CA4106 - Internet of Things Management Assignment

Michael Walsh - 17428926
Karl Hannigan - 17435332
Adrian Lackey - 17466892

GitHub Repository: https://github.com/walsm232/CA4017-IoT-Assignment
YouTube Video Demo: https://www.youtube.com/watch?v=KFZww6xRznQ

We have marked the roles and responsibilities within this document throughout using light text with brackets beside each of the headings.

Project Description

The aim of this assignment was to connect an IoT device to AWS IoT Core using their connection kit. This can then enable you to subscribe to topics in order to view messages being sent by your IoT devices. You can also use the AWS IoT Core Console in order to publish messages to your devices. We decided to also create prototypes for a potential mobile / web application which could be used to manage your IoT devices, their settings, and the access to these devices all in one place. This could be developed using the AWSIoTSDK for Python possibly with the use of Django or Flask, HTML, CSS, and Javascript. We researched IoT Core Jobs and Tunnels in order to understand them and how they work. We also decided to create a web application using Django, HTML, CSS, Bootstrap and jQuery which can be used to connect and send messages to IoT devices which is outlined in our Design section.

Roles & Responsibilities

Michael:

Michael completed the Project Timeline section of the blog in order to outline all of the work completed throughout the project. He also completed the prototypes of a potential mobile / web application with Karl using Figma and the Design section of the blog in order to outline the layout and design of an application which could be developed through the use of the AWS IoT SDK and other languages & tools. Michael also developed a simple web application using Django to show how the application could work and be used.

Karl:

Karl completed the process of connecting the IoT device to AWS IoT Core. He outlined the steps as part of this process in Deployment to the Cloud section of the blog. Karl also worked on developing the prototypes in Figma with Michael and assisted in explaining these in the Design section.

Adrian:

Adrian completed the Research section of the blog in order to outline all of the research we had to undertake in order to complete the project.

Description of Learning Outcomes (Michael):
Each of the main learning outcomes are briefly described below. These are outlined in more detail in the Research section of our document.

- ➢ *How to set up an AWS account:* We needed to learn how to set up an AWS account in order to work on the IoT Core services.
- ➢ *How to create a user and set up permissions:* We also needed to learn how to set up a User in AWS Identity and Access Management (IAM) and grant permissions to work on IoT Core services.
- ➢ *How to create a thing object through IoT Core quick connect and download the connection kit:* We also learned how to create a thing object in IoT Core's 'Quick Connect' and how to download the connection kit with certs for our thing.
- ➢ *How to configure and test the IoT device:* We also needed to learn how to configure and test our IoT device through the use of Windows Powershell / Mac Terminal.
- ➢ *How to subscribe to and publish messages through Message Queueing Telemetry Transport (MQTT):* We needed to learn how to subscribe to an MQTT topic in order to show messages being received and also how to publish to a topic through the AWS IoT Console.
- ➢ *Creating Jobs and Tunnels in the IoT Core Console:* We also needed to learn how to use the Jobs and Tunnels functionality which is found in the IoT Core console.

Project Code (Michael, Karl, Adrian):
Link: https://github.com/walsm232/CA4017-IoT-Assignment

YouTube Video Demo (Michael, Karl, Adrian):
Link: https://www.youtube.com/watch?v=KFZww6xRznQ

Blog
Project Timeline (Michael):
Week 1 (February 22nd to 28th):

- We began researching AWS IoT Core on YouTube, StackOverflow, and through the AWS documentation.
- This research has been outlined in the Research section of this blog.

Week 2 (March 1st to 7th):

- We set up the layout for our project documentation including the headings and outlining the responsibilities of each team member in order to prepare for the submission.
- Michael set up the GitHub repository so that we could begin committing and pushing changes for all group members to see.

Week 3 (March 8th to 14th):

- Karl began working on connecting a device to AWS.
- He created a thing in AWS IoT Core named 'CA4017'.

- He downloaded the connection kit for this and tested this in the IoT Core Console.
- He extracted the zipped connection kit.
- He ran 'chmod *x start.sh' in order to set up execution permissions.
- Karl then ran the 'start.sh' script found in this folder in order to begin sending messages.
- Karl then navigated to the 'Test' section of the IoT Core Console and tested the Subscribe and Publish functionality on the MQTT Test Client.
- This proved that the messages were being sent and received correctly to and from the IoT device.
- He took screenshots of this whole process in order to include these in the final document for our submission.
- We all continued working on our project documentation.

Week 4 (March 15th to 21st):

- We began working on the slides for our presentation and continued working on our project documentation for the submission.

Week 5 (March 22nd to 28th):

- Michael and Karl began researching how to create high quality prototypes to include in the Design section of this document.

Week 6 (March 29th to April 4th):

- We developed some prototypes for a potential web or mobile application for managing IoT devices. Users would be able to securely register, organize, monitor and remotely manage IoT devices in the cloud through the use of the application.

Week 7 (April 5th to April 11th):

- We finalized our slides and began practicing for our presentation.
- We decided to develop a web application using Django, HTML, Bootstrap, CSS, and jQuery.

Week 8 (April 12th to Submission Deadline):

- We recorded our demo and presentation and uploaded it to YouTube so that we could include it in our project documentation.
- We reviewed our documentation in order to ensure everything was completed correctly before our submission.
- We submitted our final document including all of the necessary headings and documentation.

Research (Adrian):

- AWS Account Setup
  To start our project we needed to create a AWS account as this was the first time for any individual in the group to create one we carried out some research this started with following some youtube videos found at Amazon Web Services (2013) by following these tutorials we were able to sign up and create a root user account. The video also suggested to create a IAM user along with the root user account.
- AWS User Creation

During the process of our project we needed to create AWS users for our AWS account and conduct research into the different user environments and the corresponding user types and their policies. In order to create a user account who can connect and interact to the AWS IoT core we discovered we needed a IAM administrator user this gives the IAM user Full access, this user then a permission policy applied, adhering to the directions given by *Set up your AWS account - AWS IoT Core* (2021) where it advises to use AdmistratorAccess.

- Thing Creation and Connection kit installation
  In order to create, register and connect a thing to aws we applied some research on youtube Electronics Innovation (2019) In this tutorial there was a selection of os system and SDK kits as we were using a Mac computer the Linux os was suited to our project and as we had more experience in python we selected this for our developer kits.

- Device Configuration
  After following the video tutorial which involved downloading and unzipping a linux connection kit.We needed to configure our device in the video this involved placing this connection kit into a folder which we could execute from the terminal we gave the folder execute permissions after which we could start the connection. After this our thing was registered and was available for continued interaction.

- Subscribing to MQTT
  To receive messages from our thing we conducted research into how to talk to our thing this led us testing our device. In the testing process we found there are many topic filters for corresponding sdk as we originally downloaded a python so we found the corresponding subscribe filter from a workshop Sacha, P. (2021) which was called sdk/test/python

- Publishing messages
  To receive messages to our device we researched into sending messages from a IoT device here we found an article on publishing MQTT to the aws IoT core at *Publish MQTT Messages to AWS IoT Core from My Device Using Python* (2020) again as per instructed we chose the sdk/test/python topic name. By then selecting publish on our IoT core we started receiving our device on our terminal.

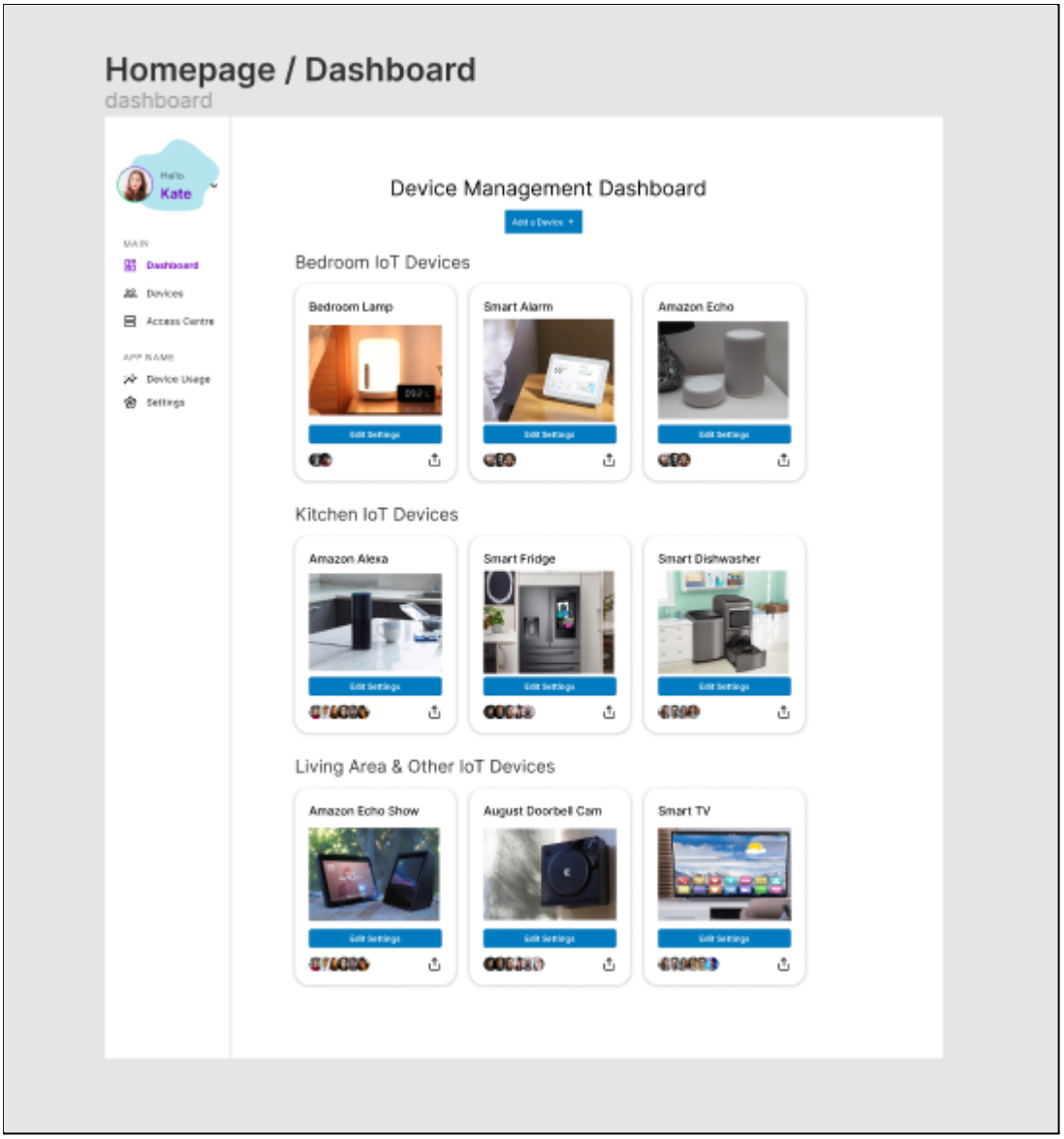
- Jobs
  We researched into AWS IoT jobs and found that a job is a remote operation that is sent to one or more IoT devices that are connected to AWS IoT and the job is executed on these devices. Jobs can be used to download and install application or firmware updates, reboot the device, rotate certificates, or perform remote troubleshooting operations. (Jobs - AWS IoT Core, 2021)

- Tunnels
  We conducted research into secure tunneling which is a feature in AWS IoT device management. We found that secure tunneling provides a secure remote access solution that directly integrates with AWS IoT to allow you to remotely access your IoT devices from anywhere. The endpoint is secured with identity and access management and communication happens over transport layer security (TLS). (Sirull, 2021)

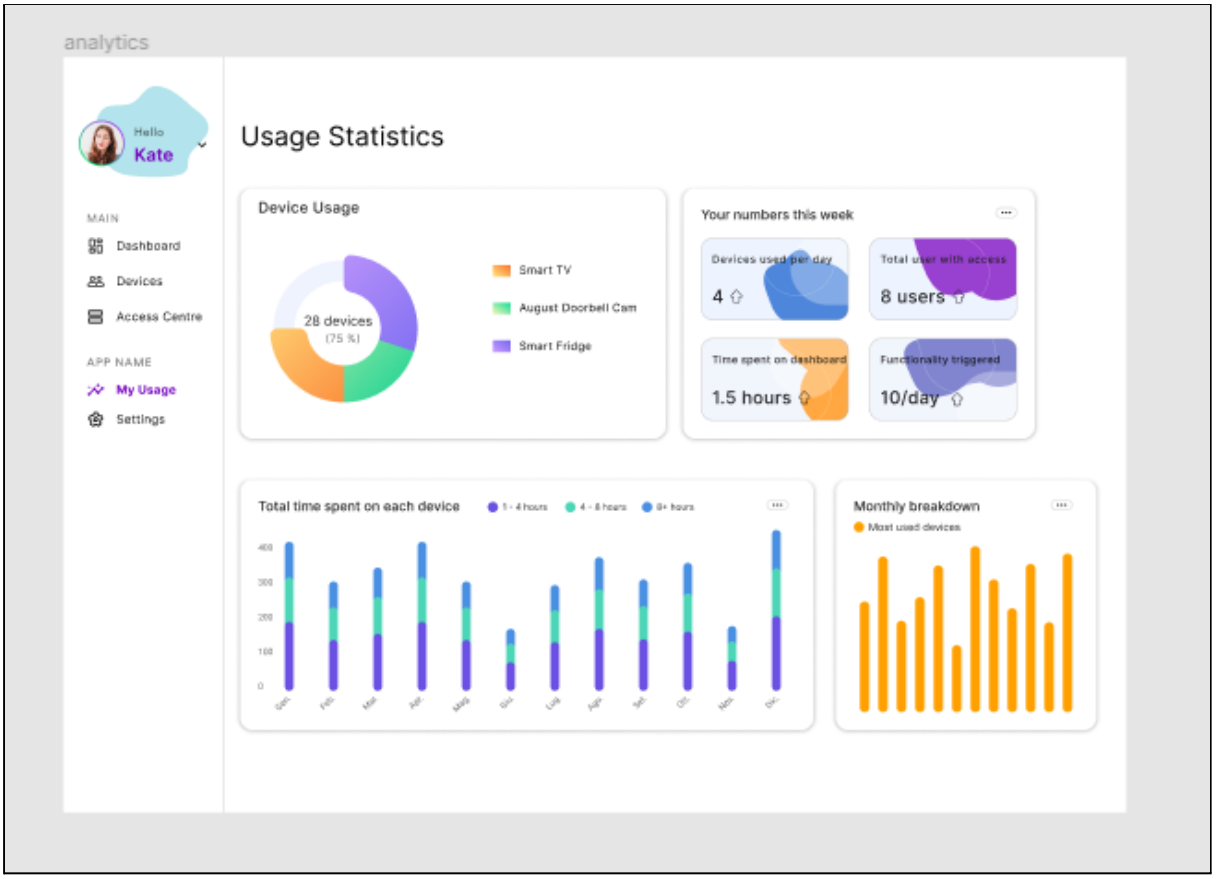
- Figma
  To display our prototype application we carried out research into web application tools, we found an interface design tool called figma. To learn the skills to appropriately design the prototype interface we would like to design for a IoT web application we followed some lesson on *Learn Design with Figma* (2021)

Design (Michael & Karl):

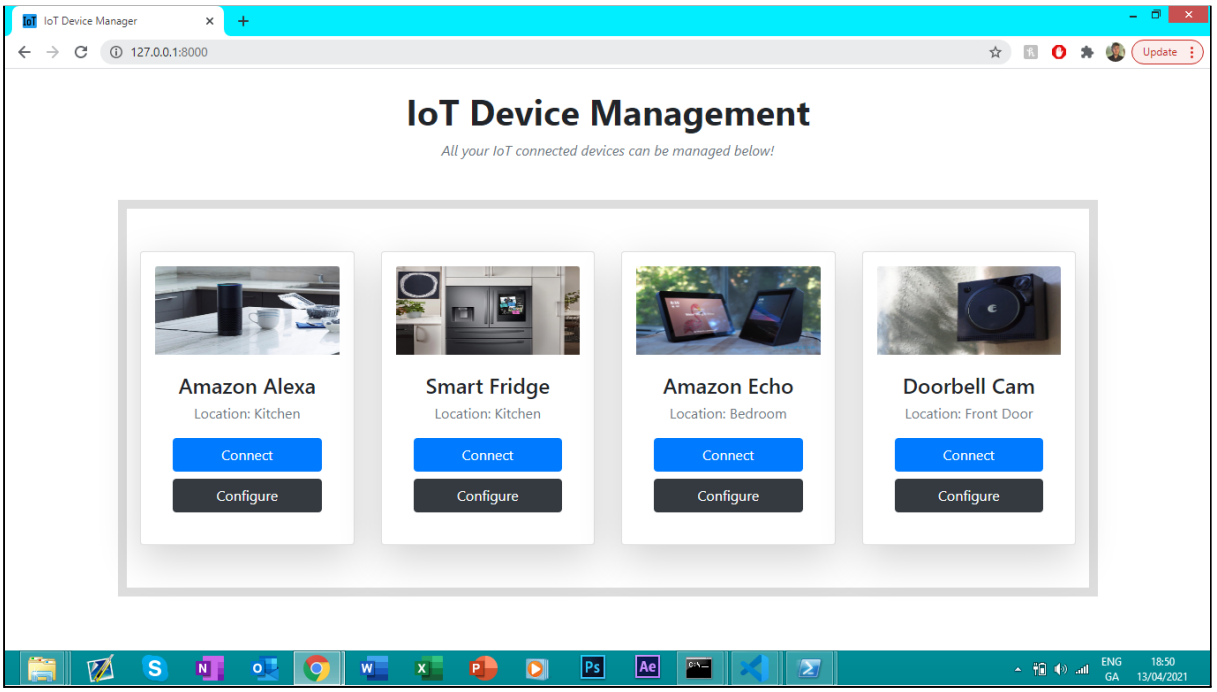
- We developed a prototype using Figma which is a web-based vector graphics editor and prototyping tool.
- This prototype could potentially be created with the use of the AWSIoTCoreSDK along with other languages and tools such as Django or Flask, HTML, CSS and Javascript.
- We first created a dashboard which could be used on a web-app or as a mobile application in order to manage IoT devices in a household. This can be seen in the image below. Users could potentially edit the settings of each IoT device in their household in order to register, organize, monitor, and remotely manage their IoT devices in the cloud. You could also potentially have functionality to see which users have access to each device as seen underneath the ‘Edit Settings’ button and also manage this through the ‘Access Centre’ as seen on the left hand side of the dashboard.



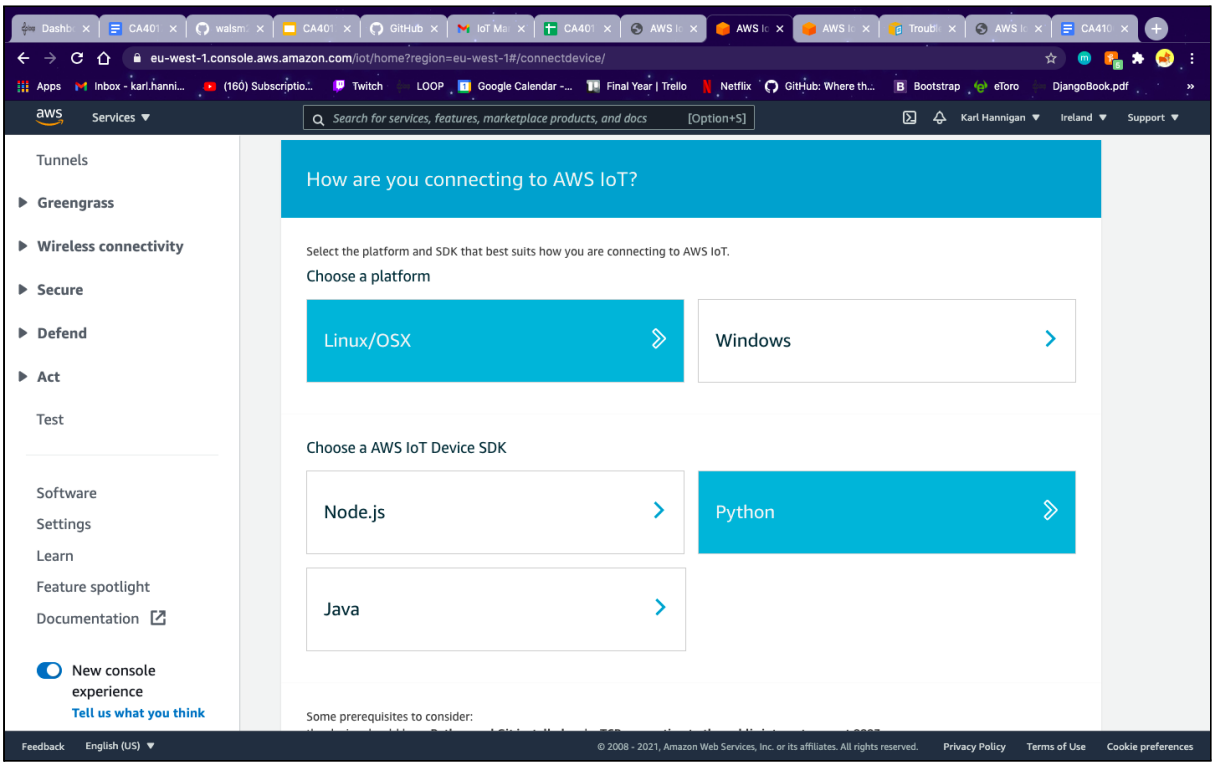
- We also created a prototype of a ‘Usage Statistics’ page where the user of the application would be able to view a range of different stats related to their IoT devices such as average number of devices used per week, total number of new users given access to devices, time spent using the IoT Device Management Dashboard, and the number of functionality triggered per day.



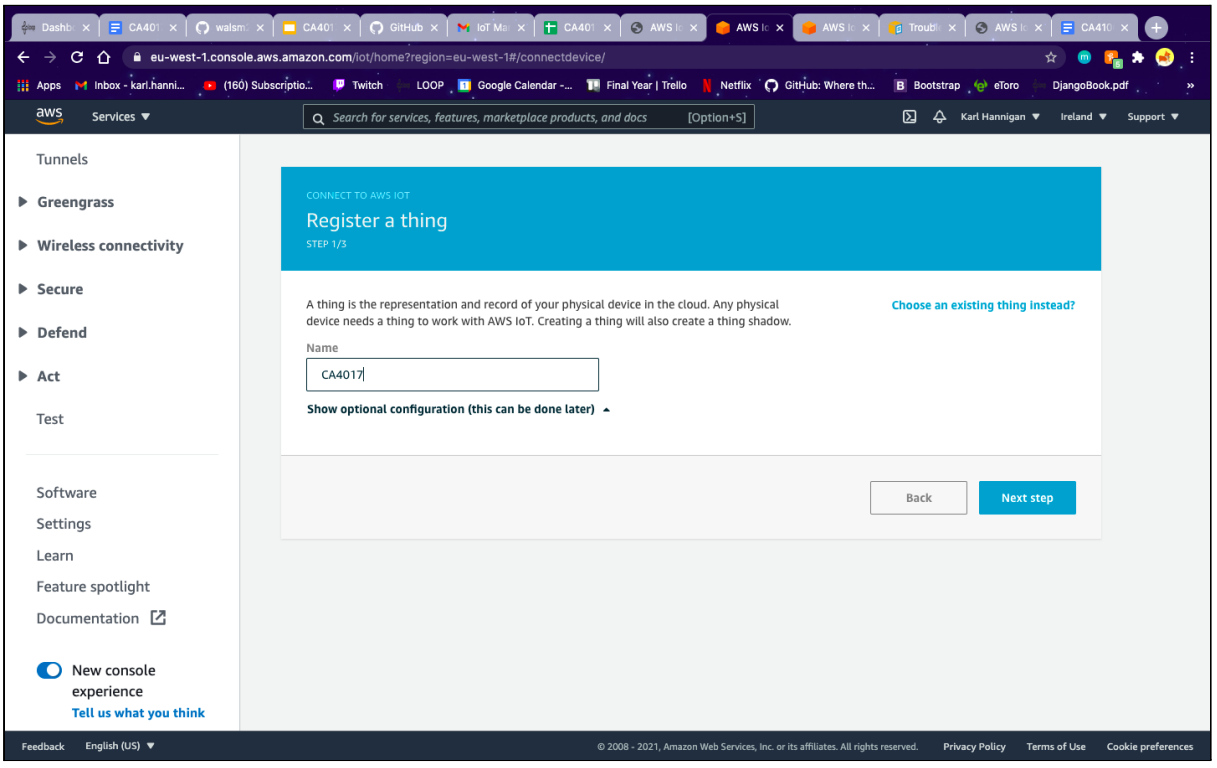
- We then decided to develop an IoT Device Management application through the use of Django, HTML, CSS, Bootstrap and jQuery. This allows users to manage their IoT devices directly through the application and send messages using MQTT straight to the AWS IoT Core console. This can be seen below. The connect button begins sending messages to AWS.



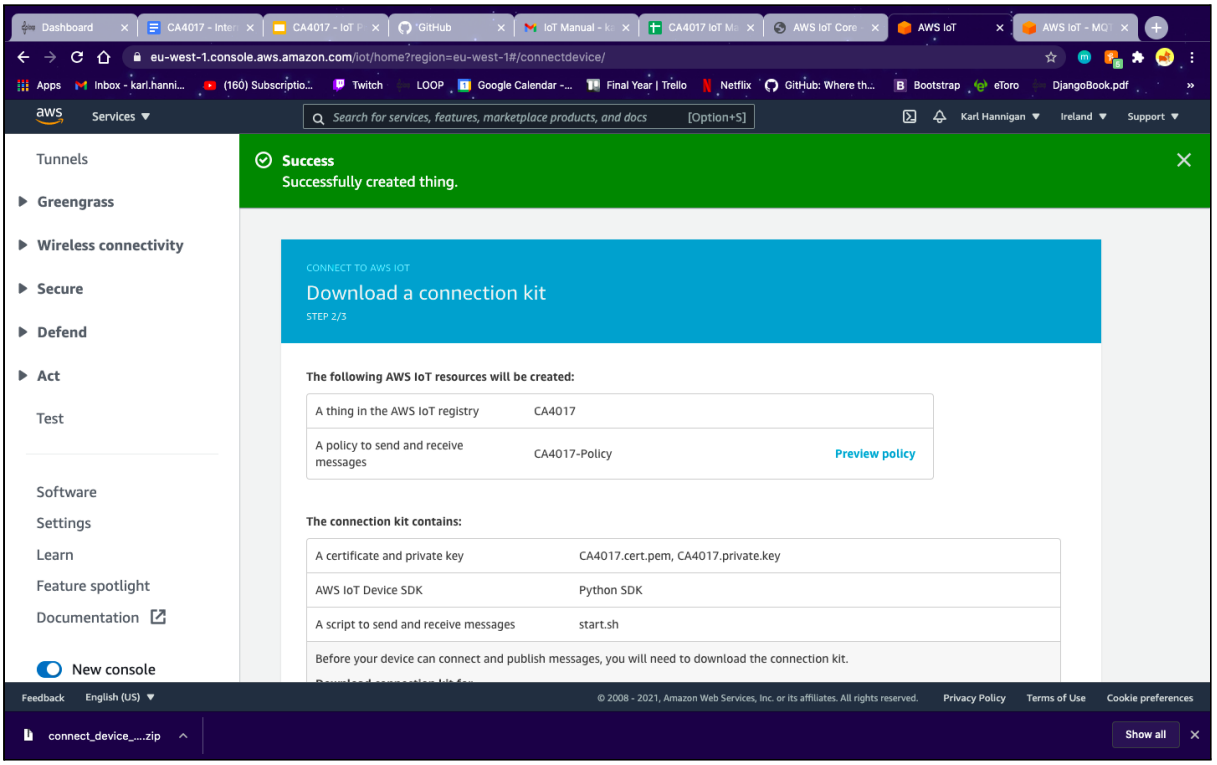
Deployment to the Cloud (Karl):
Firstly we had to choose Platform and the language of the AWS IOT Device SDK that we wanted to use. Linux and Python were the most suited to us.



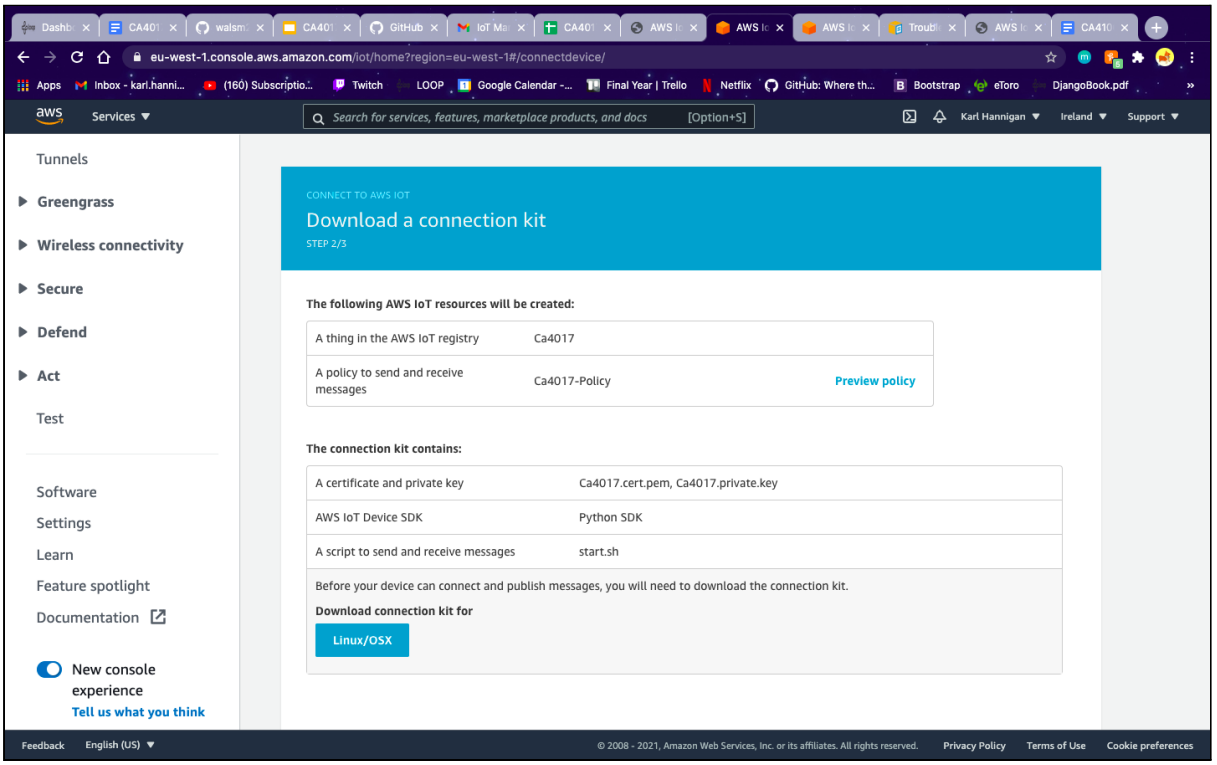
We then had to name the thing and we decided to name it CA4017 after the module code.



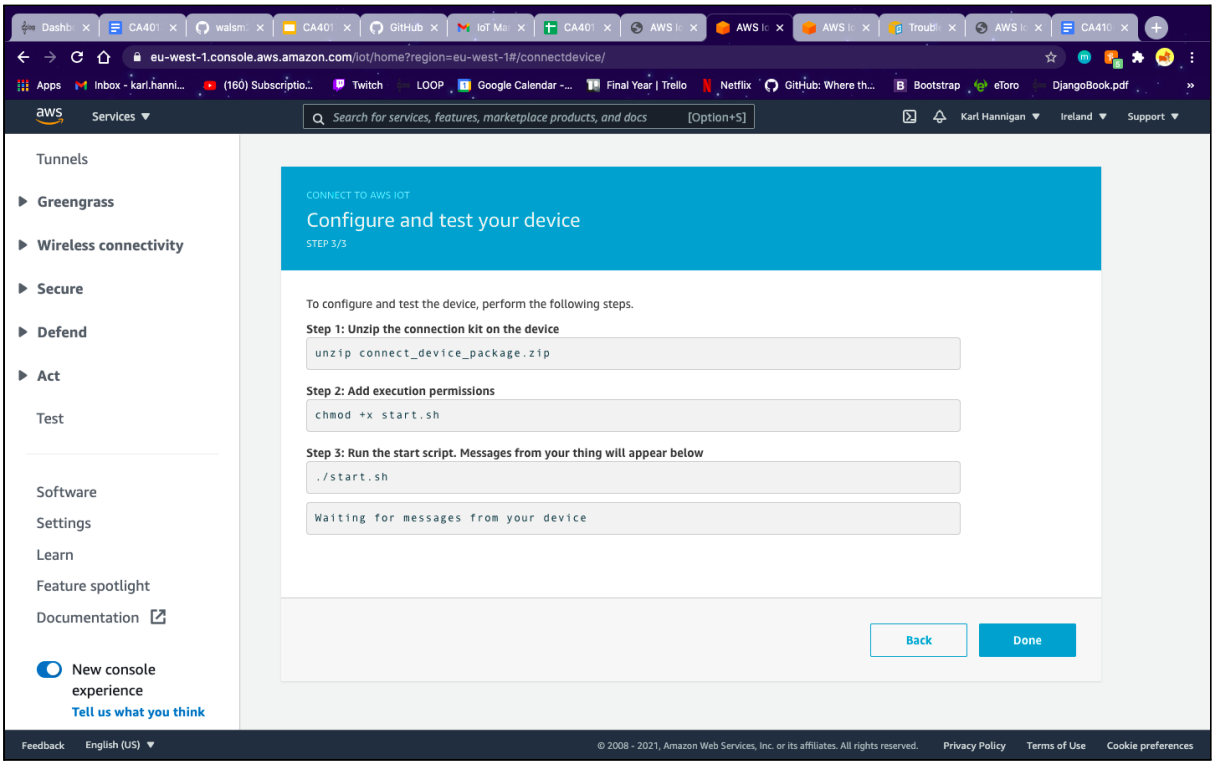
Once we had named the thing we were presented with this ‘success’ message telling us that we had successfully created the thing.



The next step was to download the connection kit on your device.



Once the connection kit was downloaded we had to use the following commands to unzip the package that was downloaded in the previous step, add execution permissions and then run the script. Screenshots of these commands are shown below.



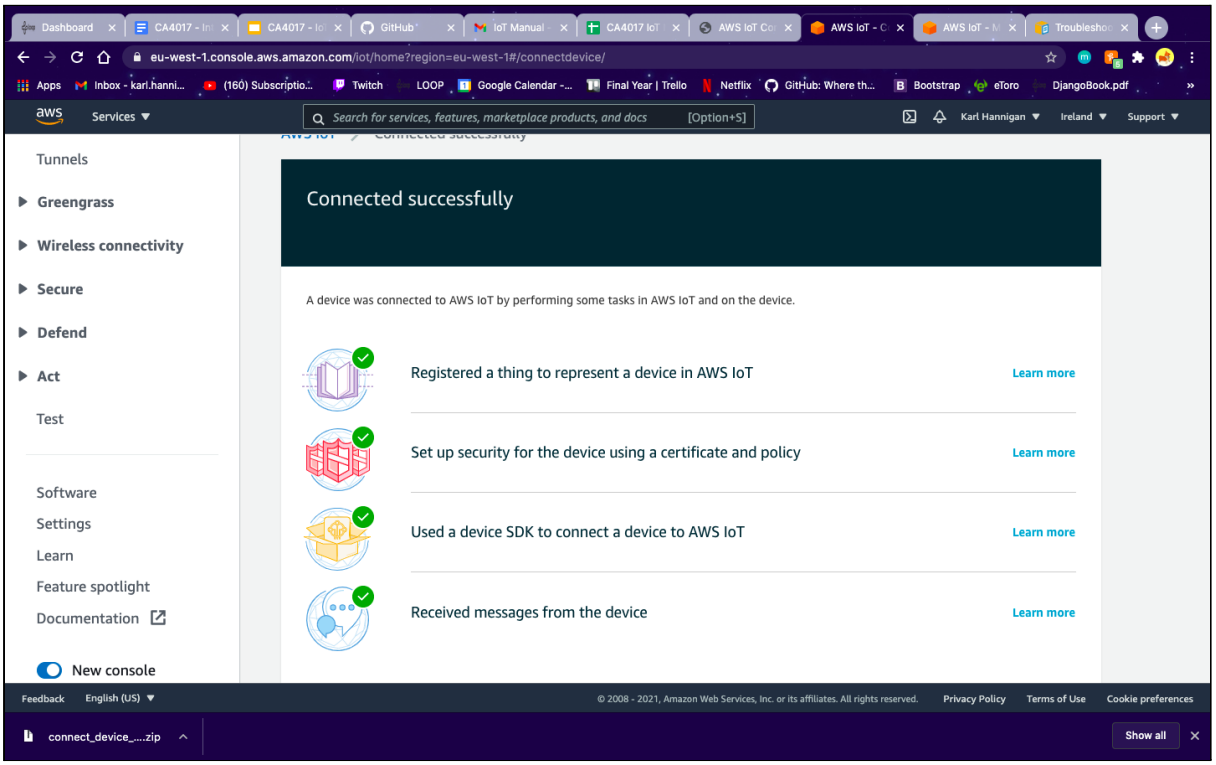
We first had to navigate to my desktop where the connection kit was downloaded to. Then navigated to the directory it was in and ran the unzip command. Then using the chmod allowed us to add execution permissions. The final step was to run the scripts using the ./start.sh command.

```
[Karls-MacBook-Air:Desktop karlhannigan$ cd Internet\ of\ Things/
[Karls-MacBook-Air:Internet of Things karlhannigan$ ls
 connect_device_package.zip
[Karls-MacBook-Air:Internet of Things karlhannigan$ unzip connect_device_package.zip
 Archive:  connect_device_package.zip
   inflating: CA4017.private.key
   inflating: CA4017.public.key
   inflating: CA4017.cert.pem
   inflating: start.sh
 Karls-MacBook-Air:Internet of Things karlhannigan$ chmod +x start.sh
[Karls-MacBook-Air:Internet of Things karlhannigan$ ./start.sh
[
 Downloading AWS IoT Root CA certificate from AWS...
   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                  Dload  Upload   Total   Spent    Left  Speed
 100  1188  100  1188    0     0   9428      0 --:--:-- --:--:-- --:--:--  9428

 Cloning the AWS SDK...
 Cloning into 'aws-iot-device-sdk-python'...
 remote: Enumerating objects: 40, done.
 remote: Counting objects: 100% (40/40), done.
 remote: Compressing objects: 100% (31/31), done.
 remote: Total 394 (delta 7), reused 26 (delta 5), pack-reused 354
 Receiving objects: 100% (394/394), 221.31 KiB | 3.07 MiB/s, done.
 Resolving deltas: 100% (164/164), done.

 Running pub/sub sample application...
```

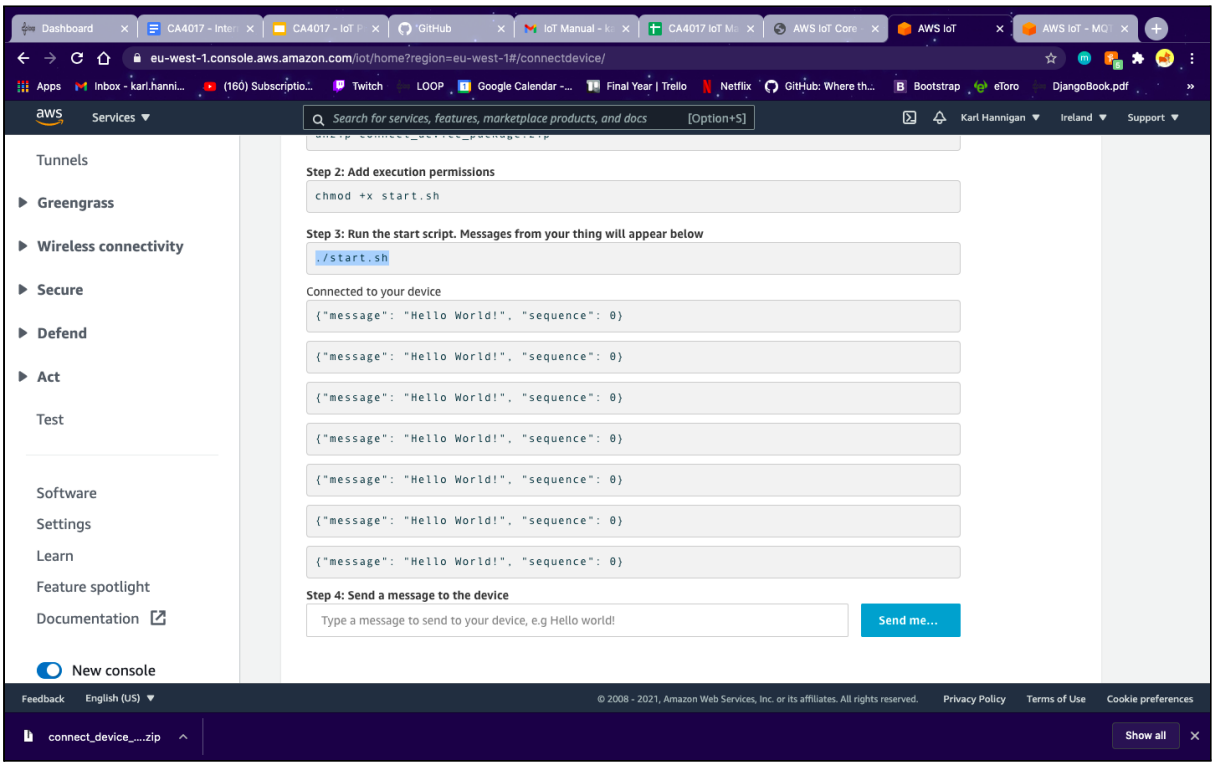
The following was outputted after running the script.

```
Received a new message:
{"message": "Hello World!", "sequence": 0}
from topic:
sdk/test/Python
--------------


2021-04-08 15:37:38,183 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2021-04-08 15:37:39,181 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
2021-04-08 15:37:39,182 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Filling in custom puback (QoS>0) event callback...
2021-04-08 15:37:39,197 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [puback] event
2021-04-08 15:37:39,199 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [puback] event
2021-04-08 15:37:39,200 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2021-04-08 15:37:39,200 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - This custom event callback is for pub/sub/unsub, removing it after invocation...
2021-04-08 15:37:39,222 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2021-04-08 15:37:39,222 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
Received a new message:
{"message": "Hello World!", "sequence": 1}
from topic:
sdk/test/Python
--------------


2021-04-08 15:37:39,222 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2021-04-08 15:37:40,219 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
2021-04-08 15:37:40,220 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Filling in custom puback (QoS>0) event callback...
2021-04-08 15:37:40,234 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [puback] event
2021-04-08 15:37:40,238 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [puback] event
2021-04-08 15:37:40,238 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2021-04-08 15:37:40,238 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - This custom event callback is for pub/sub/unsub, removing it after invocation...
2021-04-08 15:37:40,261 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2021-04-08 15:37:40,263 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
Received a new message:
{"message": "Hello World!", "sequence": 2}
from topic:
sdk/test/Python
--------------


2021-04-08 15:37:40,263 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2021-04-08 15:37:41,260 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
2021-04-08 15:37:41,260 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Filling in custom puback (QoS>0) event callback...
2021-04-08 15:37:41,275 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [puback] event
2021-04-08 15:37:41,276 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [puback] event
2021-04-08 15:37:41,276 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2021-04-08 15:37:41,276 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - This custom event callback is for pub/sub/unsub, removing it after invocation...
2021-04-08 15:37:41,302 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2021-04-08 15:37:41,306 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
Received a new message:
{"message": "Hello World!", "sequence": 3}
from topic:
sdk/test/Python
--------------
```

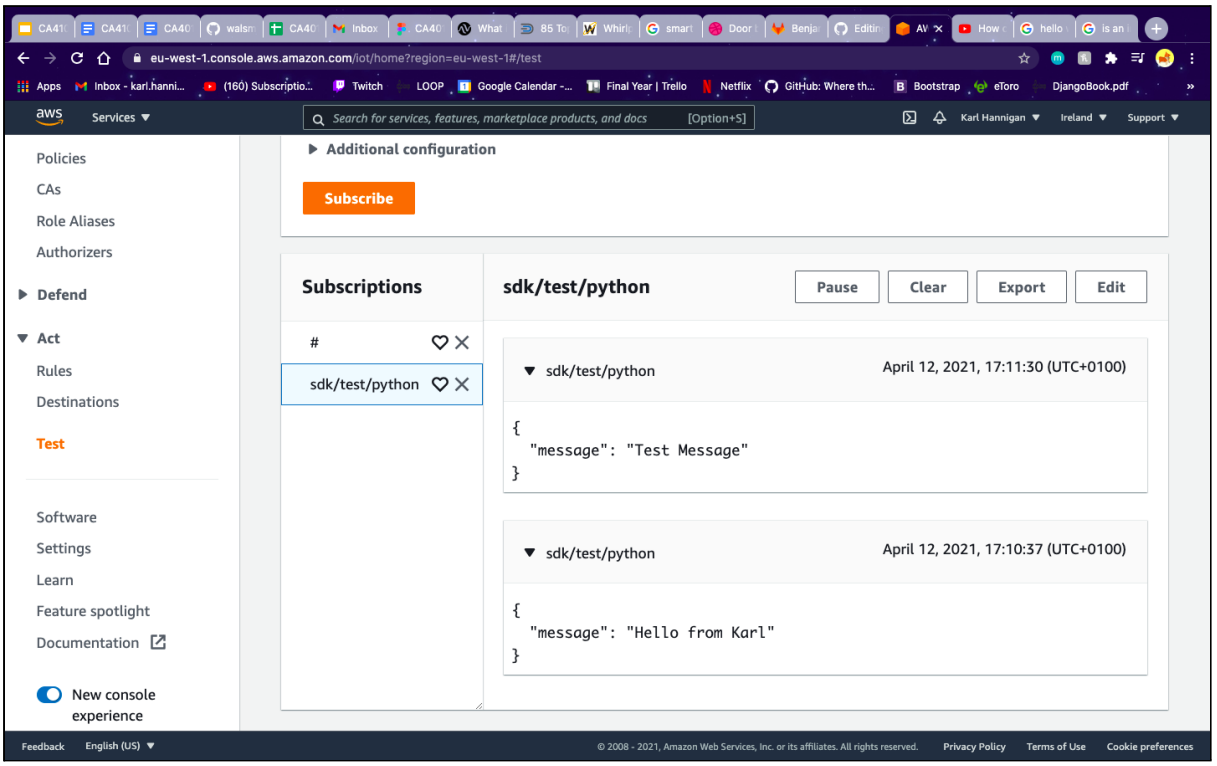
This was presented to us on the AWS console which informed us that we had successfully connected.



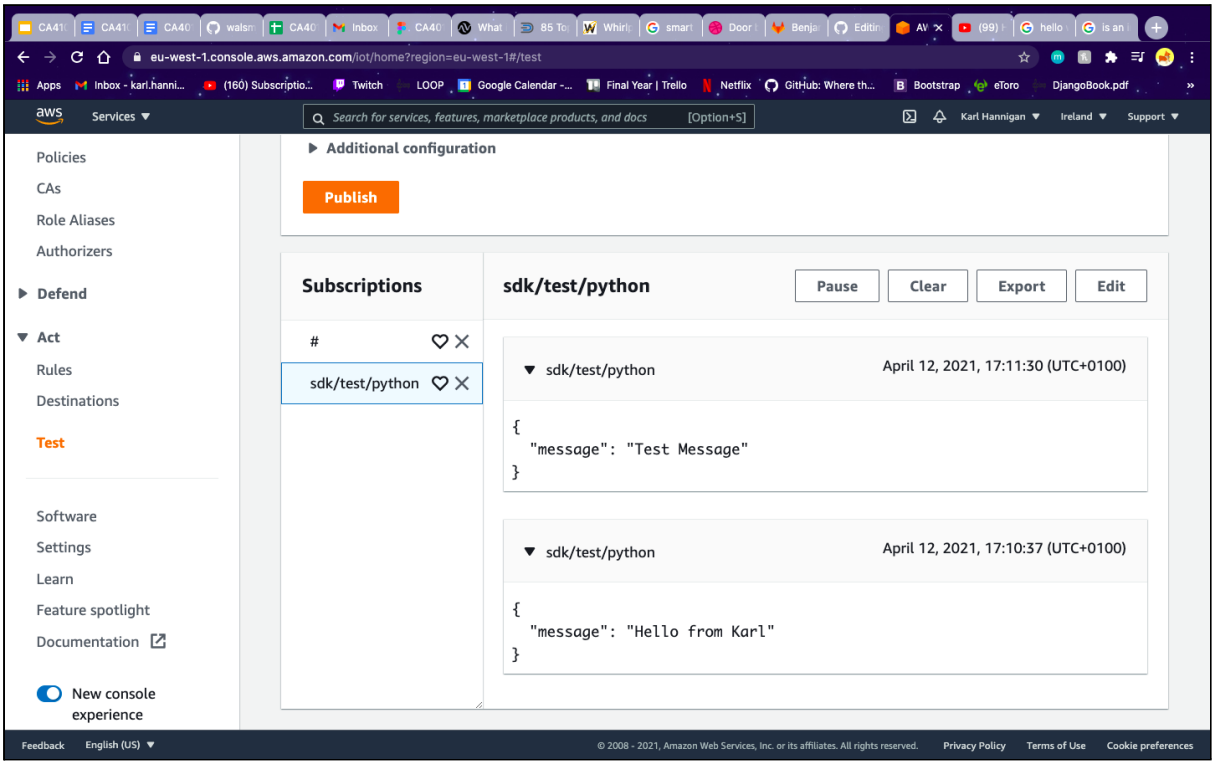
Proof of connection on the console



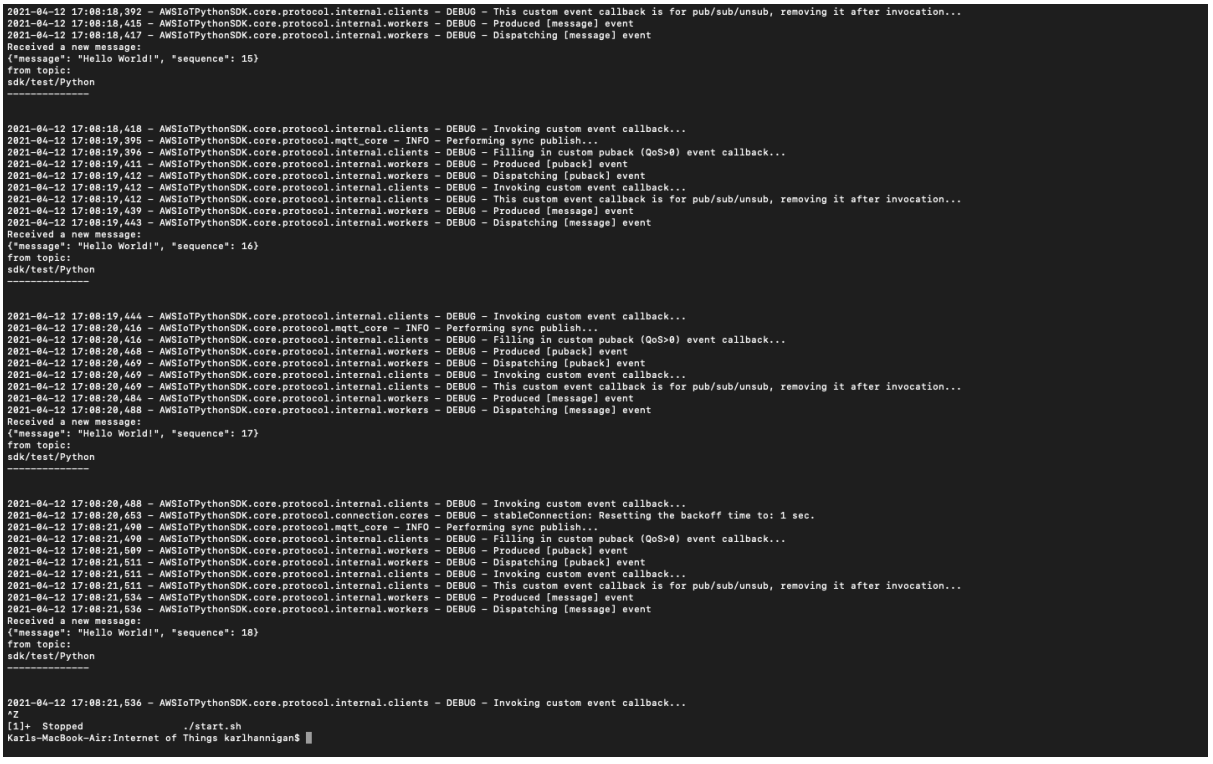
MQTT Subscribe



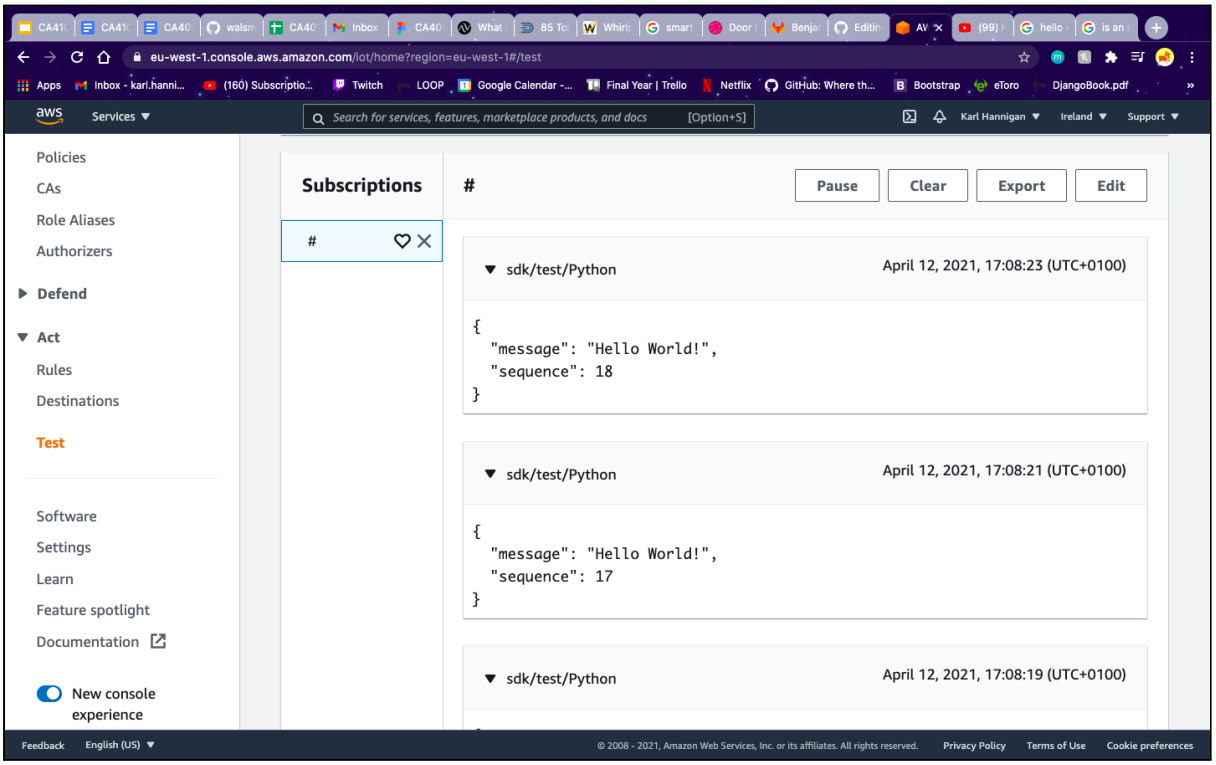
MQTT Publish



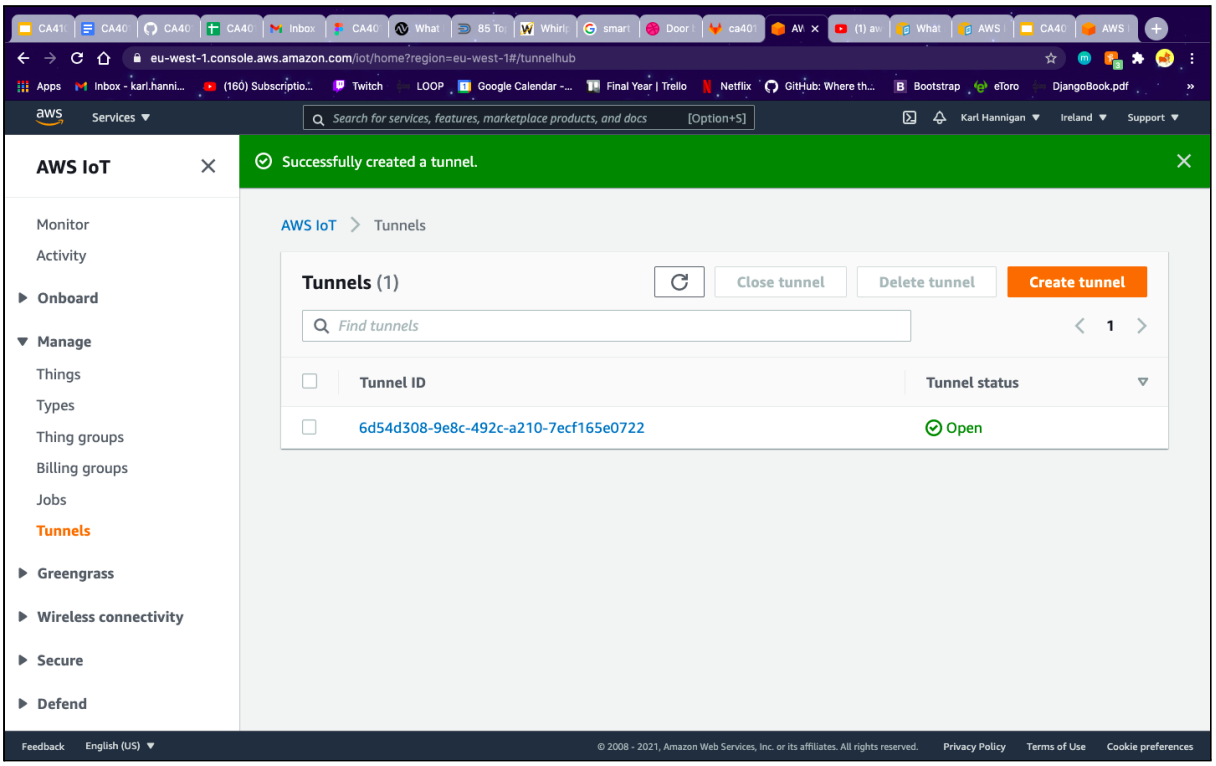
MQTT Subscribe Terminal - In order to do this we had to navigate to “things”, and click on the “interact” button. We then had to choose “connect a device” and follow the instructions. This allowed us to type custom messages which were displayed in the terminal.



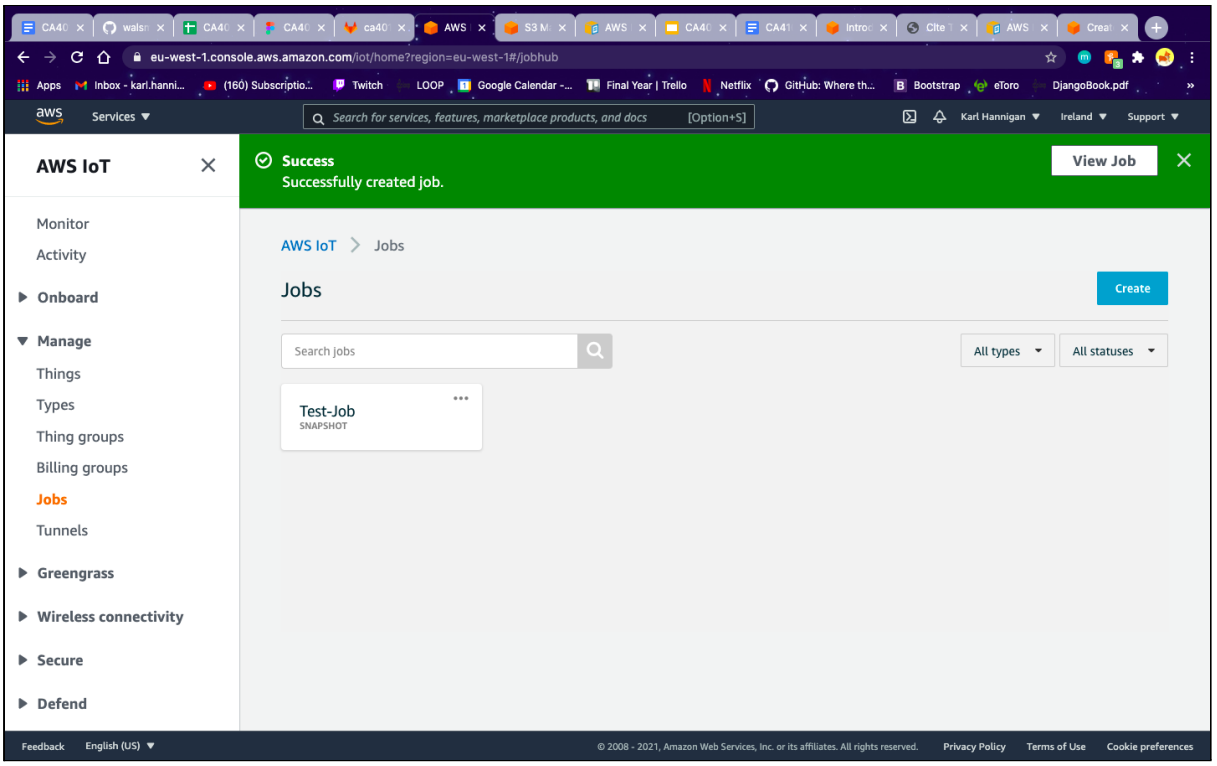
MQTT Subscribe on AWS showing messages from the terminal.



Successfully created a tunnel



Successfully Created a Job



References

Amazon Web Services (2013) *Creating an Amazon Web Services Account*. Available at: https://www.youtube.com/watch?v=WviHsoz8yHk (Accessed: 13 April 2021).

Docs.aws.amazon.com. 2021. *Jobs - AWS IoT Core*. [online] Available at: <https://docs.aws.amazon.com/iot/latest/developerguide/iot-jobs.html > [Accessed 13 April 2021].

Electronics Innovation (2019) *How to create a thing in AWS IoT Core, its Certificates & policies*. Available at: https://www.youtube.com/watch?v=0vbk8A58Qd8 (Accessed: 13 April 2021).

Learn Design with Figma (2021) *Figma*. Available at: https://www.figma.com/resources/learn-design/ (Accessed: 13 April 2021).

Sacha, P. (2021) *Send sensor data to AWS IoT :: AWS IoT SiteWise Workshop*. Available at: https://iot-sitewise.workshop.aws/aws-iot-core-basics/send-sensor-data-to-aws-iot.html (Accessed: 13 April 2021).

Set up your AWS account - AWS IoT Core (2021). Available at: https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html (Accessed: 13 April 2021)

Sirull, M., 2021. *Introducing Secure Tunneling for AWS IoT Device Management, a new secure way to troubleshoot IoT devices | Amazon Web Services*. [online] Amazon Web Services. Available at: <https://aws.amazon.com/blogs/iot/introducing-secure-tunneling-for-aws-iot-device-management-a-new-secure-way-to-troubleshoot-iot-devices /> [Accessed 13 April 2021].

Publish MQTT Messages to AWS IoT Core from My Device Using Python (2020) *Amazon Web Services, Inc.* Available at: https://aws.amazon.com/premiumsupport/knowledge-center/iot-core-publish-mqtt-messages-python/ (Accessed: 13 April 2021).