Final Project

Azure DataBox Emulator

TURI, François

Deep Azure@McKesson

Dr. Zoran B. Djordjević

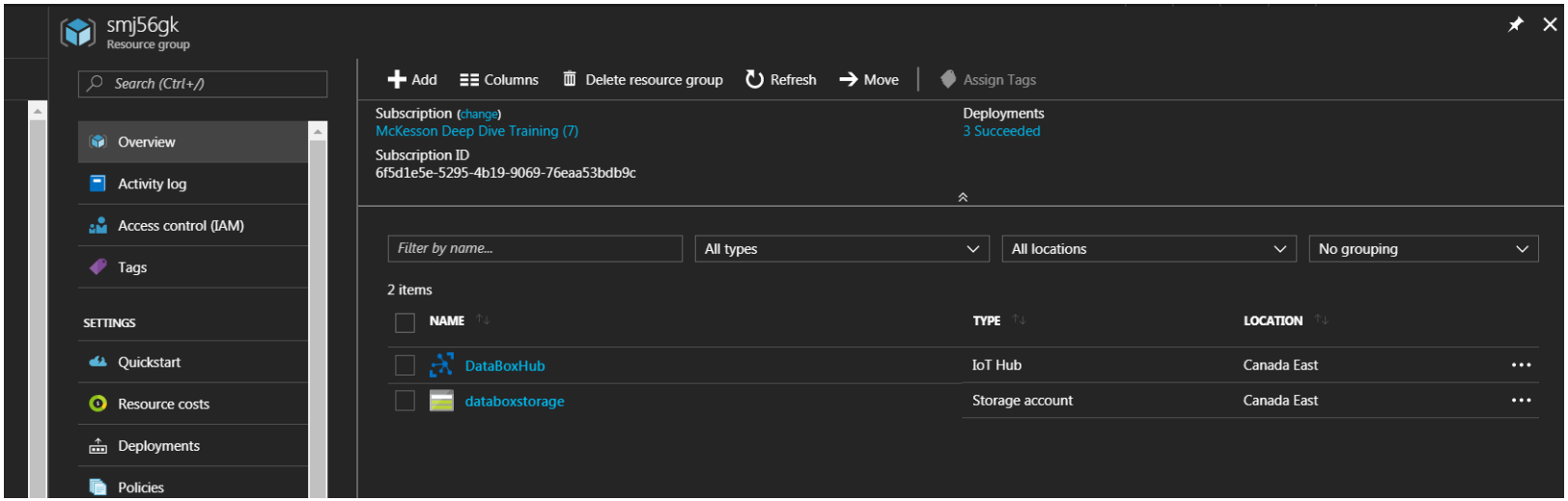
Introduction

- Microsoft is introducing a DataBox designed to upload hundreds of terrabytes from client to Azure by shipping basically a rugged computer

- Well, we are not going to play with that toy. We will create a simulator using IOT technology, geolocalisation and an emulator that will carry a single snowman photo.

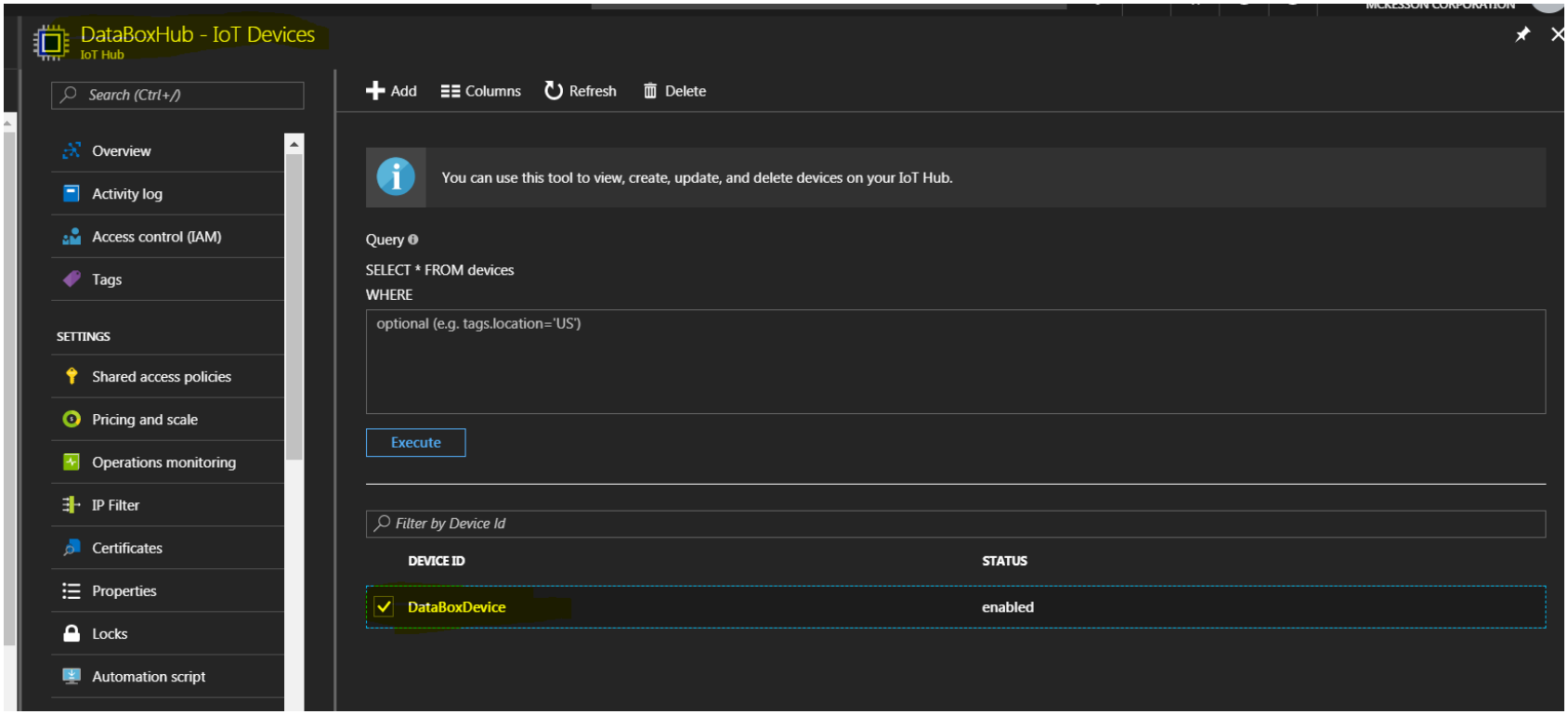
Azure Ressource-Group

- We are using two ressources:



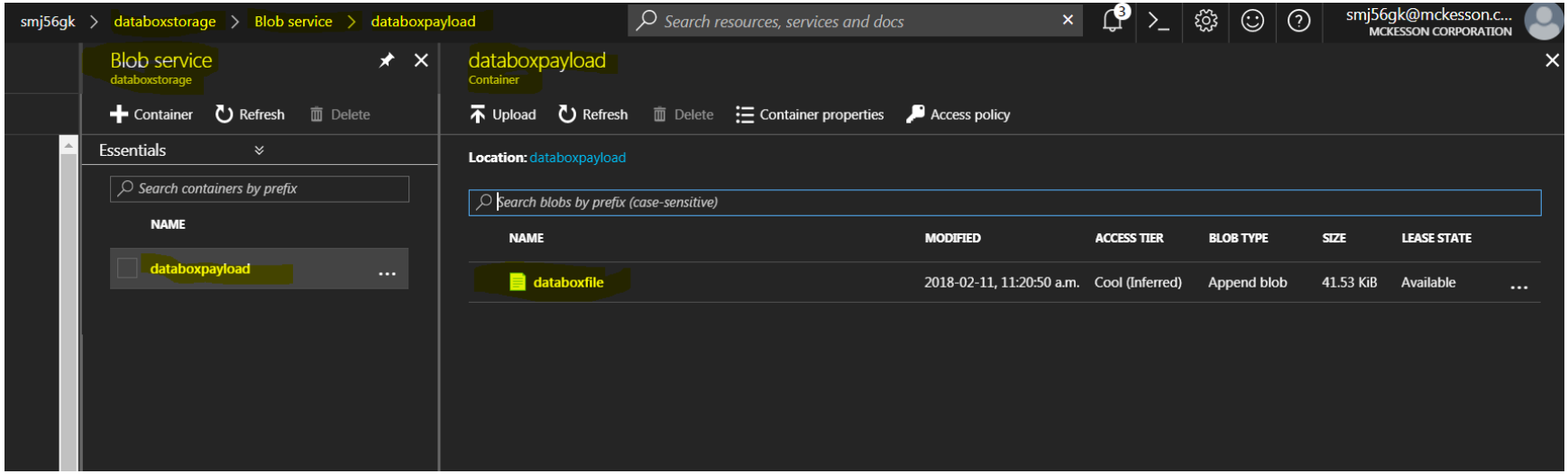
IoT DataBoxHub

- We publish a IoT Device inside the IoT DataBox Hub
- For the project we will use either the IOT hub AccessKey or The IOT device, depending of the library used



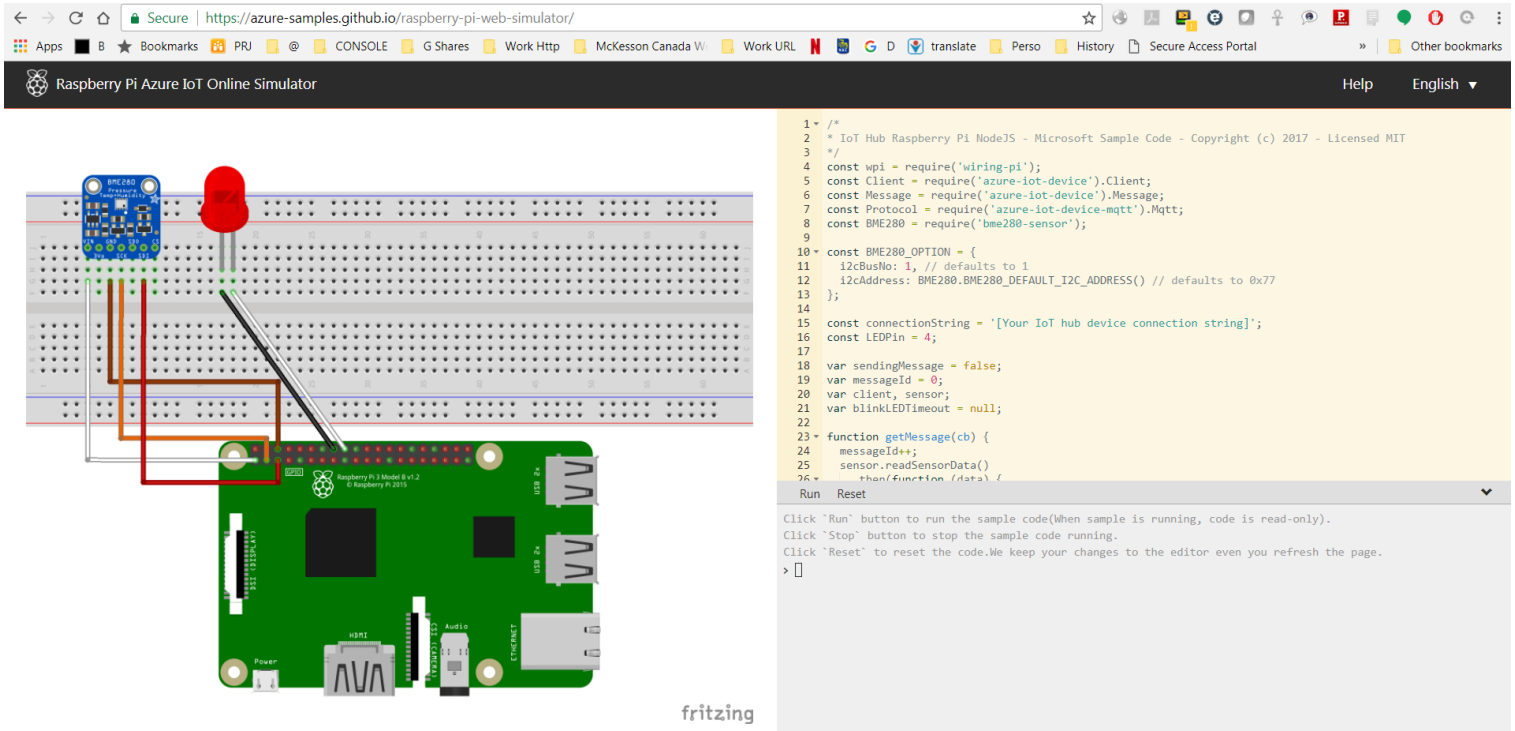
IoT Storage

- We also need a IoT Storage account, with a Blob Service, with a Blob Container (databoxpayload)
- The Blobfile databoxfile is created automatically by the code

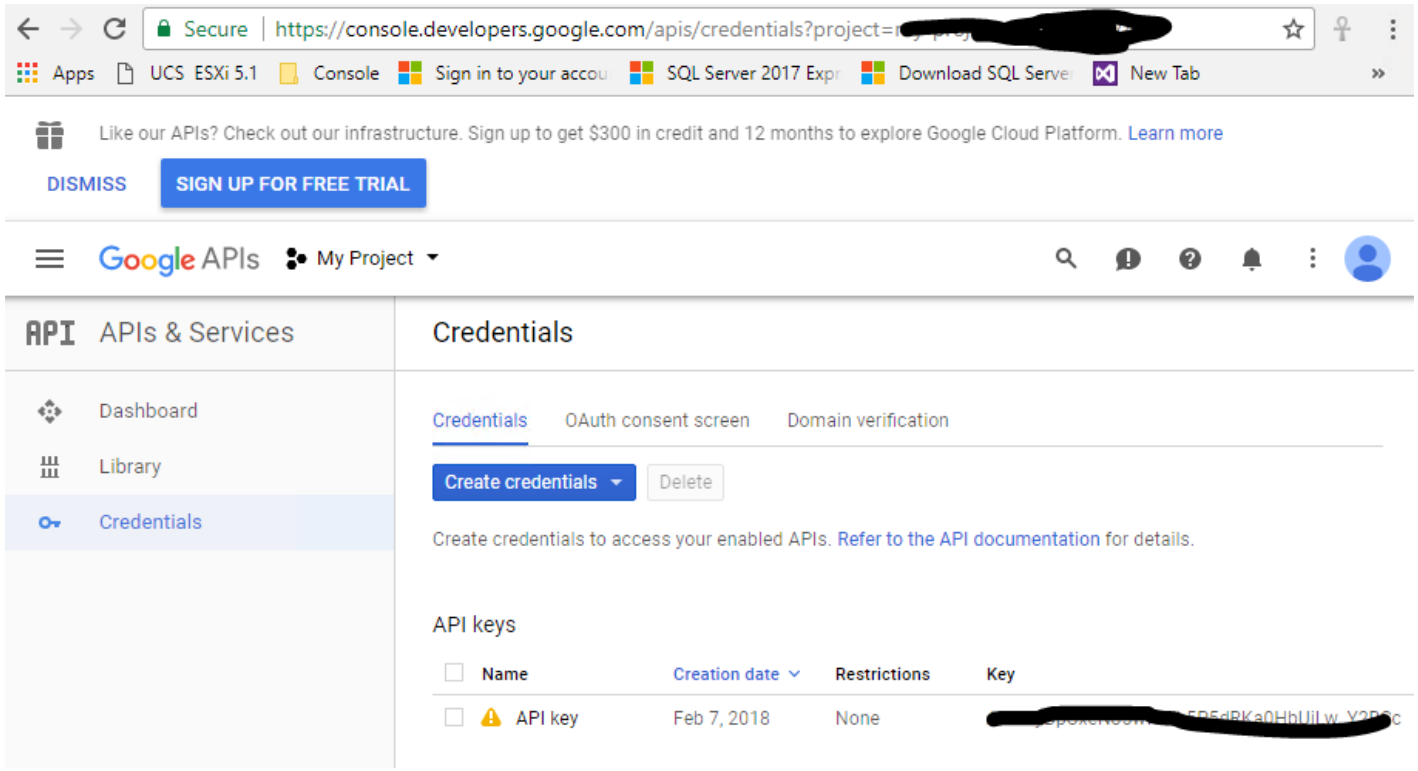


The Rasperry-Pi emulator

- There is a very nice looking raspberry emulator provided by Azure as git repository:
  We will install, and adapt this emulator to create a `functioning ` databox



Google Maps

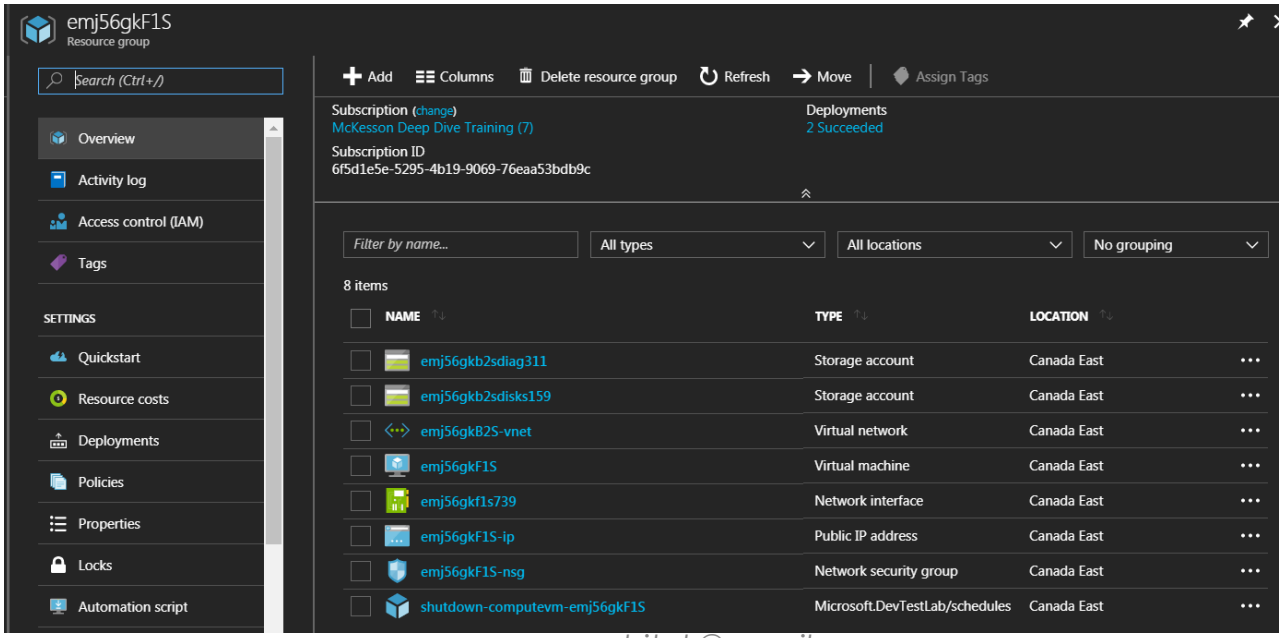
- We also need a google api Keys using this procedure https://developers.google.com/maps/documentation/javascript/get-api-key
- And visible after here:
- 

VM, Node.js host

- At last we need a machine to host the three Node.js server and also the HTTP server

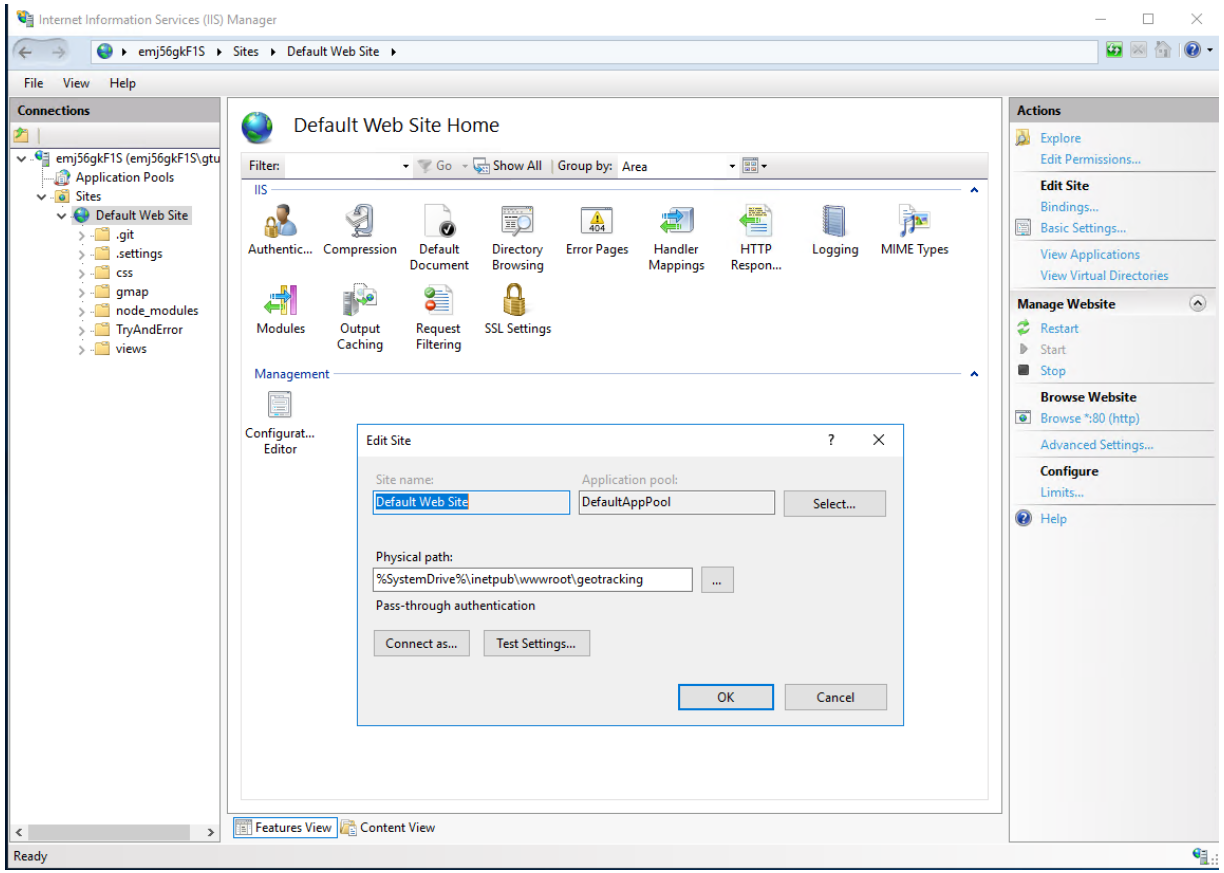
- As we use Eclipse to develop the code, we decided to use a Windows10 VM on Azure cloud

- It probably doesnt matter where the Node.JS is residing. I do believe that a redhat linux server will do the job with an Apache http server.



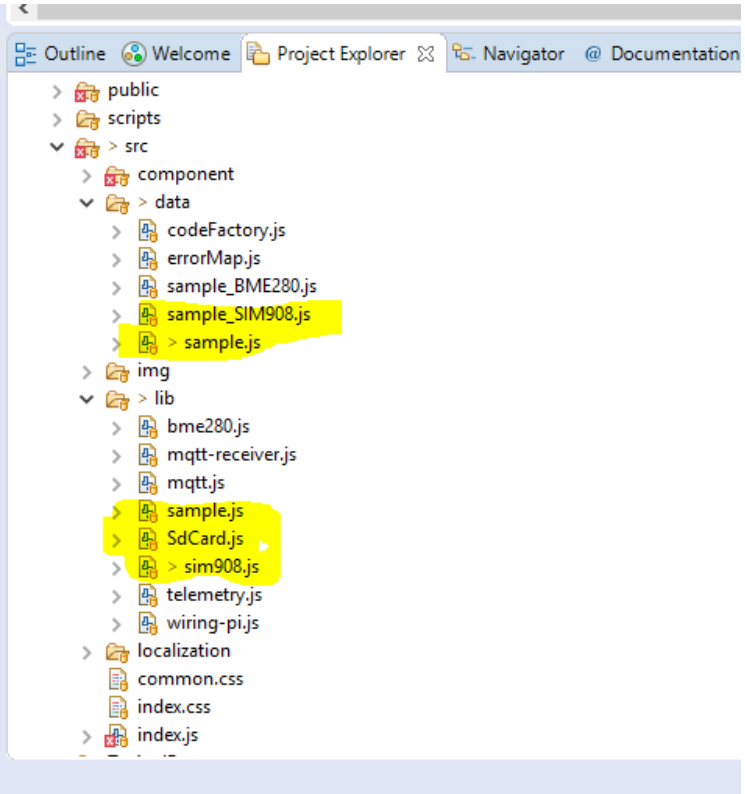
HTTP server

- We need an HTTP server to host the static pages of the tracker node.js application and the icons for the console node.js application
- For simplicity, we used IIS. The Node.js and IIS use the same root



Raspberry-Pi Emulator Technology

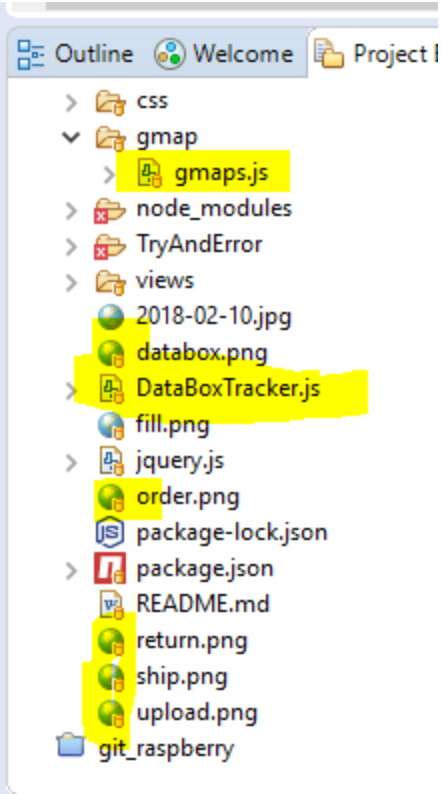
- The emulator is in fact a Native-react project **hosted** on top of a node.js
- The right part of the window is actually the heart of this product and is basically a pseudo language (using **pseudo** node.js syntax). This part will be largely changed to fit our need



The Tracker

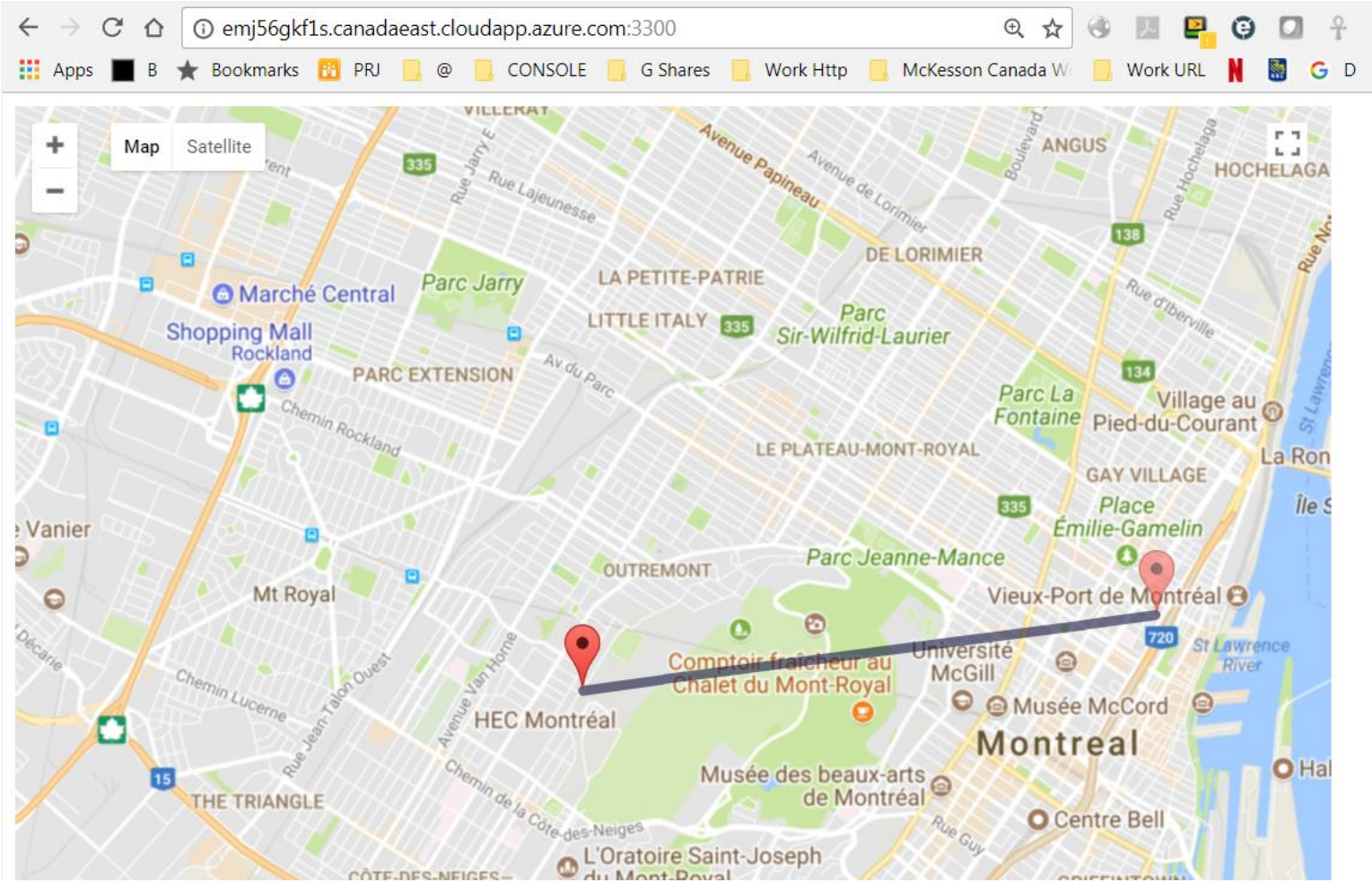
- We will use the Gmaps.js library wrapper to generate a google map with marker. Nice library ☺
- The rest of the project is a basic Node.js Express (http) & Mqttp & Azure storage
- The Azure storage is due to the fact that the raspberry written in Native-React does not support to use node.js azure library. So we have to hop on that node.js code to forward the file from the simulated Node.js disk

- We also map the IIS http server on the same directory for static images (the fill, upload icons..)



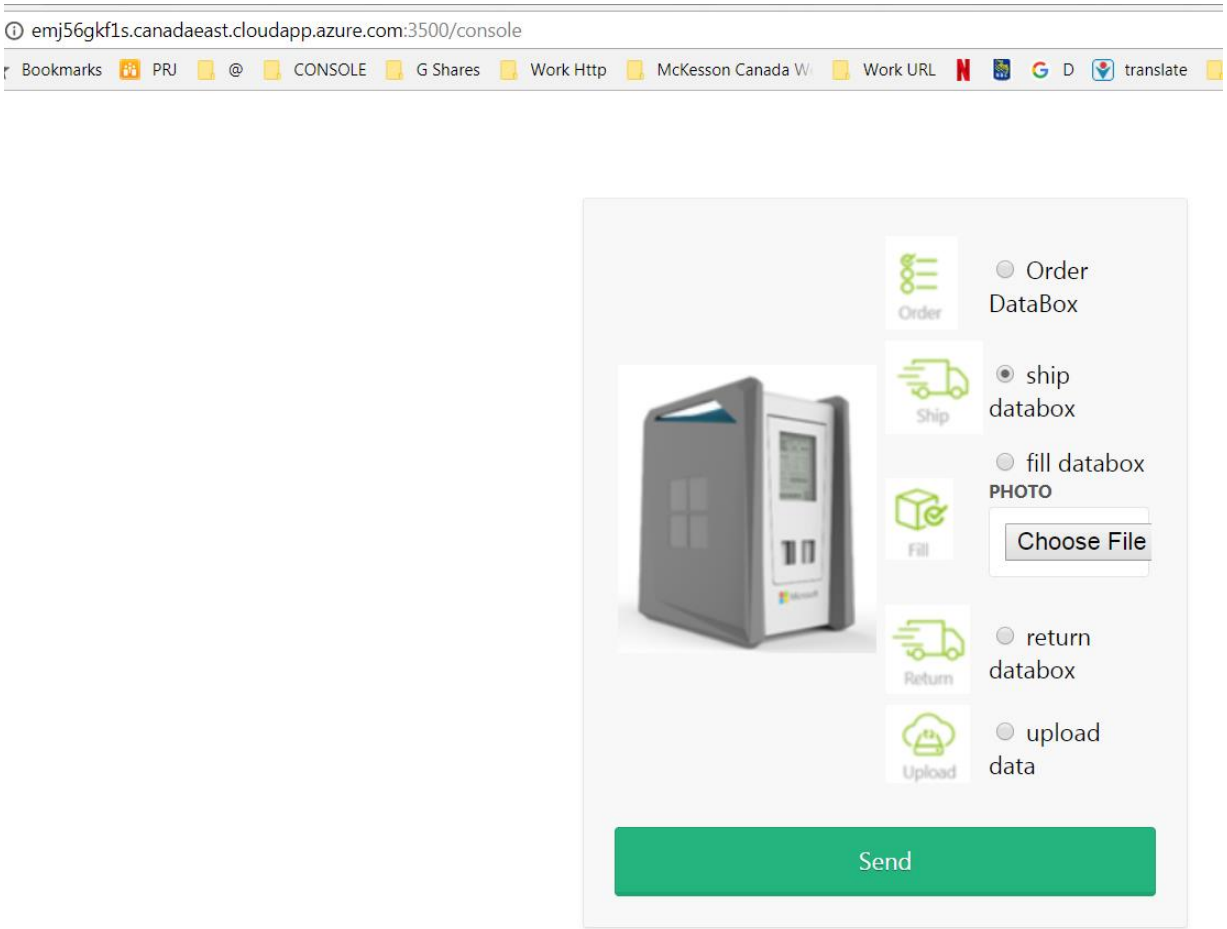
Tracker in action

- The light mark show the origin of the databox. Map can be zoomed in



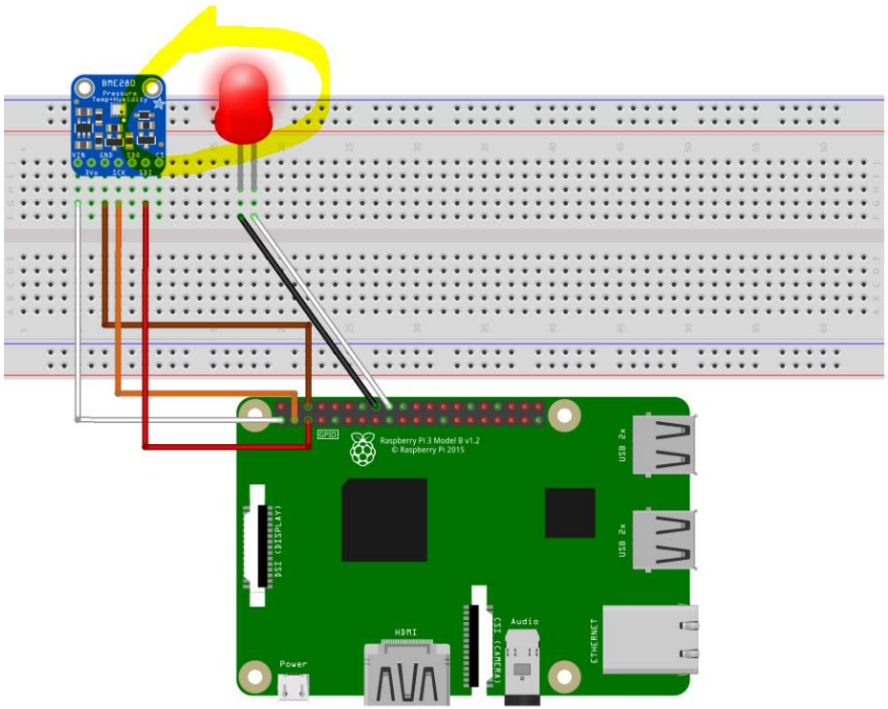
DataBox Console

- We also implemented a DataBox console (simple html forms allowing file to be uploaded). Node.js doesnt allow it to be simple…



Emulator in action

- While the databox is moving, the Led is flashing
- Console is also showing what is happening



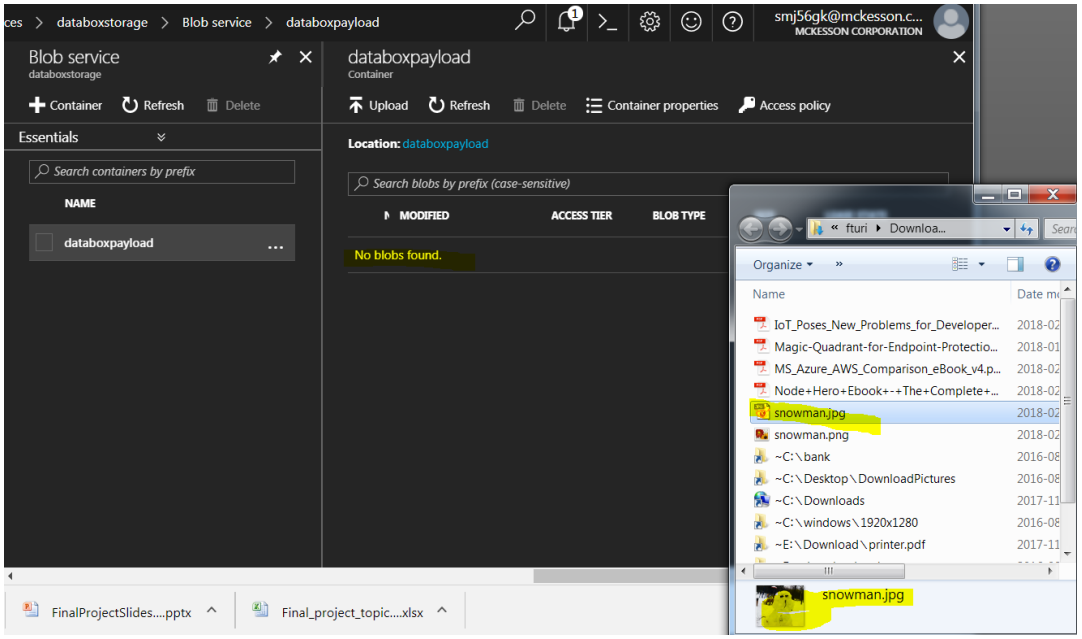

```
1  /*
2  * IoT Hub Raspberry Pi
3  */
4  const wpi = require('w
5  const Client = require
6  const Message = requir
7  const Protocol = requi
8  //const BME280 = requi
9  const SIM908 = require
10 const SDCARD = require
11 const IOTHUB = require
12
13 const SIM908_OPTION =
14   i2cBusNo: 1, // defa
15   i2cAddress: SIM908.S
16 };
17
18 const SDCARD_OPTION =
19   i2cBusNo: 2, // defa
20   i2cAddress: SDCARD.S
21 };
22
```

Stop Reset

```
order is: fill
>
file is:[object Object],jspon
>
result: ok(4)
>
msg
>
order is: return
>
result: ok(5)
>
```

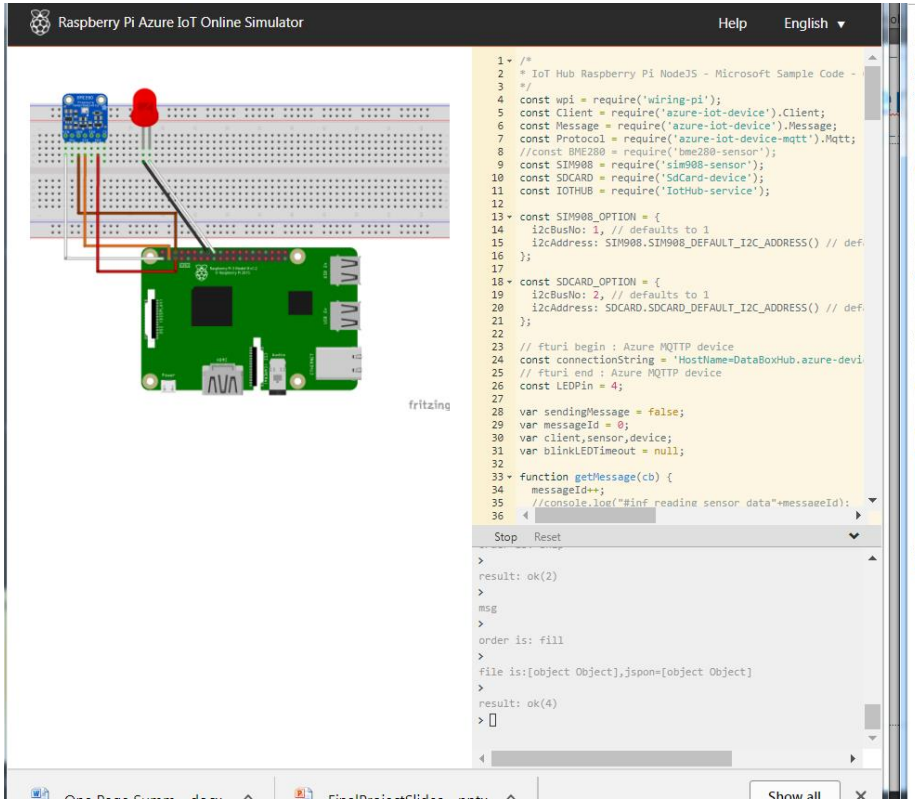
Save Private SnowMan

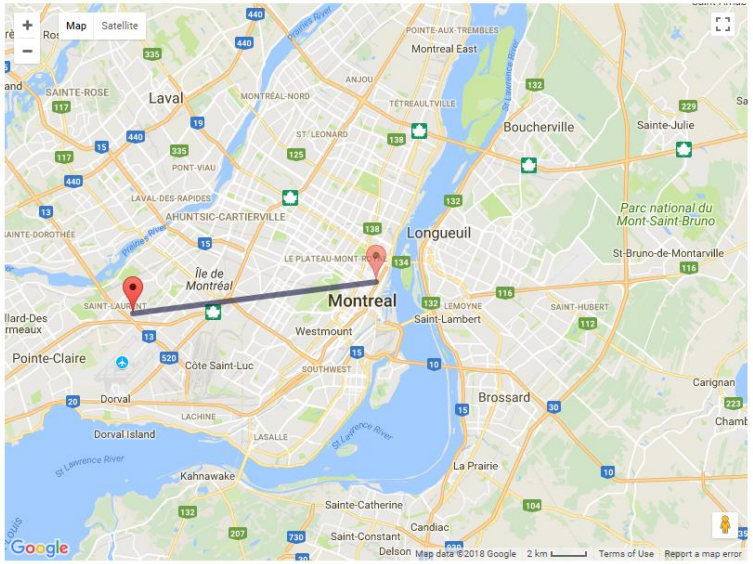
- Our mission is to upload the snowman to the databox, fly it through montreal on teh emulator SdCard and deposit it on Azure



Order, Shipping & FillUp

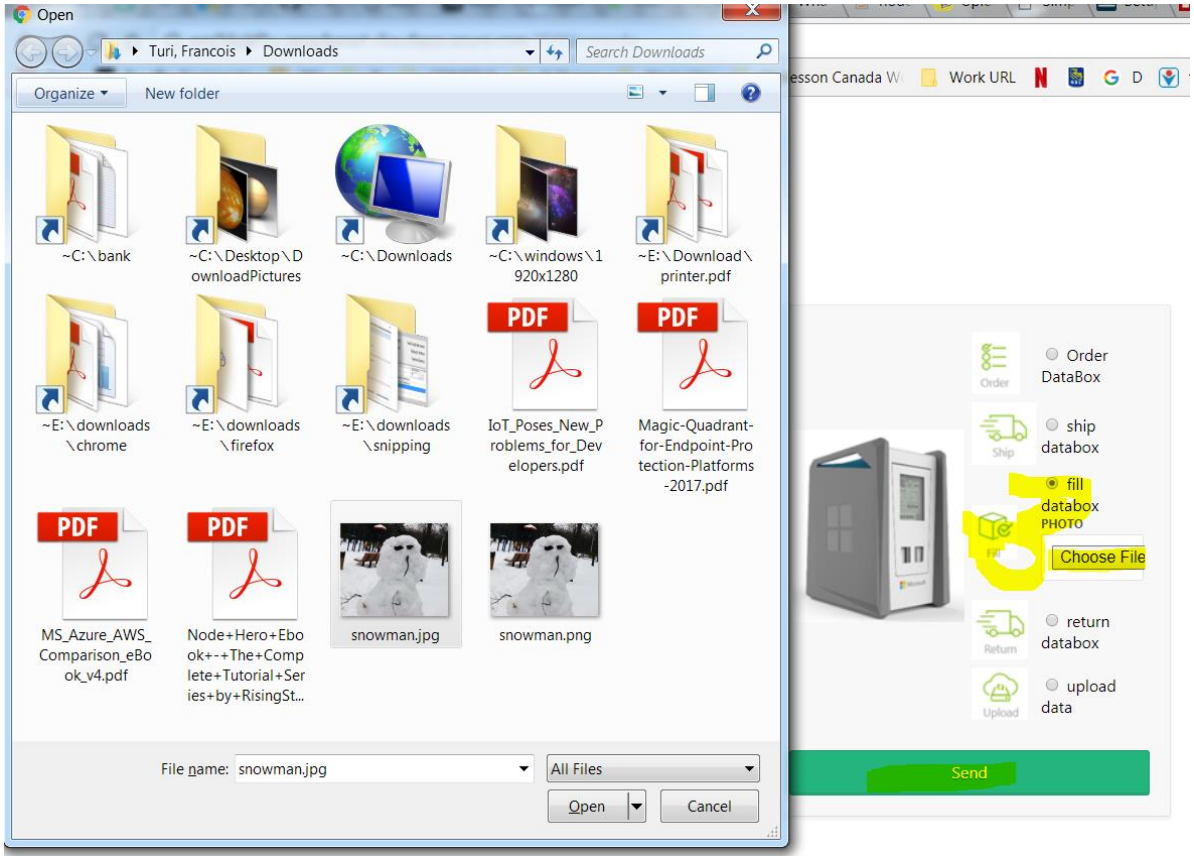
- In the first step we order the DataBox, wait for arrival and Fill it up





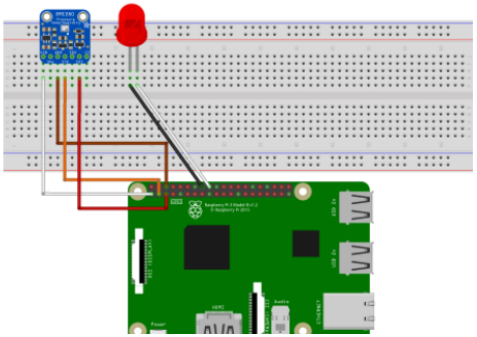
Console Fillingup

- We also embark Private Snowman on-board

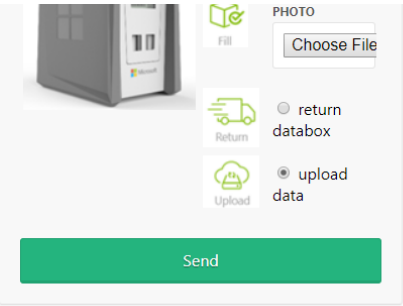


Return and Upload

- We are flying back to Viger datacenter and order an upload

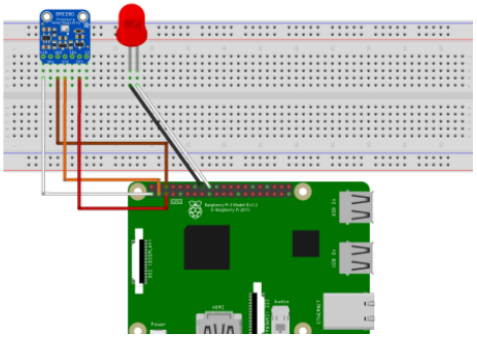


```
* IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - C
*/
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
//const BME280 = require('bme280-sensor');
const SIM908 = require('sim908-sensor');
const SDCARD = require('SdCard-device');
const IOTHUB = require('IotHub-service');

const SIM908_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: SIM908.SIM908_DEFAULT_I2C_ADDRESS() // defa
};

const SDCARD_OPTION = {
  i2cBusNo: 2, // defaults to 1
  i2cAddress: SDCARD.SDCARD_DEFAULT_I2C_ADDRESS() // defa
};

// fturi begin : Azure MQTTP device
const connectionString = 'HostName=DataBoxHub.azure-devic
// fturi end : Azure MQTTP device
```

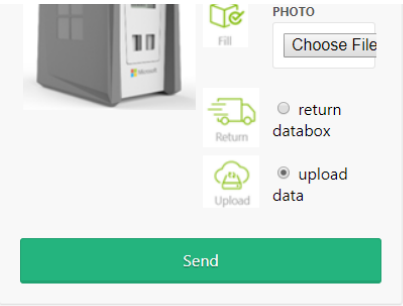


Return and Upload

- We are flying back to Viger datacenter and order an upload

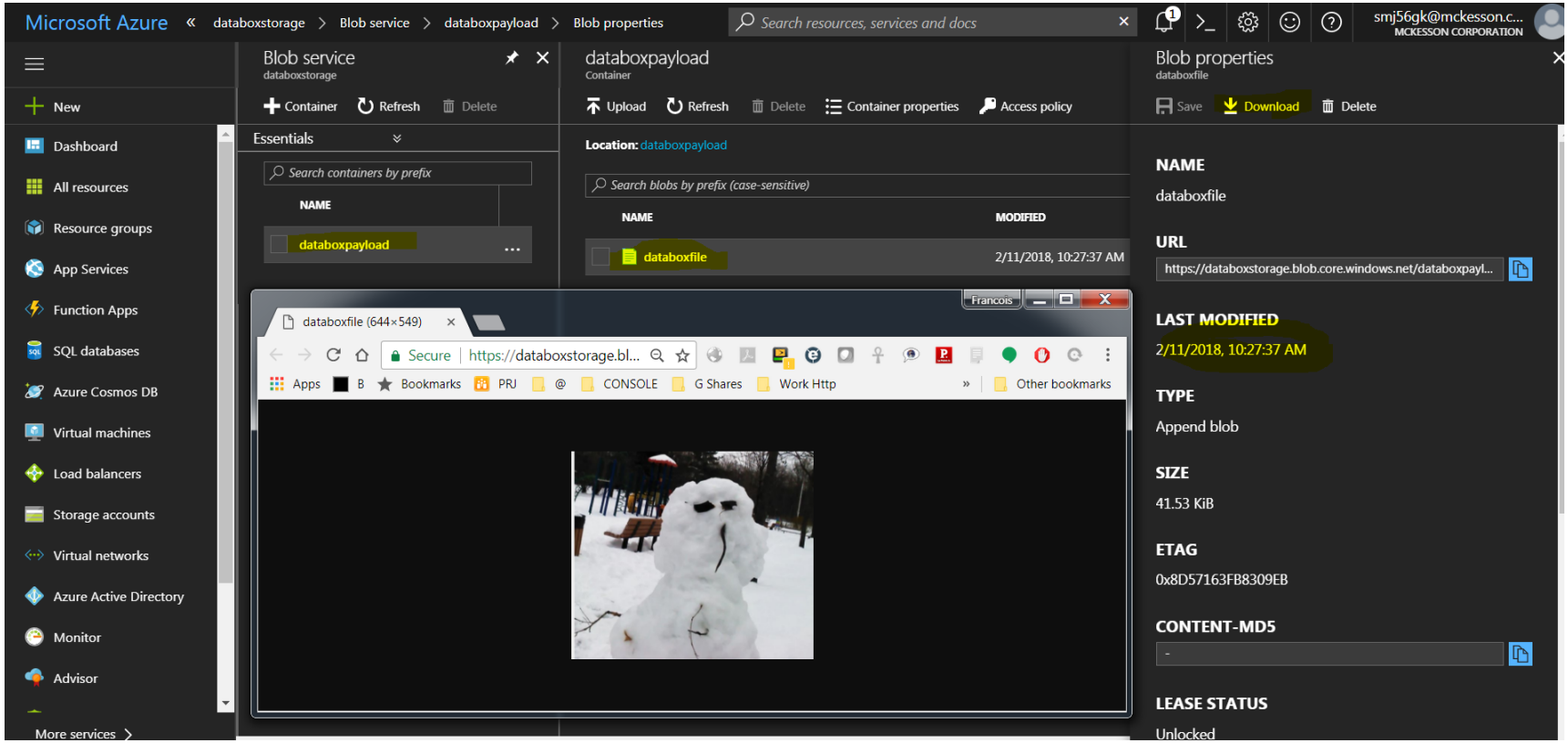


```
* IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - C
*/
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
//const BME280 = require('bme280-sensor');
const SIM908 = require('sim908-sensor');
const SDCARD = require('SdCard-device');
const IOTHUB = require('IotHub-service');

const SIM908_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: SIM908.SIM908_DEFAULT_I2C_ADDRESS() // defa
};

const SDCARD_OPTION = {
  i2cBusNo: 2, // defaults to 1
  i2cAddress: SDCARD.SDCARD_DEFAULT_I2C_ADDRESS() // defa
};

// fturi begin : Azure MQTTP device
const connectionString = 'HostName=DataBoxHub.azure-devic
// fturi end : Azure MQTTP device
```



Mission Accomplised

- Mr Snowman is safe:



YouTube URLs, GitHub URL, Last Page

- Two minute (short):

  https://youtu.be/yg7ieQeYTSo
- 15 minutes (long):

  https://youtu.be/Es7b7VV7s78
- GitHub Repository with all artifacts:
  - Raspbery emulator (branch from Azure code): https://github.com/sysarchitek/DataBox.raspberry-pi-web-simulator
  - The console: https://github.com/sysarchitek/DataBox.Console
  - The Tracker and Storage hopper-uploader https://github.com/sysarchitek/DataBox.geotracking

  - Main Documentation project
  - https://github.com/sysarchitek/DataBox-IOT-Emulator

François TURI
sysarchitek@gmail.com