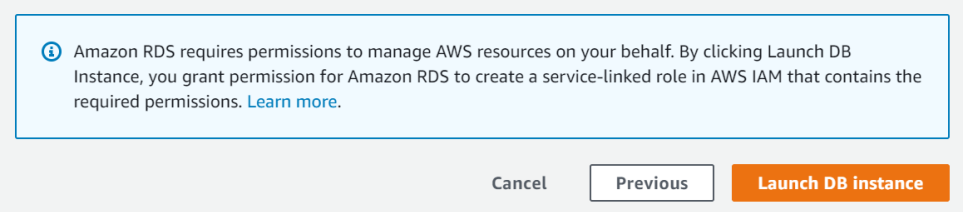



g) For the Settings portion, enter a name for 'DB instance identifier' and enter a master username and password. Take note that these information will be made use of later on. Click 'Next'.

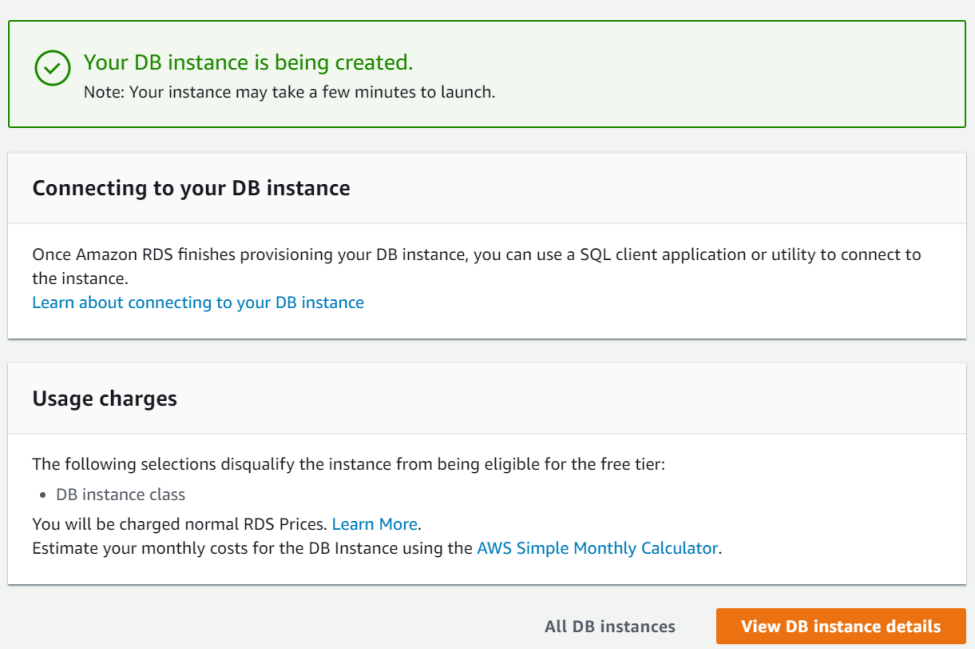



h) Leave everything as default except for the following:

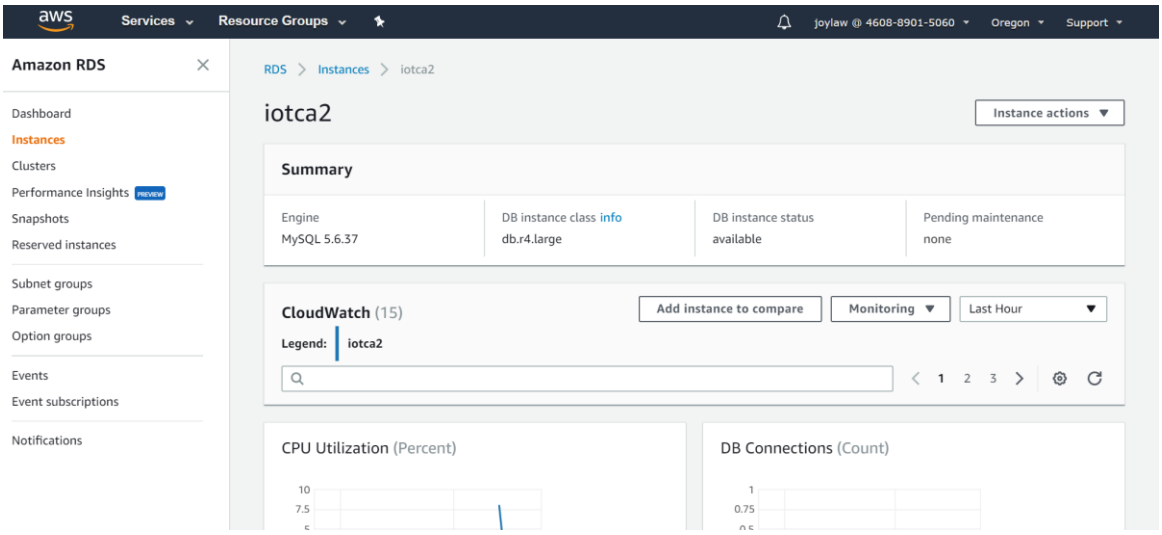
i) Click on 'Launch DB instance' after the configuration has been done.




j) If successful, you should see the following page. Click on 'View DB instance details'. It will take awhile for the database to complete its creation.




k) When it is completed, you should see the following page.

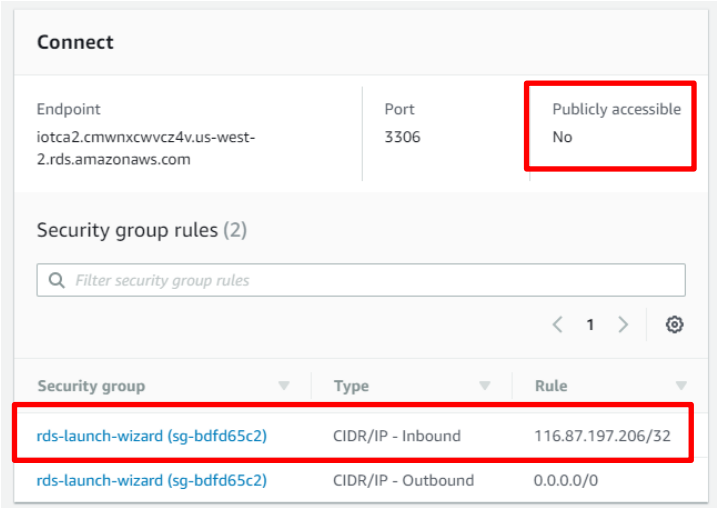


# C. Create a Database using AWS RDS

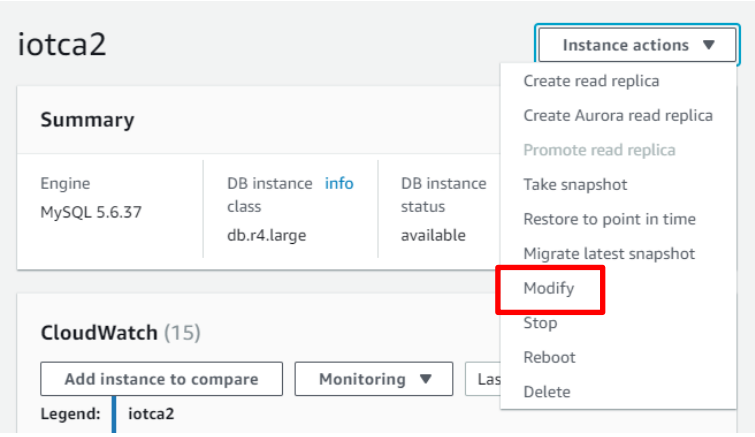
a) In your Raspberry Pi, enter the following

```
pi@joylaw:~ $ mysql -h iot.cyygylyw5wux.us-west-2.rds.amazonaws.com -u admin -p
```

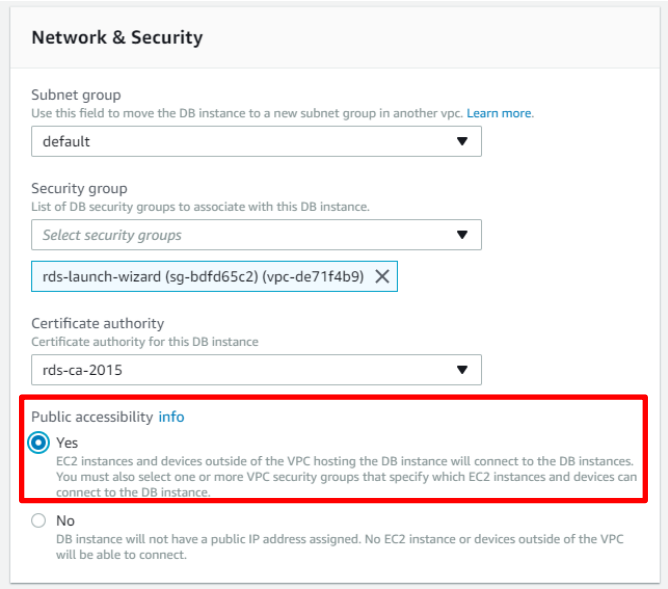
b) Replace 'iot.cyygylyw5wux.us-west-2.rds.amazonaws.com' with your own Endpoint which can be found here and 'admin' with the Master username that you have previously created. Scroll down to 'Connect' to ensure that the 'Publicly accessible' has been set to 'Yes' and the 'CIDR/IP – Inbound' rule has been set to '0.0.0.0/0'. If not, continue to the following steps. Else, continue to part i.



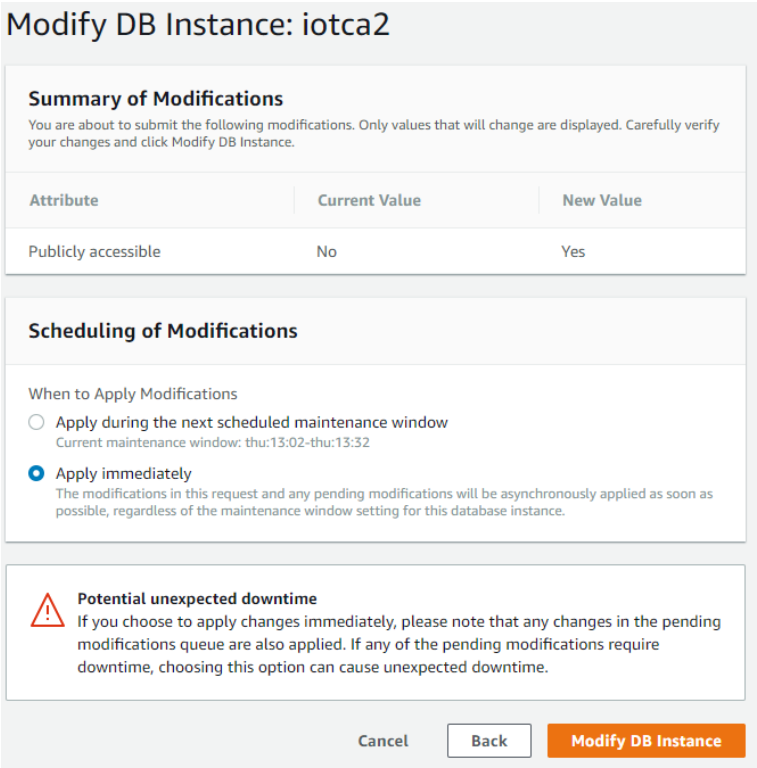

c) Scroll all the way up and click one the drop down button 'Instance actions'. Select 'Modify'.



d) Scroll down to 'Network & Security', under 'Public accessibility', pick 'Yes' and the scroll down to the bottom and click on 'Continue'.

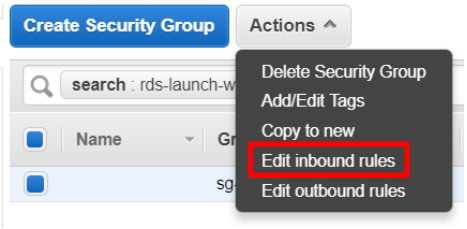



e) In this web page, select 'Apply immediately' then 'Modify DB Instance'.

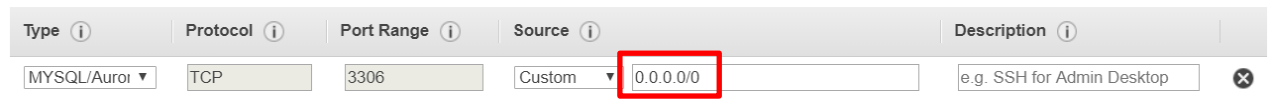


f) Under the 'Security group rules', click on th security group that corresponds to the type 'CIDR/IP – Inbound'


g) You will be redirected to another web page. From there, click on the 'Action' drop down button and select 'Edit inbound rules'.




h) Change the default to the following values and click 'Save'




i) You will then be prompted for a password, enter the Master password you have previously created. If successful, you should see the following:

```
pi@joylaw:~ $ mysql -h iotca2.cyygylyw5wux.us-west-2.rds.amazonaws.com -u admin
-p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 5.6.37-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

j) Run the command 'show databases;' to see all the databases that are available.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| iotca2             |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.22 sec)
```

k) Pick the database that you have previously created. In our case, iotca2. Run the command 'use iotca2;'

```
mysql> use iotca2;
Database changed
```

l) To create the tables needed for this project. Enter the following:

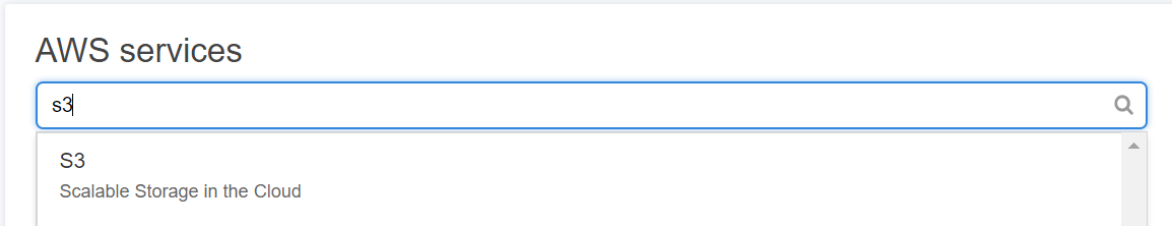
```
CREATE TABLE rfid_access (rfid_uid VARCHAR(50) NOT NULL, name
VARCHAR(255) NOT NULL, PRIMARY KEY(rfid_uid));

CREATE TABLE rfid_accesslog (id int(10) NOT NULL
AUTO_INCREMENT, date_time VARCHAR(50) NOT NULL, rfid_uid
VARCHAR(50) NOT NULL, PRIMARY KEY(id));

CREATE TABLE temp_humidity (id int(10) NOT NULL
AUTO_INCREMENT, date_time datetime NOT NULL, temperature
VARCHAR(10) NOT NULL, humidity VARCHAR(10) NOT NULL, PRIMARY
KEY(id));
```

m) Run 'show tables;' to see all the tables you have just created.

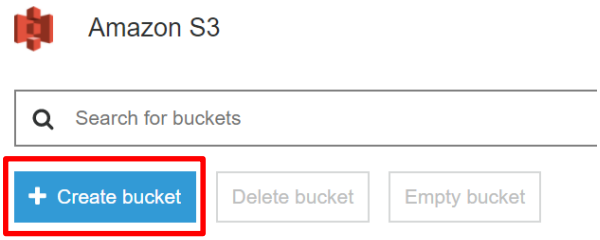
n) To quit the session, type 'quit'.

# D. Configure AWS S3

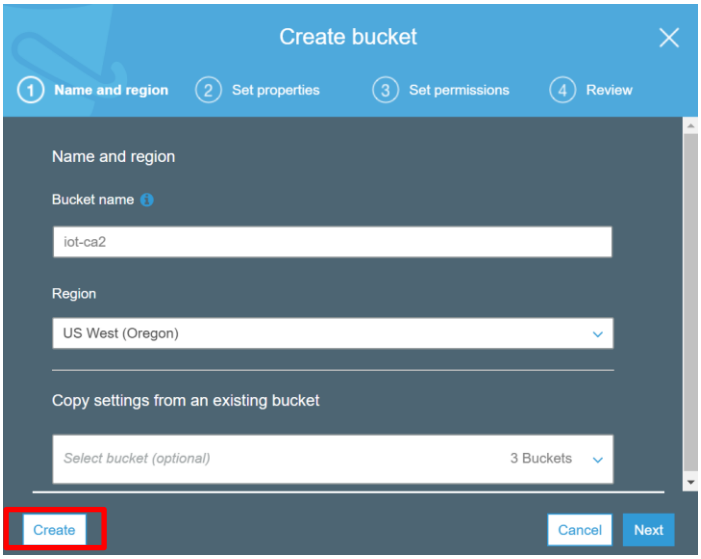
a) Go back to the home page of the console and search for the service: S3. We will be using S3 to store our images. Click on 'S3'.



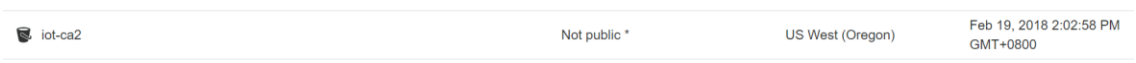

b) Click on the 'Create bucket' button.



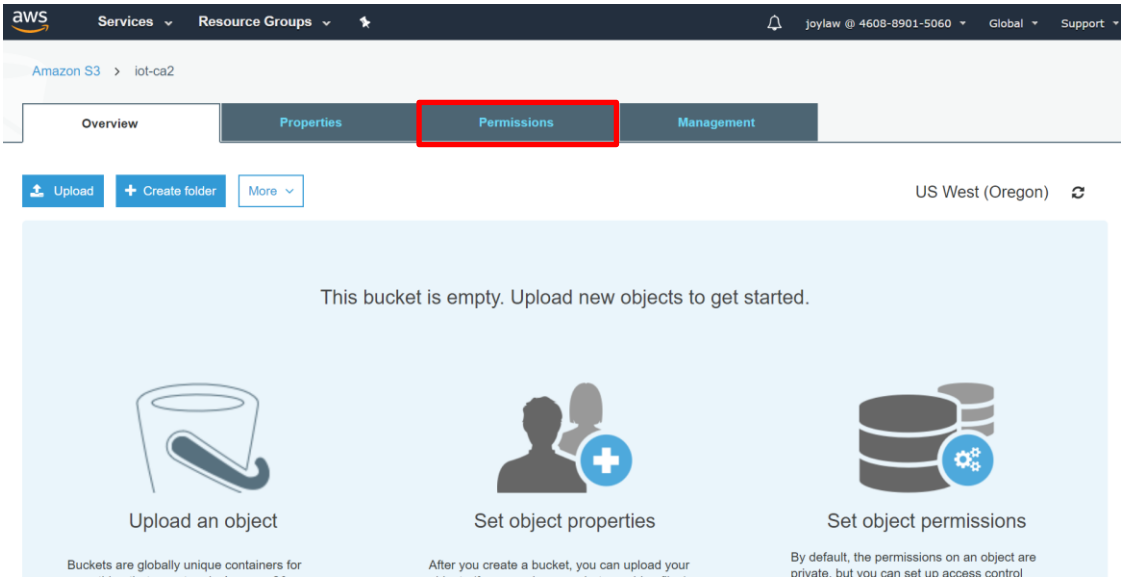

c) Enter a bucket name and leave the rest as default. Click 'Create'.




d) Once successful, you should see the following bucket created. Click on the bucket name.



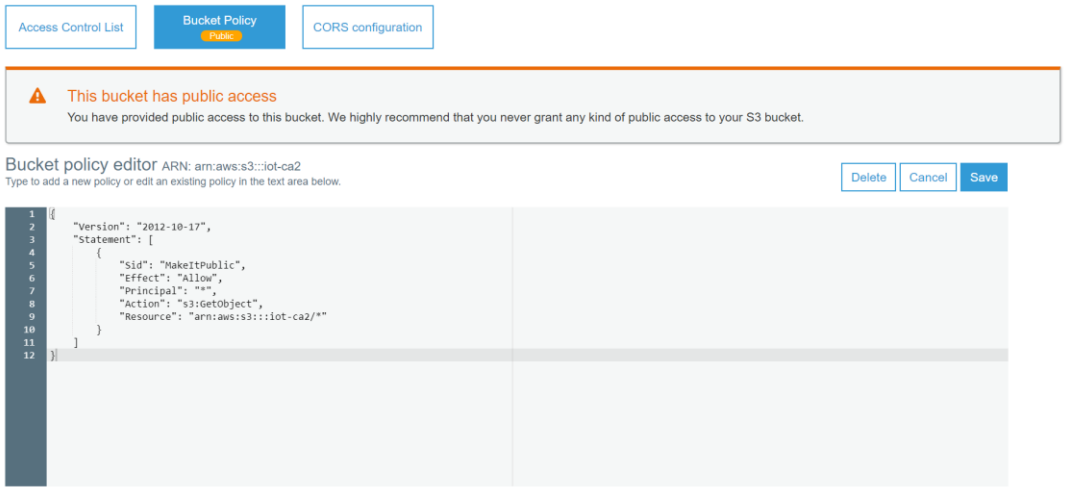
e) You will then be redirected to the following web page and click on 'Permissions'.




f) Click on 'Bucket policy' and enter the following and change "arn:aws:s3:::iot-ca2" to your own bucket ARN.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MakeItPublic",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::iot-ca2/*"
        }
    ]
}
```

g) If successful, your bucket should allow public access.



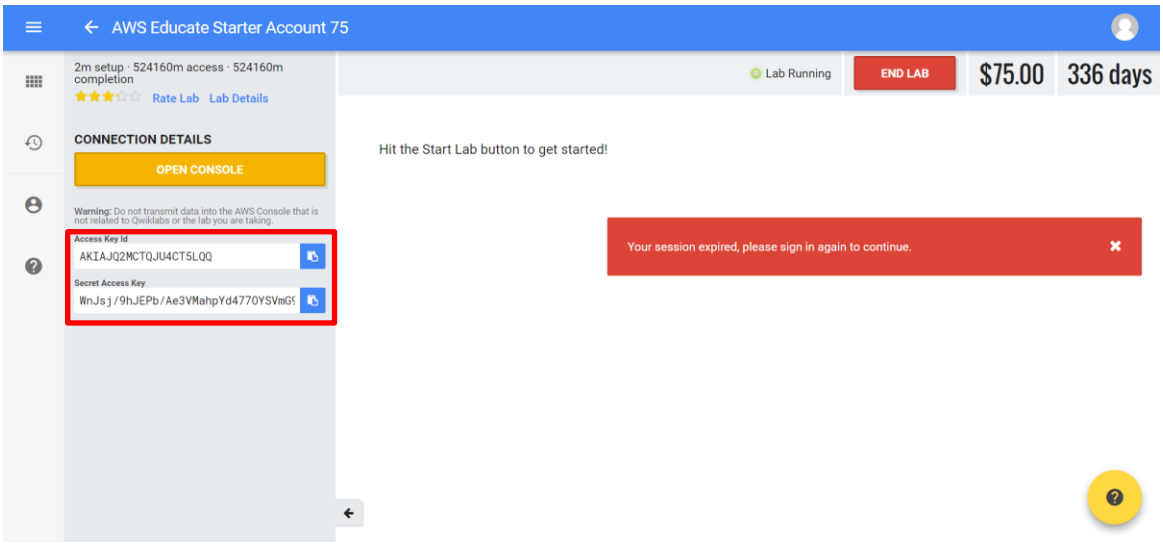
h) Go to your Raspberry Pi and run the following command to create a new folder

```
mkdir ~/assignment
```

i) Change directory to ~/assignment and type the command 'aws configure'.

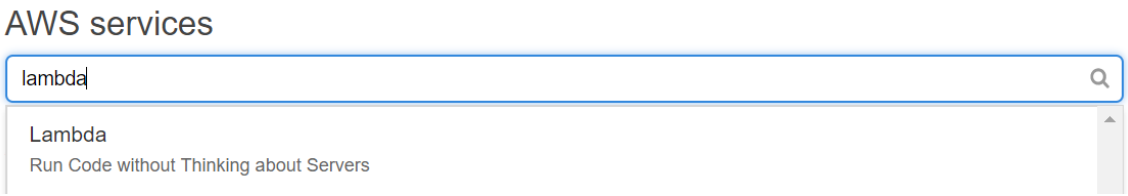
```
pi@joylaw:~ $ aws configure
AWS Access Key ID [****************5LQQ]: AKIAJQ2MCTQJU4CT5LQQ
AWS Secret Access Key [****************0M95]: WnJsj/9hJEPb/Ae3VMahpYd477OYSVmG9/
ayOM95
Default region name [None]: us-west-2
Default output format [None]:
```

j) Enter the following accordingly. Your AWS Access Key ID & AWS Secret Access Key can be retrieved from the following web page.

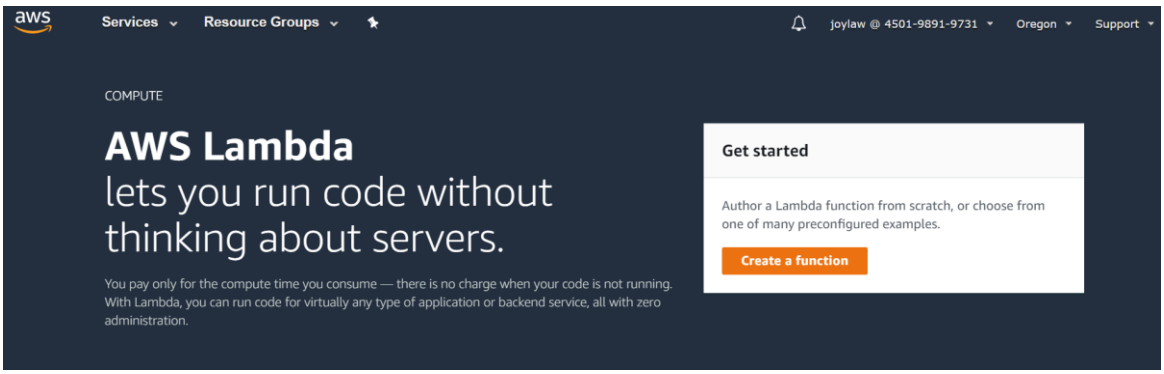


# E. Configure AWS Lambda & AWS IoT Rule

a) Go back to the home page of the console and search for the service: Lambda.

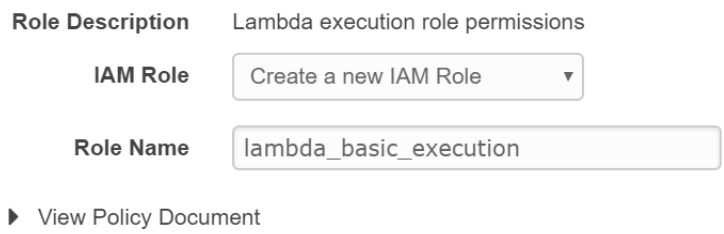



b) You will see the following web page. Click on 'Create a function'.

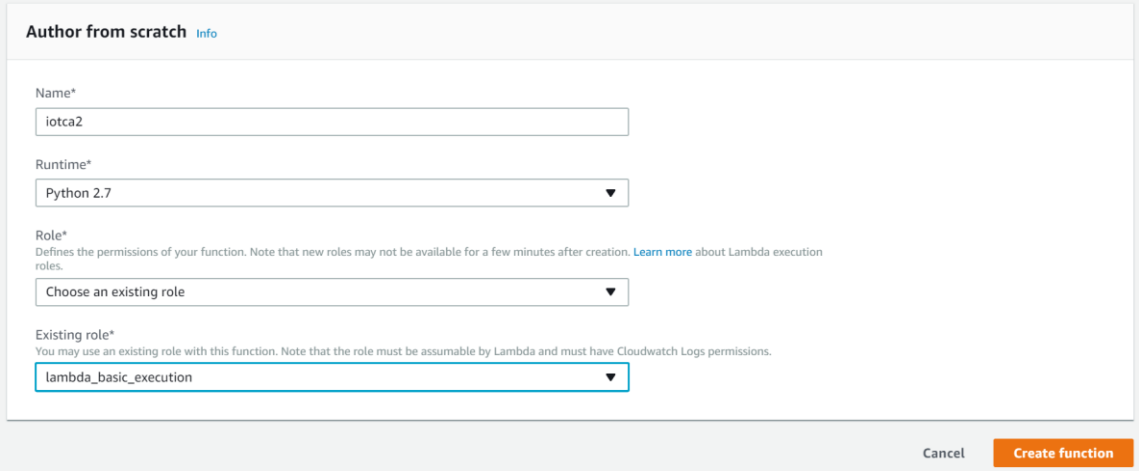



c) Scroll down to 'Role'and select 'Create a custom role'. You will then be redirected to another web page.

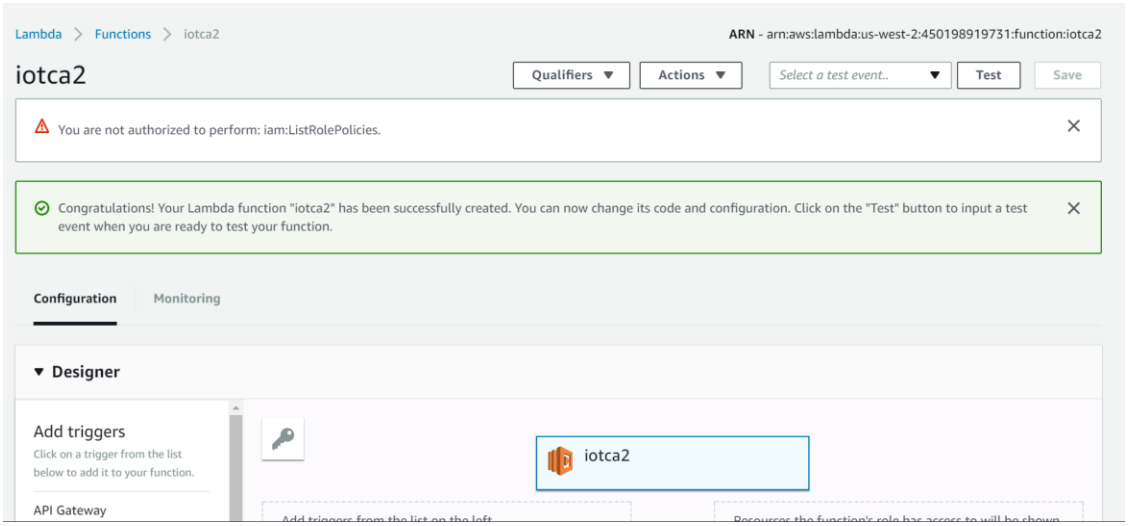

d) Accept the defaults and click 'Allow'



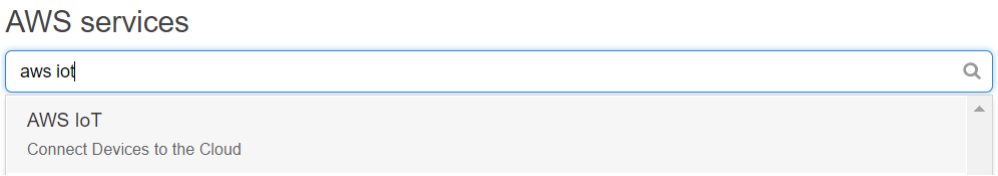
e) Go back to the previous tab and refresh. Select the following and click 'Create function'.



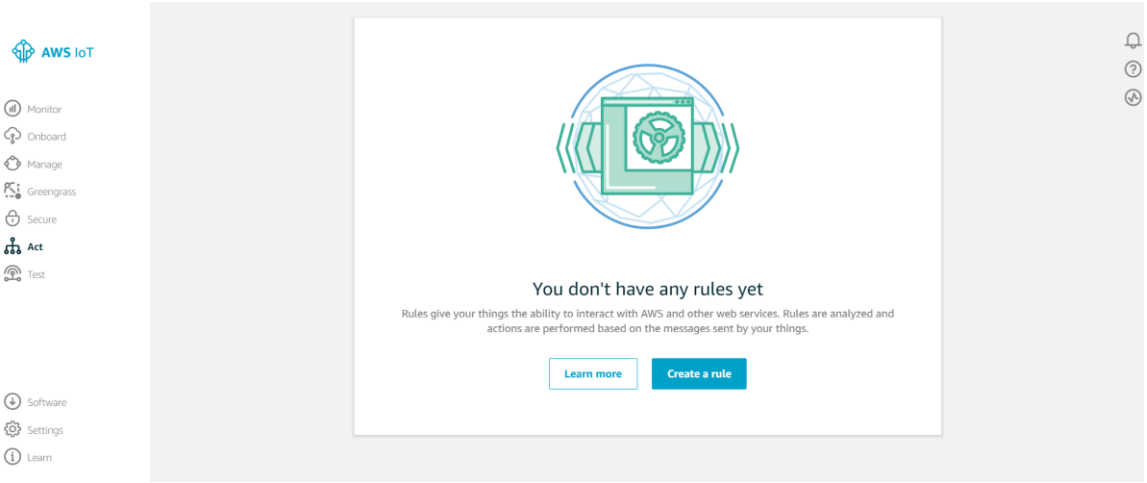

f) Once successfully created, you should see the following:



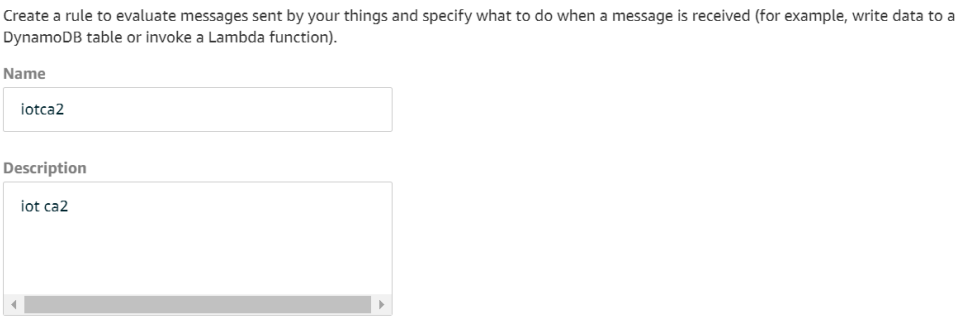

g) Go back to the home page of the console and search for the service: AWS IoT. Click on it.



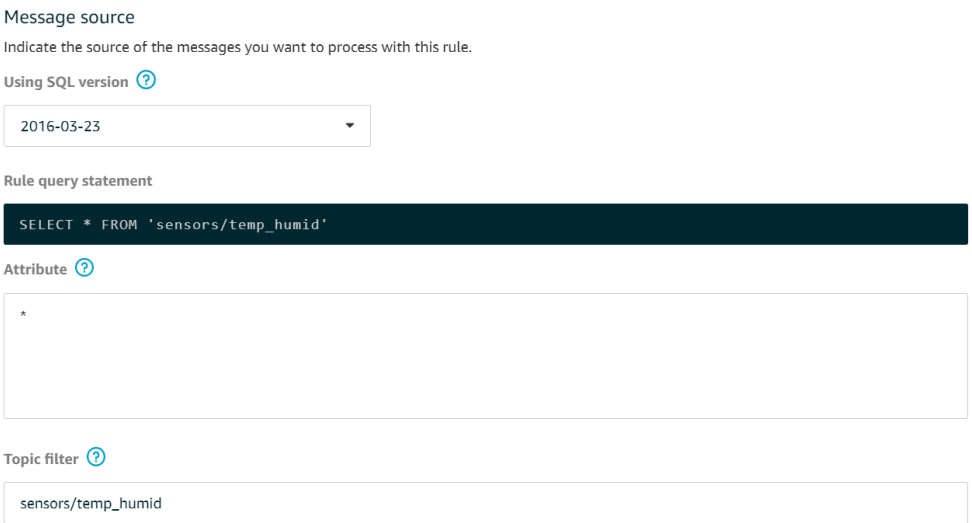
h) On the left-hand menu, click on 'Act' and you should be redirected to the following web page. Click on 'Create a rule'.



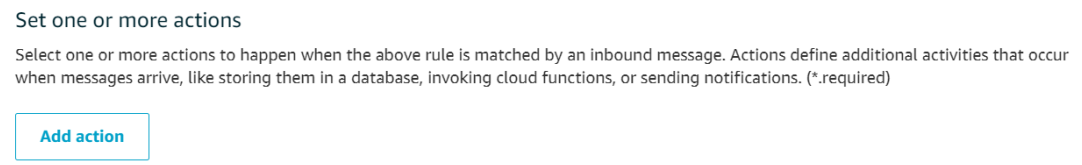

i) Enter the following accordingly and scroll down.



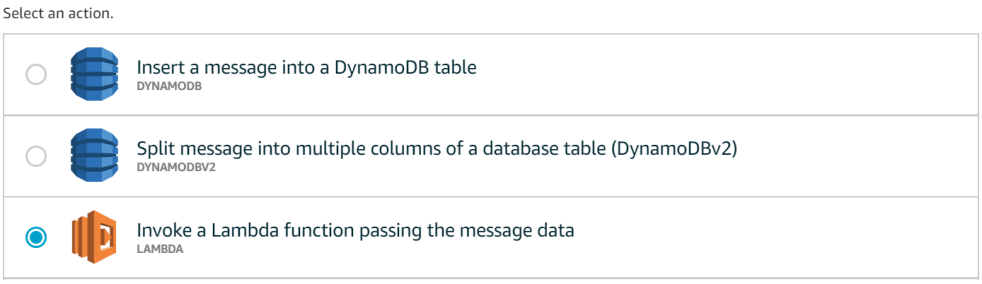

j) Under the 'Message source', enter the following:



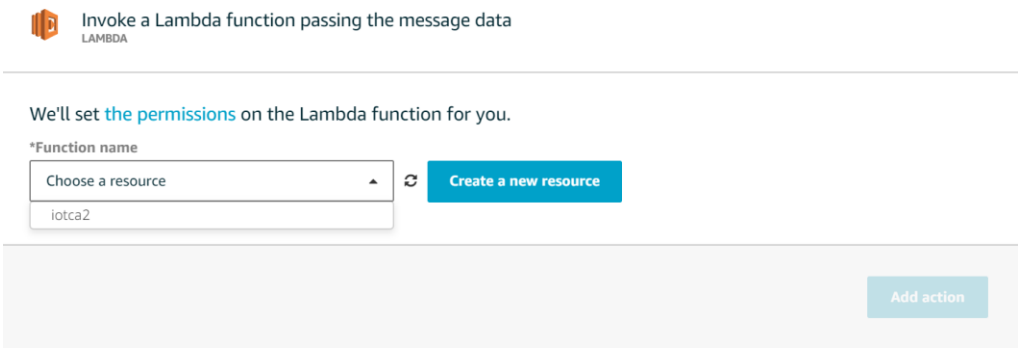
k) Scroll down and under ‘Set one or more actions’, click on ‘Add action’.



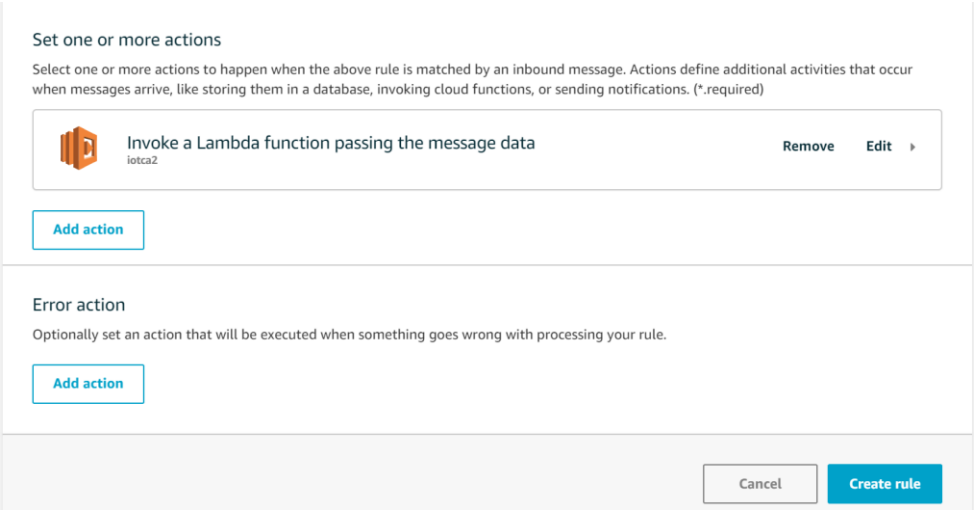

l) You will then be redirected to the following page. Select ‘Invoke a Lambda function passing the message data’ and click ‘Configure action’.



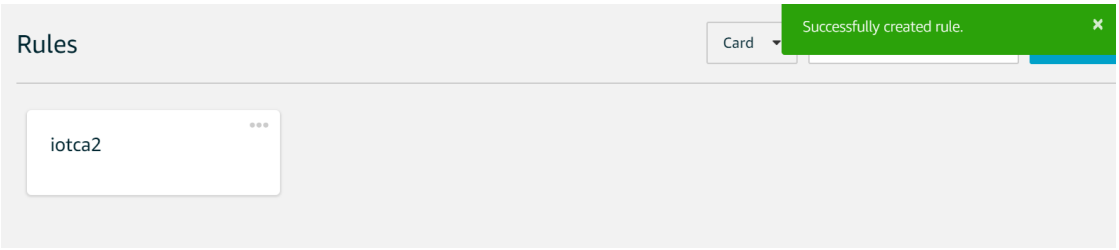

m) Click on the drop down bar for ‘Function name’. Select ‘iotca2’ and click on ‘Add action’



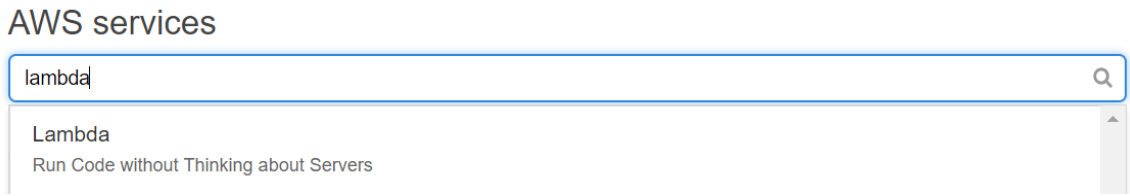

n) Scroll down and press ‘Create rule’.



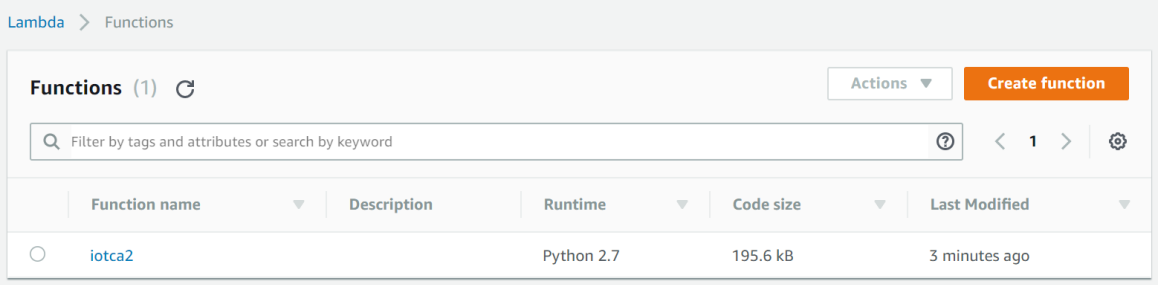
o) Once successfully created, you will see the following.



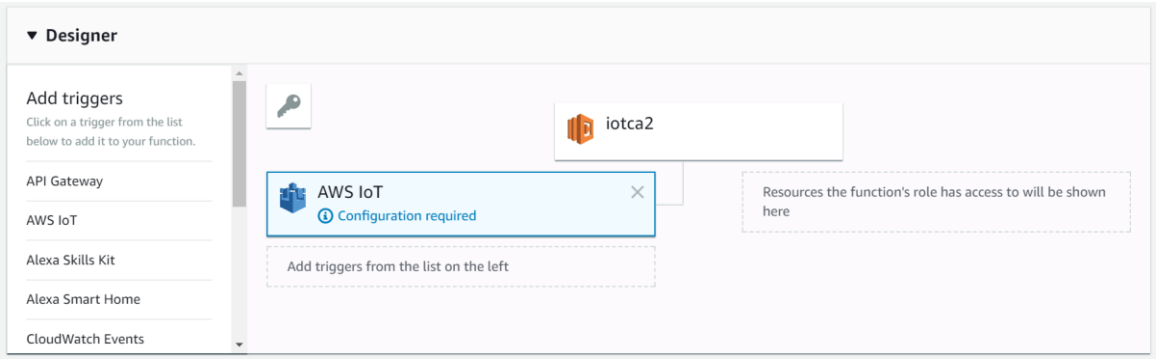

p) Go back to the home page of the console again and search for the service: Lambda.



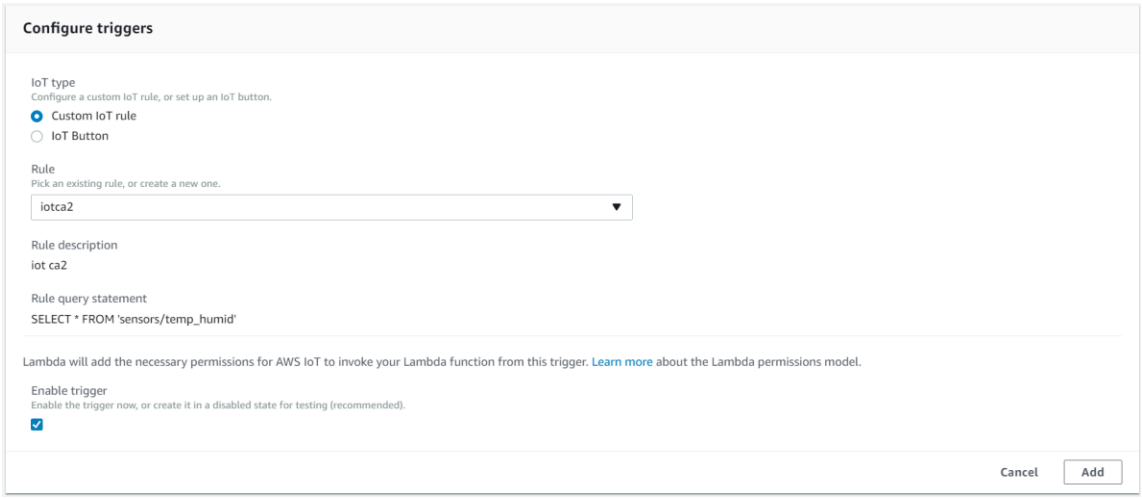

q) Under the function name, select 'iotca2'.



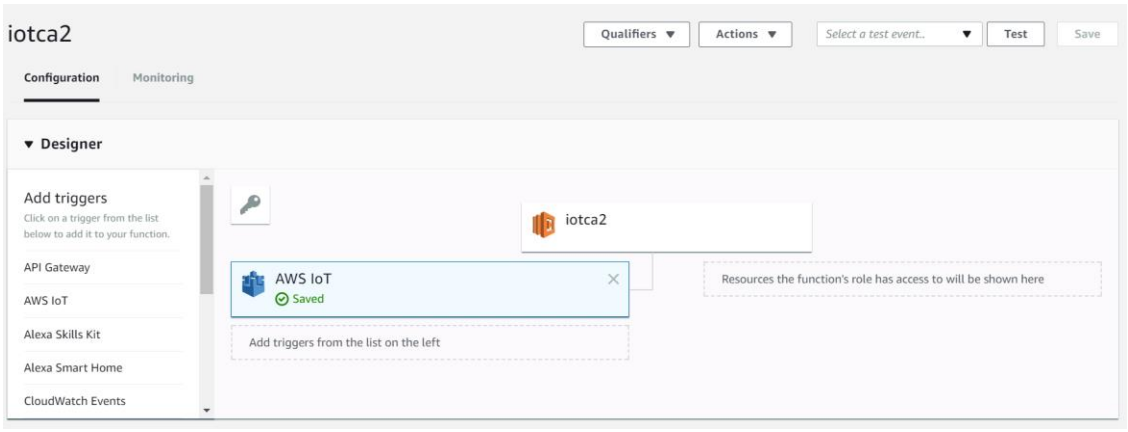

r) On the left-hand panel, under 'Add triggers', press 'AWS IoT' and you should achieve the following



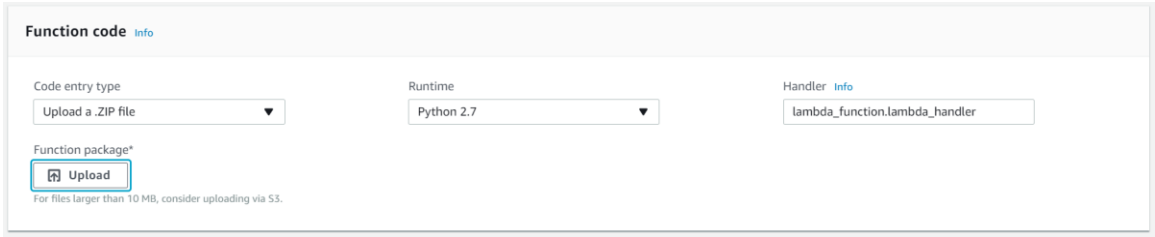
s) Scroll down to configure the trigger. Under 'Configure trigger', select 'Custom IoT rule' then select the rule 'iotca2' that you have previously created. Then click 'Add'.



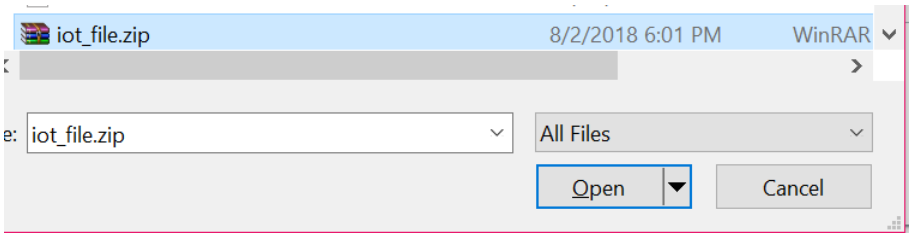

t) Afterwards, on the top right-hand corner, click on 'Save' to save the configuration. Once successful, you should see the following. Then, click on 'iotca2' and scroll down.



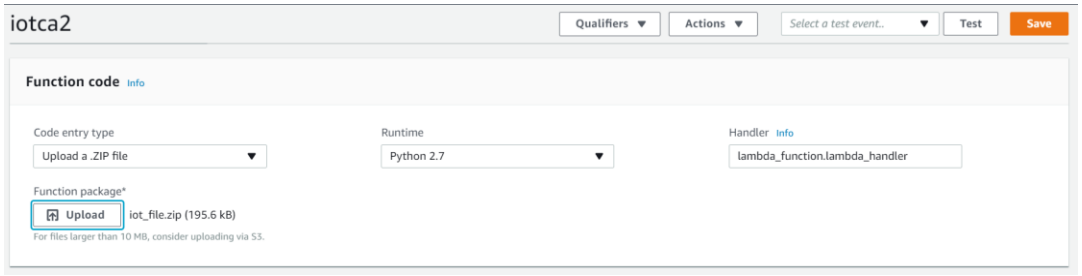

u) Under 'Code entry type', select 'Upload a .ZIP file' and click on 'Upload'.



v) 'iot_file.zip' can be downloaded under the 'resources' folder in GitHub. In your computer, search for the file 'iot_file.zip' and click 'Open'.




w) Once uploaded, on the top-right hand corner, click on 'Save'.



# Section 8
# Configure AWS IoT Rules

## A. Create an AWS Role

a) Create a file ~/assignment/iot-role-trust.json and add the following

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect": "Allow",
        "Principal": {
            "Service": "iot.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

b) Change directory to ~/assignment and run the following command:

```
aws iam create-role --role-name my-iot-role --assume-
role-policy-document file://iot-role-trust.json
```

# B. Create an AWS Policy

a) Create a file ~/assignment/iot-policy.json and add the following

```
{
    "Version": "2012-10-17",
    "Statement": [
        {

            "Effect": "Allow",
            "Action": "lambda:*",
            "Resource": "arn:aws:lambda:us-west-2:460889015060:function:*"
        },
        {

            "Effect": "Allow",
            "Action": "iot:*",
            "Resource": "*"
        }
    ]
}
```

b) Change directory to ~/assignment and run the following command:

```
aws iam put-role-policy --role-name my-iot-role --policy-
name iot-policy --policy-document file://iot-policy.json
```

# Section 9
# The Subscriber/Server

## A. Write the Code for server.py

a) Create a Python script server.py with the following code:

```
sudo nano ~/assignment/server.py
```

b) Enter the following code into server.py:

```
#Import all the libraries here
import gevent
import gevent.monkey
from gevent.pywsgi import WSGIServer
import datetime
import pymysql
import signal
import multiprocessing
from time import sleep
import os
import time
import json
from flask import jsonify
from gpiozero import LED
import sys
import threading
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from flask import Flask, request, Response, render_template
import RPi.GPIO as GPIO
import MFRC522
from flask import Flask,redirect, url_for
import picamera
import os

gevent.monkey.patch_all()

def getdatetime():
  return time.strftime('%Y-%m-%d %H:%M:%S')

led_status = None
def light_status(client, userdata, message):
  global led_status
  led_status = json.loads(message.payload)
  print led_status

def mqtt_connect():
  my_rpi = AWSIoTMQTTClient("subscriber_led")

  host = "a7gx67jy14o4h.iot.us-west-2.amazonaws.com"
  rootCAPath = "rootca.pem"
  certificatePath = "certificate.pem.crt"
  privateKeyPath = "private.pem.key"

  my_rpi.configureEndpoint(host, 8883)
  my_rpi.configureCredentials(rootCAPath, privateKeyPath,
certificatePath)
```

```
 my_rpi.configureOfflinePublishQueueing(-1)
 my_rpi.configureDrainingFrequency(2)
 my_rpi.configureConnectDisconnectTimeout(10)
 my_rpi.configureMQTTOperationTimeout(5)

 my_rpi.connect()

 #Subscribe
 my_rpi.subscribe("led/status", 1, light_status)

def dbconnect():
 try:
       db = pymysql.connect(host = "iotca2.cmwnxcwvcz4v.us-west-
2.rds.amazonaws.com", user="admin",passwd="!QWER4321", port=3306,
db="iotca2")
       print("Successfully connected to database")
       return db
 except:
       print("Error connecting to mySQL database")

def getRFID():
 #Create an object of the class MFRC522
 mfrc522 = MFRC522.MFRC522()

 #This loop keeps checking for chips.
 #If one is near it will get the UID
 try:
       while True:
             #Scan for cards
             (status,TagType)=
mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)

             #If a card is found
             if status == mfrc522.MI_OK:

                   #Get the UID of the card
                   (status,uid) = mfrc522.MFRC522_Anticoll()
                   print uid
                   return uid
 except:
       GPIO.cleanup()

def createNewUser():
 uid = getRFID()

def showRFID():
 db = dbconnect()
 curs = db.cursor()

 try:
       results = []
       curs.execute("SELECT id, date_time, name from rfid_accesslog L,
rfid_access A WHERE L.rfid_uid=A.rfid_uid order by L.id desc")
       data = curs.fetchall() #get all the results in database
       for row in data:
             results.append(row)
       curs.close()
       db.close()
       return results
 except MySQLdb.Error as e:
       print e
```

```
  except KeyboardInterrupt:
        curs.close()
        db.close()

app = Flask(__name__)

#Start: Declaring app route here [after app=Flask(__name__)]
@app.route("/")
def index():
  results = showRFID()
  msg = ""
  try:
        msg = json.loads(request.args['msg'])
        msg = msg[msg]
  except:
        pass
  templateData = {
        'results' : results,
        'msg' : msg
  }
  return render_template('index.html', **templateData)

@app.route("/readRFID")
def readRFID():
  uid = getRFID()
  return jsonify(uid=uid)

@app.route("/addUser", methods=['POST'])
def addUser():
  if request.method == 'POST':
        uid = request.form['uid']
                  name = request.form['name']
        uid = uid.replace(',', ', ')
        uid = '['+uid+']'
        try:
              db = dbconnect()
              curs = db.cursor()
              sql = "INSERT INTO rfid_access (rfid_uid, name) VALUES
(\'{0}\', \'{1}\')".format(uid, name)
              curs.execute(sql)
              msg = json.dumps({"msg":"User has been added."})
              db.commit()
        except pymysql.IntegrityError:
              msg = json.dumps({"msg":"Failed to add user. Please try
again."})
        curs.close()
        db.close()

  return redirect(url_for('.index', msg=msg))

@app.route("/newUser")
def newUser():
  return render_template('newUser.html')

# [LED(18), LED(21), LED(19), LED(13)] 0-3
@app.route("/readLED/<LED>")
def readLED(LED):
  return jsonify(response=led_status[LED])

@app.route("/writeLED/<LED>/<status>")
def writeLED(LED, status):
```

```
  led_data = {"led":LED, "status":status}

  my_rpi = AWSIoTMQTTClient("publisher_led")

  host = "a7gx67jy14o4h.iot.us-west-2.amazonaws.com"
  rootCAPath = "rootca.pem"
  certificatePath = "certificate.pem.crt"
  privateKeyPath = "private.pem.key"

  my_rpi.configureEndpoint(host, 8883)
  my_rpi.configureCredentials(rootCAPath, privateKeyPath,
certificatePath)

  my_rpi.configureOfflinePublishQueueing(-1)
  my_rpi.configureDrainingFrequency(2)
  my_rpi.configureConnectDisconnectTimeout(10)
  my_rpi.configureMQTTOperationTimeout(5)

  my_rpi.connect()

  my_rpi.publish("led/control", json.dumps(led_data), 1)

  return "Error removals R' US"

@app.route("/showImage/<access_id>")
def showImage(access_id):
  sql = "SELECT date_time, name from rfid_accesslog L, rfid_access A
WHERE L.rfid_uid=A.rfid_uid AND L.id = {}".format(access_id)

  db = dbconnect()
  curs = db.cursor()
  curs.execute(sql)

  result = curs.fetchone()

  curs.close()
  db.close()

  templateData = {
        'datetime' : result[0],
        'name': result[1]
  }
  return render_template('image.html', **templateData)

@app.route("/MQTTGraphValues")
def MQTTGraphValues():
  graphValues = [[],[],[]]
  sql = "SELECT date_time, temperature, humidity FROM temp_humidity
ORDER BY date_time DESC LIMIT 10"

  db = dbconnect()
  curs = db.cursor()
  curs.execute(sql)

  curs.close()
  db.close()

  for (datetime, temperature, humidity) in curs:
        #graphValues[0].append(datetime)
        graphValues[0].insert(0, datetime)
        #graphValues[1].append(temperature)
```

```
        graphValues[1].insert(0, temperature)
        #graphValues[2].append(humidity)
        graphValues[2].insert(0, humidity)

  return jsonify(graphdata=graphValues)

if __name__ == '__main__':
  mqtt_connect()
  try:
        print "Starting Server"
        http_server = WSGIServer(('0.0.0.0', 8010), app)
        app.debug = True
        http_server.serve_forever()
  except Exception as e:
        print(e)
  except KeyboardInterrupt:
        print "Terminate Server"
```

# B. Write the Code for index.html

a) Create a HTML file index.html with the following code:

```
sudo nano ~/assignment/templates/index.html
```

b) Enter the following code into index.html:

```
<link rel="stylesheet" href="{{ url_for('static',
filename='stylesheet.css') }}">
<!DOCTYPE html>

<head>
  <title>My Smart Home</title>

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>

  <!-- Tell the browser to be responsive to screen width -->
  <meta content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no" name="viewport">

  <!-- Bootstrap 3.3.7 -->
  <link rel="stylesheet"
href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css">

  <!-- Theme style -->
  <link rel="stylesheet" href="/static/dist/css/AdminLTE.min.css">

  <!-- AdminLTE Skins. Choose a skin from the css/skins folder instead
of downloading all of them to reduce the load. -->
  <link rel="stylesheet" href="/static/dist/css/skins/_all-
skins.min.css">

  <script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

  <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

  <!-- Google Font -->
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,6
00,700,300italic,400italic,600italic">

  <link rel="stylesheet"
href="/static/bower_components/DataTables/datatables.min.css">

  <script
src="/static/bower_components/DataTables/datatables.min.js"></script>

  <script type="text/javascript"
src="/static/dist/js/Chart.bundle.min.js"></script>

  <script type="text/javascript"
src="/static/dist/js/moment.js"></script>


  <script>
```

```
function turn_on_led(led, state){
  $.ajax({
    url: "/writeLED/"+led+"/"+state
  })
}

function rfid() {
  $.ajax({
    url: "writeRFID"
  })
}

function ledupdate(){
 $.ajax({
    url: "/readLED/0",
    success: function(result) {
      if (result.response) {
        $('input[id="red"]').prop('checked', true);
      }
    }
  })

  $.ajax({
    url: "/readLED/1",
    success: function(result) {
      if (result.response) {
        $('input[id="green"]').prop('checked', true);
      }
    }
  })

  $.ajax({
    url: "/readLED/2",
    success: function(result) {
      if (result.response) {
        $('input[id="yellow"]').prop('checked', true);
      }
    }
  })

  $.ajax({
    url: "/readLED/3",
    success: function(result) {
      if (result.response) {
        $('input[id="redtwo"]').prop('checked', true);
      }
    }
  })
}

document.addEventListener('DOMContentLoaded', function() {
  var checkbox = document.querySelector('input[id="red"]');

  checkbox.addEventListener('change', function() {
    if (checkbox.checked) {
      turn_on_led(0, "ON")
    } else {
      turn_on_led(0, "OFF")
    }
  });
```

```
    });

    document.addEventListener('DOMContentLoaded', function() {
      var checkbox = document.querySelector('input[id="green"]');

      checkbox.addEventListener('change', function() {
        if (checkbox.checked) {
          turn_on_led(1, "ON")
        } else {
          turn_on_led(1, "OFF")
        }
      });
    });

    document.addEventListener('DOMContentLoaded', function() {
      var checkbox = document.querySelector('input[id="yellow"]');

      checkbox.addEventListener('change', function() {
        if (checkbox.checked) {
          turn_on_led(2, "ON")
        } else {
          turn_on_led(2, "OFF")
        }
      });
    });

    document.addEventListener('DOMContentLoaded', function() {
      var checkbox = document.querySelector('input[id="redtwo"]');

      checkbox.addEventListener('change', function() {
        if (checkbox.checked) {
          turn_on_led(3, "ON")
        } else {
          turn_on_led(3, "OFF")
        }
      });
    });

    $(document).ready(function() {

   $('#example1').DataTable({
               "pageLength": 7,
               "dom" : "tp"
         });

      var tempchart = document.getElementById('temp-
chart').getContext('2d');
      var humiditychart = document.getElementById('humidity-
chart').getContext('2d');

      var config = {
        type: 'line',
        data: {
          labels: [],
          datasets: [{
            label: 'Temperature',
            data: [],
            backgroundColor: "rgba(255,0,0,0.4)",
            borderColor: "rgba(255,0,0,0.4)",
            fill: false
          }]
```

```
        },
        options: {
          responsive: true,
          title: {
            display: true,
            text: 'Temperature graph'
          },
      scales: {
        yAxes : [{
                    ticks : {
                        max : 35.0,
                        min : 15.0
                    }
                }]
      }
        }
      }

      var config2 = {
        type: 'line',
        data: {
          labels: [],
          datasets: [{
            label: 'Humidity',
            data: [],
            backgroundColor: "rgba(0,255,0,0.4)",
            borderColor: "rgba(0,255,0,0.4)",
            fill: false
          }]
        },
        options: {
          responsive: true,
          title: {
            display: true,
            text: 'Humidity Graph'
          },
      scales: {
        yAxes : [{
                    ticks : {
                        max : 80.0,
                        min : 20.0
                    }
                }]
      }
        }
      }

      var tempG = new Chart(tempchart, config);
      var humidityG = new Chart(humiditychart, config2);


      function dataupdate() {

        var axisLabel = [];

        $.ajax({
          url: "MQTTGraphValues",
          success: function(result) {
            var datalist = result.graphdata;

            for (var i = 0; i < datalist[0].length; i++) {
```

```
              axisLabel.push(moment(datalist[0][i]).format('h:mm:ss
a'));
            }

            tempG.data.datasets.forEach((dataset) => {
              dataset.data = datalist[1];
            });

            humidityG.data.datasets.forEach((dataset) => {
              dataset.data = datalist[2];
            });

            tempG.data.labels = axisLabel;
            humidityG.data.labels = axisLabel;

            $("#temp").text(datalist[1][datalist[1].length - 1]);
            $("#hum").text(datalist[2][datalist[2].length - 1]);
          }
        })

        tempG.update()
        humidityG.update()
      }

      // Reduce the values to increase update speed
      setInterval(dataupdate, 5000);
      //setInterval(ledupdate, 2500);
      ledupdate()
    });
  </script>
<script>
$(document).ready(function() {
{% autoescape false %}
   {% if msg %}
    alert( "{{ msg.msg }}" );
   {% endif %}
{% endautoescape %}
})
</script>
</head>

<body class="hold-transition skin-blue layout-top-nav">
  <div class="wrapper">
    <header class="main-header">
      <nav class="navbar navbar-static-top">
        <a href="/" style="float: left;">
         <img src="/static/home.jpg" alt="Home" id="home" width="50px"
height="50px">
         <h1 id="smarthome">SMART HOME</h1>
   </a>
   <a href="newUser"><button style="float:right; color: black; margin-
right: 9%; margin-top: 2%; border: none; border-radius: 2px; padding:
3px; padding-left: 8px; padding-right: 8px; background-color:
#E8E8E8;">Add New User</button></a>
      </nav>
    </header>
    <!-- Full Width Column -->
    <div class="content-wrapper">
      <div class="container">
        <!-- Main content -->
        <section class="content">
```

```
          <div class="row">
            <div class="col-lg-6">
              <div class="box box-primary">
                <div class="box-header with-border">
                  <h3 class="box-title">Home Access History</h3>
                </div>

                <div class="box-body">
                  <table id="example1"  class="table table-bordered
table-striped dataTable" data-page-length='6'>
             <thead>
                    <tr>
                      <th style="width:40%">Date (YYYY:MM:DD)</th>
              <th style="width:40%">Time (HH:mm:SS)</th>
                      <th style="width:20%">User</th>
                    </tr>
             </thead>
             <tbody>
                    {% for row in results %}
                    <tr>
                      <td>{{ row[1][0:10] }}</td>
                      <td>{{ row[1][11:13] }}:{{ row[1][14:16] }}:{{
row[1][17:19] }}</td>
                      <td><a href="showImage/{{row[0]}}">{{ row[2]
}}</a></td>
                    </tr>
                    {% endfor %}
             </tbody>
                  </table>
                </div>
         <br />
              </div>

            </div>
            <div class="col-lg-2">
              <div class="box" id="control">
                <div class="box-header with-border">
                  <h3 class="box-title">Bed Room</h3>
                </div>
                <br />
                <img src="/static/redled.jpg" alt="Red LED" id="LED"
width="80px" height="80px">
                <label class="switch">
                                                    <input
type="checkbox" id="red">
                                                    <span class="slider
round"></span>
                                                </label>

              </div>
              <div class="box" id="control">
                <div class="box-header with-border">
                  <h3 class="box-title">Living Room</h3>
                </div>
                <br />
                <img src="/static/greenled.jpg" alt="Green LED"
id="LED" width="80px" height="80px">
                <label class="switch">
                                                    <input
type="checkbox" id="green">
```

```
                                                    <span class="slider
round"></span>
                                                </label>

              </div>
            </div>
            <div class="col-lg-2">
              <div class="box" id="control">
                <div class="box-header with-border">
                  <h3 class="box-title">Toilet</h3>
                </div>
                <br />
                <img src="/static/yellowled.jpg" alt="Yellow LED"
id="LED" width="80px" height="80px">
                <label class="switch">
                                                    <input
type="checkbox" id="yellow">
                                                    <span class="slider
round"></span>
                                                </label>

              </div>
              <div class="box" id="control">
                <div class="box-header with-border">
                  <h3 class="box-title">Kitchen</h3>
                </div>
                <br />
                <img src="/static/redled.jpg" alt="Red LED" id="LED"
width="80px" height="80px">
                <label class="switch">
                                                    <input
type="checkbox" id="redtwo">
                                                    <span class="slider
round"></span>
                                                </label>

              </div>
            </div>
            <div class="col-lg-2">
              <div class="box" id="control">
                <div class="box-header with-border">
                  <h3 class="box-title">Temperature (°C)</h3>
                </div>
                <br />
                <h1 id="temp"></h1>
              </div>
              <div class="box" id="control">
                <div class="box-header with-border">
                  <h3 class="box-title">Humidity (%)</h3>
                </div>
                <br />
                <h1 id="hum"></h1>
              </div>
            </div>
          </div>
          <!-- Graphs -->
          <div class="row">
            <div class="col-lg-6">
              <div class="box">
                <canvas id="temp-chart"></canvas>
              </div>
```

```
            </div>
            <div class="col-lg-6">
              <div class="box">
                <canvas id="humidity-chart"></canvas>
              </div>
            </div>
          </div>
        </section>
        <!-- Bootstrap 3.3.7 -->
        <script
src="/static/bower_components/bootstrap/dist/js/bootstrap.min.js"></scr
ipt>
        <!-- SlimScroll -->
        <script src="/static/bower_components/jquery-
slimscroll/jquery.slimscroll.min.js"></script>
        <!-- FastClick -->
        <script
src="/static/bower_components/fastclick/lib/fastclick.js"></script>
        <!-- AdminLTE App -->
        <script src="/static/dist/js/adminlte.min.js"></script>
</body>

</html>
```

# C. Write the Code for image.html

a) Create a HTML file image.html with the following code:

```
sudo nano ~/assignment/templates/image.html
```

b) Enter the following code into image.html:

```
<link rel="stylesheet" href="{{ url_for('static',
filename='stylesheet.css') }}">
<!DOCTYPE html>

<head>
<title>My Smart Home</title>

<!-- Tell the browser to be responsive to screen width -->
<meta content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no" name="viewport">

<!-- Bootstrap 3.3.7 -->
<link rel="stylesheet"
href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css">

<!-- Theme style -->
<link rel="stylesheet" href="/static/dist/css/AdminLTE.min.css">

<!-- AdminLTE Skins. Choose a skin from the css/skins folder instead of
downloading all of them to reduce the load. -->
<link rel="stylesheet" href="/static/dist/css/skins/_all-
skins.min.css">

<script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

<script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<!-- Google Font -->
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,6
00,700,300italic,400italic,600italic">

<link type="text/css" rel="stylesheet" href="css/main.css">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>

<script type="text/javascript" src="https://code.jquery.com/jquery-
3.2.1.js"></script>

</head>

<body class="hold-transition skin-blue layout-top-nav">
   <div class="wrapper">
         <header class="main-header">
               <nav class="navbar navbar-static-top">
                     <a href="/">
                           <img src="/static/home.jpg" alt="Home"
id="home" width="50px" height="50px">
```

```
                        <h1 id="smarthome">SMART HOME</h1>
                    </a>
                </nav>
            </header>
            <!-- Full Width Column -->
            <div class="content-wrapper">
                    <div class="container">
                        <!-- Main content -->
                    <section class="content">
                                <div class="col-lg-12">
                                            <div class="box box-
primary">
                                                    <div class="box-
header with-border" id="imageTitle">
                                                            <h3 class="box-
title">User: <strong>{{name}}</strong></h3>
                                                    <br />
                                                    <h3 class="box-
title">Photo Taken on <strong>{{datetime[0:10]}}
{{datetime[11:13]}}:{{datetime[14:16]}}:{{datetime[17:19]}}</strong></h
3>
                                                    </div>
                                            <br />
                                                    <img src="https://s3-
us-west-2.amazonaws.com/accesssmarthome/{{datetime}}.jpg" alt="image"
id="image" width="720px" height="405px">
                                                    <br />
                                                    <a href="/"><h4
id="backHome">[ Back to Home Page ]</h4></a>
                                            <br />
                                            </div>
                                </div>
                    </section>
                </div>
            </div>
</body>
</html>
```

# D. Write the Code for newUser.html

a) Create a HTML file newUser.html with the following code:

```
sudo nano ~/assignment/templates/newUser.html
```

b) Enter the following code into newUser.html:

```
<link rel="stylesheet" href="{{ url_for('static',
filename='stylesheet.css') }}">
<!DOCTYPE html>

<head>
<title>My Smart Home</title>

<!-- Tell the browser to be responsive to screen width -->
<meta content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no" name="viewport">

<!-- Bootstrap 3.3.7 -->
<link rel="stylesheet"
href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css">

<!-- Theme style -->
<link rel="stylesheet" href="/static/dist/css/AdminLTE.min.css">

<!-- AdminLTE Skins. Choose a skin from the css/skins folder instead of
downloading all of them to reduce the load. -->
<link rel="stylesheet" href="/static/dist/css/skins/_all-
skins.min.css">

<script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

<script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<!-- Google Font -->
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,6
00,700,300italic,400italic,600italic">

<link type="text/css" rel="stylesheet" href="css/main.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>
<script
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js
"></script>
<style>
.center {
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 50%;
}

.close {
    color: #aaa;
    float: right;
```

```
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}
.modal-content {
    background-color: #fefefe;
    margin: 15% auto; /* 15% from the top and centered */
    padding: 20px;
    border: 1px solid #888;
    width: 20%; /* Could be more or less, depending on screen size */
    height: 30%;
    text-align: center;
}
</style>
<script>
$(document).ready(function() {
   // Modal
   var span = document.getElementsByClassName("close")[0];
   var modal = document.getElementById('myModal');
   span.onclick = function() {
       $('#myModal').modal('hide');
   }
   window.onclick = function(event) {
       if (event.target == modal) {
           $('#myModal').modal('hide');
       }
   }
   $.get( "readRFID", function( data ) {
     $('#myModal').modal('show');
    alert( "The UID detected is: " + data.uid );
     document.getElementById("uid").value = data.uid;
   });
})
</script>
</head>

<body class="hold-transition skin-blue layout-top-nav">
<div class="wrapper">
        <header class="main-header">
              <nav class="navbar navbar-static-top">
                    <a href="/">
                          <img src="/static/home.jpg" alt="Home"
id="home" width="50px" height="50px">
                          <h1 id="smarthome">SMART HOME</h1>
                    </a>
              </nav>
        </header>
        <!-- Full Width Column -->
        <div class="content-wrapper">
                    <div class="container">
                          <!-- Main content -->
                    <section class="content">
                                <div class="col-lg-12">
                                      <div class="box box-
primary">
```

```
                                                        <div class="box-
header with-border" id="imageTitle">
                                                              <h3 class="box-
title">Scan your Card</strong></h3>
                                                        </div>
                                                  <br />
                                                  <img src="/static/card.jpg"
style="width: 40%; height: auto;" class="center">
                                                        <br />
                                                        <a href="/"><h4
id="backHome">[ Back to Home Page ]</h4></a>
                                                  <br />
                                                  </div>
                                        </div>
                        </section>
                  </div>
            </div>

<div id="myModal" class="modal">
   <div class="modal-content">
         <span class="close">&times;</span>
         <form method="POST" action="addUser">
               <br />
               Username:<br><br>
               <input type="text" name="name">
               <br /> <br />
               <input type="hidden" id="uid" name="uid" value="">
               <input type="submit" value="Submit">
         </form>
   </div>
</div>
</body>
</html>
```

# E. Expected Outcome

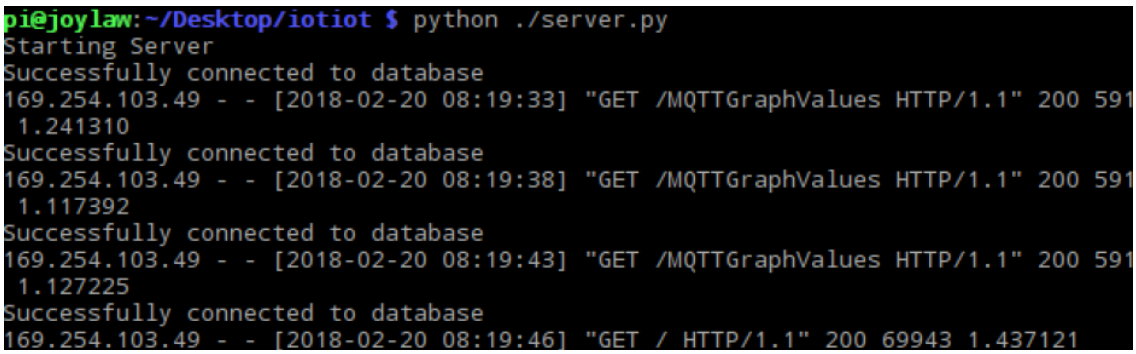
a) Change directory to ~/assignment:

```
cd ~/assignment
```

b) Run the following to start server.py:

```
python ./server.py
```

c) In your terminal, you should see the following



# Section 10
# The Publisher

## F. Write the Code for pub.py

a) Create a Python script pub.py with the following code:

```
sudo nano ~/assignment/pub.py
```

b) Enter the following code into pub.py:

```
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep
import time, datetime
import json
import Adafruit_DHT
import multiprocessing
from signal import pause
from gpiozero import LED
import MFRC522
import re
import boto3
import botocore
import RPi.GPIO as GPIO
from picamera import PiCamera
import os
import pymysql

# Returns tuple: (Humidity, Temperature)
def gethumiditytemp():
   return Adafruit_DHT.read_retry(11, 4)

def getdatetime():
   return time.strftime('%Y-%m-%d %H:%M:%S')

s3 = boto3.resource('s3')

def prepareImage():
   camera = PiCamera()
   datetime = time.strftime('%Y-%m-%d_%H-%M-%S') #Get the current date
and time
   full_path = os.getcwd() + "/"+datetime+".jpg"
   file_name = datetime+'.jpg'
   camera.capture(full_path)
   camera.close()
   return datetime, full_path, file_name

def setS3():
   bucket_name = 'accesssmarthome' # replace with your own unique bucket
name
   exists = True
   try:
         s3.meta.client.head_bucket(Bucket=bucket_name)
   except botocore.exceptions.ClientError as e:
         error_code = int(e.response['Error']['Code'])
         if error_code == 404:
               exists = False
```

```
   if exists == False:

   s3.create_bucket(Bucket=bucket_name,CreateBucketConfiguration={'Locat
ionConstraint': 'us-west-2'})
   return bucket_name

def rfid():
   mfrc522 = MFRC522.MFRC522()
   try:
         while True:
               (status,TagType)=
mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)
               if status == mfrc522.MI_OK:
                     (status,uid) = mfrc522.MFRC522_Anticoll()
                     datetime, full_path, file_name = prepareImage()
                     bucket_name = setS3()
                     db = pymysql.connect(host =
"iotca2.cmwnxcwvcz4v.us-west-2.rds.amazonaws.com",
user="admin",passwd="!QWER4321", port=3306, db="iotca2")
                     s3.Object(bucket_name,
file_name).put(Body=open(full_path, 'rb'))
                     try:
                         with db.cursor() as cur:
                               cur.execute("INSERT into rfid_accesslog
(date_time, rfid_uid) VALUES ('{0}', '{1}')".format(datetime,uid))
                               db.commit()
                     except Exception, e:
                              print (e)
                     db.close()
                     os.remove(full_path)
                     sleep(5)
   except:
         GPIO.cleanup()

LEDS = [LED(18), LED(21), LED(19), LED(13)]

def light_controller(client, userdata, message):
   data = json.loads(message.payload)

   print(data)

   if data is not None:
         led = int(data["led"])
         print led
         if led <= len(LEDS):
               if data["status"] == "ON":
                     LEDS[led].on()
               elif data["status"] == "OFF":
                     LEDS[led].off()
               else:
                     print("INVALID COMMAND")

def light_status():
   led_status = {}
   for led in LEDS:
         led_status.update({LEDS.index(led):led.is_lit})
   return led_status

def mqtt_connect():

   host = "a7gx67jy14o4h.iot.us-west-2.amazonaws.com"
```

```
   rootCAPath = "rootca.pem"
   certificatePath = "certificate.pem.crt"
   privateKeyPath = "private.pem.key"

   my_rpi = AWSIoTMQTTClient("publisher")
   my_rpi.configureEndpoint(host, 8883)
   my_rpi.configureCredentials(rootCAPath, privateKeyPath,
certificatePath)

   my_rpi.configureOfflinePublishQueueing(-1)
   my_rpi.configureDrainingFrequency(2)
   my_rpi.configureConnectDisconnectTimeout(10)
   my_rpi.configureMQTTOperationTimeout(5)

   my_rpi.connect()

   #Subscribe
   my_rpi.subscribe("led/control", 1, light_controller)

   while True:
         HT = gethumiditytemp()
         iot_data = {"datetime":str(getdatetime()),
"temperature":str(HT[1]), "humidity":str(HT[0])}
         my_rpi.publish("sensors/temp_humid", json.dumps(iot_data), 1)
         print("published ", iot_data)
         light_stat = light_status()
         my_rpi.publish("led/status", json.dumps(light_stat), 1)
         print("published ", light_stat)
         sleep(5)

if __name__ == '__main__':
   multiprocessing.Process(target=mqtt_connect).start()
   multiprocessing.Process(target=rfid).start()
   pause()
```

# G. Expected Outcome

a) Change directory to ~/assignment:

```
cd ~/assignment
```

b) Run the following to start server.py:

```
python ./pub.py
```

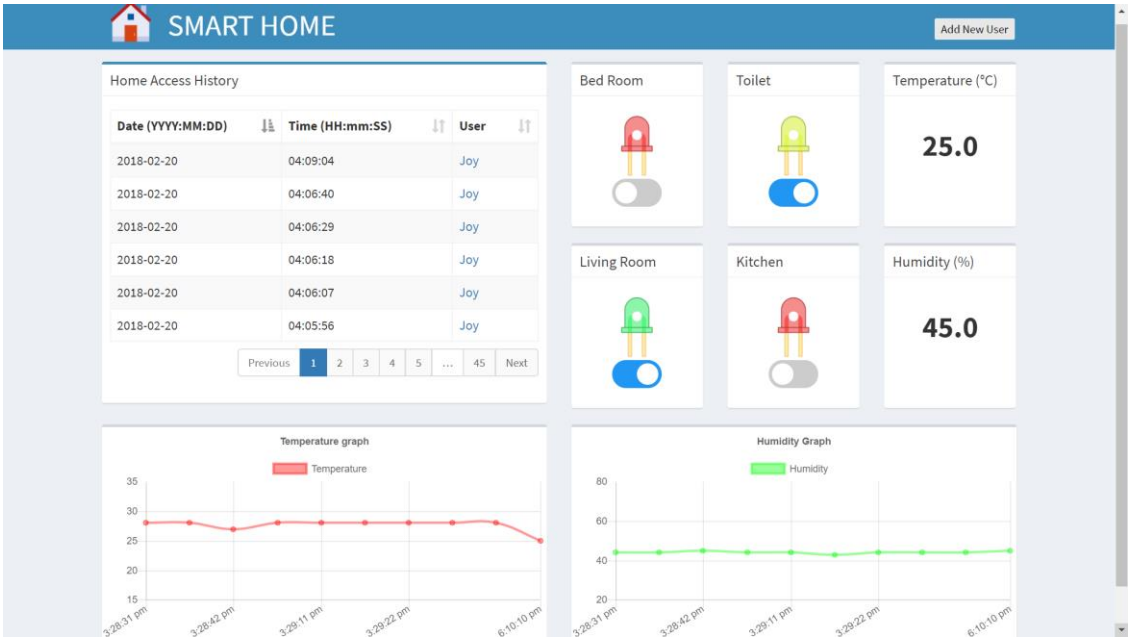
c) In your terminal, you should see the following:

```
pi@joylaw:~/Desktop/iotiot $ python ./pub.py
('published ', {'humidity': '29.0', 'temperature': '27.0', 'datetime': '2018-02-
20 08:21:49'})
('published ', {0: False, 1: False, 2: False, 3: False})
('published ', {'humidity': '29.0', 'temperature': '27.0', 'datetime': '2018-02-
20 08:21:57'})
('published ', {0: False, 1: False, 2: False, 3: False})
('published ', {'humidity': '28.0', 'temperature': '27.0', 'datetime': '2018-02-
20 08:22:05'})
('published ', {0: False, 1: False, 2: False, 3: False})
```

# H. Final Outcome (Dashboard)

a) On the Raspberry Pi that is running the server.py, using a web browser go to the following link where 169.254.134.247 is the IP address of your Raspberry Pi and 8010 is the port set in your server.py

`http://169.254.134.247:8010/`



# Section 11
# Task List

## A. Task List

| Task | Name |
|---|---|
| All RFID, Camera functions, documentation | Joy Law |
| All LED, temperature & humidity functions | Sean Phang |

**-- End of CA2 Step-by-step tutorial --**