Installation Guide for SensorApp project

Emilio Guevarra Churches

Introduction

- This project creates an IoT (internet of things) full suite which uses a Honeywell or Nova SDS-011 smoke and dust sensor attached to one or more Raspberry Pi computers for the use of detecting the level of dust and smoke particles in the air.
- There is also monitoring of temperature, humidity and air pressure through the Adafruit DHT22 sensor and Adafruit BMP180 sensor.
- Each Raspberry Pi with the above sensors attached is called a “sensor node”. The code for the web app for this project assumes there are 4 sensor nodes, but it is easy to modify the code if there are more or less than 4.
- The sensor nodes run a python program that collects data from the sensors attached and packages it in JSON format (https://json.org), and then sends that data to the Amazon IoT service (https://aws.amazon.com/iot-core/) in the cloud via the MQTT protocol (https://mqtt.org) which is used a lot for IoT communications.
- The Amazon IoT service is set up to automatically send the data it receives from each sensor node in JSON format to a Amazon DynamoDB database (https://aws.amazon.com/dynamodb/)
- A web app written in python using the Dash framework for dashboards that display information (https://dash.plot.ly) can then be run on a laptop and the web site it creates can be viewed in a web browser, or it can be deployed to a web server connected to the internet so that many people can access it. For this project Amazon Elastic Beanstalk (https://aws.amazon.com/elasticbeanstalk/) was used to make a web server with just a few commands.
- This guide only concerns the software setup. If you want to make the actual hardware or deploy the website to the internet, look in the appendix for instructions.

Sensor node setup

Requirements

- A 16GB SD Card
- An Amazon AWS account, which needs a credit card or a debit card to pay for it, although the costs for this project are very small, only a few dollars per month.
- Working wi-fi where the sensor will be deployed, with an Internet connection.
- An HDMI screen and HDMI cable for the initial RPi setup before it is connected to wi-fi. If you are using a RPi PiZeroWH, you will need a mini-to-normal HDMI adaptor
- A USB keyboard and USB mouse

Sensor Node Stage 1 RPi setup

1. Format SD Card (16GB or larger) in your laptop or home computer
2. Download latest NOOBS Distribution for RPi from https://www.raspberrypi.org/downloads/
3. Copy the NOOBS distribution files to the SD card as per these instructions: https://www.raspberrypi.org/documentation/installation/noobs.md
4. Connect the RPi to HDMI screen and keyboard/mouse via USB (you may need to use a USB hub since the PiZeroWH has only one USE socket)
5. Power up the RPi and choose Raspbian install from the menu shown.

6. Wait for Raspbian install to complete
7. From the startup menu on the RPi GUI screen, choose the RPi configuration utility and complete:
    a. Setup country/language
    b. Setup wi-fi (give it the SSID and password for your wi-fi network)
    c. Set the login password for the default pi user on the RPi
    d. Set Hostname (call it SensorX where X is the sensor number)
    e. Enable SSH logins
    f. Enable Serial port access (needed by dust sensor)
    g. Disable console over serial port (interferes with dust sensor)
    h. Enable I2C (needed by BMP180 sensor)
    i. Enable boot from console
8. From the start menu, choose to update all installed software packages via wi-fi. This may take quite a while if your internet is slow.
9. Shutdown from the start menu in the GUI. It should now be accessible over WiFi through SSH so you can remove the HDMI screen and the mouse and keyboard
10. Reboot it by powering it back on
11. Login into the RPi via SSH protocol (i.e. PuTTy) It should appear on your local wi-fi network as SensorX or SensorX.local
12. Login with username pi and the password you set
13. In the home directory for the pi user (just use "cd ~" to make sure you are there), clone the SensorApp code from GitHub using the command "git clone https://github.com/emiliog-c/SDD-Sensor-App.git"
14. If all OK, then shut it down with the command "sudo shutdown -h now" (you will need to give the login password for the pi user) and place it in the case if you are using one. Some of the cases prevent access to the SD card which is why we set up the RPi on the SD card first.

Sensor Node stage 2: AWS IoT Setup

These steps must be followed to register each sensor node as a "thing" in AWS, so you need to repeat these steps for each sensor node you are setting up.

1. Set up an AWS account
2. Login to AWS console
3. Go to IoT-Core service
4. Click on onboard
5. Click on Get Started and continue (Figure 1)
6. Choose the software package for Linux and the SDK for Python (Figure 2,3)
7. Give your device the name of SensorX wher X is the sensor node number (eg. Sensor1) (Figure 4)
8. AWS automatically sets up Public and private access keys and client certificate (these need to be installed onto the sensor node so it can authenticate itself to the AWS IoT service).
9. Download the software kit provided (Figure 5) – it should be in a file called connect_device_package.zip
10. On your laptop or home computer, start cmd from the search bar and copy the downloaded package to the RPi using this command: "pscp connect_device_package.zip pi@Sensor#.local:/home/pi" where X is the sensor node number. You will need to give the password for the pi user on the RPi. You may need to leave out the .local
11. Log in to the RPi as the pi user using PuTTY. In the home directory, unzip the package using the command "unzip connect_device_package.zip". This will places root and device certificate and the public and private key in the home directory on the RPi (Figure 6)
12. Install the three required python libraries with the command "sudo pip3 install x" where:x = AWSIoTPythonSDK or adafruit-bmp or Adafruit_DHT
13. Return to the AWS console in your web browser and click on your sensor thing.
14. Navigate to interact, and copy the HTTPS address for your sensor node (Figure 7)

15. On the RPi, change to the directory where the python code etc for the sensor node is stored using “cd SDD-Sensor-App/rpi-sensor-node/”
16. Edit the service file for the node (use the appropriate number for the node) by typinging in “nano sensor1boot.service”
17. Change the sensor number to the number you are using
18. Paste in the URL that you copied in the appropriate place
19. Change the file names in the command line argument to the correct names (i.e. /home/pi/Sensor#.private.key)
20. Change the type of sensor in the last part of the argument to either Honeywell or SDS011
21. Save the file and exit (use control-X, answer Y to save and accept the same file name)
22. Return to AWS console screen in your web browser
23. Go to the sensor management tab
24. Click security, then the certificate (Figure 8)
25. Click policies and the policy given (Figure 9)
26. Copy this into the policy document (Figure 10)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:ap-southeast-2:278221558381:topic/sdk/test/java",
        "arn:aws:iot:ap-southeast-2:278221558381:topic/sdk/test/Python",
        "arn:aws:iot:ap-southeast-2:278221558381:topic/sensors/data",
        "arn:aws:iot:ap-southeast-2:278221558381:topic/sensors/info"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:ap-southeast-2:278221558381:topicfilter/sdk/test/java",
        "arn:aws:iot:ap-southeast-2:278221558381:topicfilter/sdk/test/Python",
        "arn:aws:iot:ap-southeast-2:278221558381:topicfilter/sensors/data",
        "arn:aws:iot:ap-southeast-2:278221558381:topicfilter/sensors/info"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:ap-southeast-2:278221558381:client/Sensor1RPiZeroW",
        "arn:aws:iot:ap-southeast-2:278221558381:client/Sensor2RPiZeroW",
        "arn:aws:iot:ap-southeast-2:278221558381:client/Sensor3RPiZeroW",
        "arn:aws:iot:ap-southeast-2:278221558381:client/Sensor4RPiZeroW"
      ]
    }
  ]
}
```

27. On the RPi, type in “sudo cp sensor1boot.service /etc/systemd/system/” (use the right number instead of 1 for the sensor). This installs the system startup service file that automatically runs the python code for the sensor, and automatically restarts it if it crashes.

28. Test the sensor startup with “sudo systemctl start sensor1boot.service”
29. Check startup messages with “sudo systemctl status sensor1boot.service”
30. Wait for 10 minutes, and repeat “sudo systemctl status sensor1boot.service” checking for appropriate messages and errors. You should see sensor measurements being printed.
31. If all ok, enable script for automatic start at bootup with “sudo systemctl enable sensor1boot.service”
32. At this stage, if you reboot the sensor node using “sudo shutdown -r now” then the python file should automatically run after reboot and start sending sensor data to AWS IoT. You can use the monitor and test tabs in the AWS IoT console in your browser to check that.
33. Repeat all the above steps for each node. There are many steps but it is quite quick after you have done the first one, taking maybe 1 hour or so for each one.

Sensor Node stage 3: AWS DynamoDB database Setup

1. In the AWS console, click services and type in DynamoDB
2. Create two tables with these table names and keys, but leave everything else as default settings (Figure 11 and 12)

* Table name:	SDD-Sensors-Data
* Primary partition key:	sensorID (String)
* Primary sort key:	timestamp (String)
* Everything Default	
* Table name:	SDD-Sensors-Info
* Primary partition key:	sensorID (String)
* Primary sort key:	timestamp (String)
* Everything Default	

3. Return to IoT Core console
4. Click on Act (Figure 13)
5. Create two new rules
6. Give the rules a name, one to insert data into the sensor data table and the other to insert log messages into the info table (log)
7. Copy this into the SQL Statement for the info table
   `SELECT * FROM 'sensors/info'`
8. Copy this into the SQL Statement for the data table
   `SELECT * FROM 'sensors/data'`
9. Add the action of inserting into a DynamoDB Table (Figure 14)
10. Configure the actions according to this picture
11. Create a role in the action (Figure 15)
12. At this stage, every time the AWS IoT service receives a MQTT message from the sensor nodes with the topic “sensor/info” or “sensors/data”, the rule should cause AWS to automatically create a new record in the appropriate database table containing the JSON payload of the message. See the python code for the sensor node for details of the payload.
13. You can check this in the DynamoDB console – you should see records starting to appear in the database.

Note that it took a LOT of trial and error and head scratching to get all of the above working for the first sensor. But after that, by following the steps exactly, each extra sensor node worked first time.

Website setup

These steps should be carried out on your laptop or home computer.

1. Make sure you have Python 3.6 installed. It also works on an Apple Mac with Python 3.5. Python 3.7 should also work but that wasn’t tested. It will NOT work with Python 2.7 or 3.4 or earlier. Visit the python web site if Python is not installed.
2. From your cmd line, type the command “pip3 install x” where x is each of the following:

boto3
jsonschema
numpy
pandas
Flask
Flask-Compress
Flask-Cors
Flask-SeaSurf
dash
dash-auth
dash-bootstrap-components
dash-core-components
dash-daq==0.1.0
dash-html-components
dash-renderer
dash-table
plotly
dynamodb-json

3. These steps 3 to 6 are only needed if you are in a different AWS region from Sydney or if you used different table names in the database. Edit the sensorDataApp.py file in the SDD-Sensor-App\dash folder using Python Idle or Notepad++ or some other text editor
4. Locate the lines 64, 66 and 67.

```
# set up a handle to the database
dynamodb = session.resource('dynamodb', region_name='ap-southeast-2',)
# set up handles for the two tables: one for sensor data, one for information
messages from the sensors
dataTable = dynamodb.Table('SDD-Sensors-Data')
infoTable = dynamodb.Table('SDD-Sensors-Info')
```

5. Change your region name to your respective region
6. Insert your DynamoDB data table and info table names in line 66 and 67 respectively.
7. In the same directory, look into the app_passwords.py file.
8. Configure your usernames and passwords in there.
9. Run the web app code from command line

```
cd:\Users\(username)\Documents\GitHub\SDD-Sensor-App\dash

python sensorDashApp.py
```

10. If it is running correctly, after a while these messages will appear (with different PIN numbers):

```
Debugger PIN: 101-395-991
 * Serving Flask app "sensorDashApp" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
Running on http://127.0.0.1:8050/
Debugger PIN: 959-360-145
```

11. If so, the web site is now running on your laptop. Copy the URL from the line "Running on _____ " into your browser. The URL will probably be http://127.0.0.1:8050/
12. The site should load and you can start using it.

Appendix

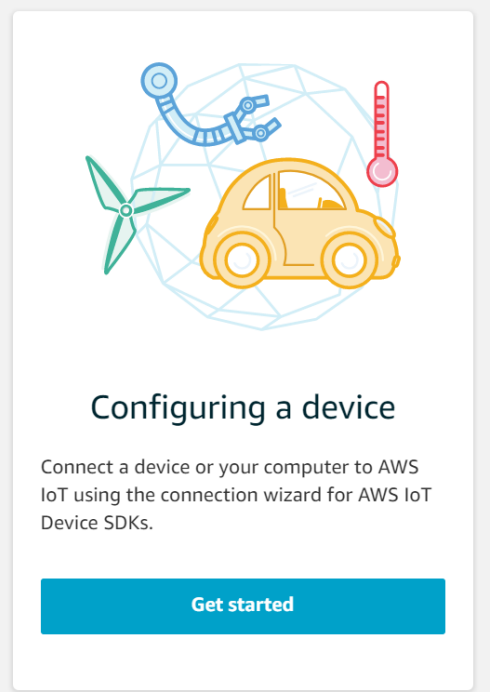

Figure 1

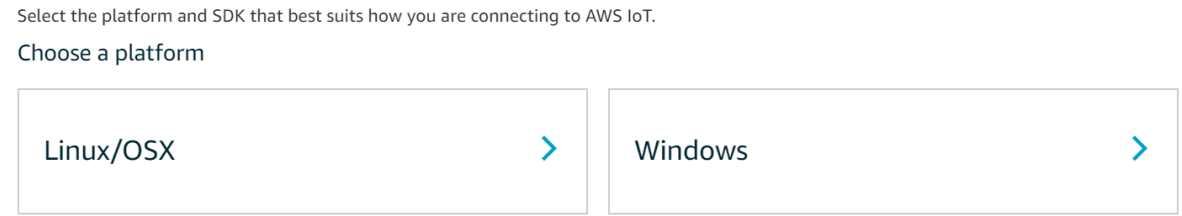

Figure 2

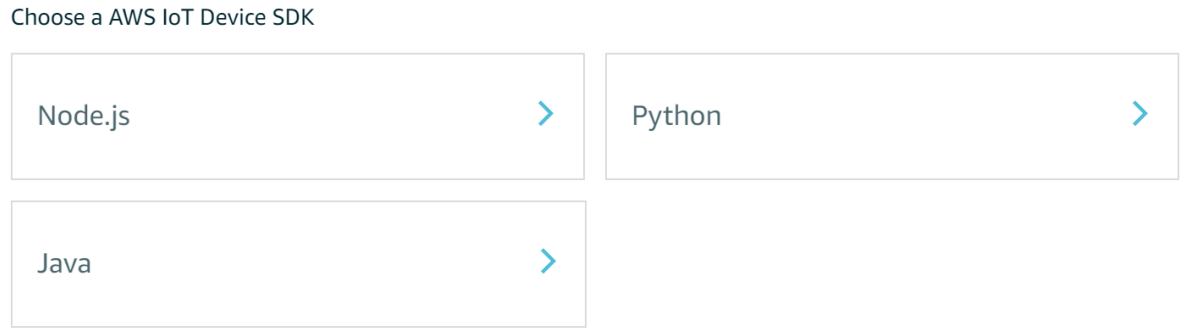

Figure 3

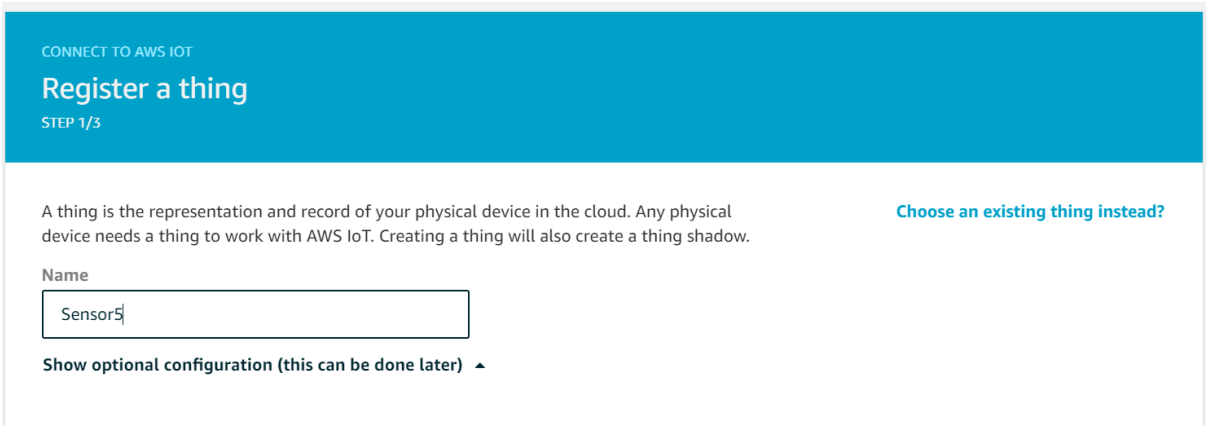

Figure 4

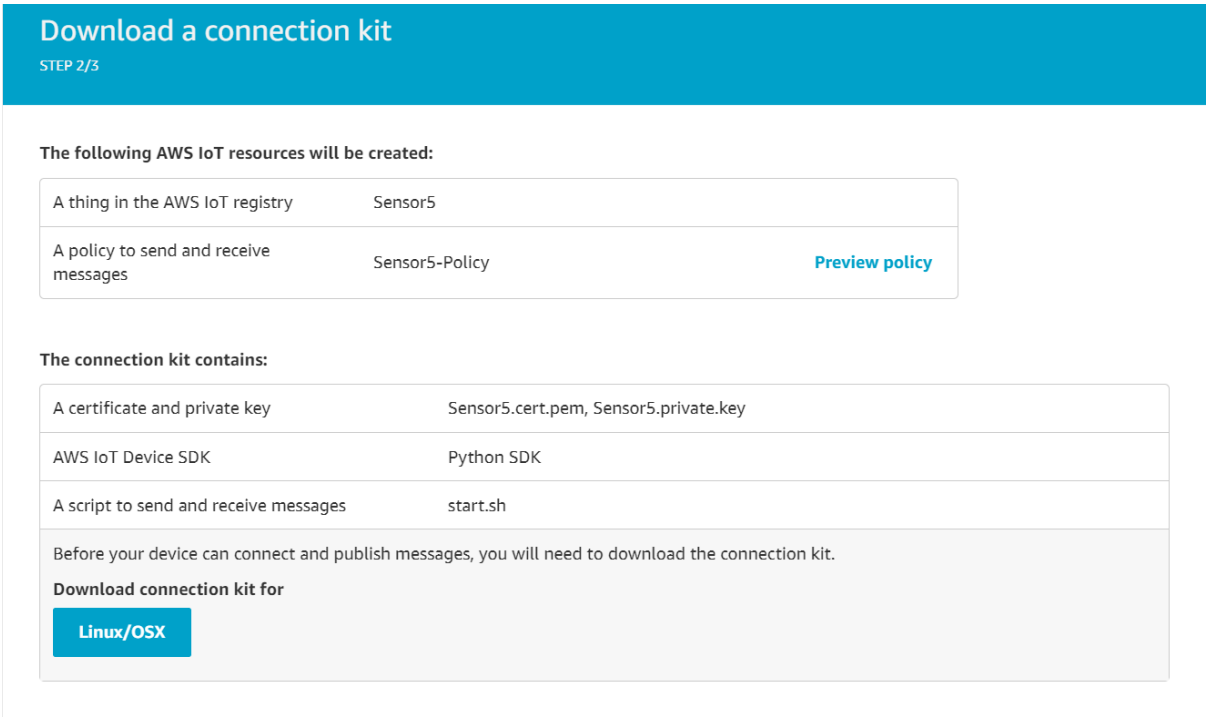

Figure 5

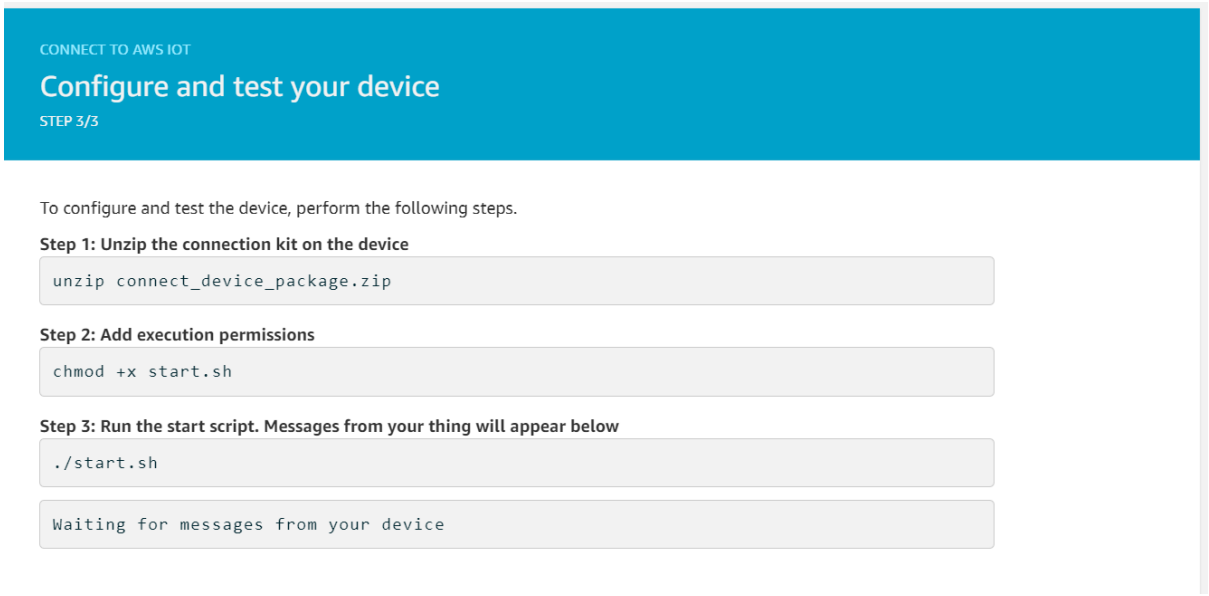

Figure 6

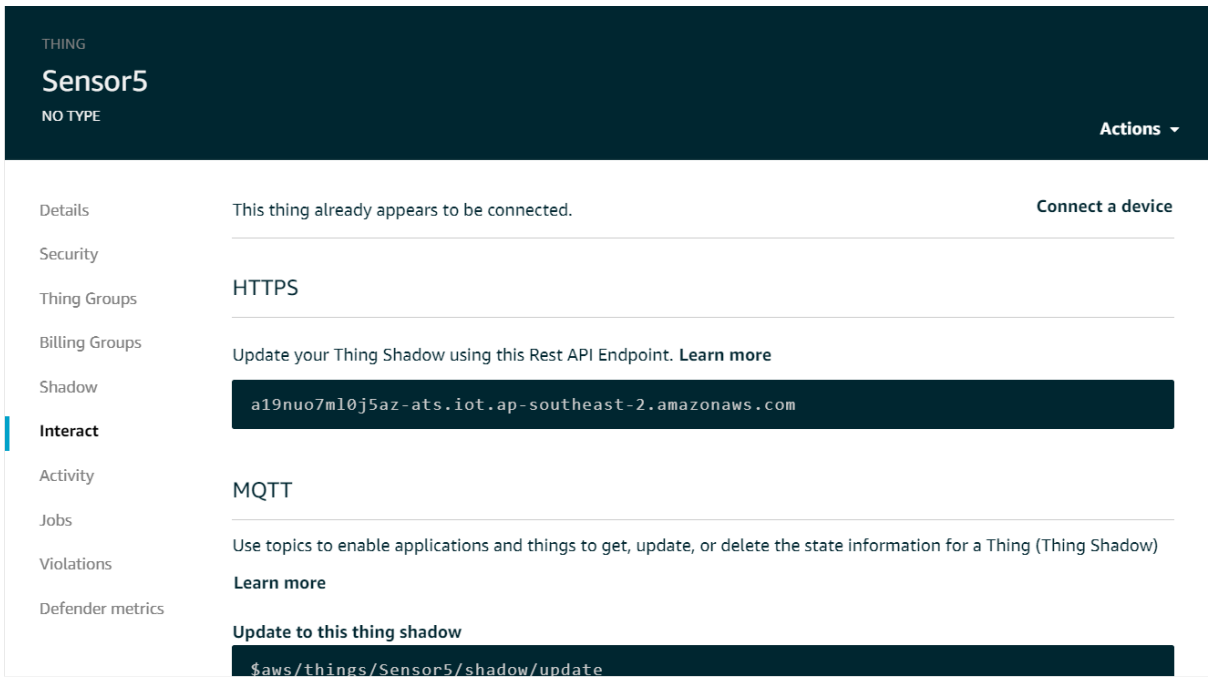

Figure 7

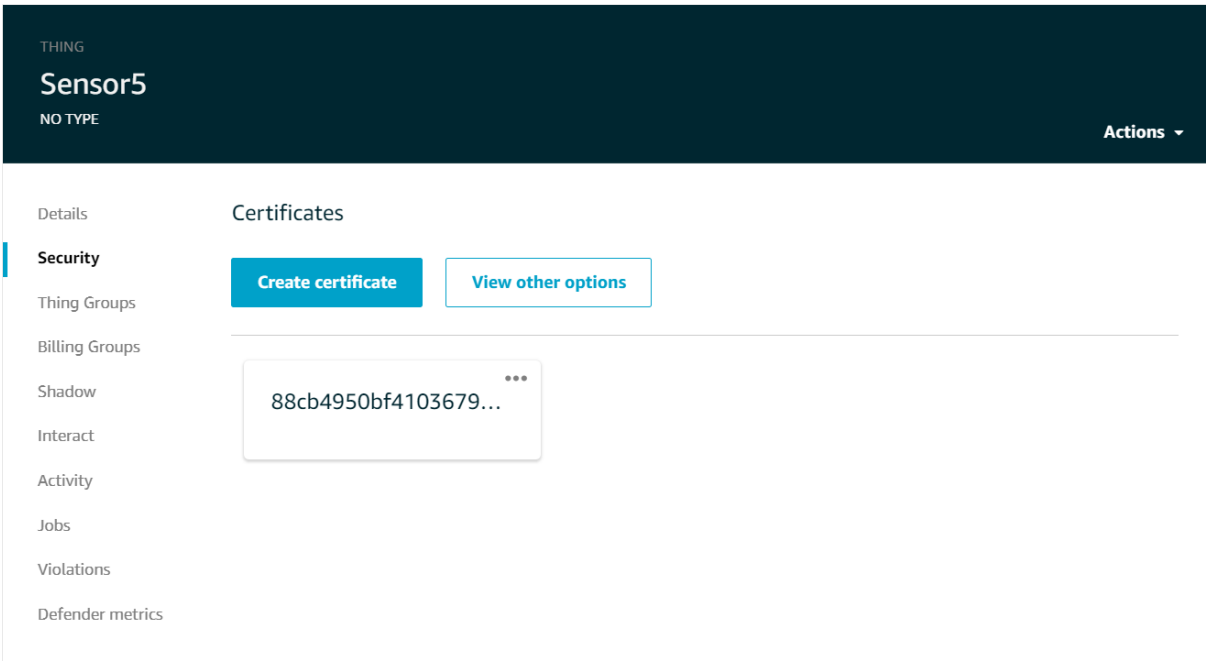

Figure 8

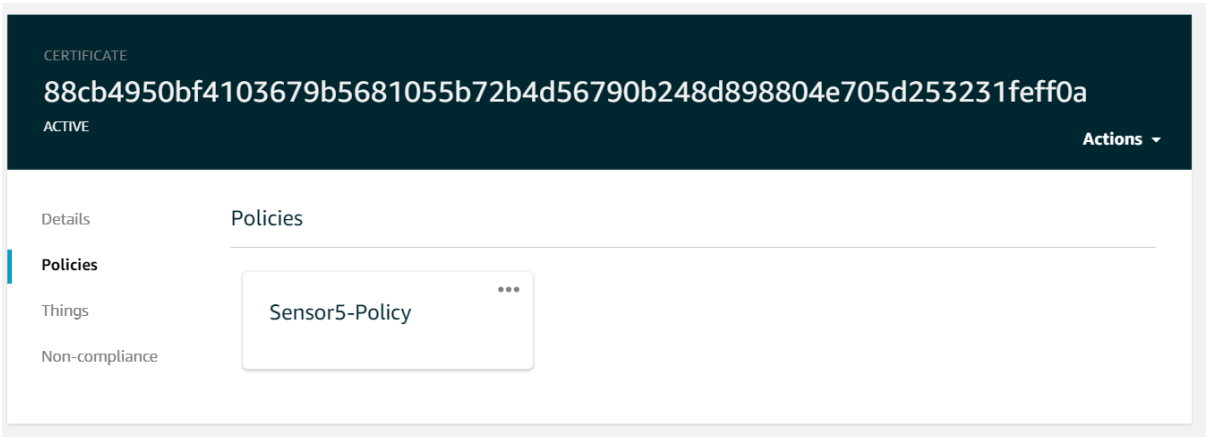

Figure 9

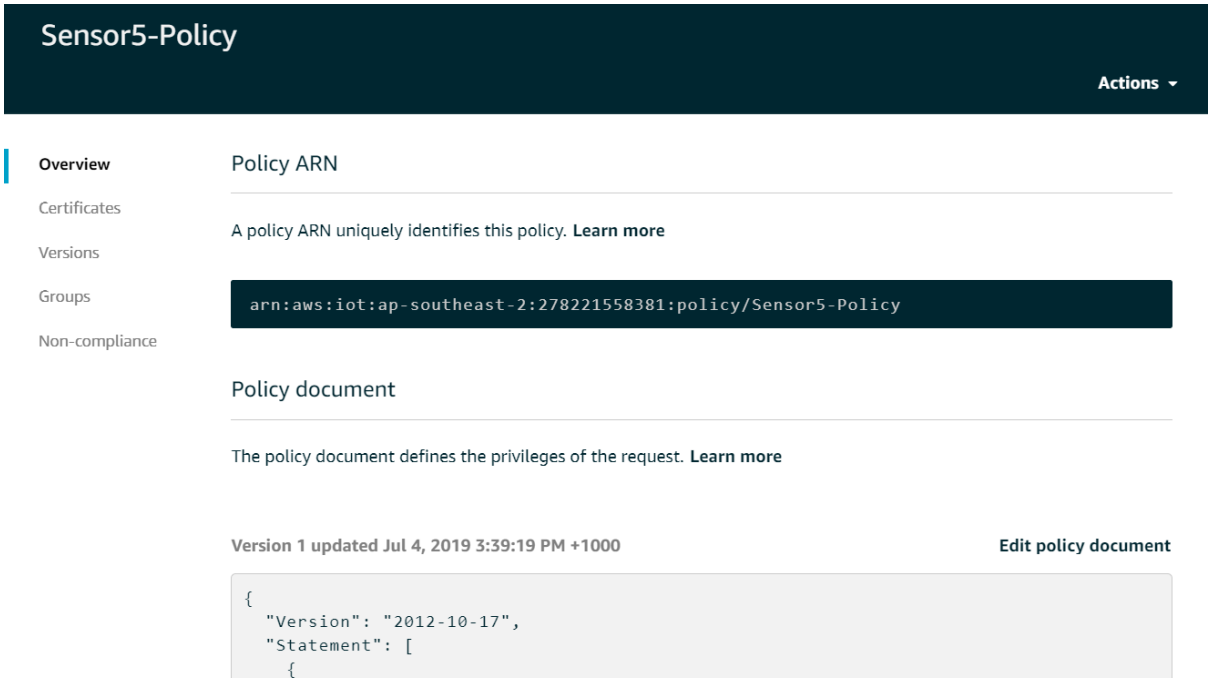

Figure 10

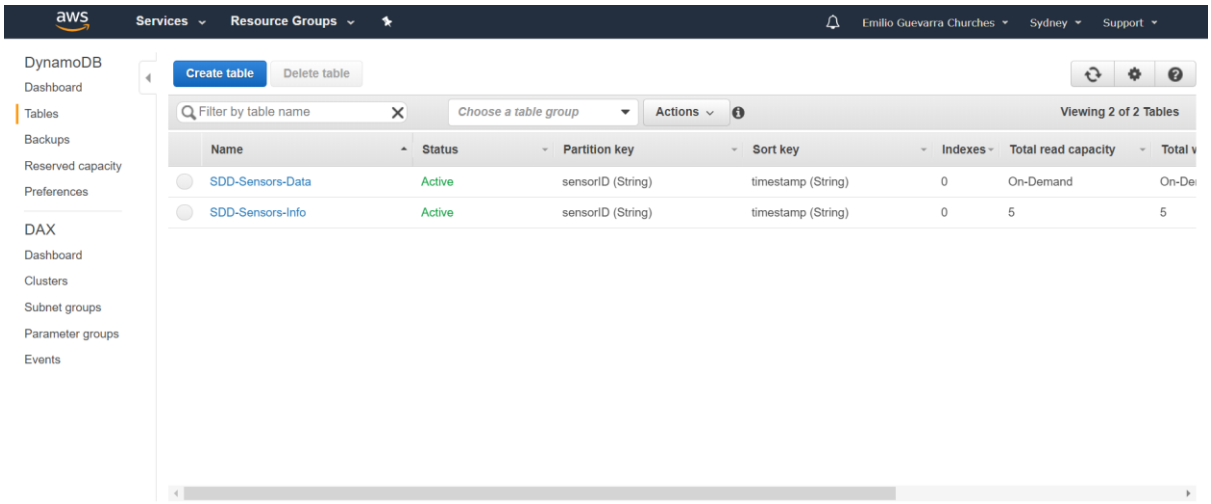

Figure 11

Table details

Table name	SDD-Sensors-Data
Primary partition key	sensorID (String)
Primary sort key	timestamp (String)
Point-in-time recovery	DISABLED **Enable**
Encryption Type	DEFAULT **Manage Encryption**
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED **Manage TTL**
Table status	Active
Creation date	June 16, 2019 at 8:52:11 PM UTC+10
Read/write capacity mode	On-Demand
Last change to on-demand mode	June 18, 2019 at 7:27:16 PM UTC+10
Provisioned read capacity units	-
Provisioned write capacity units	-
Last decrease time	-
Last increase time	June 18, 2019 at 8:25:26 AM UTC+10
Storage size (in bytes)	25.34 MB
Item count	194,573 **Manage live count**
Region	Asia Pacific (Sydney)
Amazon Resource Name (ARN)	arn:aws:dynamodb:ap-southeast-2:278221558381:table/SDD-Sensors-Data

Figure 12

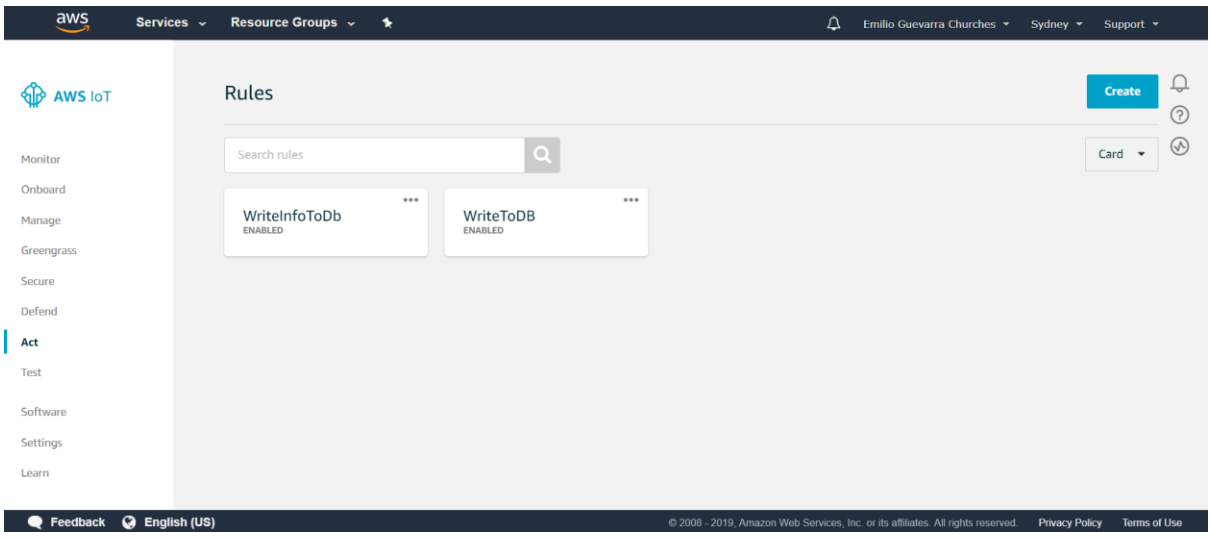

Figure 13

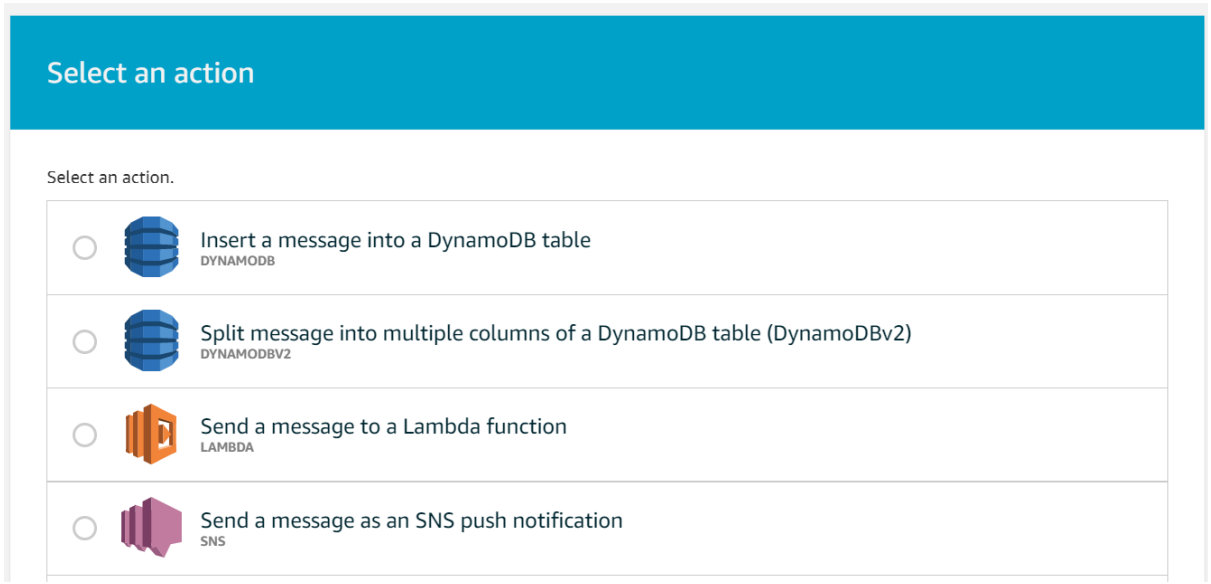

Figure 14

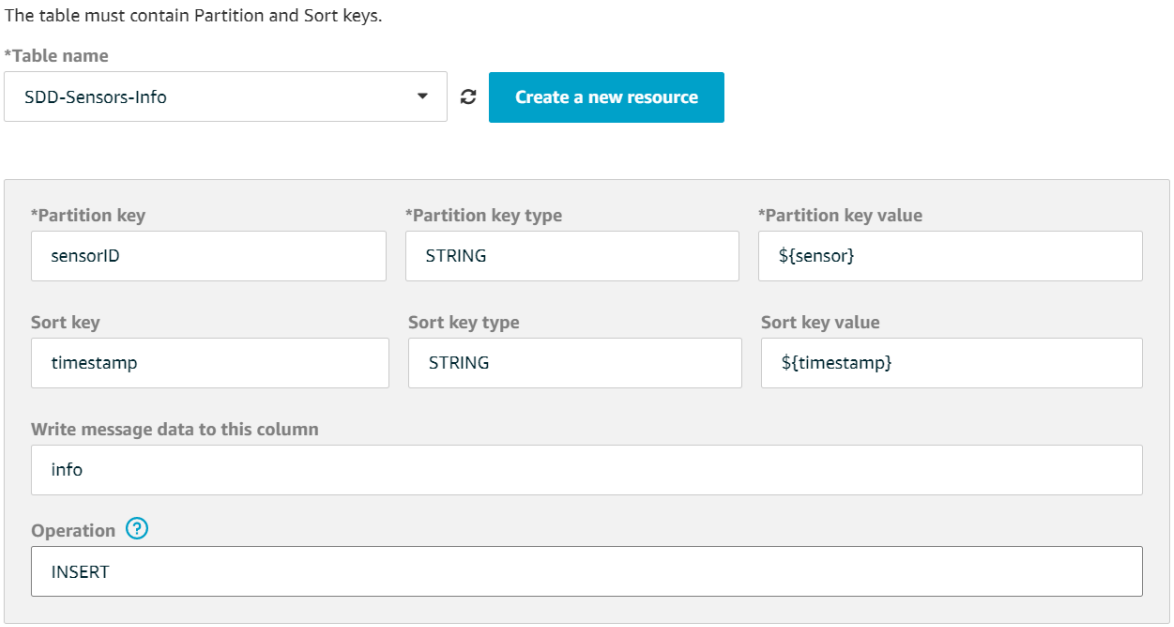

Figure 15

Hardware setup (if building)

Requirements

- 1 or more Raspberry Pi (RPi) computers with a power supply (either the official RPi supply or a USB power supply with a micro-USB connector). The RPi PiZeroWH model is recommended because it it small and cheap and has built-in wi-fi and pin headers. Most RPi models include the pin headers except the PiZeroW (no H). Not all models include built-in wi-fi. If not, you will also need a USB wi-fi dongle.
    - I purchased the RPis from here: https://core-electronics.com.au/raspberry-pi-zero-wh.html
    - A case for the RPi is optional but it looks neater and makes it easier to mount the sensor somewhere. Many cases for RPi computers are available, but you must choose a case that gives access to the RPi pins so the sensors can be connected to it. I used the Pimonori Pibow case (https://core-electronics.com.au/pimoroni-pibow-zero-w-case.html) but this case would also work and is cheaper: https://core-electronics.com.au/slim-case-for-raspberry-pi-zero.html
- A 16GB SD Card – it is cheaper to just buy these from OfficeWorks or Coles
- An SD Card Reader for your laptop so the SD card can be formatted and the NOOBS software for the RPi placed on it
- An HDMI screen and HDMI cable for the initial RPi setup before it is connected to wi-fi. I used the TV for this. If you are using a RPi PiZeroWH, you will need a mini-to-normal HDMI adaptor, like this: https://core-electronics.com.au/mini-hdmi-to-standard-hdmi-jack-adapter-for-raspberry-pi-zero.html
- A USB keyboard and USB mouse
- A Honeywell or Nova SDS-011 particulate sensor. The Honeywell sensors are easiest to obtain from major electronics parts suppliers. I ordered mine from Element14 in Australia (https://au.element14.com/honeywell/hpma115s0-xxx/particle-sensor-pm2-5-15-5v/dp/2770767). You also need connectors for the Honeywell like these https://au.element14.com/molex/51021-0800/connector-housing-rcpt-8pos/dp/1012262
    - The Nova SDS-001 sensor can be ordered on eBay from many places eg https://www.ebay.com.au/itm/ORIGINAL-NOVA-SDS011-Laser-Dust-Sensor-PM2-5-PM10-brand-new/282761937532?epid=578245929
- An Adafruit DHT22 temperature and humidity sensor. I ordered mine from here: https://core-electronics.com.au/dht22-temperature-and-relative-humidity-sensor-module.html

    - You also need a 7.5 kOhm pull-up resistor as per the DHT22 wiring diagram (see below)
- An Adafruit BMP180 temperature and air pressure sensor. I ordered mine from here: https://core-electronics.com.au/bmp180-barometric-pressure-temperature-altitude-sensor-5v-ready.html
- Some RPi/Arduino header wires, which are available from JayCar electronics stores
- The ability to solder a few connections

Hardware construction

Note that some soldering is required to connect the header pin connection wires to the DHT22 sensor and to the connector for the Honeywell or Nova dust and smoke sensor. You also need to solder the pin headers supplied onto the little BMP180 sensor board.

- The particle sensor get power and communicates with the Raspberry Pi through the serial port. For the Honeywell sensor, see the connection diagram in the datasheet for it at http://www.farnell.com/datasheets/2313714.pdf?_ga=2.50869865.1516986720.1562884160-530993158.1562884160. For the Nova sensor, the connections are labelled on the sensor board. Connect 5V input on the sensor to pin 2 (5V out) on the RPi using pin header connectors. Connect GND (ground) on the sensor to pin 9 (GND) on the RPi. Then connect pin 8 (which is serial port Receive RX) to TX (transmit) on the sensor, and pin 10 (which is serial port transmit TX) on the Rpi to RX 9receive) on the sensor.
- The Adafruit DHT22 is connected on the following pins on the RPi, as per the wiring diagram at https://learn.adafruit.com/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging/overview: pins 1 (3.3V), 11 (GPIO17) and 14 (GND). There is a 7.5 kilo ohm pullup resistor between the GPIO pin and 3.3V supply (see the wiring diagram) that must be soldered in place on the sensor connector legs. Use heat-shrink tubing to insulate all the connections so they don't short out.
- The BMP180 air pressure sensor connects via I2C protocol (high speed serial protocol). Follow the instructions on this page exactly to ensure that your sensor is working: https://tutorials-raspberrypi.com/raspberry-pi-and-i2c-air-pressure-sensor-bmp180/

At this point all the sensors should be working. However, it is recommended to complete the rest of the setup to test them before wrapping the wires in tape to make them a bit neater.

Deploy Website on internet

There are many ways to deploy the app on the internet so that others can just use it without any setup. The easiest way is to use the Amazon AWS Elastic beanstalk service, that will set up a website with just a few commands or steps. Basically I just followed the steps in this blog post (except the steps to create a dash app, because I have already done that in the steps above): https://medium.com/@austinlasseter/plotly-dash-and-the-elastic-beanstalk-command-line-89fb6b67bb79

However, the Dash app did not run and I required assistance. After adjusting the requirements.txt file that automatically installs the python libraries on the web server, and creating the additional configuration files in the .ebextensions folder, it finally worked after about 10 or 20 tries. If you follow the instructions it should work for you, but troubleshooting it is complex and you may need help.

Note that authentication is enabled when deployed on a web server, so you need to give the users of the site the URL and a username and password from the dash\app_passwords.py file. But be sure to change that file so it uses different passwords before deploying to Elastic Beanstalk, otherwise anyone will be able to log in to your web site.