**SCHOOL OF COMPUTING (SOC)**

|  |  |
| --- | --- |
| **Date of Submission:** | 18 August 2019 |

|  |  |
| --- | --- |
| **Prepared for:** | Mrs Dora Chua |

|  |  |
| --- | --- |
| **Class:** | DBIT/FT/3B/32 |

|  |  |
| --- | --- |
| **Submitted by:** |  |

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 1637735 | Keith Pang Yip Hon |
| 1748886 | Leonard Aw Wei Jia |
| 1748930 | Darryl Chua Zen Wee |

**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)**

**2017/2018 Semester 1**

**Table of Contents**

[Section 1 Overview of project 2](#_Toc16267438)

[A. Where we have uploaded our tutorial 2](#_Toc16267439)

[B. What is the application about? 2](#_Toc16267440)

[C. How does the final RPI set-up looks like? 2](#_Toc16267441)

[D. How does the web or mobile application look like? 2](#_Toc16267442)

[E. Evidence that we have met basic requirements 3](#_Toc16267443)

[F. System architecture of our system 4](#_Toc16267444)

[A. Quick-start guide (Readme first) 4](#_Toc16267445)

[Section 2 Hardware requirements 6](#_Toc16267446)

[Hardware checklist 6](#_Toc16267447)

[Section 3 Hardware set up 6](#_Toc16267448)

[Fritzing drawing 6](#_Toc16267449)

[Section 4 Software requirements 6](#_Toc16267450)

[Software 6](#_Toc16267451)

[Section 5 Mosquitto(master device) 7](#_Toc16267452)

[Section 6 MQTT(master device) 8](#_Toc16267453)

[Section 7 Index.html 14](#_Toc16267454)

[Section 8 slave 1 17](#_Toc16267455)

[Section 9 slave 2 19](#_Toc16267456)

[Section 10 Database requirements 21](#_Toc16267457)

[Database set up 21](#_Toc16267458)

[1. Sign up for AWS Educate account 21](#_Toc16267459)

[A. Set up a new AWSEducate account 22](#_Toc16267461)

[2. Sign in to AWS IoT Console 23](#_Toc16267462)

[A. Log in your AWS Educate account 24](#_Toc16267463)

[3. EC2 25](#_Toc16267464)

[A.Set up EC2 25](#_Toc16267466)

[4. Database access 30](#_Toc16267467)

[A.Instance console 30](#_Toc16267468)

# Section 1 Overview of project

* 1. Where we have uploaded our tutorial

<https://youtu.be/PP5z5ugXE3s>

<https://github.com/Leonardawj/iotec2>

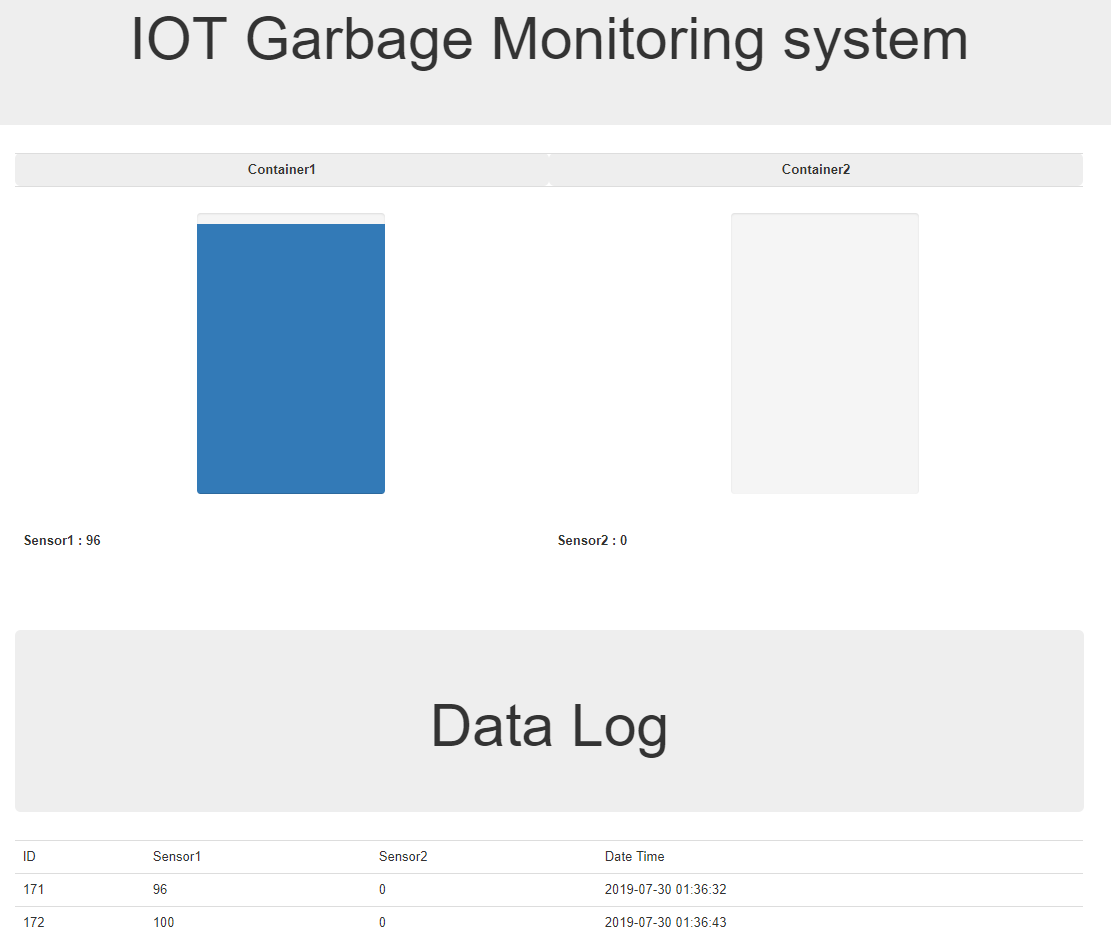
* 1. What is the application about?

Our project is IOT Garbage Monitoring system is a very innovative system which will help to keep the cities clean. This system monitors the garbage bins and informs about the level of garbage collected in the garbage bins via a web page. For this the system uses ultrasonic sensors placed over the bins to detect the garbage level and compare it with the garbage bins depth. 100 means it is empty and 0 mean it is full.

* 1. How does the final RPI set-up looks like?



* 1. How does the web or mobile application look like?

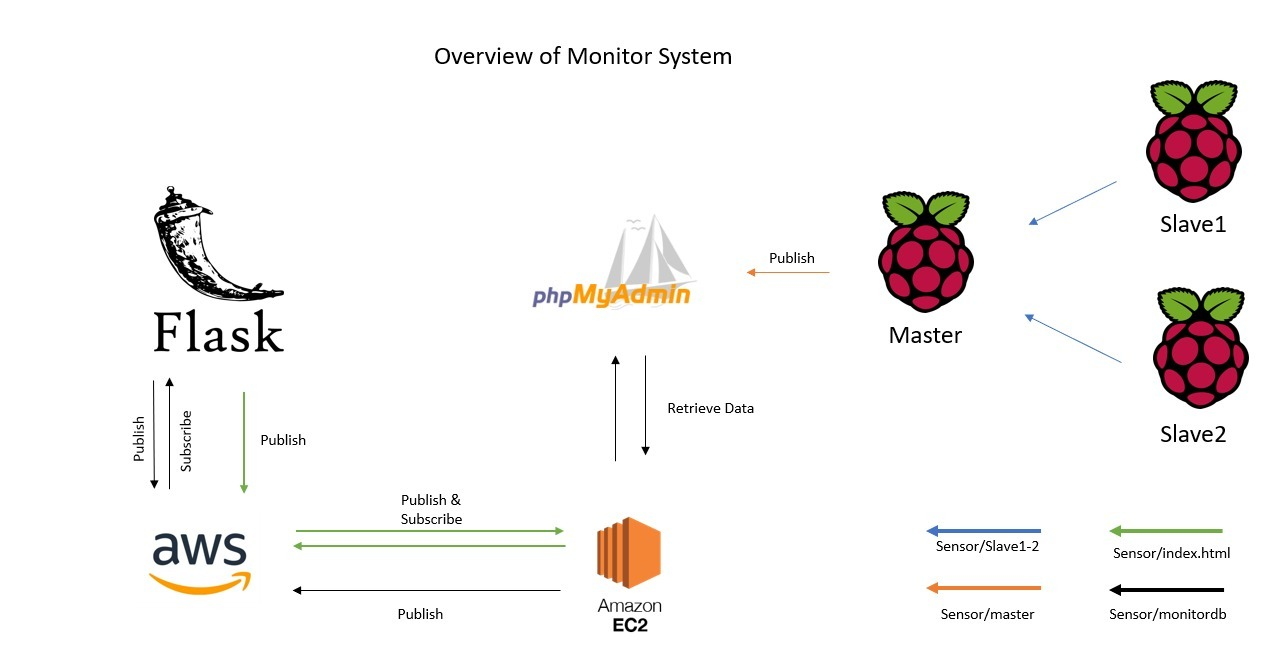


* 1. Evidence that we have met basic requirements

Provide bullet list to describe how your group has met basic requirements

|  |  |
| --- | --- |
| Requirement | Evidence |
| Used 2 sensors | Used 2 Ultrasonic sensor on slave raspiberry pi |
| Used 1 actuators | Used 1 LCD moniter on master raspiberry pi |
| Used MQTT | Our MQTT endpoint -->AWS |
| Stored data in cloud | Stored motion data in Cloudant database in AWS Cloud |
| Used cloud service | Use AWS Rekognition, hosted web server on EC2 |
| Provide real-time sensor value / status | Show the real-time value of ultrasonic sensor |
| Provide historical sensor value/ status | Show historical vlaue of ultrasonic sensor |

* 1. System architecture of our system



* 1. Quick-start guide (Readme first)

Give a few lines of basic instructions on how I need to run your app, e.g

1. Before u check you should ifconifg to check for master device for the ip address and check slave1.py and slave2.py whether the ip address in these file is the same as the ip address as the master device.
2. Modify the file mqtt\_subscriber ip address
3. First connect hardware as in Section 1 C for slave devices(2 device)
4. Then run the mqtt\_subscriber.py file and mosquitto command for web server in the master device
5. Then run slave1.py and slave2.py in the both slave device
6. To display our website u can either use the local address of the master raspiberry pi or aws ip address which is <http://ec2-3-87-62-43.compute-1.amazonaws.com/iotgarbagemonitoring/>

# Section 2 Hardware requirements

Hardware checklist

1.3 Raspiberry model b

2.2 Ultrasonic sensor

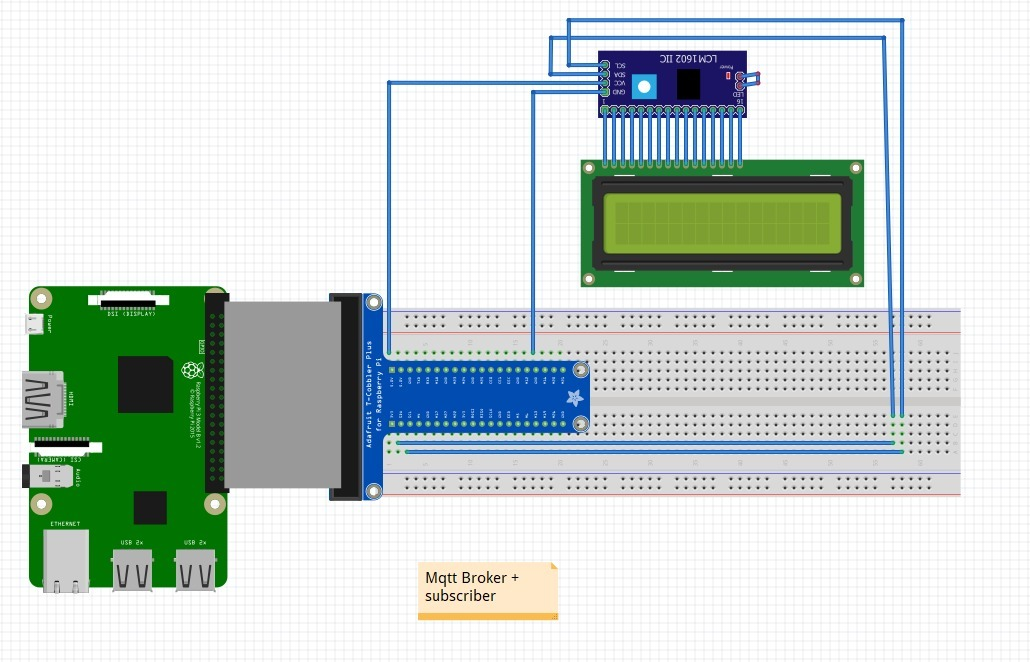
3.2 LCD Moniter

4.12 Female to male connector

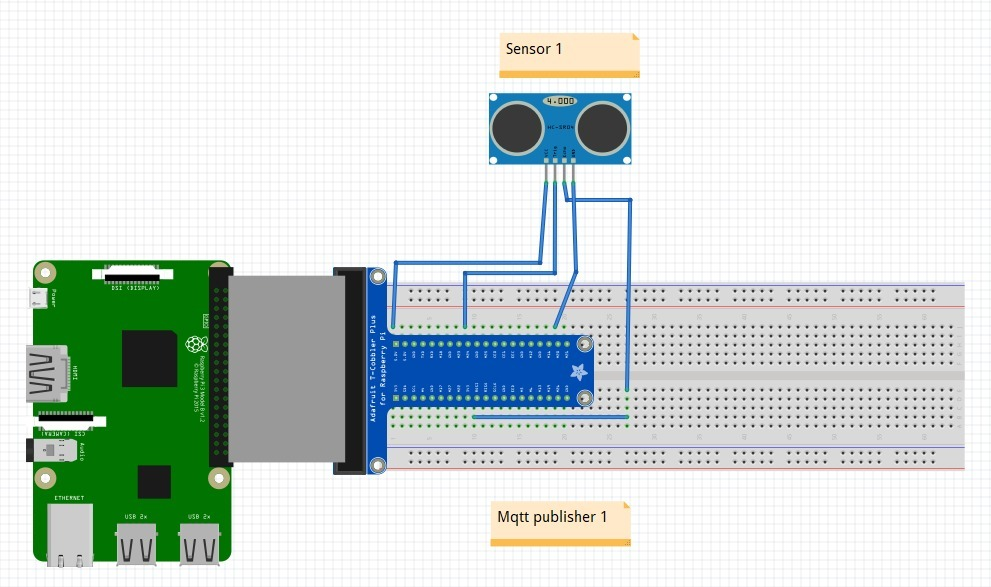
# Section 3 Hardware set up

Fritzing drawing

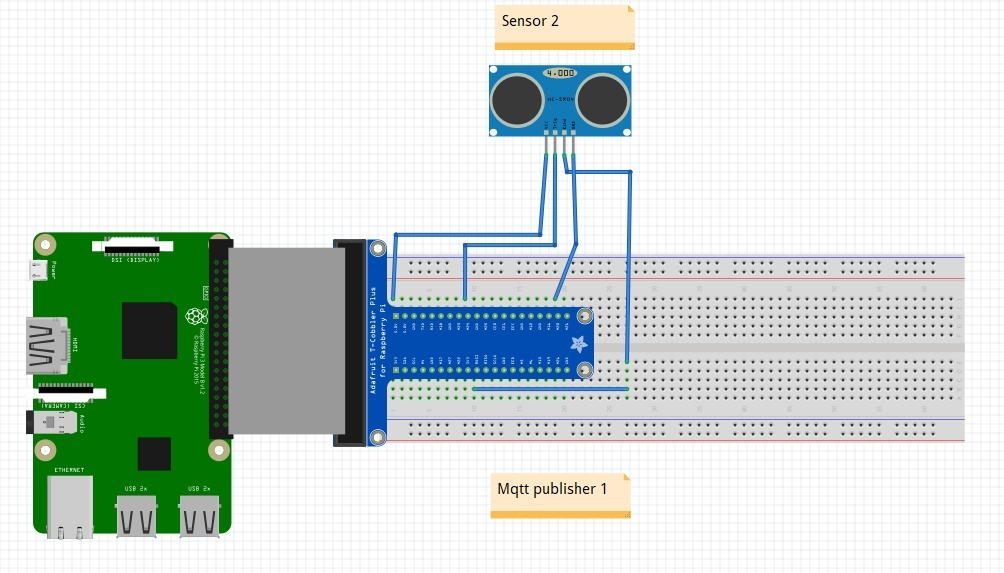
Master device



Slave 1



Slave 2



# Section 4 Software requirements

Software

* **AWS**
* **Phpmyadmin**

# Section 5 Mosquitto(master device)

|  |  |
| --- | --- |
| Open first Terminal windows | |
| In the First Terminal window, make sure the Mosquitto broker is running  mosquitto |  |
| You should see another message in the first terminal window saying another client is connected.    You should also see this message in the subscriber terminal: | |
| Troubleshoot  If u encountered error port in used.Use the following code to stop the mosquitto and then relaunch | |

# Section 6 MQTT(master device)

|  |  |
| --- | --- |
| Source code | |
| import paho.mqtt.client as mqtt  import os  import threading  from time import sleep  import MySQLdb  import datetime  import smbus  import time  # Define some device parameters  I2C\_ADDR = 0x27 # I2C device address  LCD\_WIDTH = 16 # Maximum characters per line  # Define some device constants  LCD\_CHR = 1 # Mode - Sending data  LCD\_CMD = 0 # Mode - Sending command  LCD\_LINE\_1 = 0x80 # LCD RAM address for the 1st line  LCD\_LINE\_2 = 0xC0 # LCD RAM address for the 2nd line  LCD\_LINE\_3 = 0x94 # LCD RAM address for the 3rd line  LCD\_LINE\_4 = 0xD4 # LCD RAM address for the 4th line  LCD\_BACKLIGHT = 0x08 # On  #LCD\_BACKLIGHT = 0x00 # Off  ENABLE = 0b00000100 # Enable bit  # Timing constants  E\_PULSE = 0.0005  E\_DELAY = 0.0005  #Open I2C interface  #bus = smbus.SMBus(0) # Rev 1 Pi uses 0  bus = smbus.SMBus(1) # Rev 2 Pi uses 1    MQTT\_SERVER = "localhost"  MQTT\_PATH1 = "channel1"  MQTT\_PATH2 = "channel2"  sensor1 = 0  sensor2 = 0  def lcd\_init():  # Initialise display  lcd\_byte(0x33,LCD\_CMD) # 110011 Initialise  lcd\_byte(0x32,LCD\_CMD) # 110010 Initialise  lcd\_byte(0x06,LCD\_CMD) # 000110 Cursor move direction  lcd\_byte(0x0C,LCD\_CMD) # 001100 Display On,Cursor Off, Blink Off  lcd\_byte(0x28,LCD\_CMD) # 101000 Data length, number of lines, font size  lcd\_byte(0x01,LCD\_CMD) # 000001 Clear display  time.sleep(E\_DELAY)    def lcd\_byte(bits, mode):  # Send byte to data pins  # bits = the data  # mode = 1 for data  # 0 for command  bits\_high = mode | (bits & 0xF0) | LCD\_BACKLIGHT  bits\_low = mode | ((bits<<4) & 0xF0) | LCD\_BACKLIGHT  # High bits  bus.write\_byte(I2C\_ADDR, bits\_high)  lcd\_toggle\_enable(bits\_high)  # Low bits  bus.write\_byte(I2C\_ADDR, bits\_low)  lcd\_toggle\_enable(bits\_low)  def lcd\_toggle\_enable(bits):  # Toggle enable  time.sleep(E\_DELAY)  bus.write\_byte(I2C\_ADDR, (bits | ENABLE))  time.sleep(E\_PULSE)  bus.write\_byte(I2C\_ADDR,(bits & ~ENABLE))  time.sleep(E\_DELAY)  def lcd\_string(message,line):  # Send string to display  message = message.ljust(LCD\_WIDTH," ")  lcd\_byte(line, LCD\_CMD)  for i in range(LCD\_WIDTH):  lcd\_byte(ord(message[i]),LCD\_CHR)    # The callback for when the client receives a CONNACK response from the server.  def on\_connect1(client, userdata, flags, rc):  print("Connected with result code "+str(rc))    # Subscribing in on\_connect() means that if we lose the connection and  # reconnect then subscriptions will be renewed.  client.subscribe(MQTT\_PATH1)  # The callback for when the client receives a CONNACK response from the server.  def on\_connect2(client, userdata, flags, rc):  print("Connected with result code "+str(rc))    # Subscribing in on\_connect() means that if we lose the connection and  # reconnect then subscriptions will be renewed.  client.subscribe(MQTT\_PATH2)    # The callback for when a PUBLISH message is received from the server.  def on\_message1(client, userdata, msg):  global sensor1  if msg.topic == 'channel1':  sensor1 = msg.payload  print("ch1:-"+sensor1)  def on\_message2(client, userdata, msg):  global sensor2  if msg.topic == 'channel2':  sensor2 = msg.payload  print("ch2:-"+sensor2)  def client1\_fxn(args):  args.loop\_forever()  def client2\_fxn(args):  args.loop\_forever()  def main():  global sensor1  global sensor2  print("Start")  client1 = mqtt.Client()  client1.on\_connect = on\_connect1  client1.on\_message = on\_message1    client2 = mqtt.Client()  client2.on\_connect = on\_connect2  client2.on\_message = on\_message2    client1.connect(MQTT\_SERVER, 1883, 60)  client2.connect(MQTT\_SERVER, 1883, 60)    threading.Thread(target=client1\_fxn, args=(client1,)).start()  threading.Thread(target=client2\_fxn, args=(client2,)).start()  db = MySQLdb.connect(host="ec2-3-87-62-43.compute-1.amazonaws.com", user="root", passwd="password", db="iotgarbagemonitoring")  cur=db.cursor()  lcd\_init()    while 1:  datetime1 = datetime.datetime.now()  a = int((100 \* int(sensor1))/25)  if a > 100:  a = 100  b = int((100 \* int(sensor2))/25)  if b > 100:  b = 100  line1 = "Sensor 1 : " + str(a)  line2 = "Sensor 2 : " + str(b)  # Send some test  lcd\_string(line1,LCD\_LINE\_1)  lcd\_string(line2,LCD\_LINE\_2)    try:  cur.execute("""INSERT INTO `sensor\_data`(`sensor1`, `sensor2`, `date`) VALUES (%s,%s,%s)""",(a,b,datetime1))  db.commit()  #print("DB Added")  except:  print("Db error")  db.rollback()  sleep(10)    cur.close()  db.close ()  if \_name\_ == "\_main\_":  try:  main()  except KeyboardInterrupt:  pass  finally:  lcd\_byte(0x01, LCD\_CMD) | |
| Open second Terminal windows | |
| In the Second Terminal window, make sure the MQTT is running  Once u have seen this result, this show that ur master device have suddenly connected to both of the slave device.  Sudo python mqtt\_subscriber.py |  |
| You should see another message in the first terminal window saying another client is connected and you should also see this message in the subscriber terminal: | |

# Section 7 Index.html

|  |
| --- |
| Source code |
| <?php  $servername = "localhost";  $username = "root";  $password = "dmitiot";  $dbname = "iotgarbagemonitoring";  // Create connection  $conn = new mysqli($servername, $username, $password, $dbname);  // Check connection  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }  $sql = "SELECT \*from sensor\_data ORDER BY id DESC LIMIT 1";  $result = $conn->query($sql);  if ($result->num\_rows == 1)  {  $row = $result->fetch\_assoc();  $sensor1 = $row["sensor1"];  $sensor2 = $row["sensor2"];  }  else  {  echo "0 results";  }    $conn->close();  ?>  <!DOCTYPE html>  <html lang="en">  <head>  <title>IOT Garbage Monitoring system</title>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <link rel="stylesheet" href="bootstrap.min.css">  <script src="bootstrap.min.js"></script>  <link rel="stylesheet" href="styles.css">  <meta http-equiv="refresh" content="5">  </head>  <body>  <div class="jumbotron text-center">  <h1>IOT Garbage Monitoring system</h1>  </div>  <div class="container">  <table class="table">  <tr>  <th class="jumbotron text-center">Container1</th>  <th class="jumbotron text-center">Container2</th>  </tr>  <tr>  <th>  <div class="center">  <div class="progress progress-bar-vertical">  <div class="progress-bar" role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100" <?php echo "style=\"height: ".$sensor1."%;\""; ?> > </div>  </div>  </div>  <div> <?php echo "Sensor1 : ". $sensor1; ?> </div>  </th>  <th>  <div class="center">  <div class="progress progress-bar-vertical">  <div class="progress-bar" role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100" <?php echo "style=\"height: ".$sensor2."%;\""; ?> > </div>  </div>  </div>  <div> <?php echo "Sensor2 : ". $sensor2; ?> </div>  </th>  </tr>  </table>  <br> <br> <br>  <div class="jumbotron text-center">  <h1>Data Log</h1>  </div>  <?php  $conn = new mysqli($servername, $username, $password, $dbname);  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }  $sql = 'SELECT \* FROM `sensor\_data` WHERE 1';  $result = $conn->query($sql);  if ($result->num\_rows > 0)  {  echo "<table class=\"table\">";  //echo "<tr><td>NO of Data : " . $result->num\_rows . "</td></tr>";  echo "<tr><td>ID</td><td>Sensor1</td><td>Sensor2</td><td>Date Time</td></tr>";    while($row = $result->fetch\_array(MYSQLI\_ASSOC))  {  echo "<tr><td>" . $row['id'] . "</td><td>" . $row['sensor1'] . "</td><td>" . $row['sensor2'] . "</td><td>" . $row['date'] . "</td></tr>";  }  echo "</table>";  $result->close();  }  ?>  </div>  </body>  </html> |
| This is our index.html for our website as shown on section. It is a simple interface which consistent a real time and historical dataset.Our data is obtain every 10 sec. |

# Section 8 slave 1

|  |  |
| --- | --- |
| Source code | |
| import RPi.GPIO as GPIO  from time import sleep  import paho.mqtt.client as mqtt  import time  broker\_address="192.168.1.238"  # set GPIO Pins  GPIO\_TRIGGER = 22  GPIO\_ECHO = 23  def ini\_sensor():  GPIO.cleanup()  # GPIO Mode (BOARD / BCM)  GPIO.setmode(GPIO.BCM)  # set GPIO direction (IN / OUT)  GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)  GPIO.setup(GPIO\_ECHO, GPIO.IN)  def distance():  # set Trigger to HIGH  GPIO.output(GPIO\_TRIGGER, True)  # set Trigger after 0.01ms to LOW  time.sleep(0.00001)  GPIO.output(GPIO\_TRIGGER, False)  StartTime = StopTime = time.time()  #StopTime = time.time()  # save StartTime  while GPIO.input(GPIO\_ECHO) == 0:  StartTime = time.time()  # save time of arrival  while GPIO.input(GPIO\_ECHO) == 1:  StopTime = time.time()  # time difference between start and arrival  TimeElapsed = StopTime - StartTime  # multiply with the sonic speed (34300 cm/s)  # and divide by 2, because there and back  distance = (TimeElapsed \* 34300) / 2  return distance  def main():  ini\_sensor()  client = mqtt.Client("P1") #create new instance  client.connect(broker\_address) #connect to broker  while True:  dist = distance()  dist = int(dist)  print("Measured Distance="+str(dist))  client.publish("channel1",dist) #publish  sleep(1)  if \_\_name\_\_ == "\_\_main\_\_":  main() | |
| Open Terminal windows | |
| In the Terminal window, make sure the code is running  Sudo python slave1.py |  |
| You should see the distance being measured in the following terminal  This show that our ultrasonic sensor is reading is measuring the distance of our rubbish bin fromt the top to bot. | |

# Section 9 slave 2

|  |  |
| --- | --- |
| Source code | |
| import RPi.GPIO as GPIO  from time import sleep  import paho.mqtt.client as mqtt  import time  broker\_address="192.168.1.238"  # set GPIO Pins  GPIO\_TRIGGER = 22  GPIO\_ECHO = 23  def ini\_sensor():  GPIO.cleanup()  # GPIO Mode (BOARD / BCM)  GPIO.setmode(GPIO.BCM)  # set GPIO direction (IN / OUT)  GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)  GPIO.setup(GPIO\_ECHO, GPIO.IN)  def distance():  # set Trigger to HIGH  GPIO.output(GPIO\_TRIGGER, True)  # set Trigger after 0.01ms to LOW  time.sleep(0.00001)  GPIO.output(GPIO\_TRIGGER, False)  StartTime = StopTime = time.time()  #StopTime = time.time()  # save StartTime  while GPIO.input(GPIO\_ECHO) == 0:  StartTime = time.time()  # save time of arrival  while GPIO.input(GPIO\_ECHO) == 1:  StopTime = time.time()  # time difference between start and arrival  TimeElapsed = StopTime - StartTime  # multiply with the sonic speed (34300 cm/s)  # and divide by 2, because there and back  distance = (TimeElapsed \* 34300) / 2  return distance  def main():  ini\_sensor()  client = mqtt.Client("P1") #create new instance  client.connect(broker\_address) #connect to broker  while True:  dist = distance()  dist = int(dist)  print("Measured Distance="+str(dist))  client.publish("channel1",dist) #publish sleep(1)  if \_\_name\_\_ == "\_\_main\_\_":  main() | |
| Open Terminal windows | |
| In the Terminal window, make sure the code is running  Sudo python slave2.py |  |
| You should see the distance being measured in the following terminal | |
| This show that our ultrasonic sensor is reading is measuring the distance of our rubbish bin fromt the top to bot. | |

# Section 10 Database requirements

Database set up

# Sign up for AWS Educate account



## Set up a new AWSEducate account

| No | Task | | | |
| --- | --- | --- | --- | --- |
|  | Access AWS Educate website: <https://aws.amazon.com/education/awseducate/apply/>  Select “Students” option | | | |
|  | | | | |
|  | Select Role as “Student” | | | |
|  | | | | |
|  | On the sign up form, ensure the following settings are specified, and fill in the rest of the form as appropriate and then click “Next” button. | | | |
|  | * Institution name is “**Singapore Polytechnic School of EEE**” * Email is your SP ichat email address * “Click here to select an AWS Educate Starter” account is selected | |  | |
|  | Follow the instructions to verify your account by providing a verification code that is sent to your ichat email account. | | |  |
|  | Once your AWS account is ready for use, you will receive an email similar to this one |  | | |

# Sign in to AWS IoT Console

After you have received the email confirmation from AWS, you are now ready to log into the AWS IoT console to start the lab.

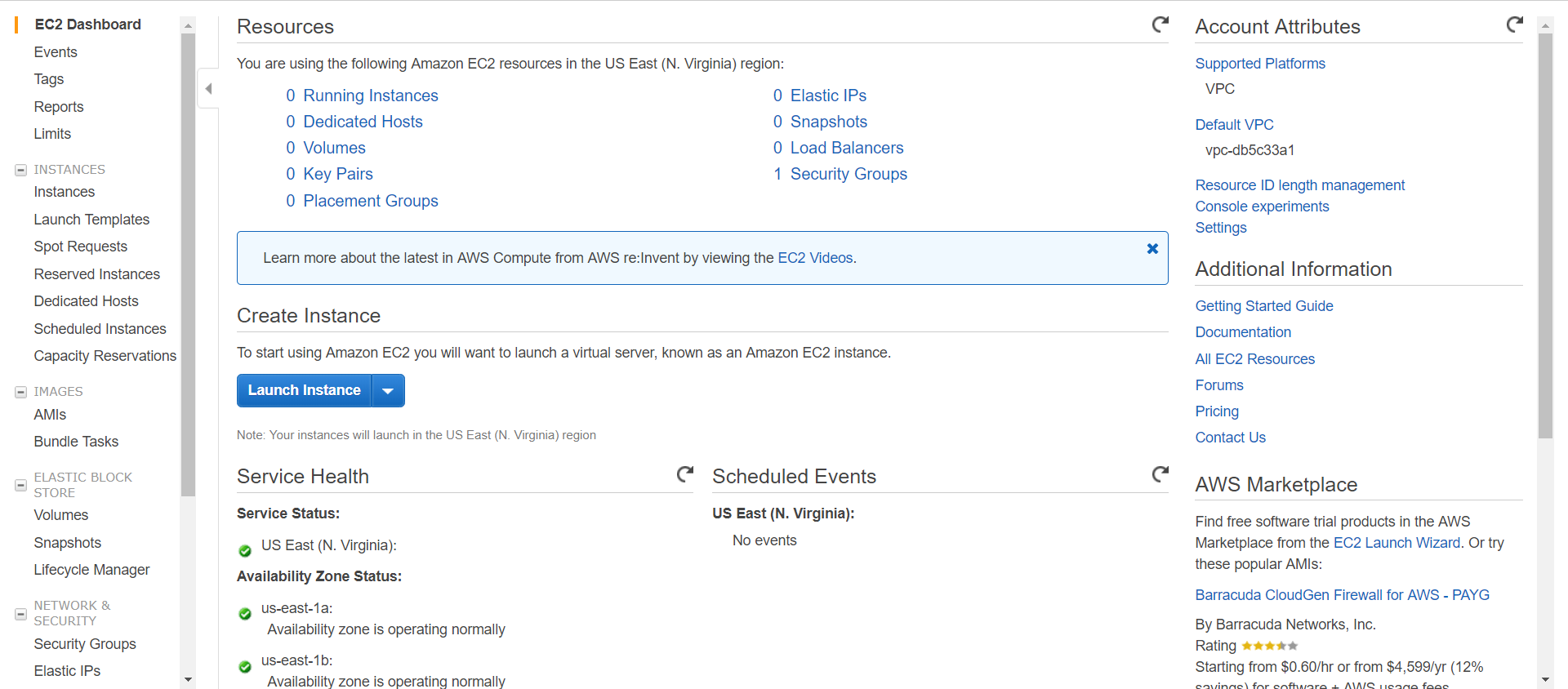
1. Log in your AWS Educate account

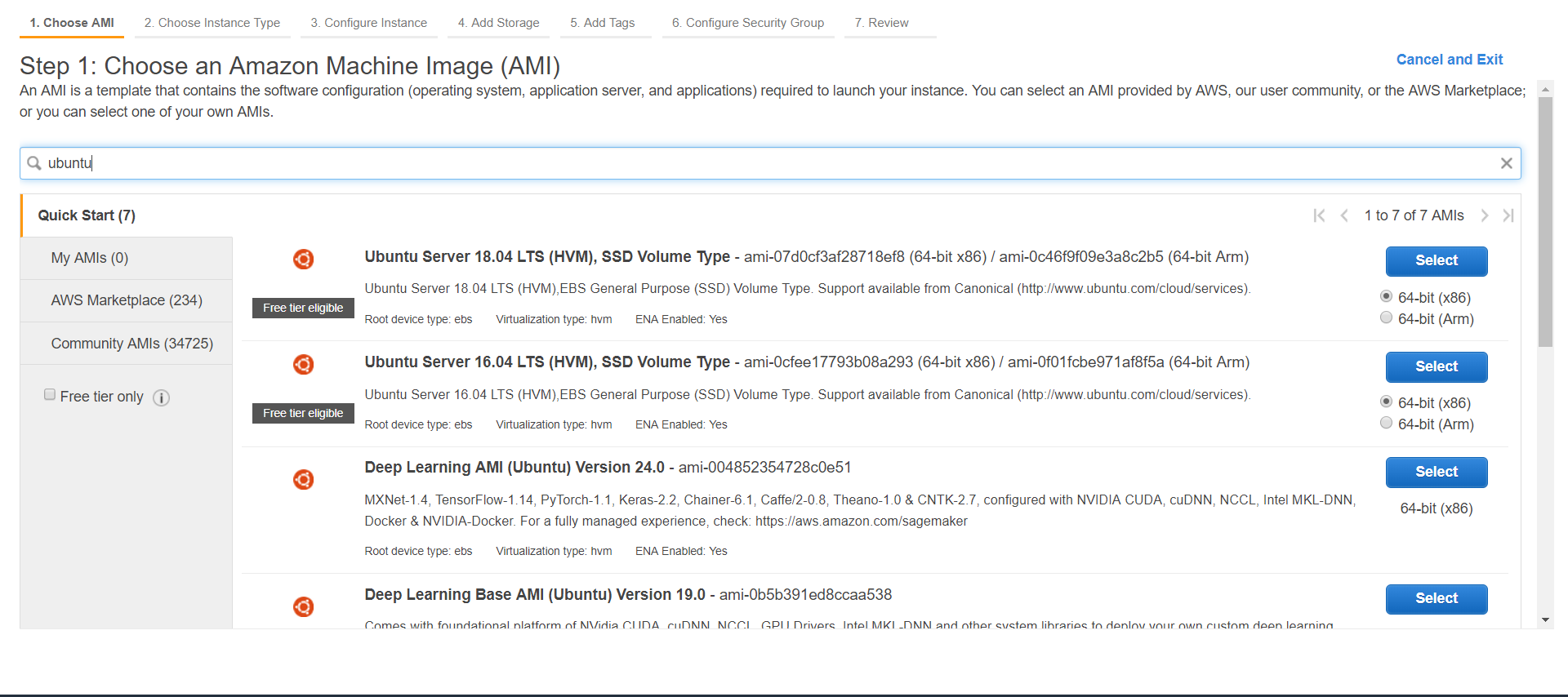
|  |  |  |
| --- | --- | --- |
|  | Log in your AWS Educate account using your ichat email address.  <https://www.awseducate.com/signin/SiteLogin> | |
|  | Click “AWS Account”  You should be brought to a screen similar to this.  Click AWS Console button. |  |
|  | After a while, you will be brought in the AWS Management Console. |  |

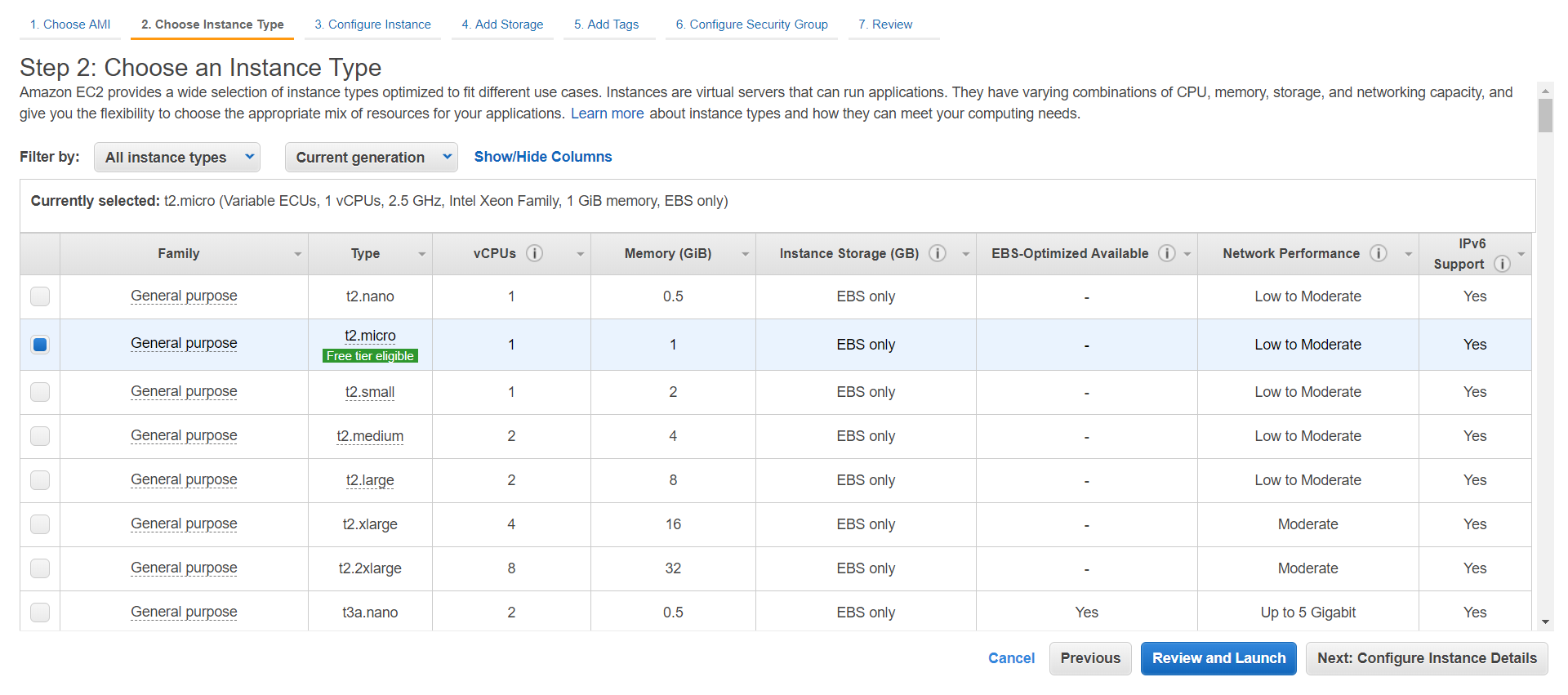
# EC2

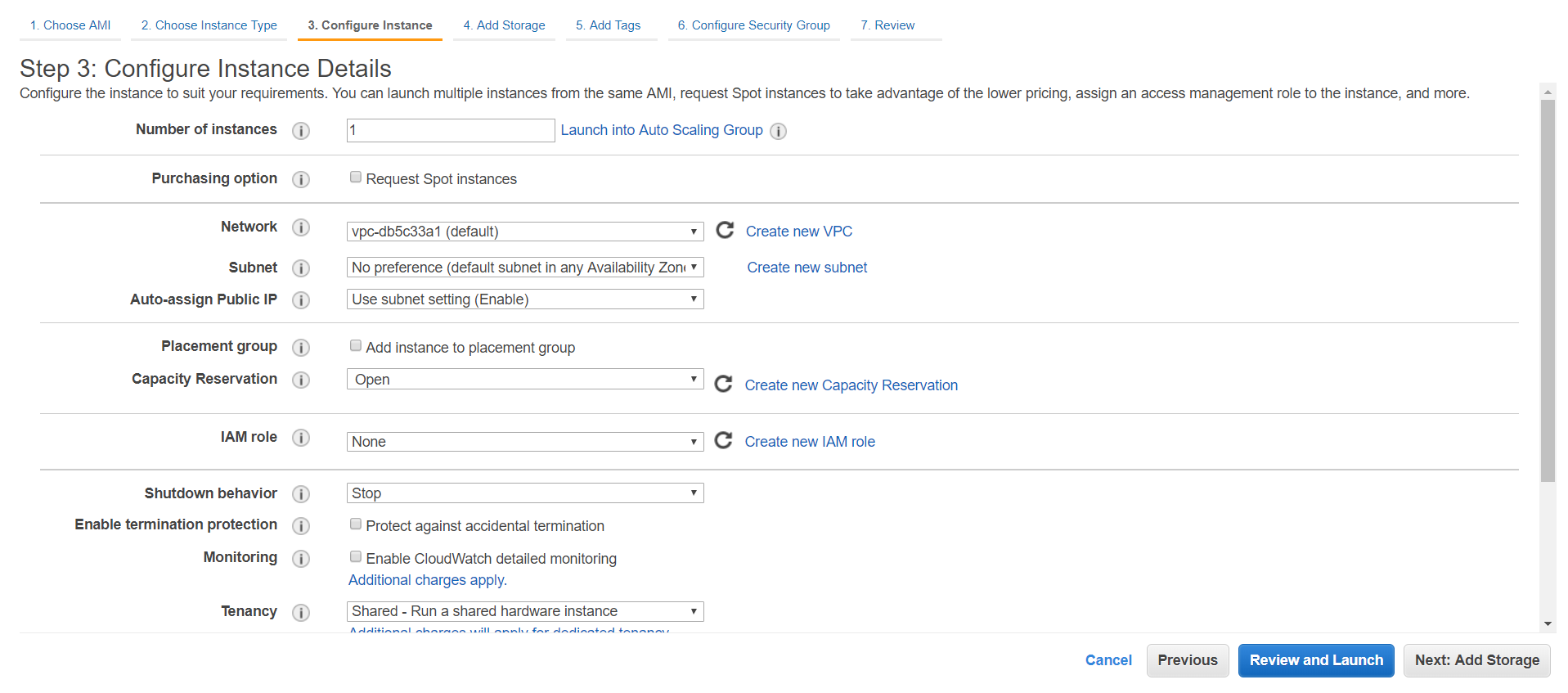


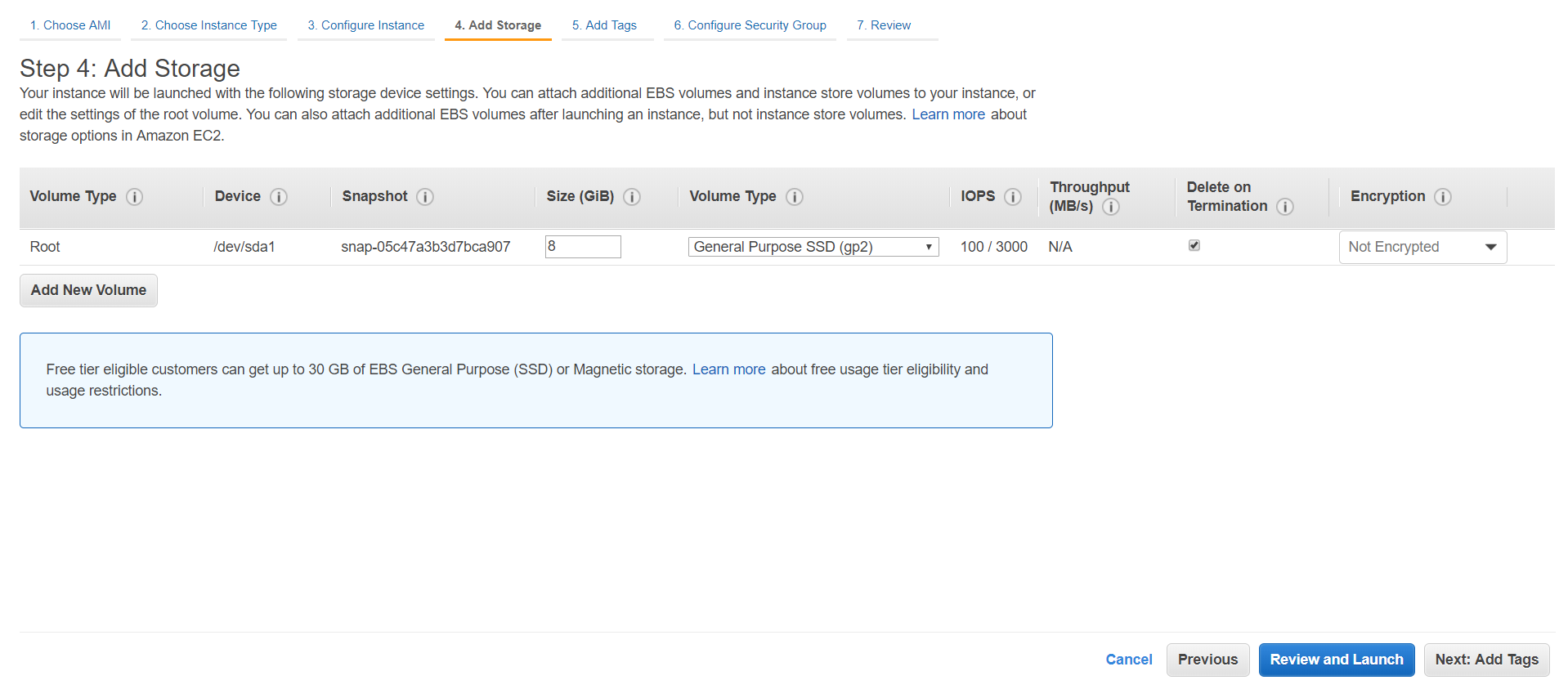
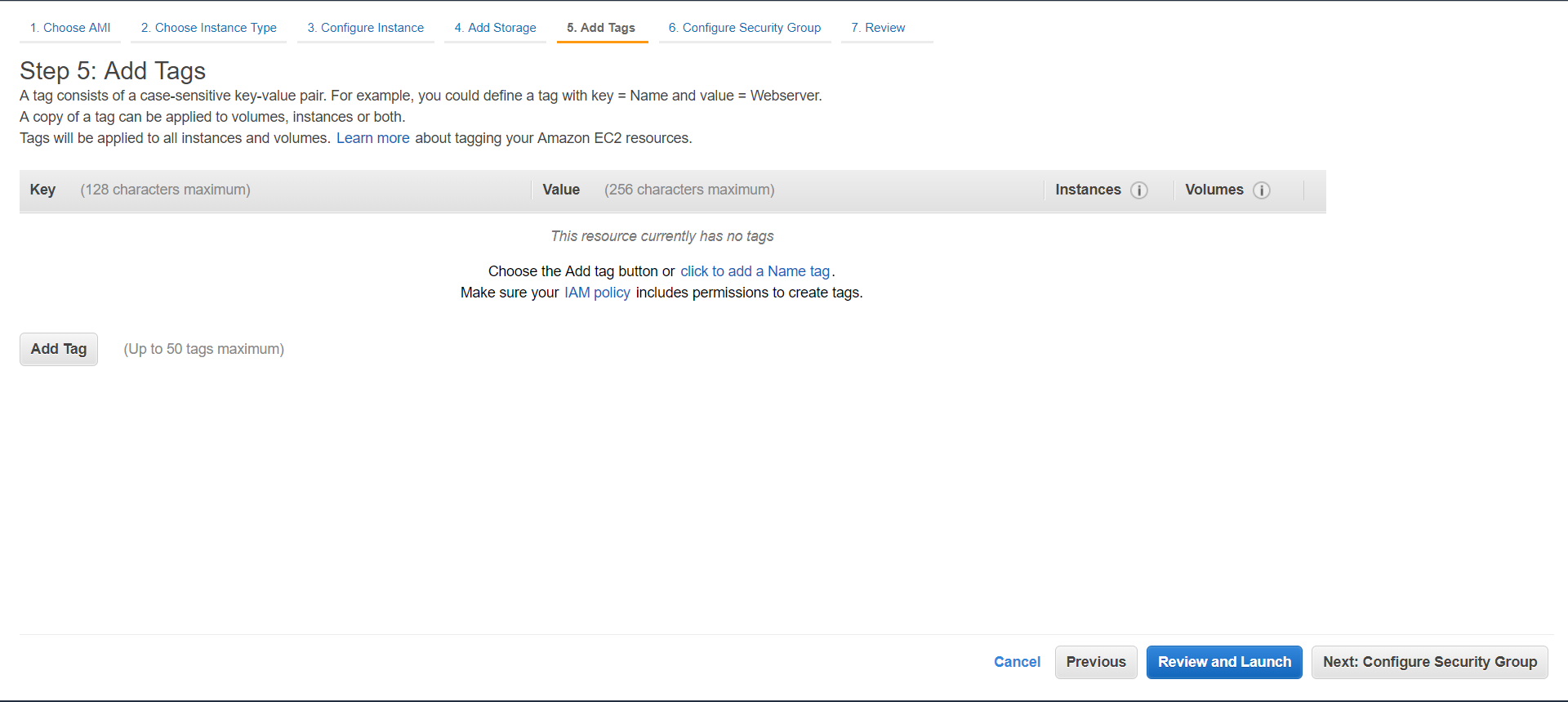
## A.Set up EC2

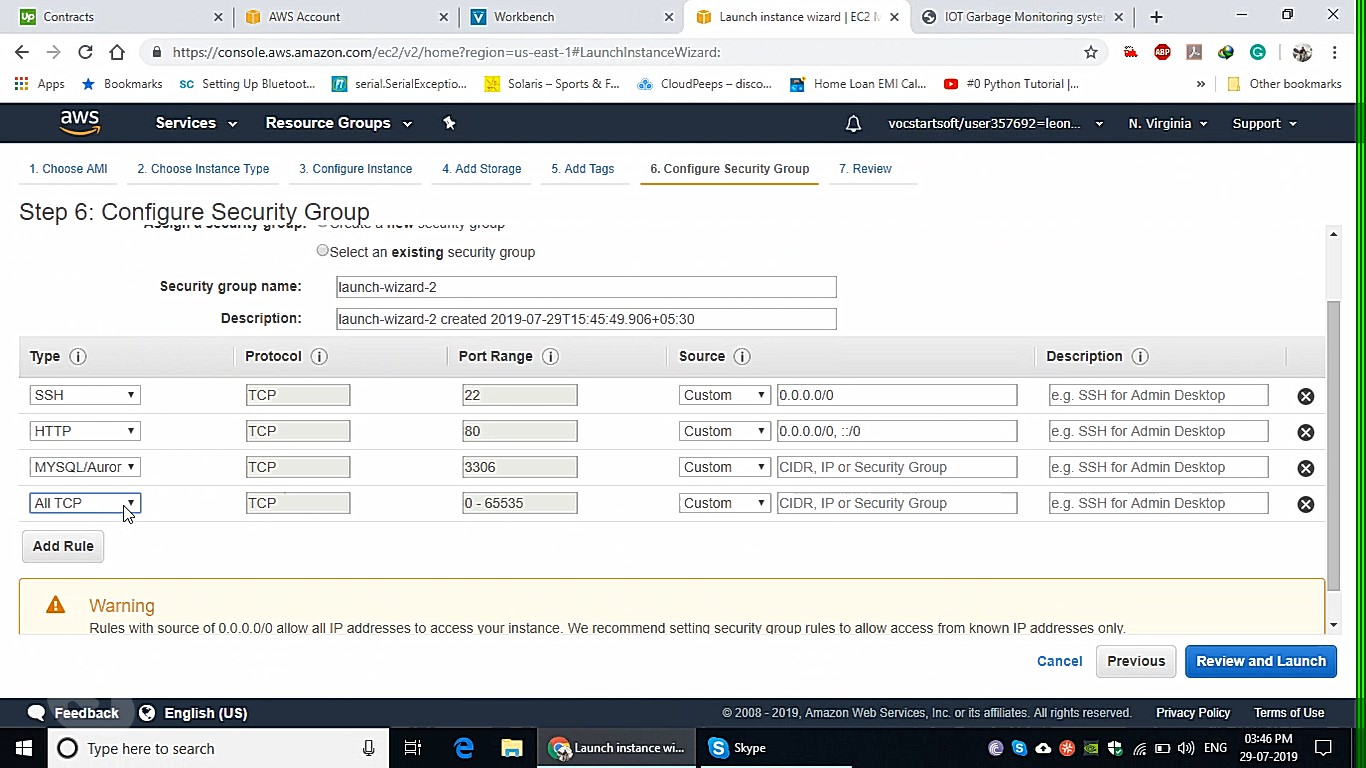


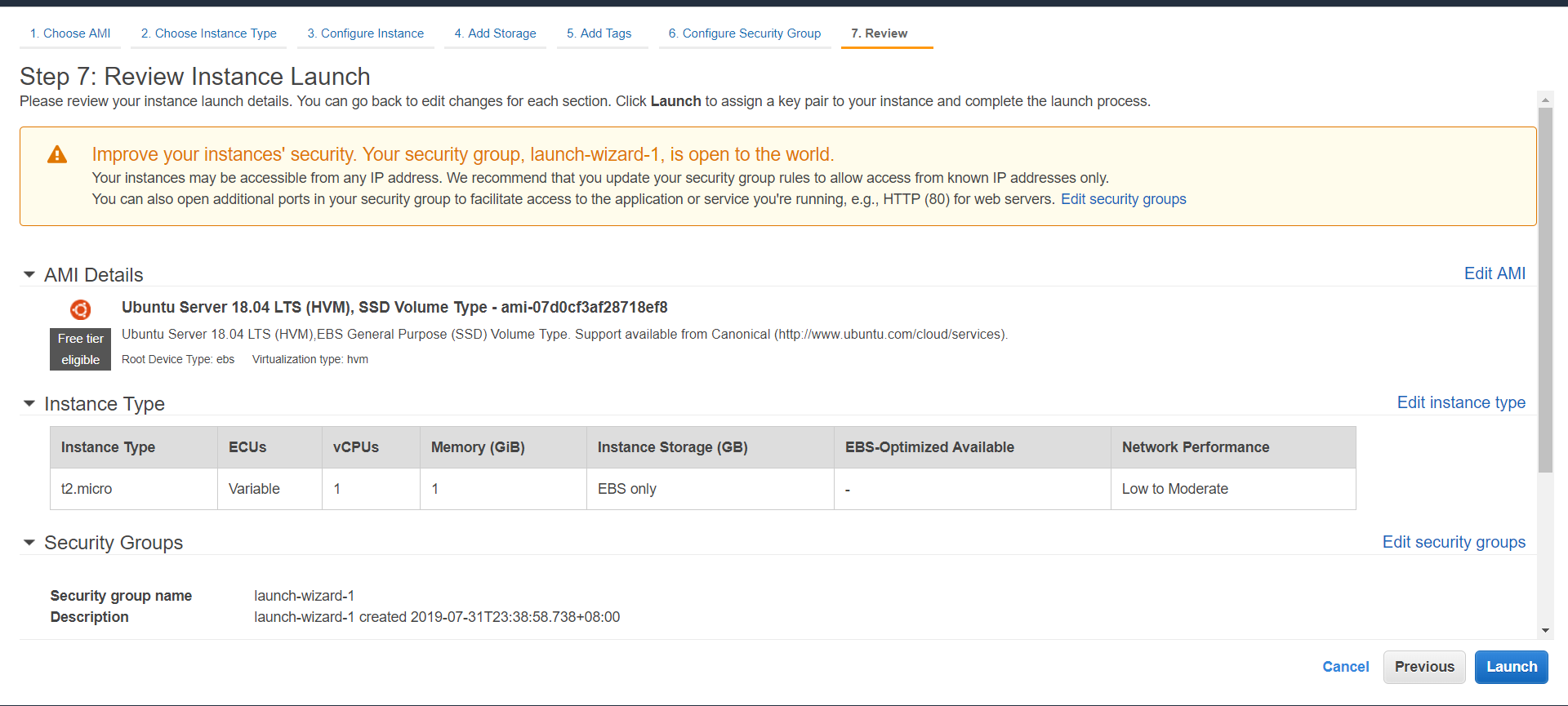










| No | Task | | | |
| --- | --- | --- | --- | --- |
|  | Access AWS Educate website: <https://aws.amazon.com/education/awseducate/apply/>  Select “Students” option | | | |
|  | | | | |
|  | Select Role as “Student” | | | |
|  | | | | |
|  | On the sign up form, ensure the following settings are specified, and fill in the rest of the form as appropriate and then click “Next” button. | | | |
|  | * Institution name is “**Singapore Polytechnic School of EEE**” * Email is your SP ichat email address * “Click here to select an AWS Educate Starter” account is selected | |  | |
|  | Follow the instructions to verify your account by providing a verification code that is sent to your ichat email account. | | |  |
|  | Once your AWS account is ready for use, you will receive an email similar to this one |  | | |

# Database access

After you are done setting up,this is what u will see as following

A.Instance console

|  |  |
| --- | --- |
|  | Url for webpage : <http://ec2-3-87-62-43.compute-1.amazonaws.com/iotgarbagemonitoring/>  url for database : <http://ec2-3-87-62-43.compute-1.amazonaws.com/phpmyadmin/> |
| <http://ec2-3-87-62-43.compute-1.amazonaws.com/phpmyadmin/>  After u have log in to ur database,create a database and named it as iotgarbagemonitoring | |
| Create a table called sensor\_data in the database u have just created | |
| SQL structure for the table | |

## Reference

AY1910s1 ST0324 IoT Practical 10

AY1910s1 ST0324 IoT Practical 11

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