**SCHOOL OF COMPUTING (SOC)**

|  |  |
| --- | --- |
| **Date of Submission:** | 23 February 2020 |

|  |  |
| --- | --- |
| **Prepared for:** | Dora chua |

|  |  |
| --- | --- |
| **Class:** | DISM/FT/3A/41 |

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

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 1703180 | Kenneth Lee Jia Le |
| 1726653 | Tang Xiu Wen Raina |

**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 3](#_Toc33462751)

[A. Where we have uploaded our tutorial 3](#_Toc33462752)

[B. What is the application about? 3](#_Toc33462753)

[C. How does the final RPI set-up looks like? 4](#_Toc33462754)

[D. How does the web or mobile application look like? 5](#_Toc33462755)

[E. System architecture of our system 5](#_Toc33462756)

[F. Evidence that we have met basic requirements 6](#_Toc33462757)

[G. Telegram Notification System 7](#_Toc33462758)

[H. Bonus features on top of basic requirements 7](#_Toc33462759)

[A. Quick-start guide (Readme first) 7](#_Toc33462760)

[Section 2 Hardware requirements 9](#_Toc33462761)

[Hardware checklist 9](#_Toc33462762)

[Hardware setup instructions 9](#_Toc33462763)

[Fritzing Diagram 10](#_Toc33462764)

[Section 3 Software Requirements 11](#_Toc33462765)

[Software checklist 11](#_Toc33462766)

[Software setup instructions 12](#_Toc33462767)

[Section 4 Source codes 12](#_Toc33462768)

[server.py 12](#_Toc33462769)

[Index.html 16](#_Toc33462770)

[WMS.py 40](#_Toc33462771)

[WMS2.py 44](#_Toc33462772)

[FMS.py 45](#_Toc33462773)

[FMS2.py 48](#_Toc33462774)

[Section 5 Task List 51](#_Toc33462775)

[Section 6 Settings up the IOT Core 52](#_Toc33462776)

[Section 7 Setting up IOT Security Policy 54](#_Toc33462777)

[Section 8 Attaching Things to Security Policies 55](#_Toc33462778)

[Section 9 Setting up AWS Roles 56](#_Toc33462779)

[Section 10 Creating Database for values from IOT 58](#_Toc33462780)

[1 Both with Device ID as Primary Key and datetime as sort key. 59](#_Toc33462781)

[Section 11 Setting IoT Rules to connect to DynamoDB 59](#_Toc33462782)

[2 Setting IoT Rules to connect to DynamoDB 59](#_Toc33462783)

[Section 12 Create S3 Bucket for Uploading Images 62](#_Toc33462784)

[3 Create S3 Bucket for Uploading Images 62](#_Toc33462785)

[4 Section 13 Lamda Setup 63](#_Toc33462786)

[a) Lamda Source Code 64](#_Toc33462787)

[Section 14 Rekognition 65](#_Toc33462788)

[Section 15 EC2 Instance 68](#_Toc33462789)

[Section 16 References 72](#_Toc33462790)

# Section 1 Overview of project

* 1. Where we have uploaded our tutorial

Fill up the Google form here to submit your links and then paste the links here of your Youtube and tutorial document here as well.

<http://bit.ly/1910s2iotca2>

|  |  |
| --- | --- |
| **Youtube** | https://youtu.be/bcoFMjM4mas |
| **Public tutorial link** | <https://github.com/rainyraina/Parrot-Monitoring-System> |

* 1. What is the application about?

This application is called the“Parrot Monitoring System” which monitors the conditions of a bird cage. It helps bird owners maintain the desired living conditions for his/her parrot(s).

Functions:  
1. Works as an alert system that informs the user about any unusual event.

* Insufficient water
* Temperature being too low
* Humidity being too high/low
  + Temperature and humidity is important during breeding season when there are eggs/hatchlings.

2. User is able to control the motor water pump to refill the water cup via the web application.

3. User is able to control the food dispenser to refill the food bowl via the web application

Sensors used:

* + 1. Ultrasonic sensor: Measures the water level of the water bowl.
    2. DHT sensor: Records the temperature and humidity of the cage.
    3. PIR sensor and Picamera: Detects motion of at the food/water bowl and takes a picture. Amazon Rekognition will identify the bird.
    4. TDS sensor: monitors the cleanliness of the water
    5. Water temperature sensor: monitors the temperature of the water. Lower temperatures promote algae growth, which is undesirable

Actuators used:

* + 1. Water Pump: Refills the water bowl upon clicking the “refill” button on web.
    2. Food dispenser: Refills the food bowl upon clicking the “refill” button on web.
  1. How does the final RPI set-up looks like?

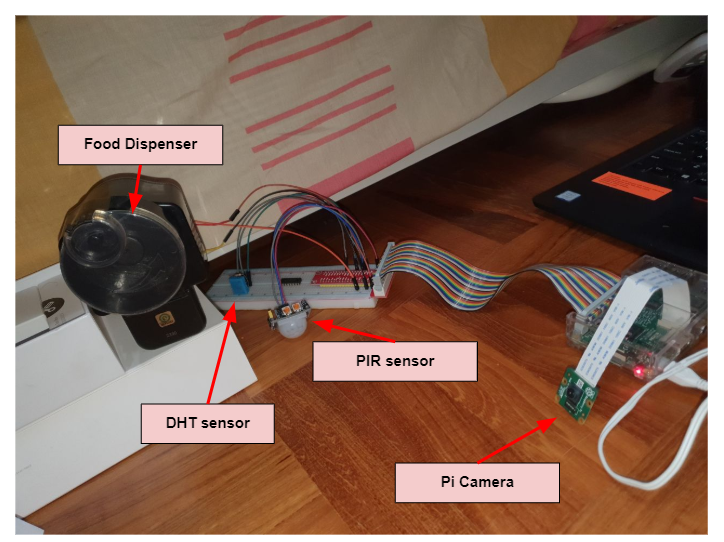


Figure 1: Raspberry Pi #1 Food Monitoring Station

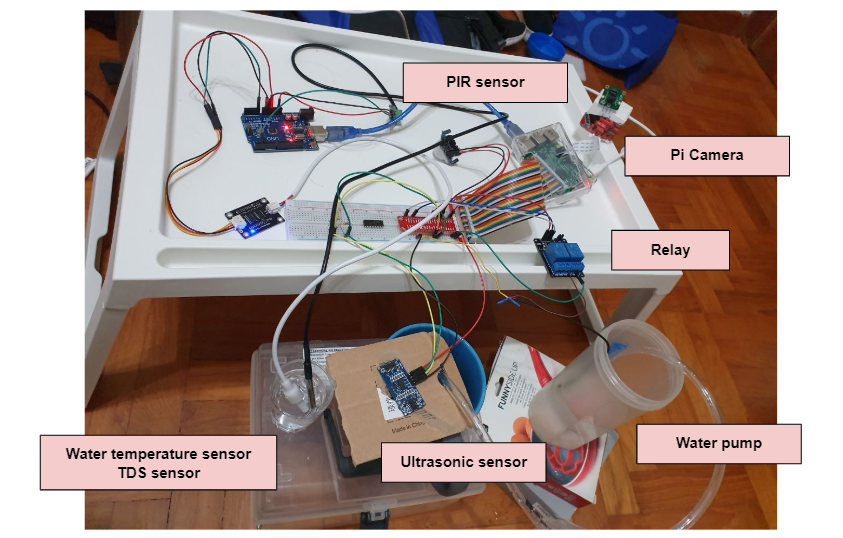
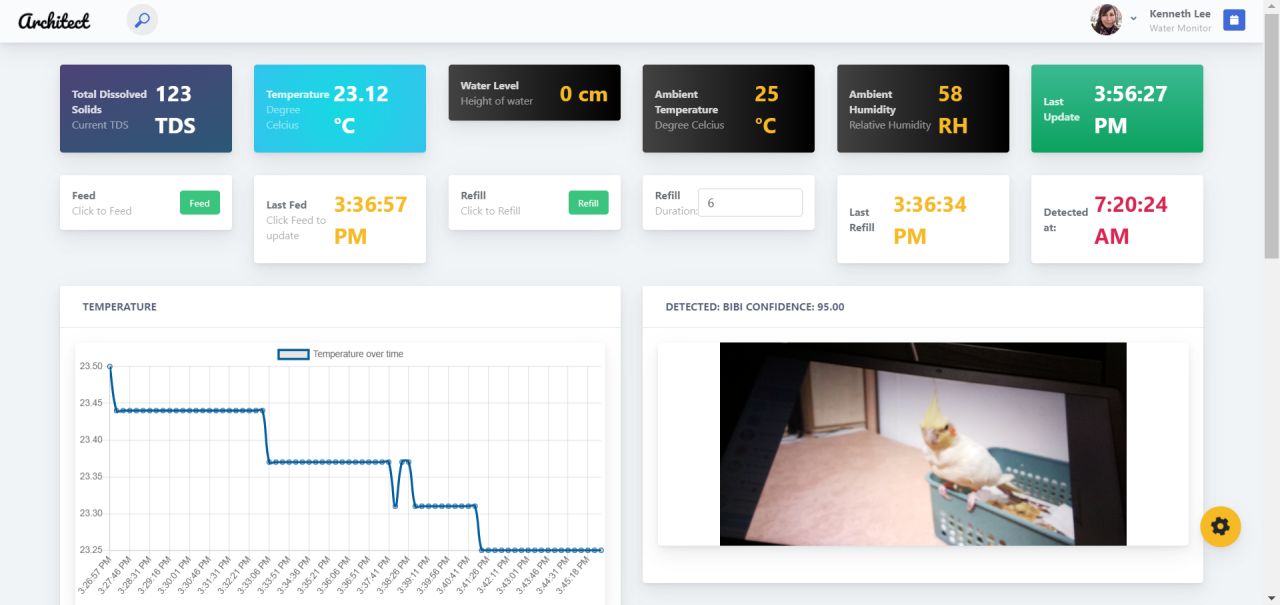
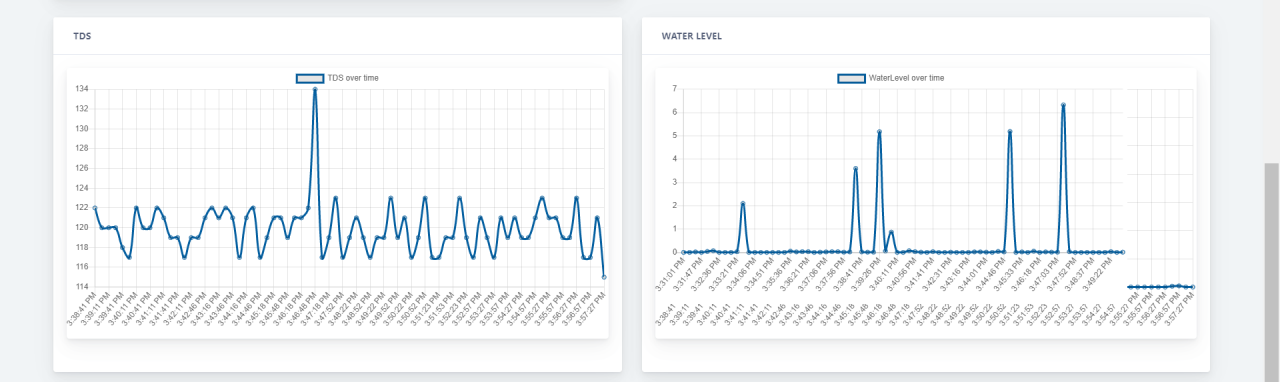


Figure 2: Raspberry Pi #2 Water Monitoring Station

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





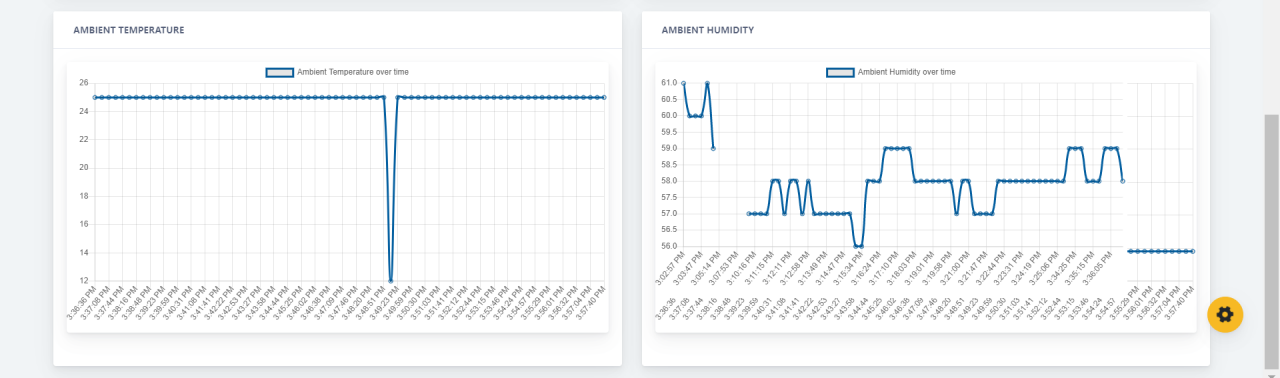


Figure 3: Web Application

* 1. System architecture of our system

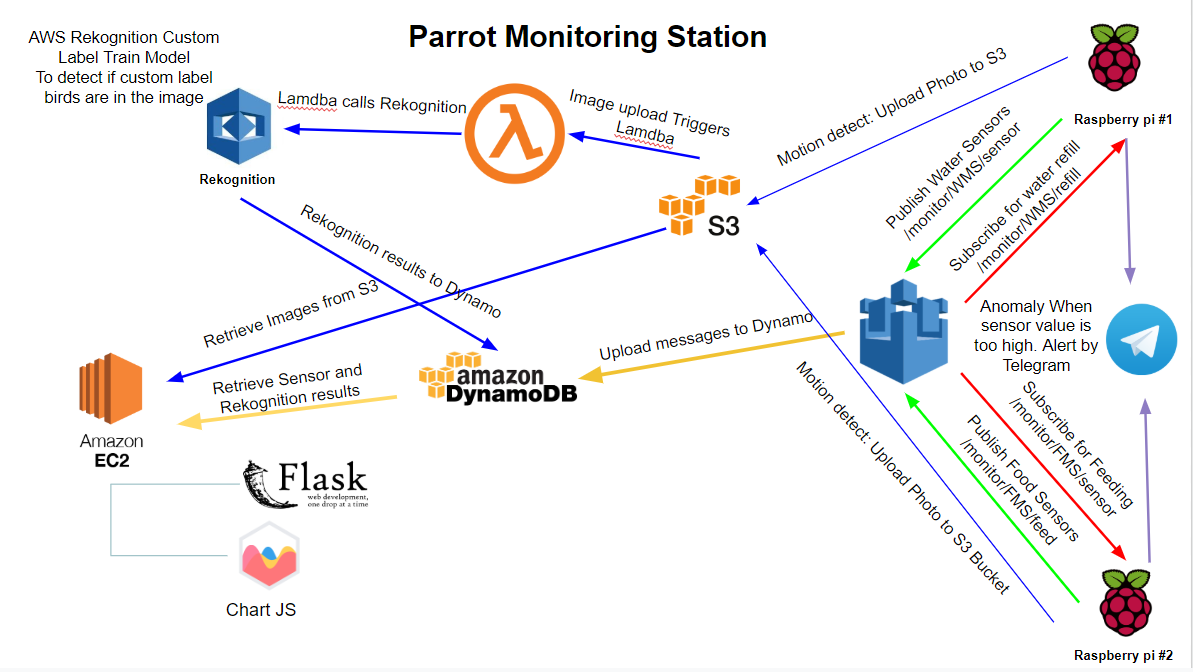


Figure 4: System Architecture

* 1. Evidence that we have met basic requirements

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

|  |  |
| --- | --- |
| Requirement | Evidence |
| Used three sensors | 1. Ultrasonic sensor  2. DHT sensor  3. Motion Sensor  Additional: motion sensor, TDS sensor, water temperature sensor  Refer to section C Figure 1 and 2 |
| Used MQTT | Our MQTT endpoint -->  /monitor/FMS/sensor  /monitor/FMS/feed  /monitor/WMS/sensor  /monitor/WMS/refill  Example of data sent through MQTT :  /monitor/WMS/sensor: Water Temperature,  TDS, Water Level  /monitor/FMS/sensor: Ambient Temperature,  Ambient Humidity |
| Stored data in cloud | Stored all sensors and images in Dynamodb as well as S3 Bucket |
| Used cloud service | Use AWS Rekognition, hosted web server on EC2, Lamda, S3, DynamoDB |
| Provide real-time sensor value / status | Show the real-time value of Ultrasonic sensor, DHT sensor, Motion sensor, TDS Sensor.  Refer to Section D Figure 3 |
| Provide historical sensor value/ status | Show historical value of TDS, Water temperature, ultrasonic, DHT sensors. Refer to Section D Figure 3 |
| Control actuator | Placed button on webpage to control water pump (using DC motor pump) and food dispenser (food dispenser) |

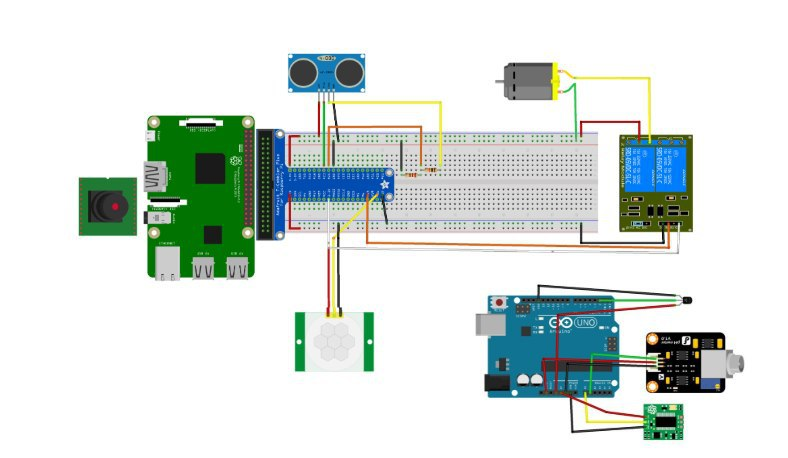
* 1. Telegram Notification System
* The telegram bot will alert the user when
  + Water bowl needs to be refilled
  + TDS above 600
  + Water temperature < 45°C or Water temperature > 18°C
  + Ambient temperature < 45°C or Water temperature > 18°C
  + Ambient Humidity below 30%
  1. Bonus features on top of basic requirements

Provide bullet list of the bonus features you have added on top of basic requirements

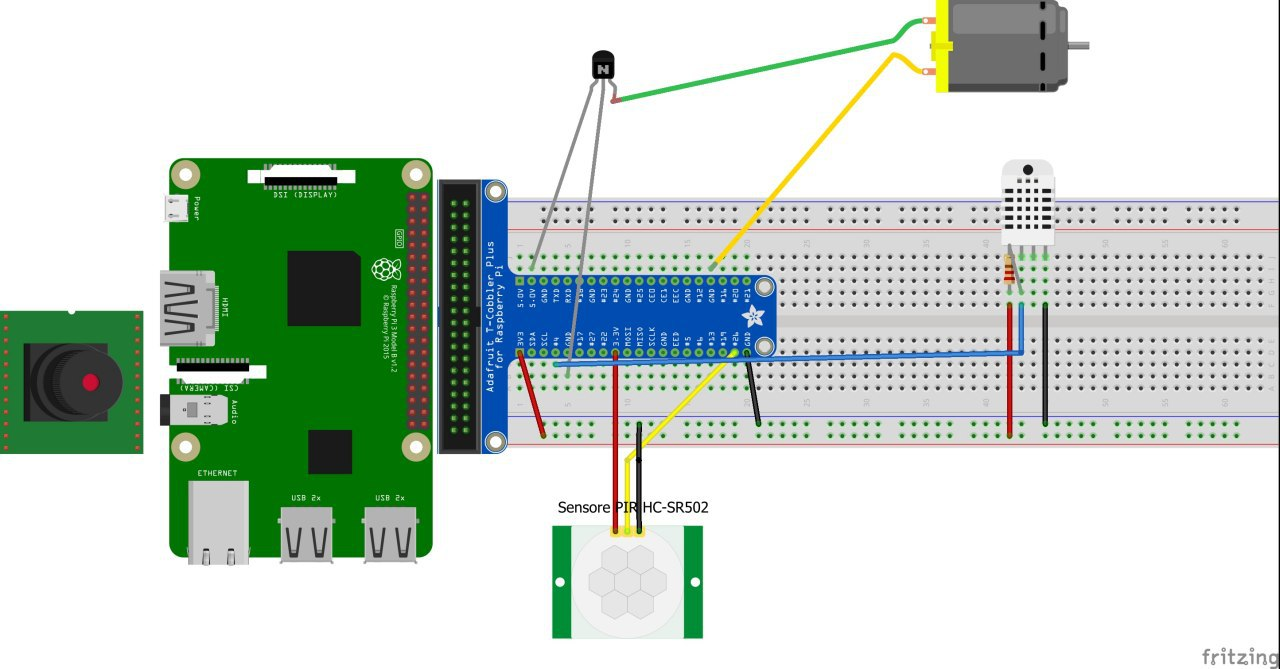
1. Using Rekongition custom label to create training model to detect Raina’s birds (Bibi)
2. Trigger Lamdba when image is uploaded to S3 bucket. Lamdba to call Rekongition and upload results to DynamoDB
3. Running Web Server on EC2.
   1. Quick-start guide (Readme first)

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

1. First connect hardware as shown below. This is for Water Monitoring Station



Raspberry pi for Food Monitoring Station



Raspberry pi for Food monitoring system

1. Collect aws credentials and add into .aws/credentials for raspberry pi and EC2
2. Run WMS.py (for Publishing sensor values) WMS2.py (for Subscribing to refill water) on Water Monitoring Station
3. Run FMS.py (for Publishing sensor values) and FMS2.py (for Subscribing to feed) on Food Monitoring Station
4. On EC2 in app directory, run Server.py
5. Browse to **ec2-3-86-16-93.compute-1.amazonaws.com:5000** for the webapplication
6. Or go to 192.168.1.59:5000 for local host and not running on EC2

# Section 2 Hardware requirements

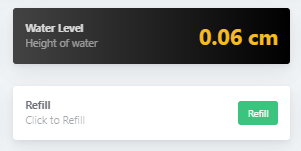
Hardware checklist

Sensors used:

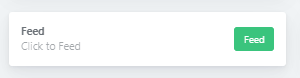
* + 1. Ultrasonic sensor: Measures the water level of the water bowl.
    2. DHT sensor: Records the temperature and humidity of the cage.
    3. 2 PIR sensors and Picameras: Detects motion of at the food/water bowl and takes a picture. Amazon Rekognition will identify the bird.
    4. TDS sensor: monitors the cleanliness of the water
    5. Water temperature sensor: monitors the temperature of the water. Lower temperatures promote algae growth, which is undesirable

Actuators used:

* + 1. Water Pump: Refills the water bowl upon clicking the “refill” button on web.



* + 1. Food dispenser: Refills the food bowl upon clicking the “feed” button on web.



Hardware setup instructions

Additional Hardware (Not provided by school):

1. TDS sensor
2. Water Temperature sensor
3. Ultrasonic sensor
4. Food dispenser
5. 2 relay module
6. Motor water pump

**Hardware Required for Food Monitoring Station:**

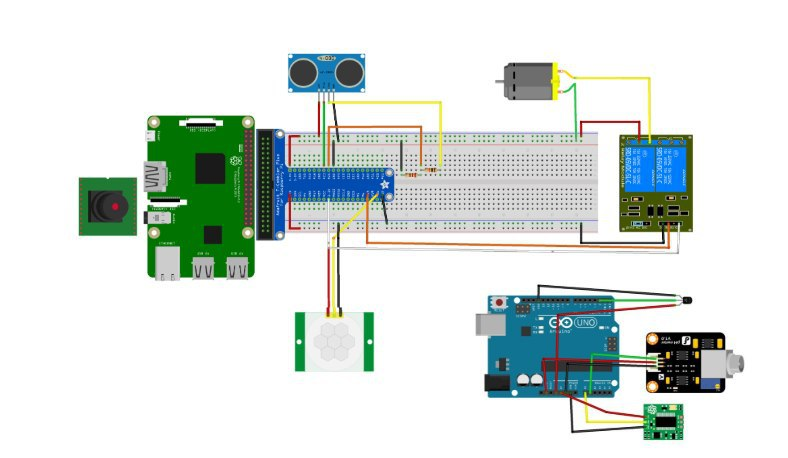
1. Raspberry pi and pi camera
2. Pir sesnor
3. DHT sensor
4. 10K resistor
5. Food dispenser

**Hardware Required for Water Monitoring Station:**

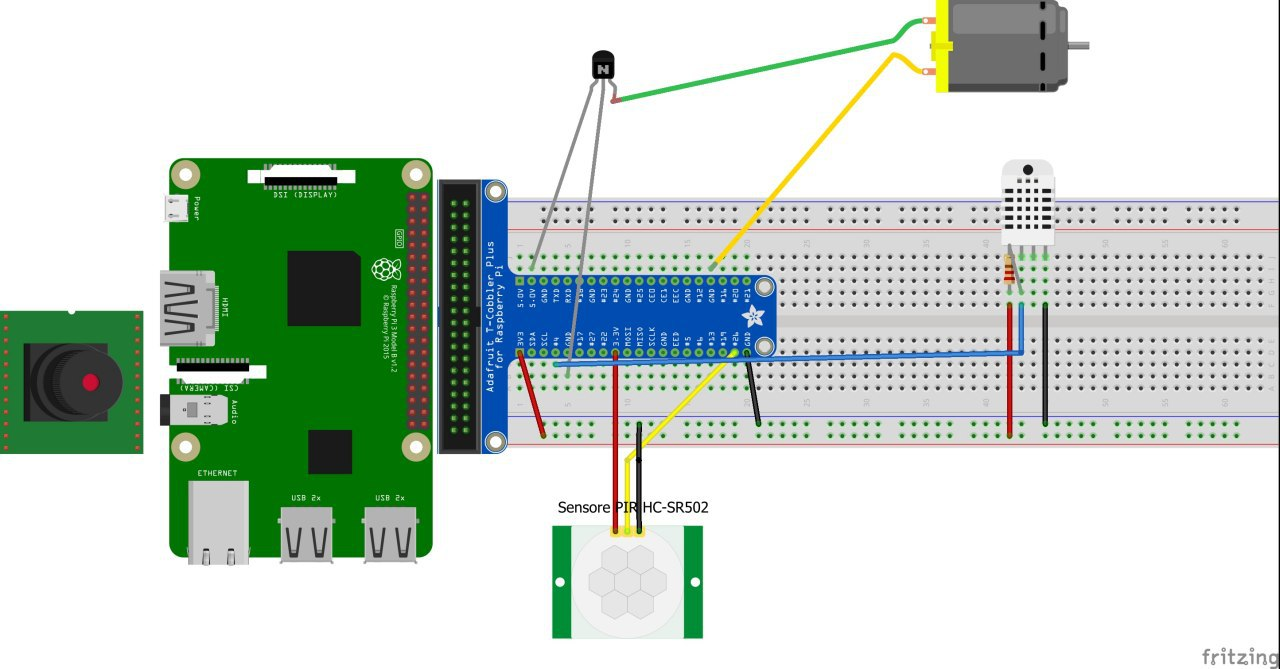
1. Raspberry pi and pi camera
2. arduino
3. Pir sensor
4. TDS sensor
5. Water temperature sensor
6. Ultrasonic sensor
7. Water motor pump
8. 10K & 220 ohm resistor

Fritzing Diagram

Fritzing Diagram to be provided along with the documents



Raspberry pi for water monitoring system



Raspberry pi for Food monitoring system

# Section 3 Software Requirements

Software checklist

1. WMS.py and FMS.py
   1. import serial
   2. import time
   3. from time import sleep
   4. import datetime
   5. import picamera
   6. import Adafruit\_DHT
   7. from gpiozero import MotionSensor
   8. import RPi.GPIO as GPIO
   9. from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
   10. import json
   11. import datetime as datetime
   12. import boto3
2. Server.py
   1. from app import app
   2. from flask import Flask, flash, redirect, render\_template, request, session, abort, jsonify
   3. import gevent
   4. import gevent.monkey
   5. import mysql.connector
   6. import RPi.GPIO as GPIO
   7. import time
   8. import base64
   9. import numpy
   10. import boto3
   11. from boto3.dynamodb.conditions import Key, Attr
   12. import sys
   13. import json
   14. import decimal
   15. from decimal import Decimal
   16. import boto3
   17. from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
   18. import datetime as datetime
   19. app = Flask(\_\_name\_\_)
   20. from gevent.pywsgi import WSGIServer
   21. gevent.monkey.patch\_all()

Software setup instructions

sudo pip install --upgrade --force-reinstall pip==9.0.3

sudo pip install AWSIoTPythonSDK --upgrade --disable-pip-version-check

sudo pip install --upgrade pip

sudo pip install awscli

sudo pip install boto3

sudo pip install botocore

# 

# Section 4 Source codes

All source codes, including Python, HTML files etc

### server.py

|  |
| --- |
| #!/usr/bin/python3  #from app import app  from flask import Flask, flash, redirect, render\_template, request, session, abort, jsonify  import gevent  import gevent.monkey  import time  import base64  import numpy  from boto3.dynamodb.conditions import Key, Attr  import sys  import json  import decimal  from decimal import Decimal  import boto3  import botocore  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  import datetime as datetime  app = Flask(\_\_name\_\_)  from gevent.pywsgi import WSGIServer  gevent.monkey.patch\_all()    #MQTT  host = "a1hdsuh2fiycts-ats.iot.us-east-1.amazonaws.com"  rootCAPath = "Server/AmazonRootCA1.pem"  certificatePath = "Server/certificate.pem.crt"  privateKeyPath = "Server/private.pem.key"    my\_rpi = AWSIoTMQTTClient("basicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)    my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  my\_rpi.connect()    def get\_latest\_file\_name(bucket\_name,prefix):  """  Return the latest file name in an S3 bucket folder.    :param bucket: Name of the S3 bucket.  :param prefix: Only fetch keys that start with this prefix (folder name).  """  s3\_client = boto3.client('s3')  objs = s3\_client.list\_objects\_v2(Bucket=bucket\_name)['Contents']  shortlisted\_files = dict()  for obj in objs:  key = obj['Key']  timestamp = obj['LastModified']  # if key starts with folder name retrieve that key  if key.startswith(prefix):  # Adding a new key value pair  shortlisted\_files.update( {key : timestamp} )  latest\_filename = max(shortlisted\_files, key=shortlisted\_files.get)  return latest\_filename    class GenericEncoder(json.JSONEncoder):  def default(self, obj):  if isinstance(obj, numpy.generic):  return numpy.asscalar(obj)  elif isinstance(obj, Decimal):  return str(obj)  elif isinstance(obj, datetime.datetime):  return obj.strftime('%Y-%m-%d %H:%M:%S')  elif isinstance(obj, Decimal):  return float(obj)  elif isinstance(obj, decimal.Decimal):  return (str(object) for object in [obj])  else:  return json.JSONEncoder.default(self, obj)  def data\_to\_json(data):  json\_data = json.dumps(data, cls=GenericEncoder)  return json\_data      def getdynamodb(tablename,DeviceID):  try:  dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')  table = dynamodb.Table(tablename)  startdate = '2020-02'  response = table.query(  KeyConditionExpression=Key('DeviceID').eq(DeviceID) & Key('datetime').begins\_with(startdate),ScanIndexForward=False)  items = response['Items']  items.reverse()  return (data\_to\_json(items))  except:  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])    def getdynamodb2():  try:  dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')  table = dynamodb.Table("Photodetect")  startdate = '2020-02'  response = table.query(  KeyConditionExpression=Key('ID').eq("1") & Key('datetime').begins\_with(startdate), ScanIndexForward=False,Limit=1)  items = response['Items']  return (data\_to\_json(items))  except:  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])    @app.route('/api/latestphoto', methods=['GET','POST'])  def latestphoto():  photoupload= getdynamodb2()  photouploadjson= json.loads(photoupload)  values=[]  for x in photouploadjson:  values.append(x['DeviceID'])  values.append(x['filename'])  values.append(x['datetime'])  values.append(x['label'])  values.append(x['confidence'])  print(values)  BUCKET\_NAME = 'uploaded-images-for-rekognition'  s3 = boto3.client('s3')  try:  s3.download\_file(BUCKET\_NAME, values[1], 'image.jpg')  with open("image.jpg", "rwb") as image\_file:  image = base64.b64encode(image\_file.read())  photo ={  "DeviceID": values[0],  "filename" : values[1],  "datetime": values[2],  "label": values[3],  "confidence": values[4],  "image" : image,  }  return jsonify(photo)    except botocore.exceptions.ClientError as e:  if e.response['Error']['Code'] == "404":  print("The object does not exist.")  else:  raise    @app.route('/')  @app.route('/index')  def index():  return render\_template('index.html')  @app.route('/api/WMSsensors', methods=['GET','POST'])  def WMSsensors():  data = getdynamodb("WaterMonitoringStation","WMS1")  return data    @app.route('/api/FMSsensors', methods=['GET','POST'])  def FMSsensors():  data = getdynamodb("FoodMonitoringStation","FMS1")  return data  @app.route('/api/updatefeed', methods=['GET','POST'])  def updatefeed():  data = getdynamodb("FeedingStation","FMS1")  return data  @app.route('/api/updaterefill', methods=['GET','POST'])  def refillstation():  data = getdynamodb("RefillStation","WMS1")  return data  @app.route('/api/refill', methods=['GET','POST'])  def publishmqttrefill():  value = request.args.get('value')  print(value)  message = {}  message["DeviceID"] = "WMS1"  now = datetime.datetime.now()  message["datetime"] = now.isoformat()  message["refill"] = value  my\_rpi.publish("monitor/WMS/refill", json.dumps(message), 1)  return(value)    @app.route('/api/feed', methods=['GET','POST'])  def publishmqttfeed():  message = {}  message["DeviceID"] = "FMS1"  now = datetime.datetime.now()  message["datetime"] = now.isoformat()  message["feed"] = "1"  my\_rpi.publish("monitor/FMS/feed", json.dumps(message), 1)  return("feed")  if \_\_name\_\_ == '\_\_main\_\_':  try:  print('Server waiting for requests')  http\_server = WSGIServer(('0.0.0.0', 5000), app)  app.debug = True  http\_server.serve\_forever()  except:  print("Exception")  import sys  print(sys.exc\_info()[0])  print(sys.exc\_info()[1]) |

### Index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta http-equiv="Content-Language" content="en">    <title>Analytics Dashboard - This is an example dashboard created using build-in elements and components.</title>  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no, shrink-to-fit=no">  <meta name="description" content="This is an example dashboard created using build-in elements and components.">  <meta name="msapplication-tap-highlight" content="no">  <link href="./static/main.css" rel="stylesheet"><style type="text/css">/\* Chart.js \*/  @-webkit-keyframes chartjs-render-animation{from{opacity:0.99}to{opacity:1}}@keyframes chartjs-render-animation{from{opacity:0.99}to{opacity:1}}.chartjs-render-monitor{-webkit-animation:chartjs-render-animation 0.001s;animation:chartjs-render-animation 0.001s;}</style>  <!--<script type="text/javascript" charset="UTF-8" src="./static/common.js.download"></script>  <script type="text/javascript" charset="UTF-8" src="./static/util.js.download"></script> -->  <!--<script src="/static/sensor.js"></script> -->  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>  <script [src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script](mailto:src=%22https://cdn.jsdelivr.net/npm/chart.js@2.8.0%22%3e%3c/script)>  <script>  function live(){  WMS()  FMS()  updatefeed()  updaterefill()  alert()  updatefeed()  setTimeout(live,20000);  };  </script>  <script>  function WMS(){  $.post("/api/WMSsensors",  function(newdata,status){  var tdslist = [];  var datetimelist= [];  var watertemplist =[]  var deviceidlist = []  var waterlevellist = []  // Adding the data in the rows  var newdata = JSON.parse(newdata);  for (i in newdata) {  datetime = newdata[i].datetime;  datetime = datetime.substring(0, 19)  jsdatetime = new Date(Date.parse(datetime));  jstime = jsdatetime.toLocaleTimeString();  tdsvalue = (newdata[i].TDS)  waterlevel = (newdata[i].WaterLevel)  watertemp = (newdata[i].WaterTemp)  datetimelist.push(jstime);  tdslist.push(tdsvalue);  waterlevellist.push(waterlevel)  watertemplist.push(watertemp)  }  var latesttds = tdslist[tdslist.length - 1];  var latestwaterlevel = waterlevellist[waterlevellist.length - 1];  var latestwatertemp = watertemplist[watertemplist.length - 1];  var latesttime = datetimelist[datetimelist.length - 1];  document.getElementById("TDS").innerHTML=latesttds +" TDS";  document.getElementById("PH").innerHTML=latestwaterlevel+" cm";  document.getElementById("Time").innerHTML=latesttime;  document.getElementById("Temp").innerHTML=latestwatertemp+" Â°C";    var ctx = document.getElementById('canvas3').getContext('2d');  var myLineChart = new Chart(ctx, {  type: 'line',  data: {  labels: datetimelist,  datasets: [{  label: 'TDS over time',  data: tdslist,  "fill": false,  "borderColor":"rgb(5, 96, 161)",  }],    },  }  )  var ctx = document.getElementById('canvas1').getContext('2d');  var myLineChart = new Chart(ctx, {  type: 'line',  data: {  labels: datetimelist,  datasets: [{  label: 'Temperature over time',  data: watertemplist,  "fill": false,  "borderColor":"rgb(5, 96, 161)",  }],    },  }  )  var ctx = document.getElementById('canvas2').getContext('2d');  var myLineChart = new Chart(ctx, {  type: 'line',  data: {  labels: datetimelist,  datasets: [{  label: 'WaterLevel over time',  data: waterlevellist,  "fill": false,  "borderColor":"rgb(5, 96, 161)",  }],    },  }  )  }  );  };  </script>  <script>  function FMS(){  $.post("/api/FMSsensors",  function(newdata,status){  var AHlist = [];  var datetimelist= [];  var deviceidlist = []  var ATlist = []  // Adding the data in the rows  var newdata = JSON.parse(newdata);  for (i in newdata) {  datetime = newdata[i].datetime;  datetime = datetime.substring(0, 19)  jsdatetime = new Date(Date.parse(datetime));  jstime = jsdatetime.toLocaleTimeString();  AH = (newdata[i].AH)  AT = (newdata[i].AT)  datetimelist.push(jstime);  AHlist.push(AH);  ATlist.push(AT)  }  var latestat = ATlist[ATlist.length - 1];  var latestah = AHlist[AHlist.length - 1];  document.getElementById("AT").innerHTML=latestat +" Â°C";  document.getElementById("AH").innerHTML=latestah+" RH";  var ctx = document.getElementById('canvas4').getContext('2d');  var myLineChart = new Chart(ctx, {  type: 'line',  data: {  labels: datetimelist,  datasets: [{  label: 'Ambient Temperature over time',  data: ATlist,  "fill": false,  "borderColor":"rgb(5, 96, 161)",  }],    },  }  )  var ctx = document.getElementById('canvas5').getContext('2d');  var myLineChart = new Chart(ctx, {  type: 'line',  data: {  labels: datetimelist,  datasets: [{  label: 'Ambient Humidity over time',  data: AHlist,  "fill": false,  "borderColor":"rgb(5, 96, 161)",  }],    },  }  )    }  );  };  </script>  <script>  function feed(){  $.post("/api/feed",  function(data,status){  if (data=== "feed"){  document.getElementById("feed").innerHTML="Fed";  }  });  setTimeout(updatefeed,3000)  };  </script>  <script>  function updatefeed(){  $.post("/api/updatefeed",  function(data,status){  var newdata = JSON.parse(data);  var datetime = newdata[newdata.length - 1].datetime;  datetime = datetime.substring(0, 19)  jsdatetime = new Date(Date.parse(datetime));  jstime = jsdatetime.toLocaleTimeString();  document.getElementById("updatefeed").innerHTML=jstime;  console.log(jstime);  });  };  </script>  <script>  function refill(){  var number=document.getElementById("value").value;  const regex = RegExp('[0-9]');  if (regex.test(number) === true){  $.get('/api/refill', {value: number},  function(returnedData){  console.log(returnedData);  });  setTimeout(updaterefill,3000)  }  else {  alert("Please enter a single digit");  }  };  </script>  <script>  function updaterefill(){  $.post("/api/updaterefill",  function(data,status){  var newdata = JSON.parse(data);  var refill = newdata[newdata.length - 1].refill;  var datetime = newdata[newdata.length - 1].datetime;  datetime = datetime.substring(0, 19)  jsdatetime = new Date(Date.parse(datetime));  jstime = jsdatetime.toLocaleTimeString();  document.getElementById("lastrefill").innerHTML=jstime;  });  };  </script>  <script>  function alert(){  $.post("/api/latestphoto",  function(anomaly,status){  var date = anomaly.datetime  datetime = date.substring(0, 19)  jsdatetime = new Date(Date.parse(datetime));  jstime = jsdatetime.toLocaleTimeString();  document.getElementById("anomalytext").innerHTML = "Detected: "+anomaly.label + " Confidence: " + anomaly.confidence;  //document.getElementById("anomalytext2").innerHTML = anomaly.filename;  document.getElementById("anomalydate").innerHTML = jstime;  document.getElementById("image").src="data:image/jpeg;base64,"+anomaly.image;  });  };  </script>    </head>  <body onload="live()">  <div class="app-container app-theme-white body-tabs-shadow fixed-sidebar fixed-header closed-sidebar-mobile closed-sidebar">  <div class="app-header header-shadow">  <div class="app-header\_\_logo">  <div class="logo-src"></div>  <div class="header\_\_pane ml-auto">  <div>  <button type="button" class="hamburger close-sidebar-btn hamburger--elastic" data-class="closed-sidebar">  <span class="hamburger-box">  <span class="hamburger-inner"></span>  </span>  </button>  </div>  </div>  </div>  <div class="app-header\_\_mobile-menu">  <div>  <button type="button" class="hamburger hamburger--elastic mobile-toggle-nav">  <span class="hamburger-box">  <span class="hamburger-inner"></span>  </span>  </button>  </div>  </div>  <div class="app-header\_\_menu">  <span>  <button type="button" class="btn-icon btn-icon-only btn btn-primary btn-sm mobile-toggle-header-nav">  <span class="btn-icon-wrapper">  <i class="fa fa-ellipsis-v fa-w-6"></i>  </span>  </button>  </span>  </div> <div class="app-header\_\_content">  <div class="app-header-left">  <div class="search-wrapper">  <div class="input-holder">  <input type="text" class="search-input" placeholder="Type to search">  <button class="search-icon"><span></span></button>  </div>  <button class="close"></button>  </div>  <ul class="header-menu nav">  <button type="button" class="btn btn-success" id="update" onclick="live()" hidden='hidden' >Update</button>  <h6 id="status"> </h6>  </ul>  </div>  <div class="app-header-right">  <div class="header-btn-lg pr-0">  <div class="widget-content p-0">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="btn-group">  <a data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" class="p-0 btn">  <img width="42" class="rounded-circle" src="./static/assets/images/avatars/1.jpg" alt="">  <i class="fa fa-angle-down ml-2 opacity-8"></i>  </a>  <div tabindex="-1" role="menu" aria-hidden="true" class="dropdown-menu dropdown-menu-right">  <button type="button" tabindex="0" class="dropdown-item">User Account</button>  <button type="button" tabindex="0" class="dropdown-item">Settings</button>  <h6 tabindex="-1" class="dropdown-header">Header</h6>  <button type="button" tabindex="0" class="dropdown-item">Actions</button>  <div tabindex="-1" class="dropdown-divider"></div>  <button type="button" tabindex="0" class="dropdown-item">Dividers</button>  </div>  </div>  </div>  <div class="widget-content-left ml-3 header-user-info">  <div class="widget-heading">  Kenneth Lee  </div>  <div class="widget-subheading">  Water Monitor  </div>  </div>  <div class="widget-content-right header-user-info ml-3">  <button type="button" class="btn-shadow p-1 btn btn-primary btn-sm show-toastr-example">  <i class="fa text-white fa-calendar pr-1 pl-1"></i>  </button>  </div>  </div>  </div>  </div> </div>  </div>  </div> <div class="ui-theme-settings">  <button type="button" id="TooltipDemo" class="btn-open-options btn btn-warning">  <i class="fa fa-cog fa-w-16 fa-spin fa-2x"></i>  </button>  <div class="theme-settings\_\_inner">  <div class="scrollbar-container ps ps--active-y">  <div class="theme-settings\_\_options-wrapper">  <h3 class="themeoptions-heading">Layout Options  </h3>  <div class="p-3">  <ul class="list-group">  <li class="list-group-item">  <div class="widget-content p-0">  <div class="widget-content-wrapper">  <div class="widget-content-left mr-3">  <div class="switch has-switch switch-container-class" data-class="fixed-header">  <div class="switch-animate switch-on">  <input type="checkbox" checked="" data-toggle="toggle" data-onstyle="success">  </div>  </div>  </div>  <div class="widget-content-left">  <div class="widget-heading">Fixed Header  </div>  <div class="widget-subheading">Makes the header top fixed, always visible!  </div>  </div>  </div>  </div>  </li>  <li class="list-group-item">  <div class="widget-content p-0">  <div class="widget-content-wrapper">  <div class="widget-content-left mr-3">  <div class="switch has-switch switch-container-class" data-class="fixed-sidebar">  <div class="switch-animate switch-on">  <input type="checkbox" checked="" data-toggle="toggle" data-onstyle="success">  </div>  </div>  </div>  <div class="widget-content-left">  <div class="widget-heading">Fixed Sidebar  </div>  <div class="widget-subheading">Makes the sidebar left fixed, always visible!  </div>  </div>  </div>  </div>  </li>  <li class="list-group-item">  <div class="widget-content p-0">  <div class="widget-content-wrapper">  <div class="widget-content-left mr-3">  <div class="switch has-switch switch-container-class" data-class="fixed-footer">  <div class="switch-animate switch-off">  <input type="checkbox" data-toggle="toggle" data-onstyle="success">  </div>  </div>  </div>  <div class="widget-content-left">  <div class="widget-heading">Fixed Footer  </div>  <div class="widget-subheading">Makes the app footer bottom fixed, always visible!  </div>  </div>  </div>  </div>  </li>  </ul>  </div>  <h3 class="themeoptions-heading">  <div>  Header Options  </div>  <button type="button" class="btn-pill btn-shadow btn-wide ml-auto btn btn-focus btn-sm switch-header-cs-class" data-class="">  Restore Default  </button>  </h3>  <div class="p-3">  <ul class="list-group">  <li class="list-group-item">  <h5 class="pb-2">Choose Color Scheme  </h5>  <div class="theme-settings-swatches">  <div class="swatch-holder bg-primary switch-header-cs-class" data-class="bg-primary header-text-light">  </div>  <div class="swatch-holder bg-secondary switch-header-cs-class" data-class="bg-secondary header-text-light">  </div>  <div class="swatch-holder bg-success switch-header-cs-class" data-class="bg-success header-text-dark">  </div>  <div class="swatch-holder bg-info switch-header-cs-class" data-class="bg-info header-text-dark">  </div>  <div class="swatch-holder bg-warning switch-header-cs-class" data-class="bg-warning header-text-dark">  </div>  <div class="swatch-holder bg-danger switch-header-cs-class" data-class="bg-danger header-text-light">  </div>  <div class="swatch-holder bg-light switch-header-cs-class" data-class="bg-light header-text-dark">  </div>  <div class="swatch-holder bg-dark switch-header-cs-class" data-class="bg-dark header-text-light">  </div>  <div class="swatch-holder bg-focus switch-header-cs-class" data-class="bg-focus header-text-light">  </div>  <div class="swatch-holder bg-alternate switch-header-cs-class" data-class="bg-alternate header-text-light">  </div>  <div class="divider">  </div>  <div class="swatch-holder bg-vicious-stance switch-header-cs-class" data-class="bg-vicious-stance header-text-light">  </div>  <div class="swatch-holder bg-midnight-bloom switch-header-cs-class" data-class="bg-midnight-bloom header-text-light">  </div>  <div class="swatch-holder bg-night-sky switch-header-cs-class" data-class="bg-night-sky header-text-light">  </div>  <div class="swatch-holder bg-slick-carbon switch-header-cs-class" data-class="bg-slick-carbon header-text-light">  </div>  <div class="swatch-holder bg-asteroid switch-header-cs-class" data-class="bg-asteroid header-text-light">  </div>  <div class="swatch-holder bg-royal switch-header-cs-class" data-class="bg-royal header-text-light">  </div>  <div class="swatch-holder bg-warm-flame switch-header-cs-class" data-class="bg-warm-flame header-text-dark">  </div>  <div class="swatch-holder bg-night-fade switch-header-cs-class" data-class="bg-night-fade header-text-dark">  </div>  <div class="swatch-holder bg-sunny-morning switch-header-cs-class" data-class="bg-sunny-morning header-text-dark">  </div>  <div class="swatch-holder bg-tempting-azure switch-header-cs-class" data-class="bg-tempting-azure header-text-dark">  </div>  <div class="swatch-holder bg-amy-crisp switch-header-cs-class" data-class="bg-amy-crisp header-text-dark">  </div>  <div class="swatch-holder bg-heavy-rain switch-header-cs-class" data-class="bg-heavy-rain header-text-dark">  </div>  <div class="swatch-holder bg-mean-fruit switch-header-cs-class" data-class="bg-mean-fruit header-text-dark">  </div>  <div class="swatch-holder bg-malibu-beach switch-header-cs-class" data-class="bg-malibu-beach header-text-light">  </div>  <div class="swatch-holder bg-deep-blue switch-header-cs-class" data-class="bg-deep-blue header-text-dark">  </div>  <div class="swatch-holder bg-ripe-malin switch-header-cs-class" data-class="bg-ripe-malin header-text-light">  </div>  <div class="swatch-holder bg-arielle-smile switch-header-cs-class" data-class="bg-arielle-smile header-text-light">  </div>  <div class="swatch-holder bg-plum-plate switch-header-cs-class" data-class="bg-plum-plate header-text-light">  </div>  <div class="swatch-holder bg-happy-fisher switch-header-cs-class" data-class="bg-happy-fisher header-text-dark">  </div>  <div class="swatch-holder bg-happy-itmeo switch-header-cs-class" data-class="bg-happy-itmeo header-text-light">  </div>  <div class="swatch-holder bg-mixed-hopes switch-header-cs-class" data-class="bg-mixed-hopes header-text-light">  </div>  <div class="swatch-holder bg-strong-bliss switch-header-cs-class" data-class="bg-strong-bliss header-text-light">  </div>  <div class="swatch-holder bg-grow-early switch-header-cs-class" data-class="bg-grow-early header-text-light">  </div>  <div class="swatch-holder bg-love-kiss switch-header-cs-class" data-class="bg-love-kiss header-text-light">  </div>  <div class="swatch-holder bg-premium-dark switch-header-cs-class" data-class="bg-premium-dark header-text-light">  </div>  <div class="swatch-holder bg-happy-green switch-header-cs-class" data-class="bg-happy-green header-text-light">  </div>  </div>  </li>  </ul>  </div>  <h3 class="themeoptions-heading">  <div>Sidebar Options</div>  <button type="button" class="btn-pill btn-shadow btn-wide ml-auto btn btn-focus btn-sm switch-sidebar-cs-class" data-class="">  Restore Default  </button>  </h3>  <div class="p-3">  <ul class="list-group">  <li class="list-group-item">  <h5 class="pb-2">Choose Color Scheme  </h5>  <div class="theme-settings-swatches">  <div class="swatch-holder bg-primary switch-sidebar-cs-class" data-class="bg-primary sidebar-text-light">  </div>  <div class="swatch-holder bg-secondary switch-sidebar-cs-class" data-class="bg-secondary sidebar-text-light">  </div>  <div class="swatch-holder bg-success switch-sidebar-cs-class" data-class="bg-success sidebar-text-dark">  </div>  <div class="swatch-holder bg-info switch-sidebar-cs-class" data-class="bg-info sidebar-text-dark">  </div>  <div class="swatch-holder bg-warning switch-sidebar-cs-class" data-class="bg-warning sidebar-text-dark">  </div>  <div class="swatch-holder bg-danger switch-sidebar-cs-class" data-class="bg-danger sidebar-text-light">  </div>  <div class="swatch-holder bg-light switch-sidebar-cs-class" data-class="bg-light sidebar-text-dark">  </div>  <div class="swatch-holder bg-dark switch-sidebar-cs-class" data-class="bg-dark sidebar-text-light">  </div>  <div class="swatch-holder bg-focus switch-sidebar-cs-class" data-class="bg-focus sidebar-text-light">  </div>  <div class="swatch-holder bg-alternate switch-sidebar-cs-class" data-class="bg-alternate sidebar-text-light">  </div>  <div class="divider">  </div>  <div class="swatch-holder bg-vicious-stance switch-sidebar-cs-class" data-class="bg-vicious-stance sidebar-text-light">  </div>  <div class="swatch-holder bg-midnight-bloom switch-sidebar-cs-class" data-class="bg-midnight-bloom sidebar-text-light">  </div>  <div class="swatch-holder bg-night-sky switch-sidebar-cs-class" data-class="bg-night-sky sidebar-text-light">  </div>  <div class="swatch-holder bg-slick-carbon switch-sidebar-cs-class" data-class="bg-slick-carbon sidebar-text-light">  </div>  <div class="swatch-holder bg-asteroid switch-sidebar-cs-class" data-class="bg-asteroid sidebar-text-light">  </div>  <div class="swatch-holder bg-royal switch-sidebar-cs-class" data-class="bg-royal sidebar-text-light">  </div>  <div class="swatch-holder bg-warm-flame switch-sidebar-cs-class" data-class="bg-warm-flame sidebar-text-dark">  </div>  <div class="swatch-holder bg-night-fade switch-sidebar-cs-class" data-class="bg-night-fade sidebar-text-dark">  </div>  <div class="swatch-holder bg-sunny-morning switch-sidebar-cs-class" data-class="bg-sunny-morning sidebar-text-dark">  </div>  <div class="swatch-holder bg-tempting-azure switch-sidebar-cs-class" data-class="bg-tempting-azure sidebar-text-dark">  </div>  <div class="swatch-holder bg-amy-crisp switch-sidebar-cs-class" data-class="bg-amy-crisp sidebar-text-dark">  </div>  <div class="swatch-holder bg-heavy-rain switch-sidebar-cs-class" data-class="bg-heavy-rain sidebar-text-dark">  </div>  <div class="swatch-holder bg-mean-fruit switch-sidebar-cs-class" data-class="bg-mean-fruit sidebar-text-dark">  </div>  <div class="swatch-holder bg-malibu-beach switch-sidebar-cs-class" data-class="bg-malibu-beach sidebar-text-light">  </div>  <div class="swatch-holder bg-deep-blue switch-sidebar-cs-class" data-class="bg-deep-blue sidebar-text-dark">  </div>  <div class="swatch-holder bg-ripe-malin switch-sidebar-cs-class" data-class="bg-ripe-malin sidebar-text-light">  </div>  <div class="swatch-holder bg-arielle-smile switch-sidebar-cs-class" data-class="bg-arielle-smile sidebar-text-light">  </div>  <div class="swatch-holder bg-plum-plate switch-sidebar-cs-class" data-class="bg-plum-plate sidebar-text-light">  </div>  <div class="swatch-holder bg-happy-fisher switch-sidebar-cs-class" data-class="bg-happy-fisher sidebar-text-dark">  </div>  <div class="swatch-holder bg-happy-itmeo switch-sidebar-cs-class" data-class="bg-happy-itmeo sidebar-text-light">  </div>  <div class="swatch-holder bg-mixed-hopes switch-sidebar-cs-class" data-class="bg-mixed-hopes sidebar-text-light">  </div>  <div class="swatch-holder bg-strong-bliss switch-sidebar-cs-class" data-class="bg-strong-bliss sidebar-text-light">  </div>  <div class="swatch-holder bg-grow-early switch-sidebar-cs-class" data-class="bg-grow-early sidebar-text-light">  </div>  <div class="swatch-holder bg-love-kiss switch-sidebar-cs-class" data-class="bg-love-kiss sidebar-text-light">  </div>  <div class="swatch-holder bg-premium-dark switch-sidebar-cs-class" data-class="bg-premium-dark sidebar-text-light">  </div>  <div class="swatch-holder bg-happy-green switch-sidebar-cs-class" data-class="bg-happy-green sidebar-text-light">  </div>  </div>  </li>  </ul>  </div>  <h3 class="themeoptions-heading">  <div>Main Content Options</div>  <button type="button" class="btn-pill btn-shadow btn-wide ml-auto active btn btn-focus btn-sm">Restore Default  </button>  </h3>  <div class="p-3">  <ul class="list-group">  <li class="list-group-item">  <h5 class="pb-2">Page Section Tabs  </h5>  <div class="theme-settings-swatches">  <div role="group" class="mt-2 btn-group">  <button type="button" class="btn-wide btn-shadow btn-primary btn btn-secondary switch-theme-class" data-class="body-tabs-line">  Line  </button>  <button type="button" class="btn-wide btn-shadow btn-primary active btn btn-secondary switch-theme-class" data-class="body-tabs-shadow">  Shadow  </button>  </div>  </div>  </li>  </ul>  </div>  </div>  <div class="ps\_\_rail-x" style="left: 0px; bottom: 0px;"><div class="ps\_\_thumb-x" tabindex="0" style="left: 0px; width: 0px;"></div></div><div class="ps\_\_rail-y" style="top: 0px; height: 754px; right: 0px;"><div class="ps\_\_thumb-y" tabindex="0" style="top: 0px; height: 528px;"></div></div></div>  </div>  </div>  <div class="app-main">  <div class="app-main\_\_outer">  <div class="app-main\_\_inner">  </div>  <div class="row">  <div class="col-md-5 col-xl-2">  <div class="card mb-3 widget-content bg-midnight-bloom">  <div class="widget-content-wrapper text-white">  <div class="widget-content-left">  <div class="widget-heading">Total Dissolved Solids</div>  <div class="widget-subheading">Current TDS</div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-white"><span id="TDS"></span></div>  </div>  </div>  </div>  </div>  <div class="col-md-5 col-xl-2">  <div class="card mb-3 widget-content bg-arielle-smile">  <div class="widget-content-wrapper text-white">  <div class="widget-content-left">  <div class="widget-heading">Temperature</div>  <div class="widget-subheading">Degree Celcius</div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-white"><span id="Temp"> </span></div>  </div>  </div>  </div>  </div>  <div class="col-md-5 col-xl-2">  <div class="card mb-3 widget-content bg-premium-dark">  <div class="widget-content-wrapper text-white">  <div class="widget-content-left">  <div class="widget-heading">Water Level</div>  <div class="widget-subheading">Height of water</div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-warning"><span id="PH"></span></div>  </div>  </div>  </div>  </div>  <div class="col-md-5 col-xl-2">  <div class="card mb-3 widget-content bg-premium-dark">  <div class="widget-content-wrapper text-white">  <div class="widget-content-left">  <div class="widget-heading">Ambient Temperature</div>  <div class="widget-subheading">Degree Celcius</div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-warning"><span id="AT"></span></div>  </div>  </div>  </div>  </div>  <div class="col-md-5 col-xl-2">  <div class="card mb-3 widget-content bg-premium-dark">  <div class="widget-content-wrapper text-white">  <div class="widget-content-left">  <div class="widget-heading">Ambient Humidity</div>  <div class="widget-subheading">Relative Humidity</div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-warning"><span id="AH"></span></div>  </div>  </div>  </div>  </div>  <div class="col-md-5 col-xl-2">  <div class="card mb-3 widget-content bg-grow-early">  <div class="widget-content-wrapper text-white">  <div class="widget-content-left">  <div class="widget-heading">Last Update</div>  <div class="widget-subheading"></div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-white"><span id="Time"></span></div>  </div>  </div>  </div>  </div>  </div>  <div class="row">  <div class="col-md-6 col-xl-2">  <div class="card mb-3 widget-content">  <div class="widget-content-outer">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="widget-heading">Feed</div>  <div class="widget-subheading" id="feed">Click to Feed</div>  </div>  <div class="widget-content-right">  <button type="button" class="btn btn-success" onclick="feed()" >Feed</button>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-6 col-xl-2">  <div class="card mb-3 widget-content">  <div class="widget-content-outer">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="widget-heading">Last Fed</div>  <div class="widget-subheading">Click Feed to update</div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-warning" id="updatefeed"></div>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-6 col-xl-2">  <div class="card mb-3 widget-content">  <div class="widget-content-outer">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="widget-heading">Refill</div>  <div class="widget-subheading" id="feed">Click to Refill</div>  </div>  <div class="widget-content-right">  <button type="button" class="btn btn-success" onclick="refill()" >Refill</button>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-6 col-xl-2">  <div class="card mb-3 widget-content">  <div class="widget-content-outer">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="widget-heading">Refill</div>  <div class="widget-subheading">Duration:</div>  </div>  <div class="widget-content-right">  <input type="text" class="form-control" pattern="[0-9]" id="value">  </div>  </div>  </div>  </div>  </div>  <div class="col-md-6 col-xl-2">  <div class="card mb-3 widget-content">  <div class="widget-content-outer">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="widget-heading">Last Refill</div>  <div class="widget-subheading"></div>  </div>  <div class="widget-content-right">  <div class="widget-numbers text-warning" id="lastrefill"></div>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-6 col-xl-2">  <div class="card mb-3 widget-content">  <div class="widget-content-outer">  <div class="widget-content-wrapper">  <div class="widget-content-left">  <div class="widget-heading">Detected at:</div>  <div id="anomalytext2" class="widget-subheading"></div>  </div>  <div class="widget-content-right">  <div id="anomalydate"class="widget-numbers text-danger"></div>  </div>  </div>  </div>  </div>  </div>  </div>  <div class="row">  <div class="col-md-12 col-lg-6">  <div class="mb-3 card">  <div class="card-header-tab card-header-tab-animation card-header">  <div class="card-header-title">  <i class="header-icon lnr-apartment icon-gradient bg-love-kiss"> </i>  Temperature  </div>  </div>  <div class="card-body">  <div class="tab-content">  <div class="tab-pane fade show active" id="tabs-eg-77">  <div class="card mb-3 widget-chart widget-chart2 text-left w-100">  <div class="widget-chat-wrapper-outer">  <div class="widget-chart-wrapper widget-chart-wrapper-lg opacity-10 m-0"><div class="chartjs-size-monitor" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: hidden; pointer-events: none; visibility: hidden; z-index: -1;"><div class="chartjs-size-monitor-expand" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:1000000px;height:1000000px;left:0;top:0"></div></div><div class="chartjs-size-monitor-shrink" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:200%;height:200%;left:0; top:0"></div></div></div>  <canvas id="canvas1" width="542" height="271" class="chartjs-render-monitor" style="display: block; height: 217px; width: 434px;"></canvas>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-12 col-lg-6">  <div class="mb-3 card">  <div class="card-header-tab card-header-tab-animation card-header">  <div class="card-header-title">  <i class="header-icon lnr-apartment icon-gradient bg-love-kiss"> </i>  <div id = "anomalytext"></div>  </div>  </div>  <div class="card-body">  <div class="tab-content">  <div class="tab-pane fade show active" id="tabs-eg-77">  <div class="card mb-3 widget-chart widget-chart2 text-left w-100">  <div class="widget-chat-wrapper-outer">  <div class="widget-chart-wrapper widget-chart-wrapper-lg opacity-10 m-0"><div class="chartjs-size-monitor" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: hidden; pointer-events: none; visibility: hidden; z-index: -1;"><div class="chartjs-size-monitor-expand" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:1000000px;height:1000000px;left:0;top:0"></div></div><div class="chartjs-size-monitor-shrink" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:200%;height:200%;left:0; top:0"></div></div></div>  <img width="542" height="271" id="image" src="" style="display: block; margin-left:auto; margin-right:auto;">  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <div class="row">  <div class="col-md-12 col-lg-6">  <div class="mb-3 card">  <div class="card-header-tab card-header-tab-animation card-header">  <div class="card-header-title">  <i class="header-icon lnr-apartment icon-gradient bg-love-kiss"> </i>  TDS  </div>  </div>  <div class="card-body">  <div class="tab-content">  <div class="tab-pane fade show active" id="tabs-eg-77">  <div class="card mb-3 widget-chart widget-chart2 text-left w-100">  <div class="widget-chat-wrapper-outer">  <div class="widget-chart-wrapper widget-chart-wrapper-lg opacity-10 m-0"><div class="chartjs-size-monitor" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: hidden; pointer-events: none; visibility: hidden; z-index: -1;"><div class="chartjs-size-monitor-expand" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:1000000px;height:1000000px;left:0;top:0"></div></div><div class="chartjs-size-monitor-shrink" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:200%;height:200%;left:0; top:0"></div></div></div>  <canvas id="canvas3" width="542" height="271" class="chartjs-render-monitor" style="display: block; height: 217px; width: 434px;"></canvas>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-12 col-lg-6">  <div class="mb-3 card">  <div class="card-header-tab card-header-tab-animation card-header">  <div class="card-header-title">  <i class="header-icon lnr-apartment icon-gradient bg-love-kiss"> </i>  Water Level  </div>  </div>  <div class="card-body">  <div class="tab-content">  <div class="tab-pane fade show active" id="tabs-eg-77">  <div class="card mb-3 widget-chart widget-chart2 text-left w-100">  <div class="widget-chat-wrapper-outer">  <div class="widget-chart-wrapper widget-chart-wrapper-lg opacity-10 m-0"><div class="chartjs-size-monitor" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: hidden; pointer-events: none; visibility: hidden; z-index: -1;"><div class="chartjs-size-monitor-expand" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:1000000px;height:1000000px;left:0;top:0"></div></div><div class="chartjs-size-monitor-shrink" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:200%;height:200%;left:0; top:0"></div></div></div>  <canvas id="canvas2" width="542" height="271" class="chartjs-render-monitor" style="display: block; height: 217px; width: 434px;"></canvas>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <div class="row">  <div class="col-md-12 col-lg-6">  <div class="mb-3 card">  <div class="card-header-tab card-header-tab-animation card-header">  <div class="card-header-title">  <i class="header-icon lnr-apartment icon-gradient bg-love-kiss"> </i>  Ambient Temperature  </div>  </div>  <div class="card-body">  <div class="tab-content">  <div class="tab-pane fade show active" id="tabs-eg-77">  <div class="card mb-3 widget-chart widget-chart2 text-left w-100">  <div class="widget-chat-wrapper-outer">  <div class="widget-chart-wrapper widget-chart-wrapper-lg opacity-10 m-0"><div class="chartjs-size-monitor" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: hidden; pointer-events: none; visibility: hidden; z-index: -1;"><div class="chartjs-size-monitor-expand" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:1000000px;height:1000000px;left:0;top:0"></div></div><div class="chartjs-size-monitor-shrink" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:200%;height:200%;left:0; top:0"></div></div></div>  <canvas id="canvas4" width="542" height="271" class="chartjs-render-monitor" style="display: block; height: 217px; width: 434px;"></canvas>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <div class="col-md-12 col-lg-6">  <div class="mb-3 card">  <div class="card-header-tab card-header-tab-animation card-header">  <div class="card-header-title">  <i class="header-icon lnr-apartment icon-gradient bg-love-kiss"> </i>  Ambient Humidity  </div>  </div>  <div class="card-body">  <div class="tab-content">  <div class="tab-pane fade show active" id="tabs-eg-77">  <div class="card mb-3 widget-chart widget-chart2 text-left w-100">  <div class="widget-chat-wrapper-outer">  <div class="widget-chart-wrapper widget-chart-wrapper-lg opacity-10 m-0"><div class="chartjs-size-monitor" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: hidden; pointer-events: none; visibility: hidden; z-index: -1;"><div class="chartjs-size-monitor-expand" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:1000000px;height:1000000px;left:0;top:0"></div></div><div class="chartjs-size-monitor-shrink" style="position:absolute;left:0;top:0;right:0;bottom:0;overflow:hidden;pointer-events:none;visibility:hidden;z-index:-1;"><div style="position:absolute;width:200%;height:200%;left:0; top:0"></div></div></div>  <canvas id="canvas5" width="542" height="271" class="chartjs-render-monitor" style="display: block; height: 217px; width: 434px;"></canvas>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <!--</div> -->    <div class="row">  <div class="col-md-12">    </div>  </div>    </div>  </div>  </body>  </html> |

### WMS.py

|  |
| --- |
| import serial  import time  from time import sleep  import datetime  import picamera  import Adafruit\_DHT  from gpiozero import MotionSensor  import RPi.GPIO as GPIO  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  import json  import datetime as datetime  import boto3  import botocore  import requests    # Water Level Sensor  TRIG = 23  ECHO = 24  GPIO.setwarnings(False)  GPIO.setmode(GPIO.BCM)  GPIO.setup(TRIG,GPIO.OUT)  GPIO.setup(ECHO,GPIO.IN)    #Motion Sensor  pir = MotionSensor(17, sample\_rate=5,queue\_len=1)  lastsent = time.time()    #Water Temp and TDS  ser = serial.Serial('/dev/ttyUSB0',9600)    #MQTT  host = "a1hdsuh2fiycts-ats.iot.us-east-1.amazonaws.com"  rootCAPath = "WMS/AmazonRootCA1.pem"  certificatePath = "WMS/certificate.pem.crt"  privateKeyPath = "WMS/private.pem.key"    #Telegram  bot\_token = '960746603:AAFea1wjoRls\_fOz6N624zPKQEBOhgIVCag'  bot\_chatID = '239725290'      my\_rpi = AWSIoTMQTTClient("basicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)    my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(20) # 10 sec  my\_rpi.configureMQTTOperationTimeout(20) # 5 sec  my\_rpi.connect()    def publishmqtt(watertemp,tds,waterlevel):  message = {}  message["DeviceID"] = "WMS1"  now = datetime.datetime.now()  message["datetime"] = now.isoformat()  message["WaterTemp"] = watertemp  message["TDS"] = tds  message["WaterLevel"] = waterlevel  my\_rpi.publish("monitor/WMS/sensor", json.dumps(message), 1)  def motiondetected():  timestring = time.strftime("%Y-%m-%d\_%H\_%M\_%S", time.localtime())  print ("Bird detected at water Bowl at: " +timestring + "\nTaking photo...")  photo = "/home/pi/Desktop/photo/photo\_"+timestring+".jpg"  filename="WMS1\_"+timestring+".jpg"  with picamera.PiCamera() as camera:  camera.capture(photo)  print("photo taken")  uploadToS3(filename,photo)    def uploadToS3(filename,full\_path):  s3 = boto3.resource('s3') # Create an S3 resource  exists = True    try:  s3.meta.client.head\_bucket(Bucket="uploaded-images-for-rekognition")  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="uploaded-images-for-rekognition" ,CreateBucketConfiguration={'LocationConstraint': 'us-east-1'})    # Upload the file  s3.Object("uploaded-images-for-rekognition", filename).put(Body=open(full\_path, 'rb'))  print("File uploaded")  def checkwaterlevel():  GPIO.output(TRIG, False)  GPIO.output(TRIG, True)  time.sleep(0.00001)  GPIO.output(TRIG, False)  pulse\_start = 0  pulse\_end = 0  while GPIO.input(ECHO)==0:  pulse\_start = time.time()  while GPIO.input(ECHO)==1:  pulse\_end = time.time()  pulse\_duration = pulse\_end - pulse\_start #time difference between start and end  dist = (pulse\_duration \* 1280)/2    #To find the water level  distance = 6.6 - dist  if distance > 0:  #print("Water Level: " + str("%.2f" % round(distance,2)))  height=(float("%.2f" % round(distance,2)))  return(height)  else:  height = 0  return(height)    def send(bot\_message):  print("Sending Telegram Message")  send\_text = '<https://api.telegram.org/bot>' + bot\_token + '/sendMessage?chat\_id=' + bot\_chatID + '&parse\_mode=Markdown&text=' + bot\_message  try:  response = requests.get(send\_text)  return response.json()  except:  print("Unable to send to Telegram")  else:  print("Message sent 5 minutes ago")    while True:  read\_serial=ser.readline()  data = read\_serial.split()  Temp = data[0]  TDS = data[1]  print("Temp: "+Temp+"C")  print("TDS: " + TDS)  waterlevel=checkwaterlevel()  print(waterlevel)  publishmqtt(Temp,TDS,str(waterlevel))  noti=[]  if ((Temp > 45) or (Temp < 18)):  noti.append("Water temperature undesirable")  if (TDS > 600):  noti.append("TDS more than 600")  if (waterlevel < 3):  noti.append("Need to refill water bowl")  if (len(noti) != 0):  currenttime = time.time()  difference = currenttime - lastsent  if (difference > 300): # Will send only every 5 minutes  lastsent = time.time() #update last sent  for x in noti:  send(x)  if pir.motion\_detected:  currenttime = time.time()  difference = currenttime - lastsent  if (difference > 300):  motiondetected()  lastsent = time.time()  else:  print("photo already taken!")  sleep(15) |

### WMS2.py

|  |
| --- |
| from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  import RPi.GPIO as GPIO  import time  from time import sleep  import datetime  import json  import datetime as datetime  from multiprocessing import Process    import string, random    def rand\_str\_gen(size=20):  lettersal = ''.join(random.choice(string.ascii\_letters) for i in range(size))  lettersd = ''.join(random.choice(string.digits) for i in range(size))  lettersp = ''.join(random.choice(string.punctuation) for i in range(size))  letter = str(lettersal) + str(lettersd) + str(lettersp)  return ''.join(random.choice(letter) for i in range(size))    #Water pump  GPIO.setwarnings(False)  GPIO.setmode(GPIO.BCM)  GPIO.setup(13,GPIO.OUT)    #MQTT  host = "a1hdsuh2fiycts-ats.iot.us-east-1.amazonaws.com"  rootCAPath = "WMS/AmazonRootCA1.pem"  certificatePath = "WMS/certificate.pem.crt"  privateKeyPath = "WMS/private.pem.key"      def customCallback(client, userdata, message):  payload = json.loads(message.payload)  if payload["DeviceID"] == "WMS1":  refillduration=float(payload["refill"])  refill(refillduration)      def subscribe():  my\_rpi = AWSIoTMQTTClient("basicPubSub" + rand\_str\_gen())  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  my\_rpi.connect()  my\_rpi.subscribe("monitor/WMS/refill", 1, customCallback)  while True:  print("Waiting for call...")  sleep(5)  def refill(refillduration):  GPIO.output(13, 0)  print("Start")  sleep(refillduration)  GPIO.output(13, 1)  print("Stop")  sleep(1)  print("Refilling for: "+ str(refillduration))  return ("water refilled")      if \_\_name\_\_ == '\_\_main\_\_':  subscribe\_proc = Process(name='subscribe', target=subscribe)  subscribe\_proc.start()  subscribe\_proc.join() |

### FMS.py

|  |
| --- |
| import Adafruit\_DHT  from gpiozero import MotionSensor  import RPi.GPIO as GPIO  import picamera  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  import json  import threading  import datetime as datetime  from time import sleep  import time  import boto3  import botocore  import requests    #Motion Sensor  pir = MotionSensor(17, sample\_rate=5,queue\_len=1)  lastsent = time.time()    lastsent1 =time.time()  #Food Feeder  GPIO.setup(18,GPIO.OUT)    #S3 Bucket  BUCKET = 'arn:aws:s3:::uploaded-images-for-rekognition' # replace with your own unique bucket name  location = {'LocationConstraint': 'us-east-1'}      #MQTT  host = "a23gkxvm8h23di-ats.iot.us-east-1.amazonaws.com"  rootCAPath = "FMS/AmazonRootCA1.pem"  certificatePath = "FMS/certificate.pem.crt"  privateKeyPath = "FMS/private.pem.key"    #Telegram  bot\_token = '960746603:AAFea1wjoRls\_fOz6N624zPKQEBOhgIVCag'  bot\_chatID = '239725290'      my\_rpi = AWSIoTMQTTClient("basicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)    my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec    # Connect and subscribe to AWS IoT  my\_rpi.connect()    def publishmqtt(Temperature,Humidity):  message = {}  message["DeviceID"] = "FMS1"  now = datetime.datetime.now()  message["datetime"] = now.isoformat()  message["AT"] = Humidity  message["AH"] = Temperature  my\_rpi.publish("monitor/FMS/sensor", json.dumps(message), 1)  def tempandhum():  temperature, humidity = Adafruit\_DHT.read\_retry(11, 27)  publishmqtt(temperature ,humidity)  return(temperature, humidity)  def motiondetected():  timestring = time.strftime("%Y-%m-%d\_%H\_%M\_%S", time.localtime())  print ("Bird detected at food Bowl at: " +timestring + "\nTaking photo...")  photo = "/home/pi/Desktop/photo/photo\_"+timestring+".jpg"  filename="FMS1\_"+timestring+".jpg"  with picamera.PiCamera() as camera:  camera.capture(photo)  print("photo taken")  uploadToS3(filename,photo)    def uploadToS3(filename,full\_path):  s3 = boto3.resource('s3') # Create an S3 resource  exists = True    try:  s3.meta.client.head\_bucket(Bucket="uploaded-images-for-rekognition")  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="uploaded-images-for-rekognition" ,CreateBucketConfiguration={'LocationConstraint': 'us-east-1'})    # Upload the file  s3.Object("uploaded-images-for-rekognition", filename).put(Body=open(full\_path, 'rb'))  print("File uploaded")    def send(bot\_message):  print("Sending Telegram Message")  send\_text = '<https://api.telegram.org/bot>' + bot\_token + '/sendMessage?chat\_id=' + bot\_chatID + '&parse\_mode=Markdown&text=' + bot\_message  try:  response = requests.get(send\_text)  return response.json()  except:  print("Unable to send to Telegram")  else:  print("Message sent 5 minutes ago")    while True:  temperature, humidity = tempandhum()  print("Temperature:" + str(temperature))  print("Humidity:" + str(humidity))  print(humidity<28)  noti=[]  if ((temperature > 45) or (temperature < 18)):  noti.append("Water temperature undesirable")  if (humidity < 28):  noti.append("Humidity too low.")  if (len(noti) != 0):  currenttime1 = time.time()  difference1 = currenttime1 - lastsent1  print(difference1)  if (difference1 > 10): # Will send only every 5 minutes  lastsent1 = time.time() #update last sent  for x in noti:  send(x)    if pir.motion\_detected:  currenttime = time.time()  difference = currenttime - lastsent  if (difference > 300):  motiondetected()  lastsent = time.time()  else:  print("photo already taken!")  sleep(15) |

### FMS2.py

|  |
| --- |
| from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  import RPi.GPIO as GPIO  import time  from time import sleep  import datetime  import json  import datetime as datetime  from multiprocessing import Process    import string, random    def rand\_str\_gen(size=20):  lettersal = ''.join(random.choice(string.ascii\_letters) for i in range(size))  lettersd = ''.join(random.choice(string.digits) for i in range(size))  lettersp = ''.join(random.choice(string.punctuation) for i in range(size))  letter = str(lettersal) + str(lettersd) + str(lettersp)  return ''.join(random.choice(letter) for i in range(size))    #Food Feeder  GPIO.setwarnings(False)  GPIO.setmode(GPIO.BCM)  GPIO.setup(18,GPIO.OUT)      #MQTT  host = "a23gkxvm8h23di-ats.iot.us-east-1.amazonaws.com"  rootCAPath = "FMS/AmazonRootCA1.pem"  certificatePath = "FMS/certificate.pem.crt"  privateKeyPath = "FMS/private.pem.key"    def customCallback(client, userdata, message):  payload = json.loads(message.payload)  if payload["DeviceID"] == "FMS1":  feed()      def subscribe():  my\_rpi = AWSIoTMQTTClient("basicPubSub" + rand\_str\_gen())  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  my\_rpi.connect()  my\_rpi.subscribe("monitor/FMS/feed", 1, customCallback)  while True:  print("Waiting for call...")  sleep(5)  def feed():  GPIO.output(18, 0)  print("Start")  sleep(1)  GPIO.output(18, 1)  print("Stop")  sleep(1)  print("Fed")  return ("Fed")    if \_\_name\_\_ == '\_\_main\_\_':  subscribe\_proc = Process(name='subscribe', target=subscribe)  subscribe\_proc.start()  subscribe\_proc.join() |

# Section 5 Task List

A table listing members names and the parts of the assignment they worked on

|  |  |  |
| --- | --- | --- |
| Name of member | Part of project worked on | Contribution percentage |
| Kenneth Lee Jia Le | Lamdba, EC2, Water monitoring Station, Dynamodb, Chart JS,  Webserver | 50% |
| Raina Tang Xiu Wen | Rekognition, Food Monitoring Station, IoT Mqtt, S3 Bucket | 50% |
|  |  |  |

# Section 6 Settings up the IOT Core

There are 3 IOT Things we are creating. Bird-WMS-1 and BirdFMS1 belongs to Raspberry Pi. And Server is for EC2 Server. To publish to MQTT when needed to refill or feed. Save the certificate, Public and Private Key repectively in their folders.

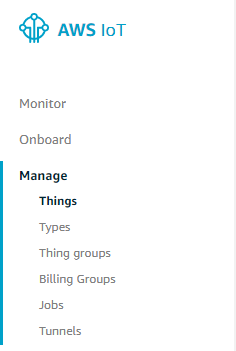
Server --> Server

BirdFMS1 --> FMS

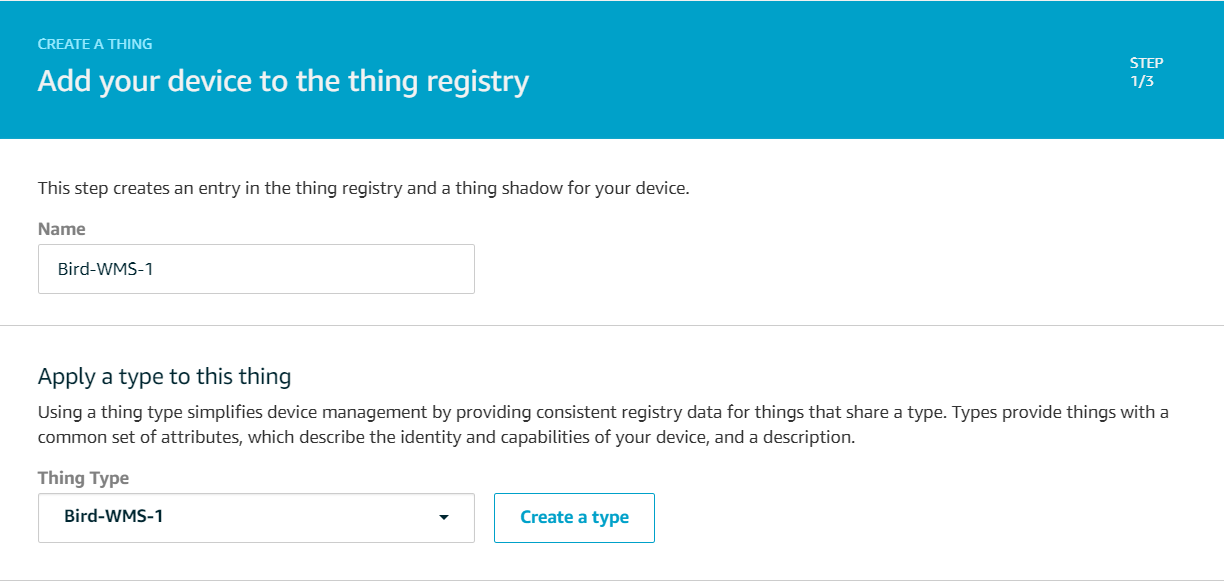
Bird-WMS-1 --> WMS



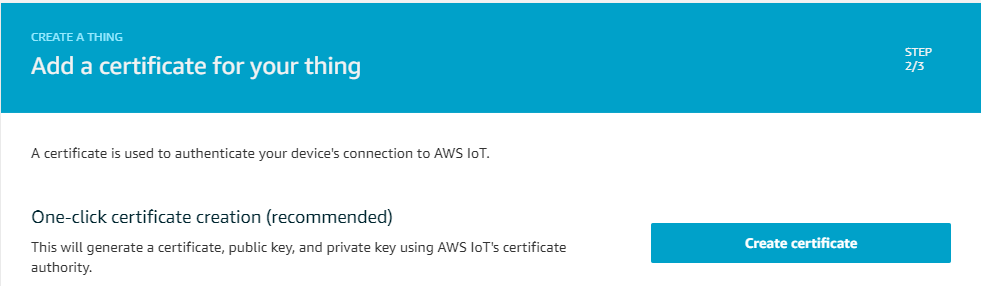
1. Manage > Things



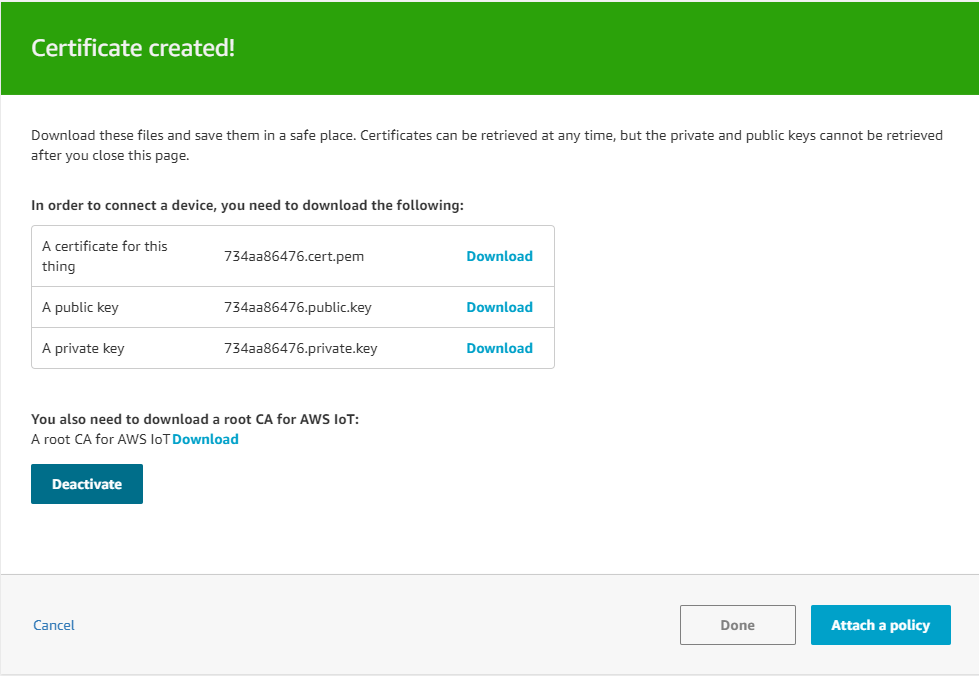
1. Create a Thing in IOTCore



1. Creating IOT Thing Certificate



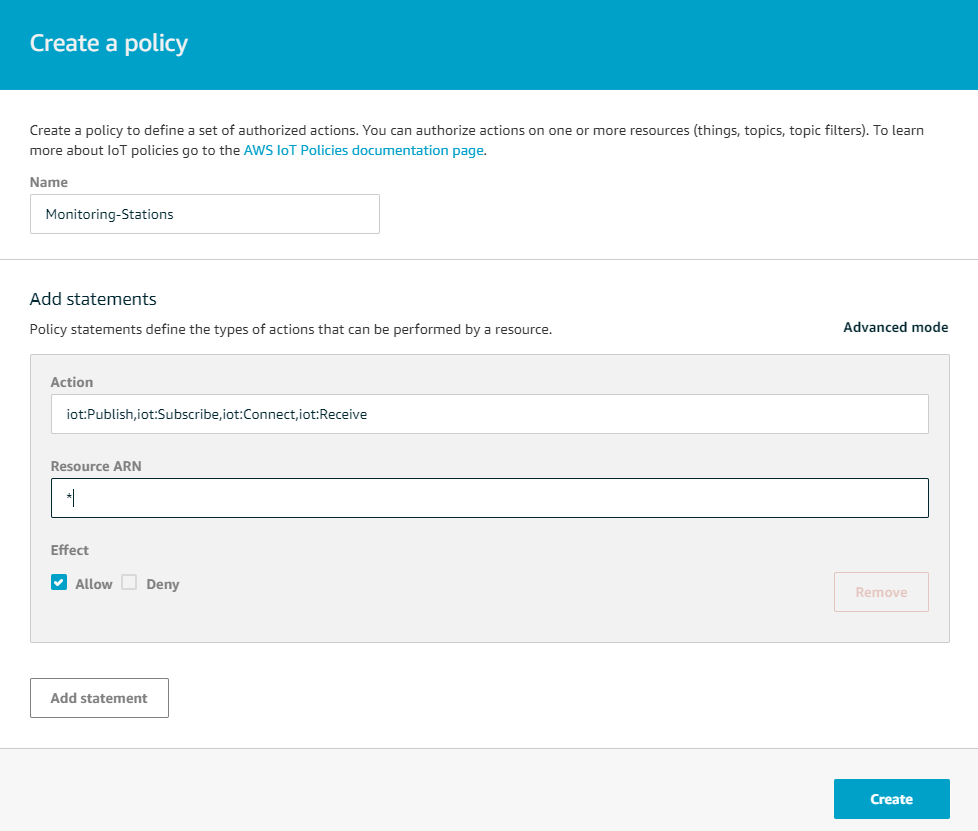
1. Download the Certificate and Activate the Root CA for AWS IOT



Download Certificate, Public Key and Private Key.

# Section 7 Setting up IOT Security Policy

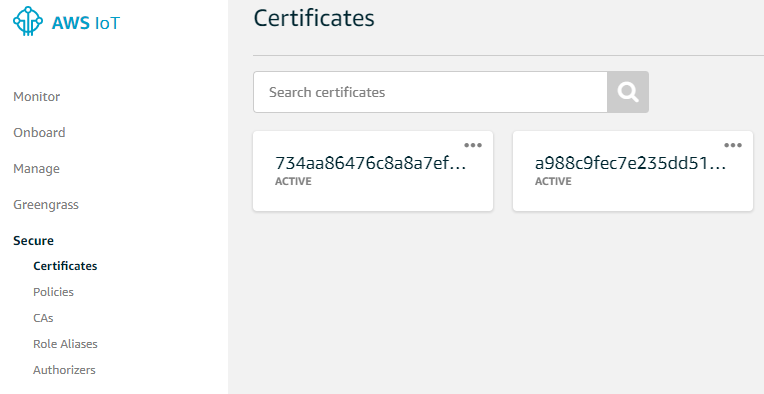
1. Create a Policy



1. Click “Create”

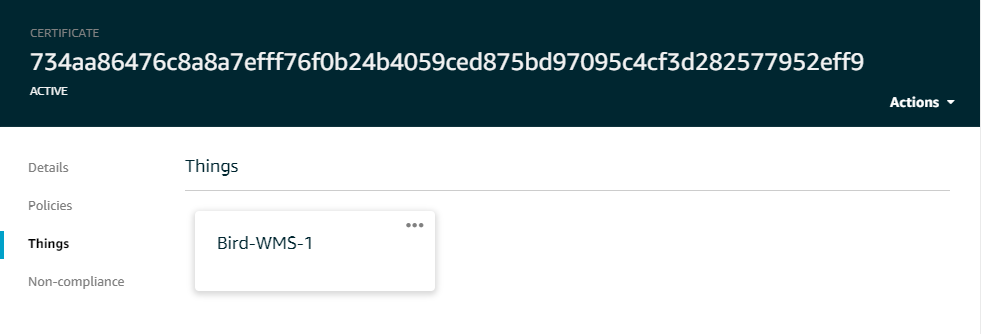


# Section 8 Attaching Things to Security Policies



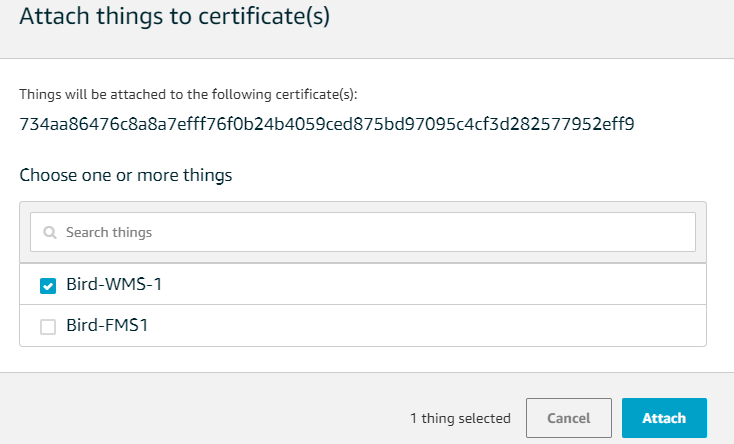
As for now both share the same security policy.

Select Either Certificate and select Things. You will be able to see which Certificate this Thing belongs to.

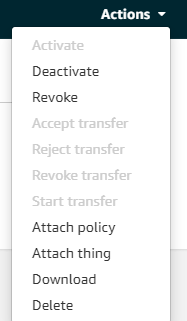


Selection actions and choose Attach things

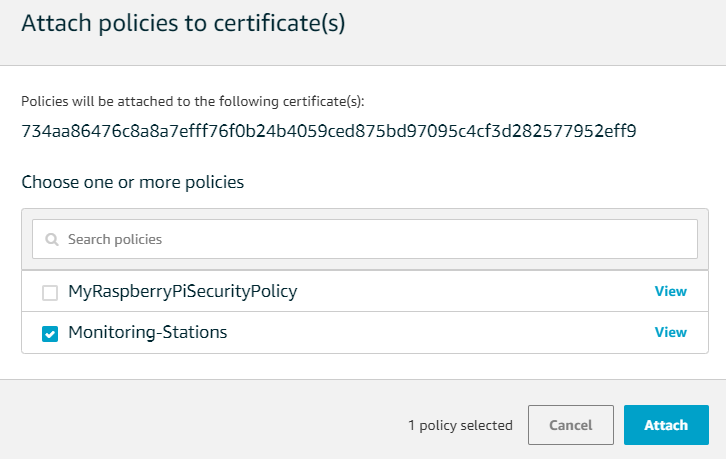
Select the respective things from the Things Tab



Selection actions and choose attach policy



Select the Monitoring-Stations Policy and click Attach



Selection actions and choose Attach things

Select the respective things from the Things Tab

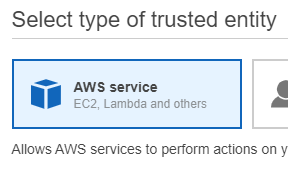
Repeat for the second Certificate with the Bird-FMS1 adn Server

# Section 9 Setting up AWS Roles

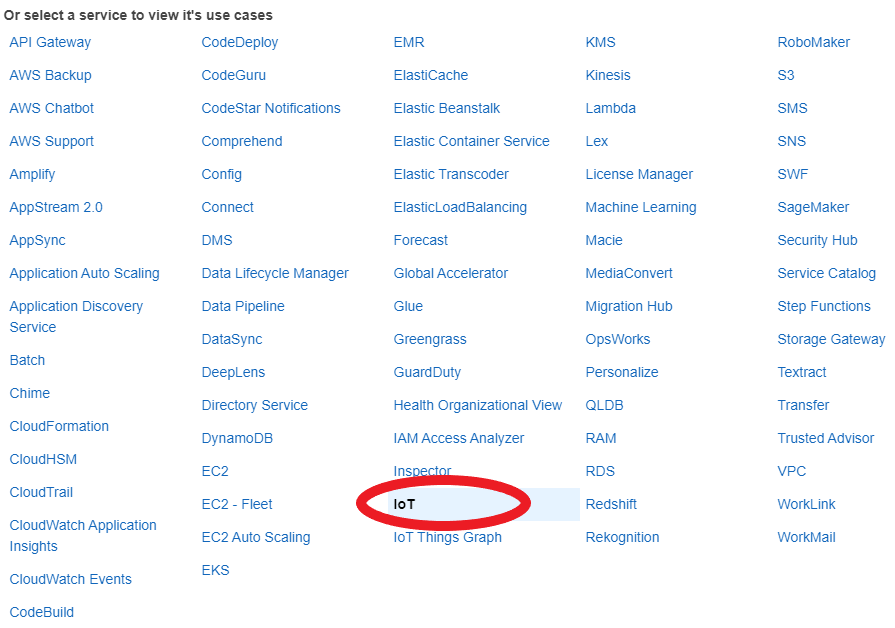
Go to IAM → Roles



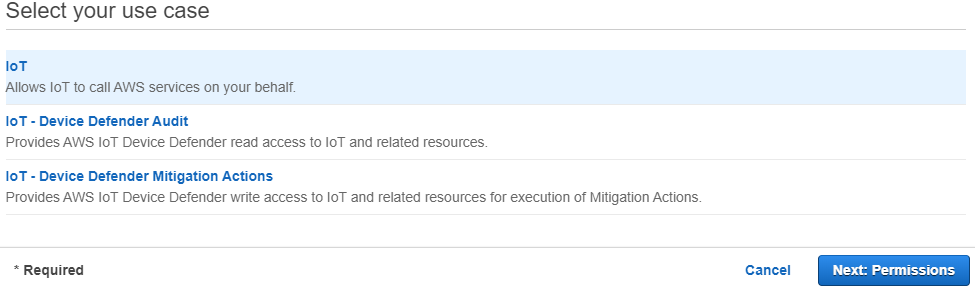
Since we are all using AWS Systems choose AWS Service



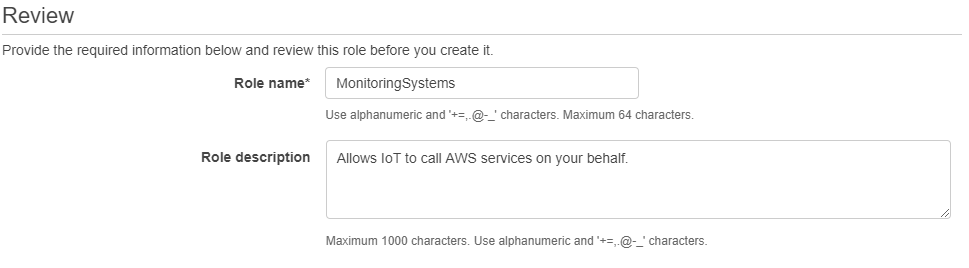
Select IoT



Select Iot  and click Next:Permissions





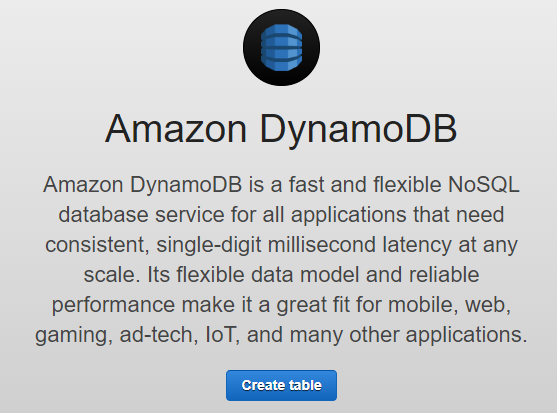


Click Create Role

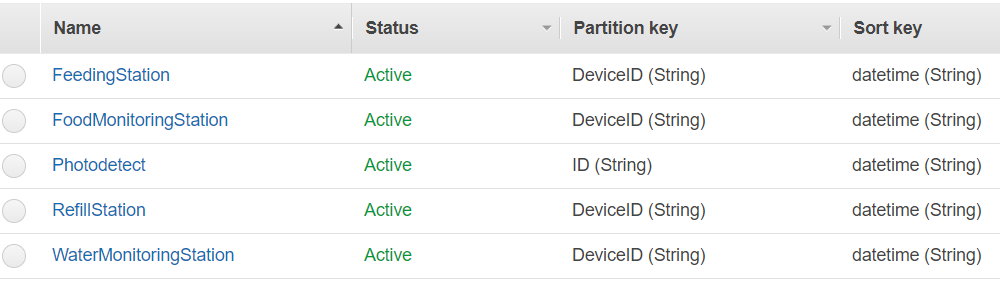


# Section 10 Creating Database for values from IOT

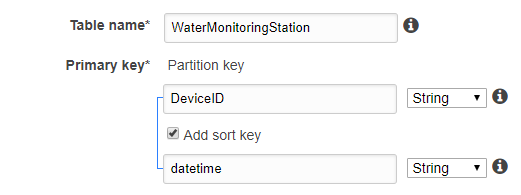
Using DynamoDB. Create Table



Create 4 Tables. FeedingStation, FoodMonitoringStation, RefillStation, WaterMonitoringStation, Photodetect



# Both with Device ID as Primary Key and datetime as sort key.



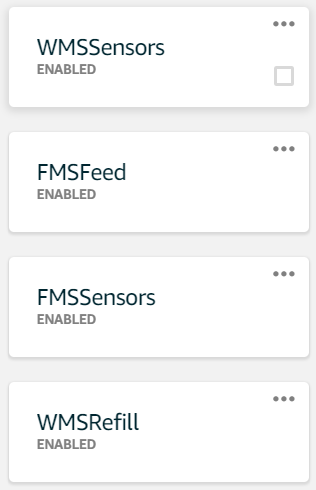


Create the same for all except for Photodetect. The partition key will be ID instead of DeviceID

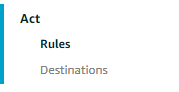
# Section 11 Setting IoT Rules to connect to DynamoDB

# Setting IoT Rules to connect to DynamoDB

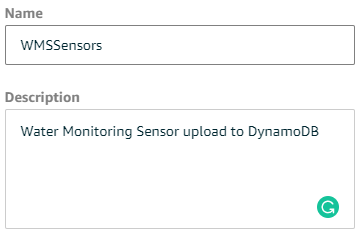
There will be 4 Acts



|  |  |  |  |
| --- | --- | --- | --- |
|  | Rule Query Statement | Tablename | Roles |
| WMSSensors | /monitor/WMS/sensor | WaterMonitoringStation | Monitoring Systems |
| WMSRefill | /monitor/WMS/refill | RefillStation | Monitoring Systems |
| FMSSensors | /monitor/FMS/sensor | FoodMonitoringStation | Monitoring Systems |
| FMSFeed | /monitor/FMS/feed | FeedingStation | Monitoring Systems |



Set up Water Monitoring Sensors



Under Rule Query Statement



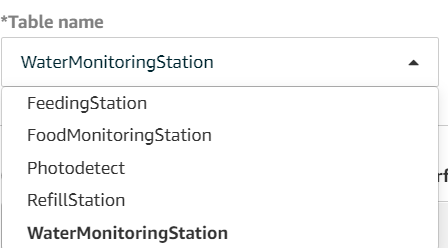
Add actions



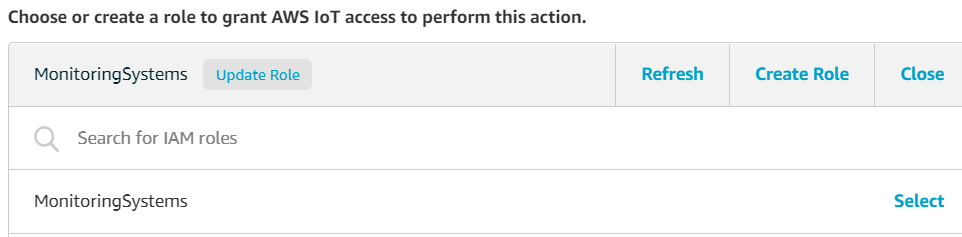




Select Table for this case it will be WaterMonitoring System



Select Roles Montoring Station.





Repeat for the next 3 acts

# Section 12 Create S3 Bucket for Uploading Images

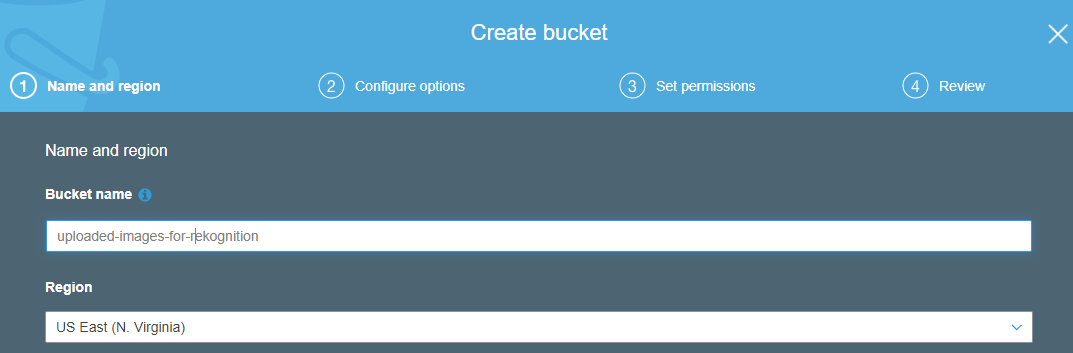
# Create S3 Bucket for Uploading Images

Amazon S3

Create Bucket

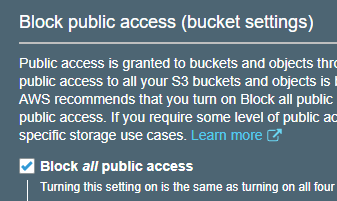


Enter the Bucket name





Block Public Access



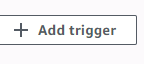
Take note of Bucket name

# Section 13 Lamda Setup

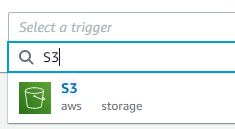
AWS Lamda Create Function



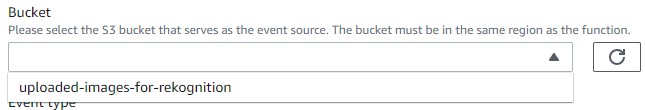
In Designer click Add Trigger



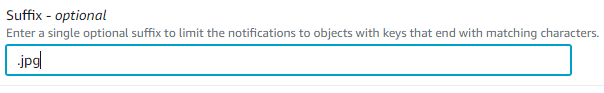
Search For S3



Select the Bucket we created



Suffix, we are looking for jpg images only



Tick Enable Trigger



 And Add

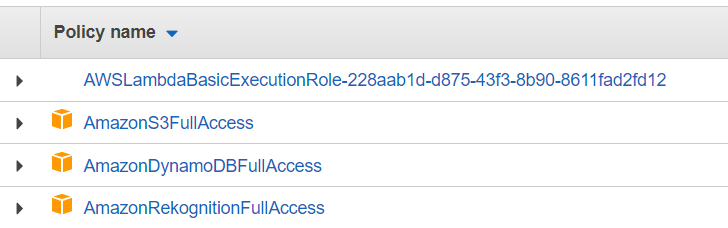


To give Lamda permission to access for S3,Rekonigition as well as Dynamodb.

Go to IAM Roles.

Create Role for photoupload

Attach these policies.



1. Lamda Source Code

import json

import boto3

from datetime import datetime

from decimal import Decimal

bucket = "uploaded-images-for-rekognition"

model='arn:aws:rekognition:us-east-1:964394112301:project/BirdRecognition/version/BirdRecognition.2020-02-23T01.59.29/1582394369894'

min\_confidence=60

def lambda\_handler(event, context):

if event:

file\_obj = event["Records"][0]

bucketname = str(file\_obj['s3']['bucket']['name'])

filename = str(file\_obj['s3']['object']['key'])

print("Filename: ", filename)

client=boto3.client('rekognition')

#Call DetectCustomLabels

response = client.detect\_custom\_labels(Image={'S3Object': {'Bucket': bucket, 'Name': filename}},MinConfidence=min\_confidence,ProjectVersionArn=model)

# calculate and display bounding boxes for each detected custom label

print('Detected custom labels for ' + filename)

Labellist = []

confidencelist = []

for customLabel in response['CustomLabels']:

labelling=(customLabel['Name'])

Labellist.append(labelling)

confidencevalue=(customLabel['Confidence'])

confidencelist.append(confidencevalue)

if len(Labellist) > 0:

highest= confidencelist.index(max(confidencelist))

highlabel=(Labellist[highest])

highconfidence=(confidencelist[highest])

filenamelist=filename.split("\_")

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('Photodetect')

now = datetime.now()

response = table.put\_item(

Item={

'ID': "1",

'datetime': now.isoformat(),

'DeviceID': filenamelist[0],

'filename': filename,

'label': highlabel,

'confidence': Decimal(highconfidence)

})

print("Successful Upload")

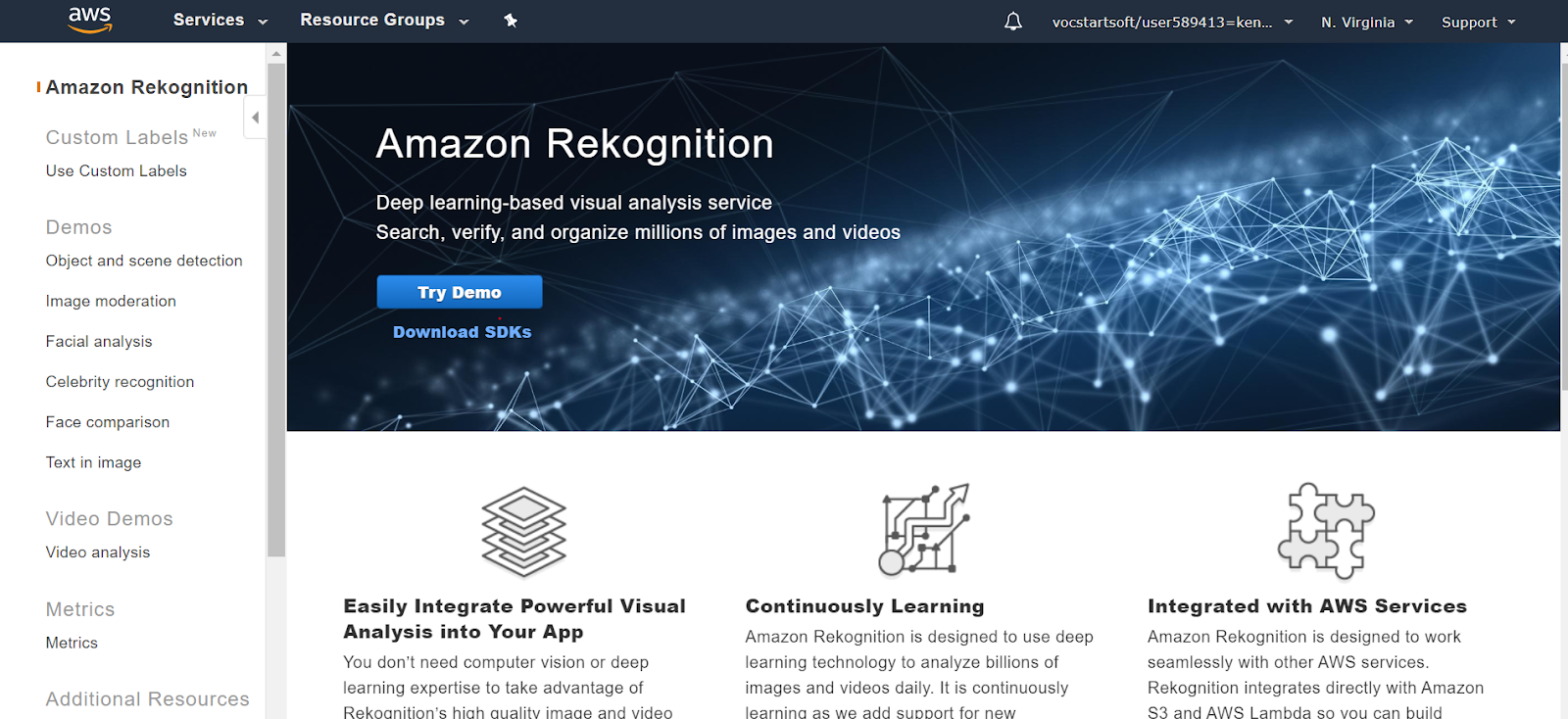
else:

print("No Birds Detected")

# Section 14 Rekognition

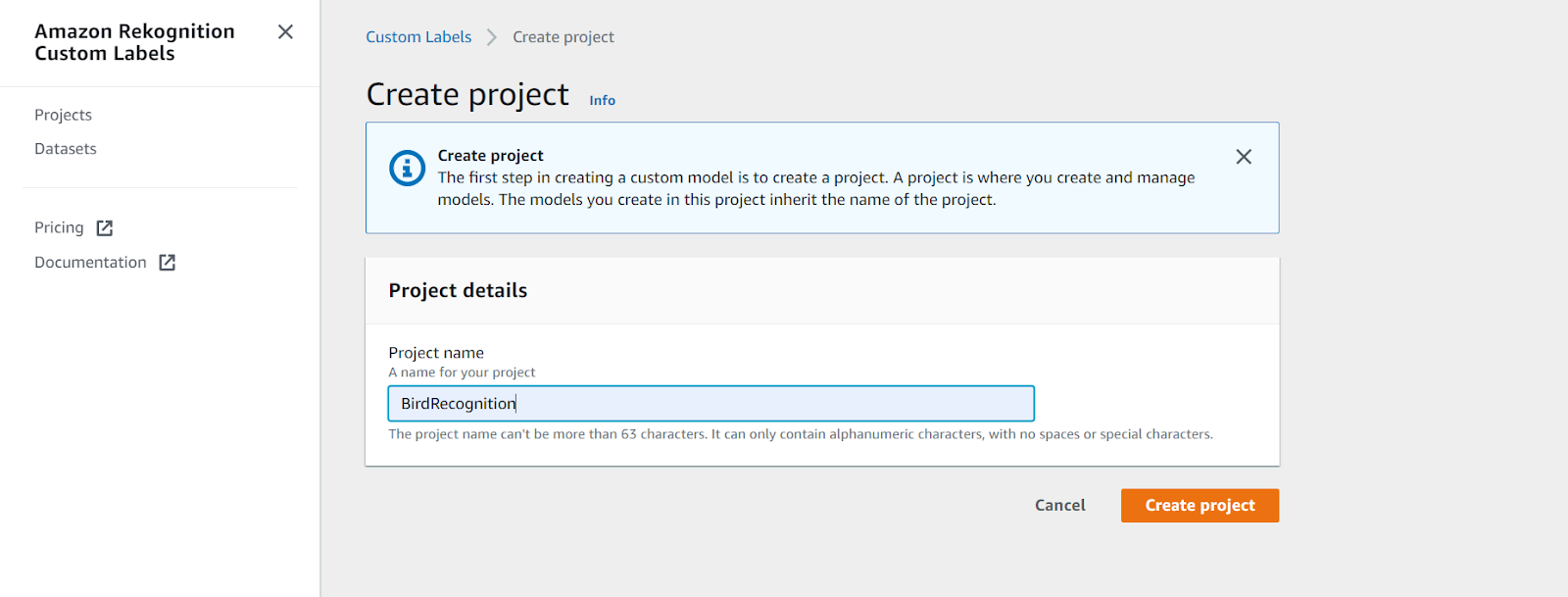
Amazon Rekognition identifies objects, people, text, scenes, and activities in images and videos. For this case, we are using Amazon Rekognition to identify 2 different birds, BooBoo and BiBi.

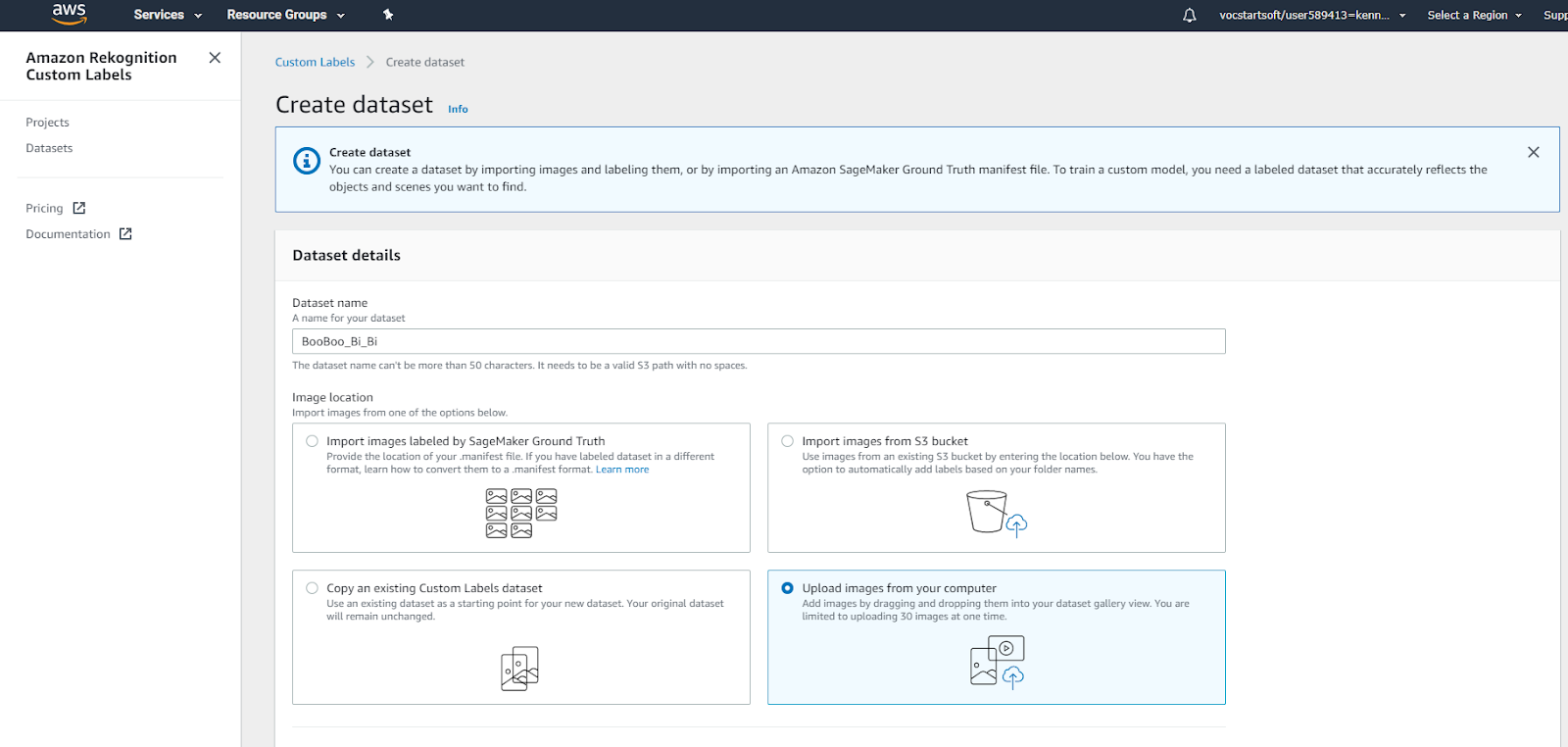
1. Custom labels



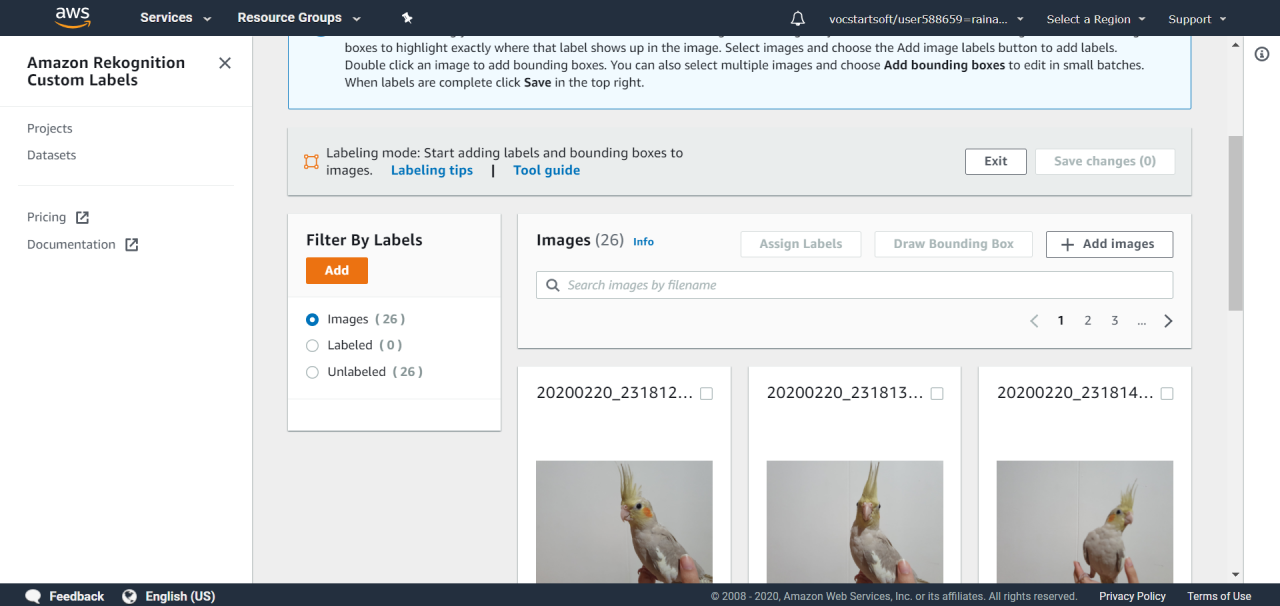
1. Firstly, we create a new project “BirdRecognition”

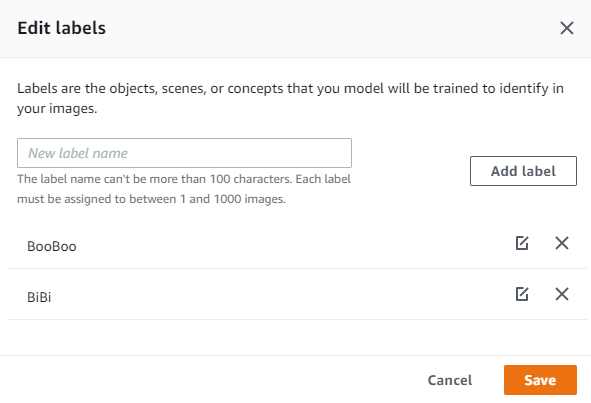




1. Next, we create a Dataset called “BooBoo\_BiBi” and click on “Upload images from your computer”
2. We need to add “front view” photos of the birds into the dataset



1. 
2. Create the labels “BooBoo” and “BiBi”. These labels are to help identify the birds.



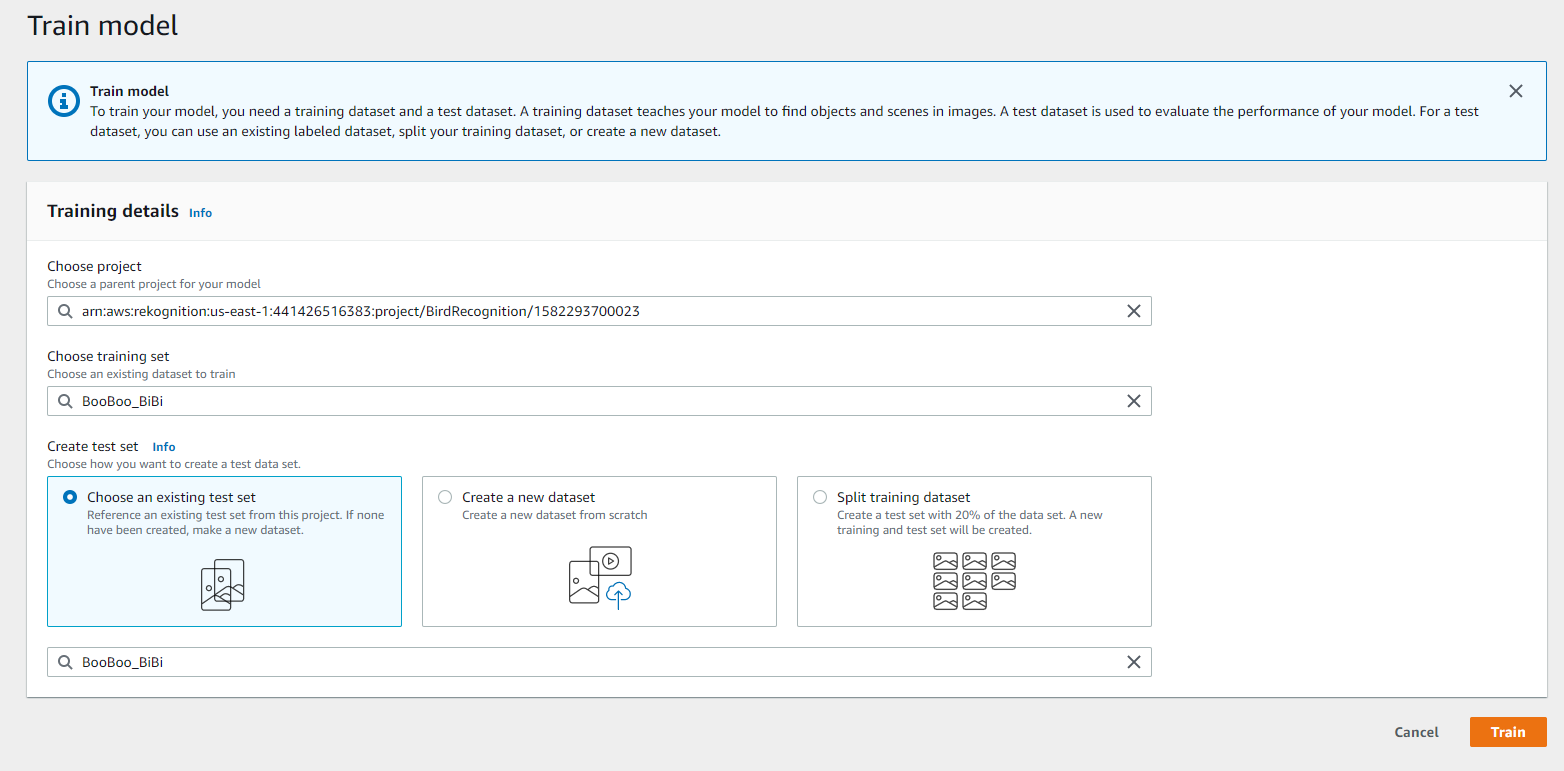
1. Click on “Draw Bounding box” to narrow the photo down to just the bird

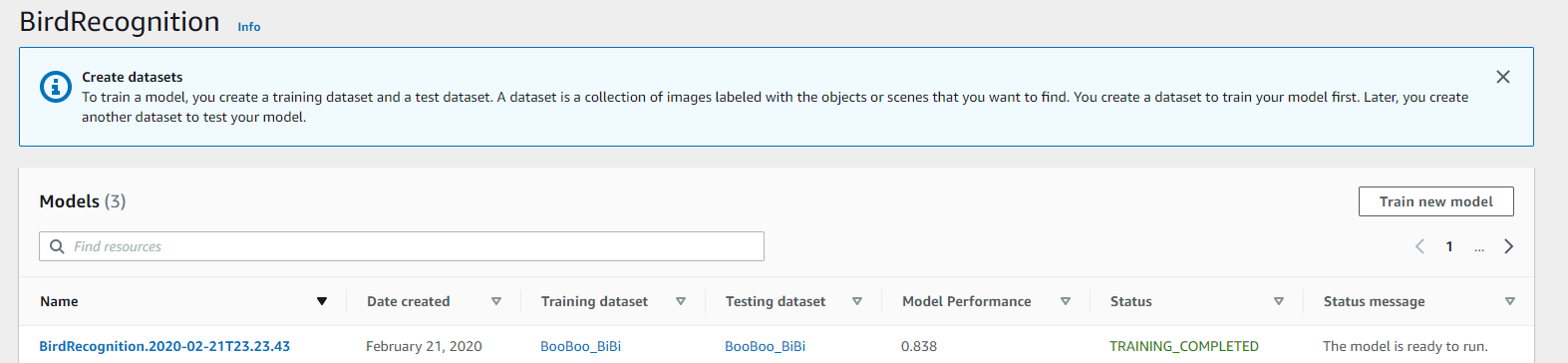


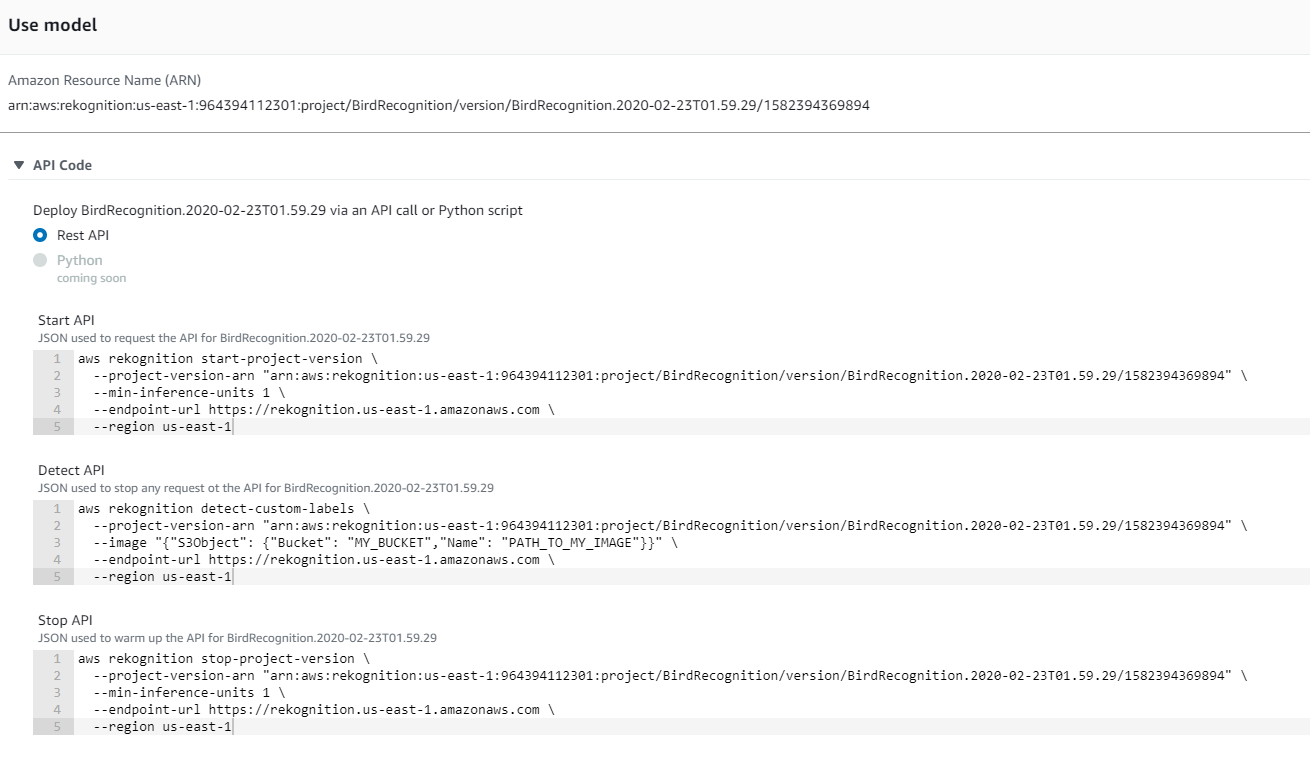
1. After labelling the photos, click “save changes” and “Train Model”



1. Choose BooBoo\_BiBi to start training





To Start Model get the project arn provided in the 

Make sure you have downloaded awscli as well as updated you AWS Credentials

In Terminal or cmd run to start model

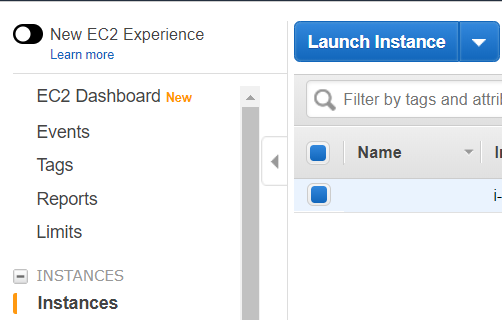
Replace project arn with your model arn

aws rekognition start-project-version --project-version-arn "<Project-arn>" --min-inference-units 1 --region us-east-1

to stop change start-project-version to stop-project-version

# Section 15 EC2 Instance

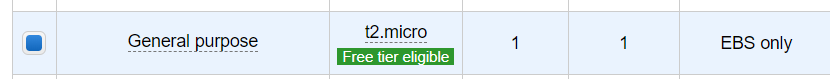
EC2 Instances. Click Launch Instance



Select Amazon linux 2 AMI HVM



Select Instance Type t2 micro



Click Next



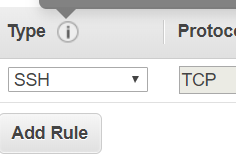




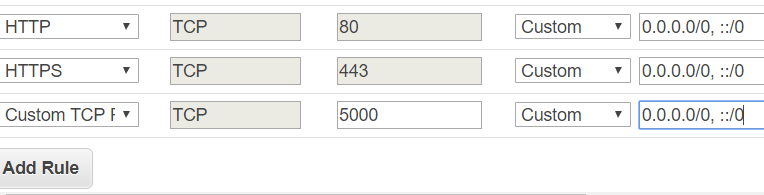


To allow connection, Configure Security Group to allow port 80, 443, and 5000 for flask.

Click add Rule



Add HTTP, HTTPS, Custom TCP Rule. In custom TCP Rule add port 5000 and allow all sources 0.0.0.0/0, ::/0

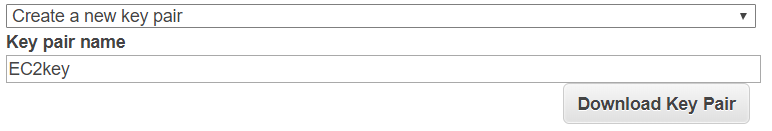


Click Review and Launch to start instance





Creating an new Key pair to connect to SSH

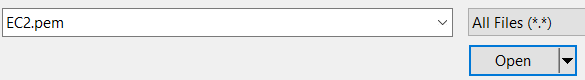


Download Key Pair and Launch instances

Go to PUTTYGEN



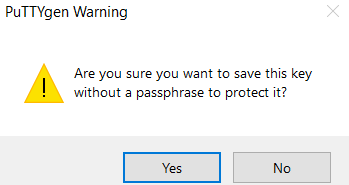
Load pem file to PuttyGen



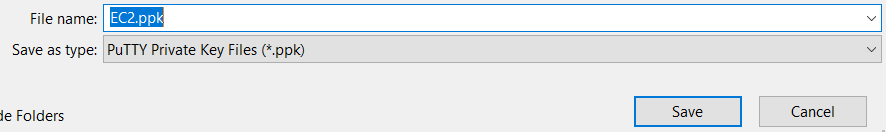
Click Save Private Key and save it as EC2.ppk



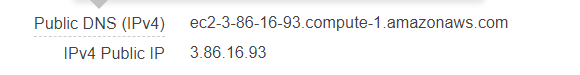
Click YES



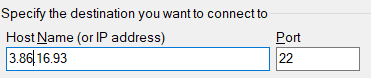
Save ppk



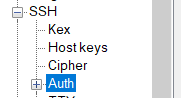
From EC2 instance, get IP or DNS



Enter into Putty



On the Left hand side click on SSH followed by auth



Add Private key and open



Login as : ec2-user

Library Installation for EC2

sudo yum update -y

sudo yum install -y httpd

curl -O <https://bootstrap.pypa.io/get-pip.py>

python get-pip.py --user

sudo pip install awscli

pip install flask

pip install gevent

pip install numpy

pip install botocore

pip install boto3

pip install AWSIotMQTTClient

pip install AWSIotPythonSDK

mkdir .aws

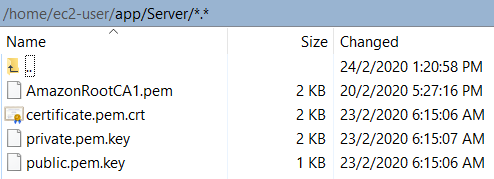
Touch .aws/credentials

Add your aws credentials into the aws credentials file

To run Server.

Unzip app

Add your Certificate and Private Key into app/server



In directory app,

Run

Python Server.py

# Section 16 References

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