Lab3 -- Building a IoT Data Pipeline: Raspberry Pi to Analytics

This Lab is based on Google codelabs “Building a Serverless Data Pipeline: IoT to Analytics” https://codelabs.developers.google.com/codelabs/iot-data-pipeline/index.html

In this Lab, you are going to build a data pipeline that starts with an Internet of Things (IoT) device, utilizes a message queue to receive and deliver data, leverages a serverless function to move the data to a data warehouse and then create a dashboard that displays the information. A Raspberry Pi 3 Model B+ with a SenseHat sensor will be used for the IoT device and several components of the Google Cloud Platform will form the data pipeline.

Introduction

In this laboratory you will stream data collected by the SenseHat sensors, installed in a Raspberry PI board. The Raspberry Pi board will run a Linux based system, and it will stream the data to a Google Cloud Platform project.

Objectives

In this lab, you learn how to perform the following tasks:

- Create Cloud Pub/Sub topics and subscriptions
- Use IoT Core to create a registry
- Authorize devices to publish data in your Pub/Sub topic
- Setup your own data pipeline in Google Cloud Platform
- Stream change in button state (events)
- Stream sensor data periodically
- Run the MQTT Application on Raspberry Pi
- Store the data using a BigQuery table from Google Cloud Platform
- Visualize the collected data using the Data Studio Dashboard

Prerequisites

- A Google Cloud Platform account
- Raspberry Pi 3 Model B
- SenseHat for Raspberry Pi
- MicroSD card for Raspbian (8GB+ recommended)
- MicroSD card reader
- USB keyboard
- MicroUSB to USB-A cable
- Power adapter for Raspberry Pi 3
- Monitor with HDMI input
- HDMI cable

In addition, having access to a computer monitor or TV with HDMI input, HDMI cable, keyboard and a mouse is assumed. Also, it is assumed you have configured the Raspberry Pi to connect to the Internet and reach the Google Cloud Platform website.

Instructions

There are quite a large number of steps to complete this lab, but each step is not difficult.

Get free access to the Google cloud

To get free access to the Google cloud, you need to redeem the following coupon:

> Dear Students,
>
> Here is the URL you will need to access in order to request a Google Cloud Platform coupon. You will be asked to provide your school email address and name. An email will be sent to you to confirm these details before a coupon is sent to you.
>
> Student Coupon Retrieval Link
>
> You will be asked for a name and email address, which needs to match the domain. A confirmation email will be sent to you with a coupon code.
> You can request a coupon from the URL and redeem it until: 18/05/2021
> Coupon valid through: 18/01/2022
> You can only request ONE code per unique email address.

Set up the Raspberry Pi and the SenseHat

To complete this part of the Laboratory, it is assumed you have a running Raspberry Pi system. In other words, you should connect the Raspberry Pi to an HDMI Monitor, plug in the keyboard and mouse, power it on, install the system (Raspbian is prefered), and should be able of doing the initial configuration (as stated below).

Connect your Raspberry Pi

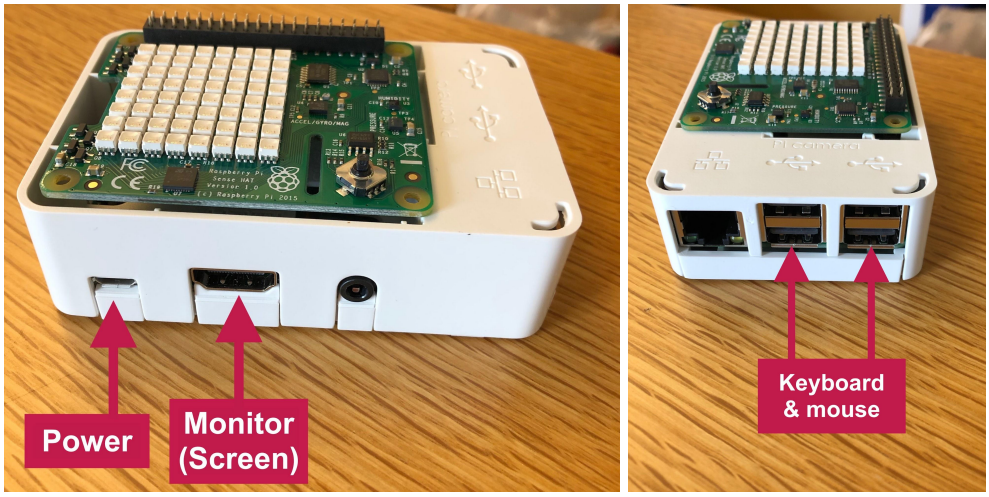
What you will need:

- USB Keyboard and mouse (available at LAB E1)
- A computer screen with HDMI input (available at LAB E1)
- Your Raspberry Pi Kit (Raspberry Pi + Power supply + micro SD Card)

Let's prepare you setup:

1. Insert the SD card into the micro SD card slot at the underside of your Raspberry Pi (actually, it is already installed)

2. Connect the mouse to a USB port on the Raspberry Pi (it doesn't matter which one)
3. Connect the keyboard in the same way (it doesn't matter which USB port you will use)
4. Find the HDMI cable from the computer screen and connect it to the HDMI port on your Raspberry Pi (notice that it has a large, flat side on top)
5. Plug the power supply into a power outlet, and connect it to the Raspberry Pi micro USB power port (Notice that the Raspberry Pi's micro USB power port has a longer flat side on top. Also notice that the Raspberry Pi doesn't have a power switch and as soon as you connect it to a power outlet, it will turn on)



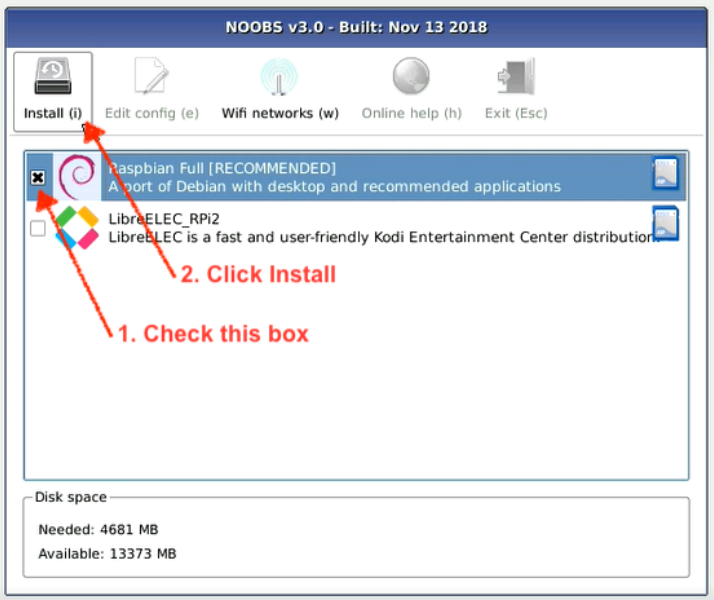
For more detailed instructions, please visit the officil Raspberry Pi website clicking this link https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/4

System installation

If this is the first time you're starting your Raspberry Pi, you will see a system installation screen (the NOOBS installer). This software will walk you through installing the Raspbian operating system (OS) which is a Debian Linux based system.

When the installer has loaded, it will offer you a choice of which OS to install. We will install the Raspbian SO.

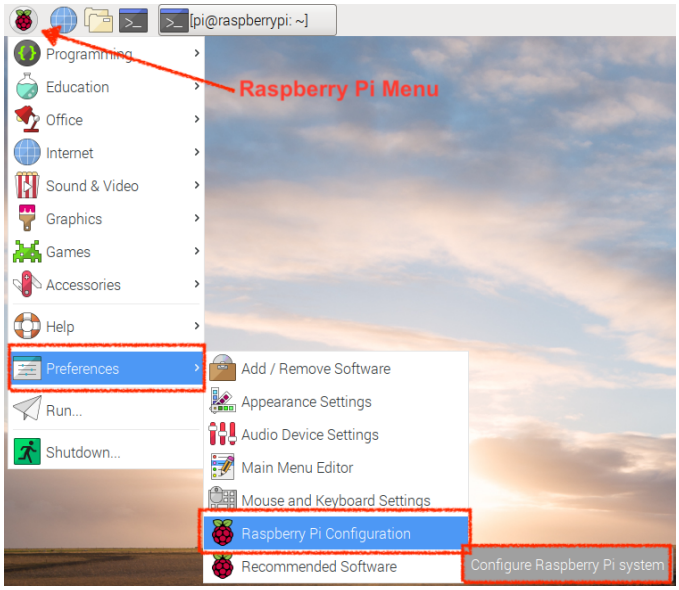
1. Check the box for Raspbian (the box should be market with an X)
2. Click Install
3. Click Yes in the warning dialogue box, and wait until the installer finishes the OS installation. The Raspbian installation process will take some time.
4. When Raspbian has been installed, click OK. Your Raspberry Pi will restart, and Raspbian will then boot up
5. After a few seconds the Raspbian Desktop will appear.



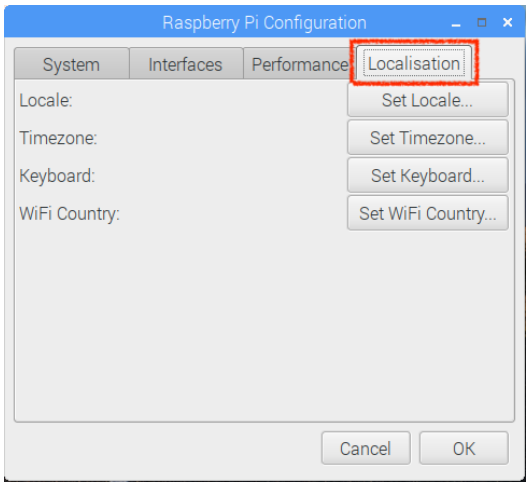
Initial Configuration (Language, Keyboard and Timezone)

When you start your Raspberry Pi for the first time, the **Welcome to Raspberry Pi** application will pop up. Just click **Cancel**.

1. Click on the **Raspberry Pi menu** on the upper left corner of the screen, go to **Preferences** and click on the **Raspberry Pi Configuration**



2. On the **Raspberry Pi Configuration** window, click in the tab **Localization**



3. Click **Set Locale**, select the values according to the following table and click **OK**

Property	**Value (type or select)**
Language	en (English)
Country	GB (United Kingdom)
Character Set	UTF-8

4. Click **Select Timezone** and select **Europe** for the **Area** field and **Stockholm** for the **Location**. Click **OK**

5. Click **Set Keyboard**, select the values according to the following table and click **OK**

Property	**Value (type or select)**
Model	Generic 105-key (Intl) PC
Layout	Swedish
Variant	Swedish

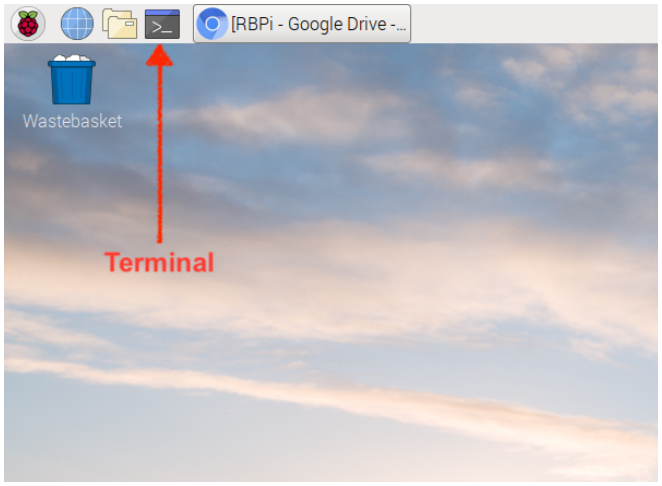
6. Click **Select WiFi Country** and select **SE Sweden** for **Country**. Click OK

Connect to the EDUROAM Wireless Network (at HH)

It is assumed that you have access to the "Eduroam" network. If you don't have access, please ask for help. If you can access the internet from your Raspberry Pi board, this means you have already done this before, and you can safely skip this step.

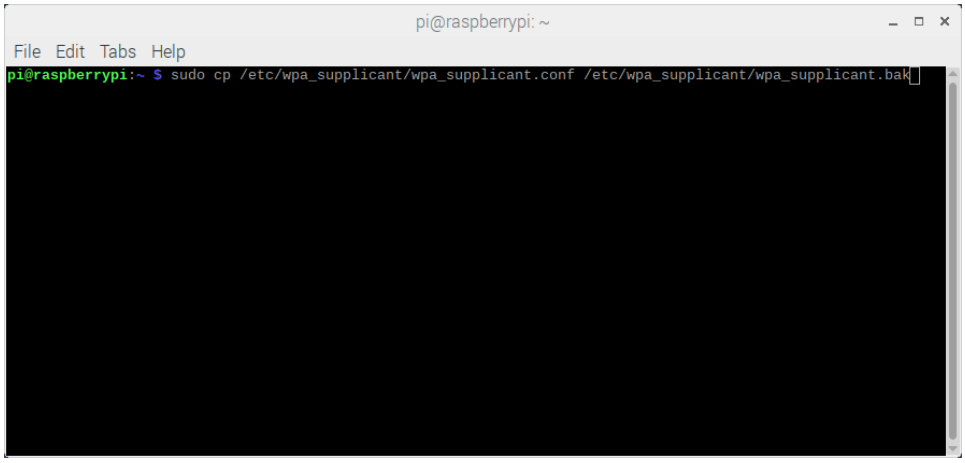
In this step, you will use some command line tools (e.g., the Nano text editor). Don't worry if this is your first time using the command line, the steps are easy to follow and you probably wouldn't face any difficulties.

1. Click the **Terminal** icon to open a terminal window



2. Before you edit the configuration file, it is always a good idea to keep a copy of the original file. In case something goes wrong, you can restore the original file. In the Linux command line, you will use the command "**cp**" (which means "copy") to copy a file. The **"sudo"** (before "cp") command will give you administrator power (i.e., it will allow you to modify system configuration files. On the **Terminal**, type the following command and press **Enter**:

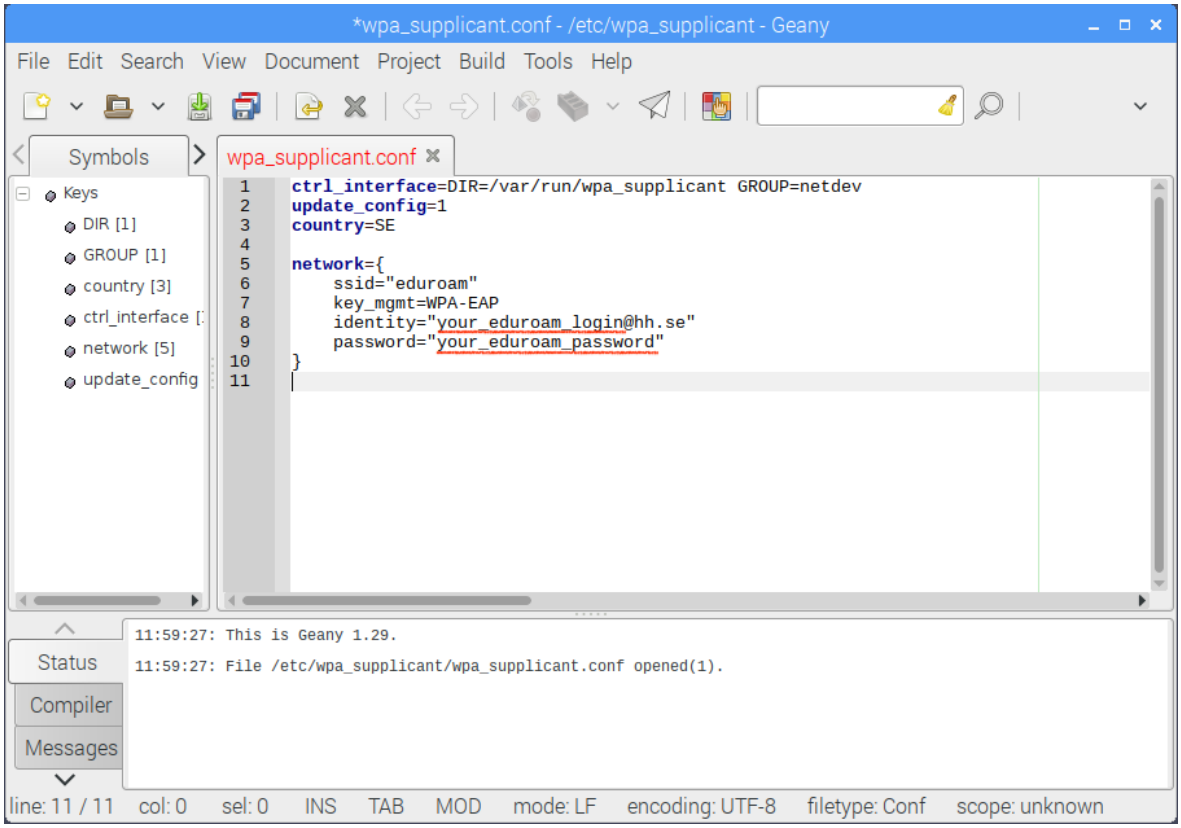
```
sudo cp /etc/wpa_supplicant/wpa_supplicant.conf
/etc/wpa_supplicant/wpa_supplicant.bak
```



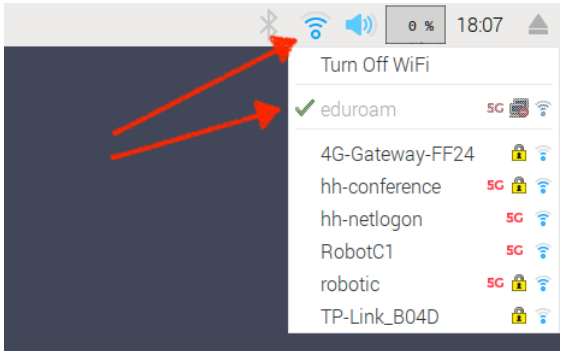
3. Now you will edit the original file using the Geany editor. Type the following command in the terminal to open the file with the Nano editor:

```
sudo geany /etc/wpa_supplicant/wpa_supplicant.conf
```

4. The Geany editor will show the content of the file **/etc/wpa_supplicant/wpa_supplicant.conf**, and you will see something like this in the text edit area



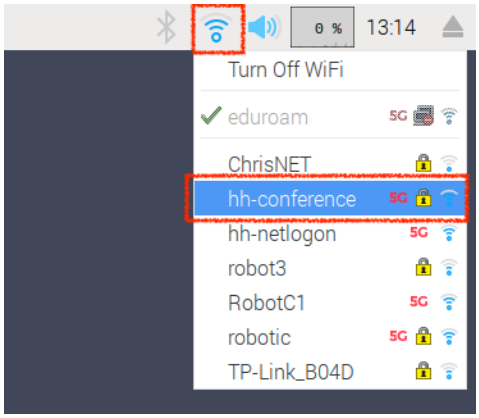
5. Edit the file to look like the file in the picture above, adding the **ssid**, **key_mgmt**, **identity** and **password** configurations. Don't forget to replace **your_eduroam_login** and **your_eduroam_password** by your Eduroam login details
6. If you have finished editing the file, press CTRL + S (Press the "S" key while you hold the Control key pressed) to save the changes
7. Press CTRL + Q (Press the "Q" key while you hold the Control key pressed) to close the Geany editor (or click on the Close window button)
8. After the change you will need to restart the Raspberry Pi so the changes take effect. You can restart the Raspberry Pi using the Terminal: type **reboot** and press the Enter key
9. Check the WiFi connection details



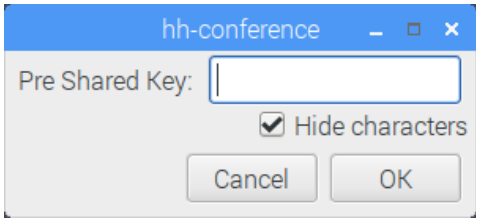
10. Check if your Raspberry Pi can reach the Internet (using the Chromium browser, for example)

Connect to a Wireless Network (non EDUROAM)

1. Connect the Raspberry Pi system to a wireless network at your home, for example, should be easy, and usually you should be able to configure the network connection using the graphical interface (i.e., without editing any file).
2. From the main screen (in your Raspberry Pi), click on the **Network icon** (in the top right corner of the screen)
3. Select the network you want connect by clicking in the network name (**SSID**)



4. If the network required authentication, a new window asking for the "**Pre Shared Key**" should appears. Type your wireless network password and click on the "**OK**" button



5. If you see the message "**WiFi disabled - country not set**", click the text "**Click here to set WiFi country**" and select "**SE Sweden**" for the country. Click "**OK**"
6. Sometimes you need to remove the old WiFi configuration to the RaspbianOS (the system running in the RaspberryPi) be able to configure a new WiFi network. If you are facing problems to connect to your WiFi network at home, try to remove the EDUROAM configuration. Edit the file "**wpa_supplicant.conf**" using the Geany editor with the command "**sudo geany /etc/wpa_supplicant/wpa_supplicant.conf**" in the **Terminal** (See EDUROAM configuration topic). Delete the entire "network={ … }" block of code.

Extra…

For those that want to work remotely using ssh, you need to turn on ssh on the RaspberryPi by issuing the following commands:

```
sudo systemctl enable ssh

sudo systemctl start ssh
```

Figure out the IP address of the RaspberryPi by issuing the command:

```
hostname -I
```

Then using the returned IP address to remotely log in:

```
ssh pi@NNN.NNN.NNN.NNN
```

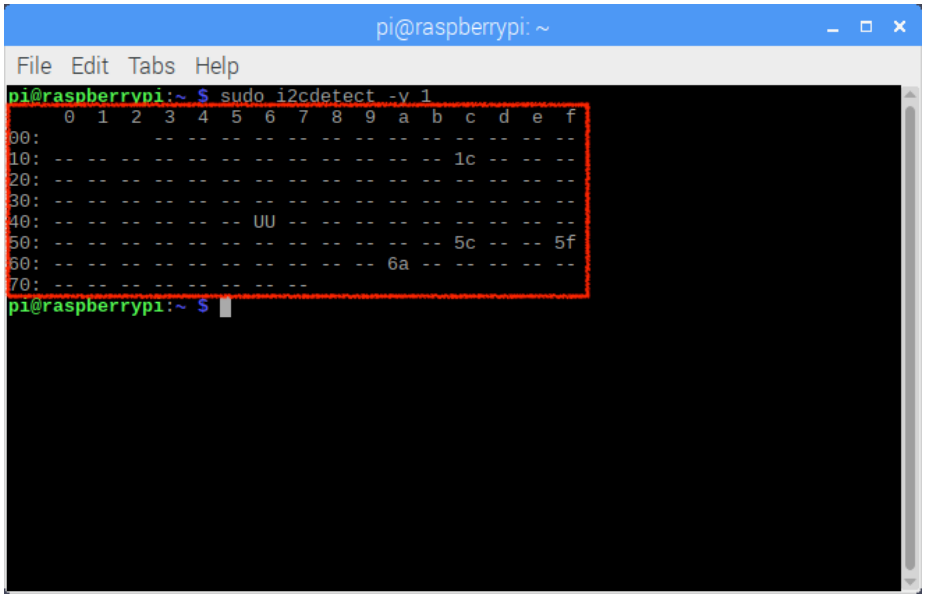
Verifying the SenseHat installation

As you will use sensors from the SenseHat board, it is a good idea to test whether the Raspberry Pi board is able to communicate with the SenseHat board. The SenseHat sensors operate over the I2C bus. You can use the tool "**i2cdetect**" from the command line to check if the communication between the Raspberry Pi board and the sensors is working.

1. Click the **Terminal** icon to open a terminal window
2. Type the following command in the Terminal window and press Enter

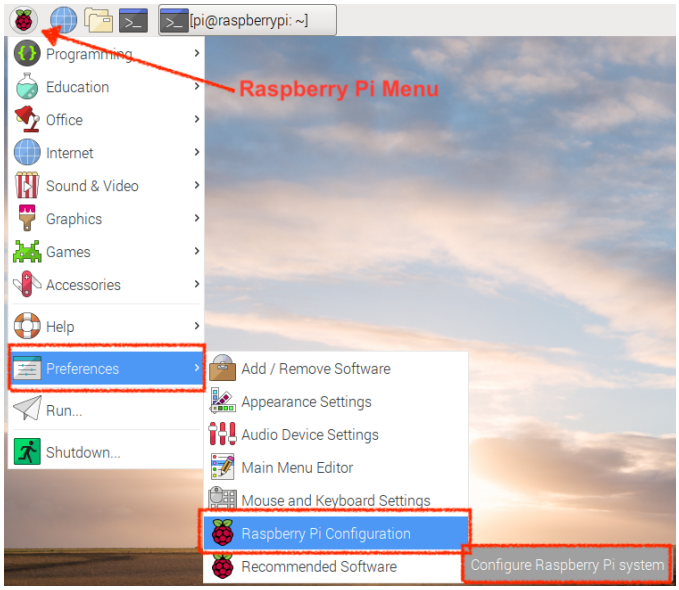
```
sudo i2cdetect -y 1
```

3. The command should print out the following information:



4. If you get the message ""Error: Could not open file `/dev/i2c-1' or `/dev/i2c/1': No such file or directory" as the result of the command, it means the I2C communication isn't working. In this case you have to enable the I2C interface.

a. Click on the **Raspberry Pi menu** on the upper left corner of the screen, go to **Preferences** and click on the **Raspberry Pi Configuration**
b. On the **Raspberry Pi Configuration** window, click on the **Interfaces** tab

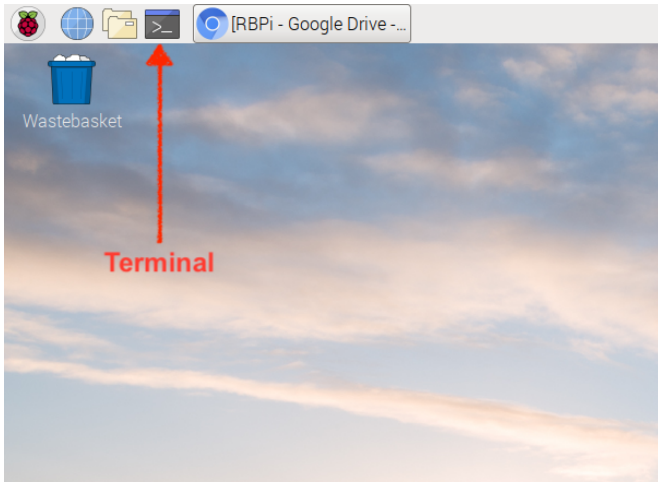


c. Locate the I2C and SPI interfaces
d. Check the value "**Enabled**" for both of them (I2C and SPI)
e. Click "**OK**"
f. Run the "**i2cdetect**" command again

Install the necessary software on Raspberry Pi

In this step you will install the required software to connect to the Google Cloud Platform. We will use the the command line (**Terminal**) to do the installation.

1. Click the **Terminal** icon to open a terminal window



2. Let's make sure that the list of places your Pi will be getting its libraries from is current. Sometimes these repositories move around, get deprecated, etc, so it's always good to

run this when starting a project. Type the following command on the Terminal window and press Enter:

```
sudo apt-get update
```

3. The apt-get command will print out some information and should end with the message **“Reading package lists... Done”**.
4. Using the Terminal, install the following packages / software. Enter one command at a time and press enter after each command. Press “**Y**” when you see the message “Do you want to continue? [Y/n]” to confirm that you want to continue with the installation of the package / software.

```
sudo apt-get install build-essential

sudo apt-get install libssl-dev

sudo apt-get install python-dev

sudo apt-get install libffi-dev
```

5. We will use the Python Language to collect data from the Raspberry Pi sensors and to send it to the cloud. Our software will send the data using the MQTT protocol, therefore, we will need the library that allows us to use that protocol, **paho-mqtt**. To install the **paho-mqtt** library for Python, we will use another tool, called “**pip**”. Run:

```
sudo pip install paho-mqtt
```

6. The Google Cloud Platform authenticates the devices using the JWT standard instead of user/password which is a more secure way of authentication. To handle the authentication the python code will use the **pyjwt** library. To install the pyjwt library, run the command

```
sudo pip install pyjwt
```

7. You probably will get the message “Requirement already satisfied: pyjwt in /usr/lib/python2.7/dist-packages”. The library was installed previously.
8. The cryptography library is also required for authentication. Run the following command to install it:

```
sudo pip install cryptography
```

9. And finally, just to make sure you have the last version of the SenseHat library. You will use this library to get the data from the SenseHat sensors. This library is usually pre installed in the Raspberry Pi system. However you can safely run the install command again. If the installed version is outdated, the install command will try to update the installed version. In the **Terminal** window type the following:

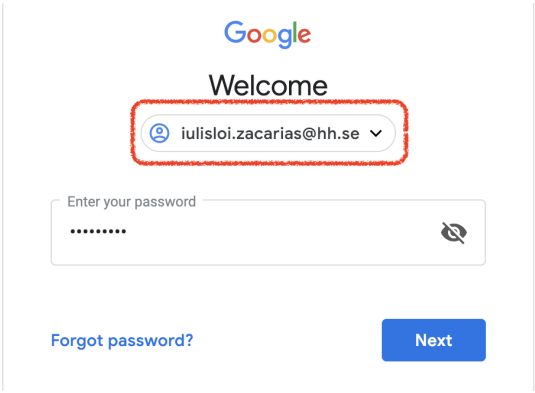
```
sudo apt-get install sense-hat
```

That’s all. Now, all the required software is installed in the system.

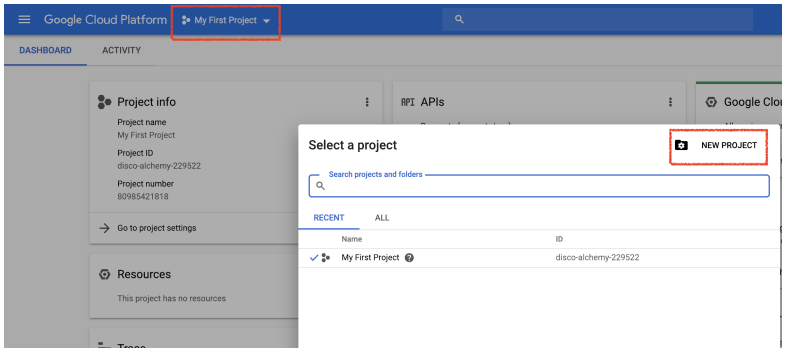
Create a new project at Google Cloud Platform

In this section, we will set up a project in Google Cloud Platform to store the data acquired by your Raspberry Pi board.

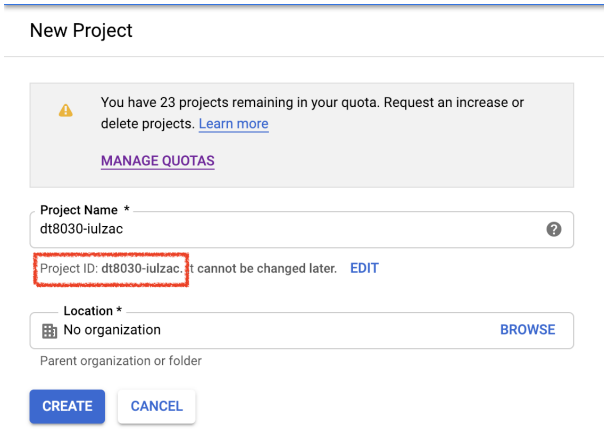
1. Sign-in to Google Cloud Platform console (console.cloud.google.com) using the same Google Account used in the previous Labs



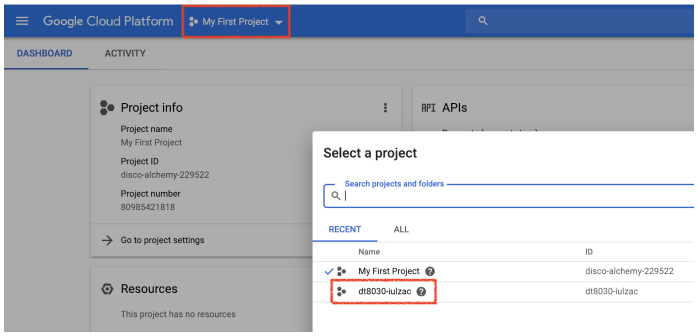
2. Once you have logged in, **create a new project** by clicking in the default project name, on the blue bar at the top of the page and by clicking in the New Project button



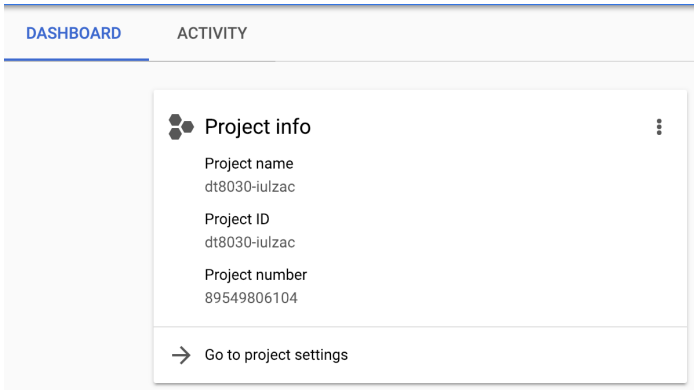
3. Fill in the project name field and take note of your project ID (i.e. Project ID: _____) as it will be needed later. The project ID needs to be a unique name across all Google Cloud projects. You should name your project following the format "**proj-userID-X**", replacing the "userID" part by your user id (login) at the Halmstad University, and replacing the X by a number, starting from 1. If you need to create a second project, increase the number used for the "X" part.
   Example:
   My student email at the Halmstad University is "iulzac2018@student.hh.se". My first project at Google Cloud Platform should be "**proj-iulzac2018-1**". The Project ID for the second project should be "**proj-iulzac2018-2**" and so on.



4. Click **Create**
5. Change to your new project by clicking on the Project Name at the top blue bar and select your new project



6. If you forget your project ID, it is located in the Cloud Console in the Home area



Enable APIs (*Optional*)

In this lab, we will be using Pub/Sub, Dataflow, and IoT Core APIs, so we'll need to enable those APIs.

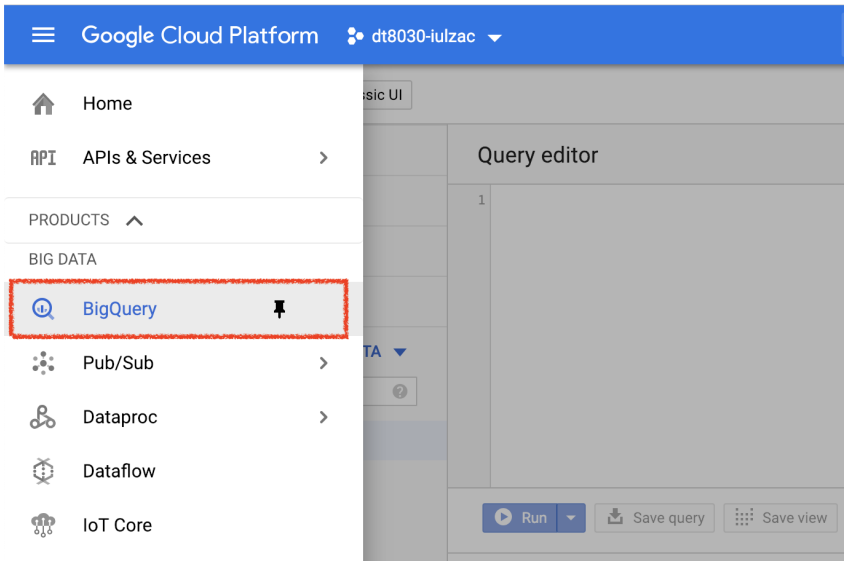
1. **Select APIs & Services** from the **Navigation menu**.
2. Click on Enable APIs and Services
3. In the Search Bar, type in PubSub

4. Click on the resulting tile that says "Cloud Pub/Sub API" and, on the following page, "Enable API"
5. Repeat this process for Dataflow, Compute Engine and IoT Core

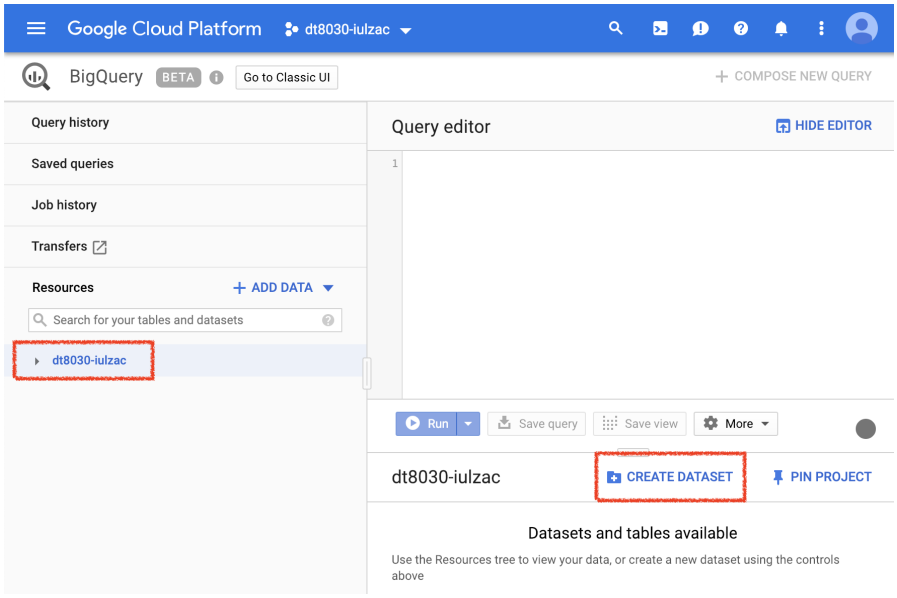
Create a BigQuery table

BigQuery is a serverless, highly scalable, low cost enterprise data warehouse and will be an ideal option to store data being streamed from IoT devices while also allowing an analytics dashboard to query the information.
Let's create a table that will hold all of the IoT environment data (for this laboratory, the button press events on the SenseHat “joystick”).

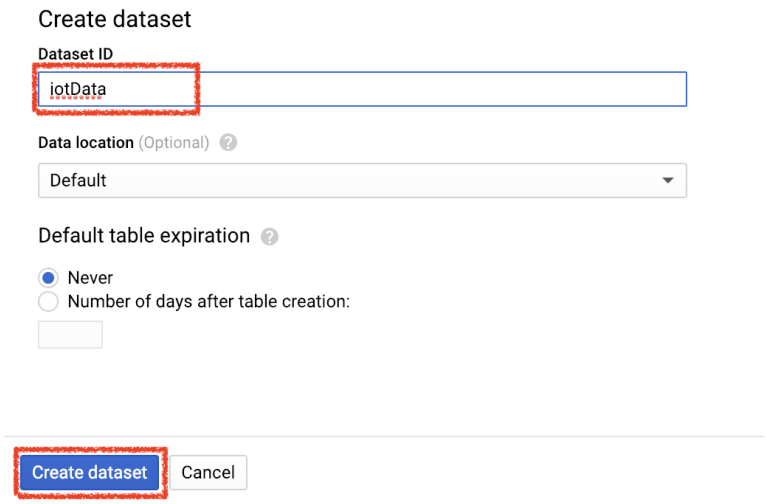
1. Select **BigQuery** from the **Navigation menu**



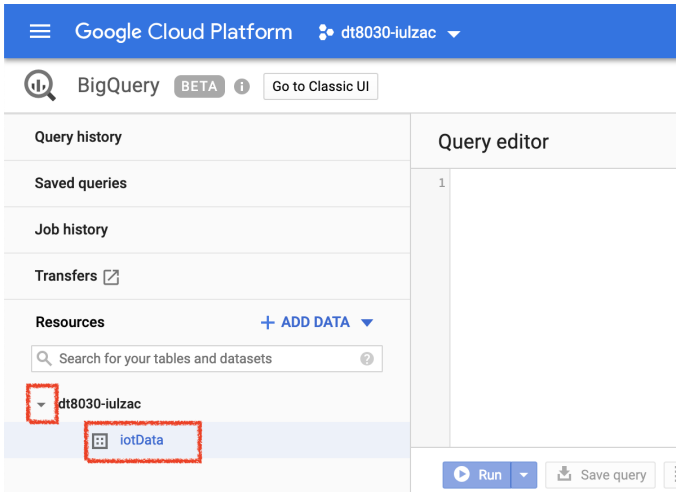
2. On the left panel, select (click) your project name. After you select your project, the button Create Dataset will appear on the bottom right of your screen



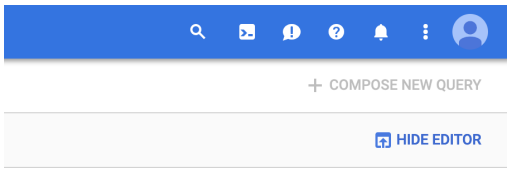
3. Type "**iotData**" in the Dataset ID field and click **Create Dataset**

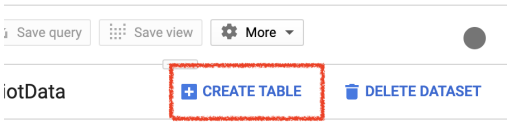


4. On the right panel, expand your project by clicking in the small arrow in front of your project name and click in the dataset you have create in the previous step (**iotData**)



5. On the right panel, click the **Create table**

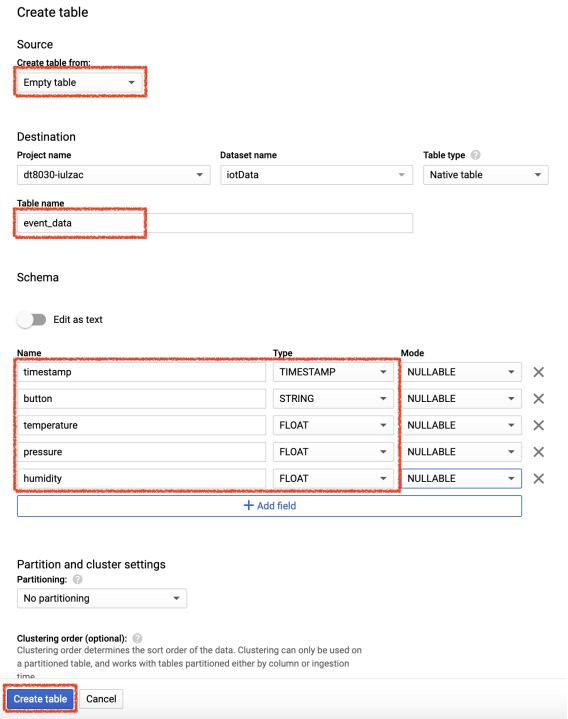




6. In the Source Data section, select **Empty table** for the **Create table from** field
7. *Pay attention when entering the information. Any spelling errors for the field names or the table name can cause issues when Cloud Functions attempts to add data later. The field names are exactly the same field names that we are using in our code (See Start sensing and streaming data from Raspberry Pi). Therefore, if you want to add a new sensor to your project, you should create a new field in the BigQuery table, and use exactly the same name in the JSON object that you will send in the payload of the MQTT message.*
8. In the **Destination** section, type **event_data** for the **Table name** field
9. In the **Schema** section, click "**+ Add field**". Enter **timestamp** for the field name and set the field's **Type** to **TIMESTAMP**
10. Click "**+ Add field**" again. Type the text "**button**" for the field name and set the field's Type to **STRING**

11. Click “**+ Add field**” again. Type the text “**temperature**” for the field name and set the field's Type to **FLOAT**
12. Click “**+ Add field**” again. Type the text “**pressure**” for the field name and set the field's Type to **FLOAT**
13. Click “**+ Add field**” again. Type the text “**humidity**” for the field name and set the field's Type to **FLOAT**
14. Leave the other defaults unmodified. Click **Create table**

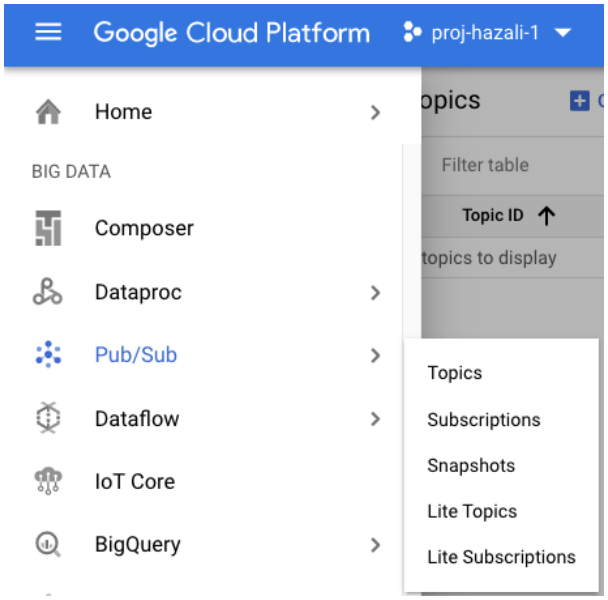
Field name	Type	Mode
timestamp	TIMESTAMP	NULLABLE
button	STRING	NULLABLE
temperature	FLOAT	NULLABLE
pressure	FLOAT	NULLABLE
humidity	FLOAT	NULLABLE



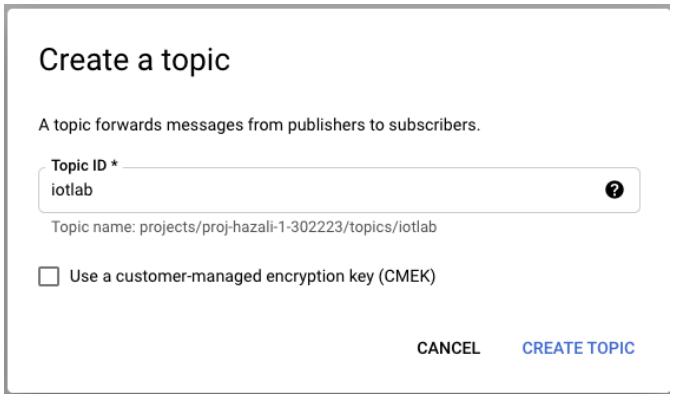
Create a Cloud Pub/Sub topic

To define a new Cloud Pub/Sub topic:

1. In the GCP Console, go to **Navigation menu** > **Pub/Sub** > **Topics**



2. If you see an Enable API prompt, click the **Enable API button**
3. Click **Create Topic**. The Create a topic dialog shows you a partial URL path, consisting of projects/ followed by your project name and a trailing slash, then topics/ . Confirm that the project name is the one you noted above
4. Give this string as your **Topic ID**: "**iotlab**"

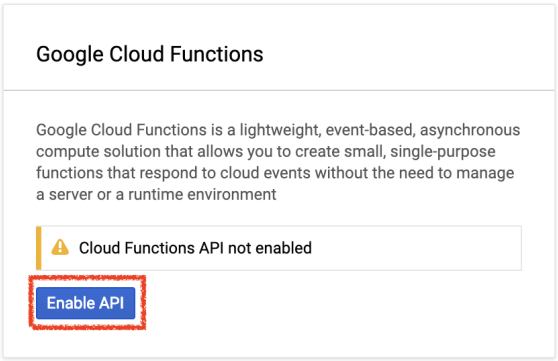


5. Then click **CREATE TOPIC**.

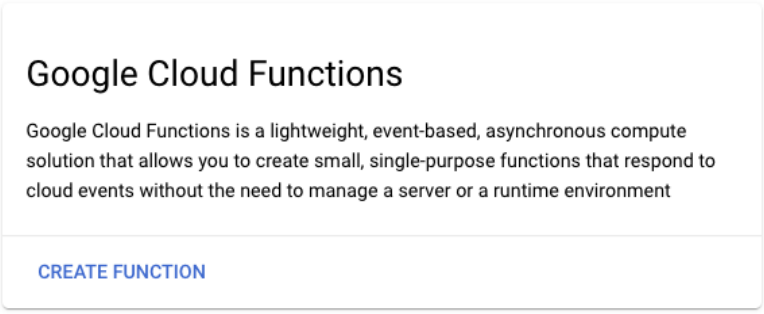
Create a Cloud Function

Cloud computing has made possible fully serverless models of computing where logic can be spun up on-demand in response to events originating from anywhere. For this lab, a Cloud Function will start each time a message is published to the **iotlab** Pub/Sub topic, will read the message and then store it in BigQuery.

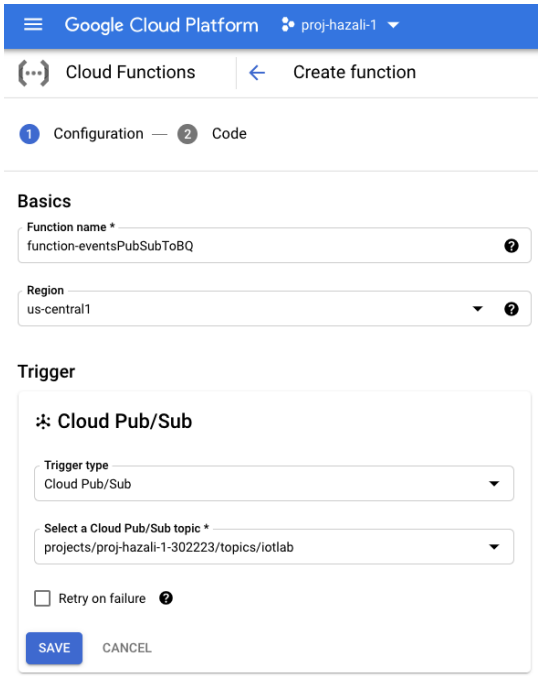
1. From the **Navigation menu**, select **Cloud Functions**
2. If you see an API message, click on the **Enable API** button



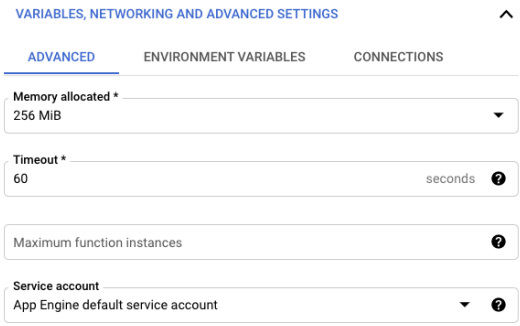
3. Click on the **Create function** button



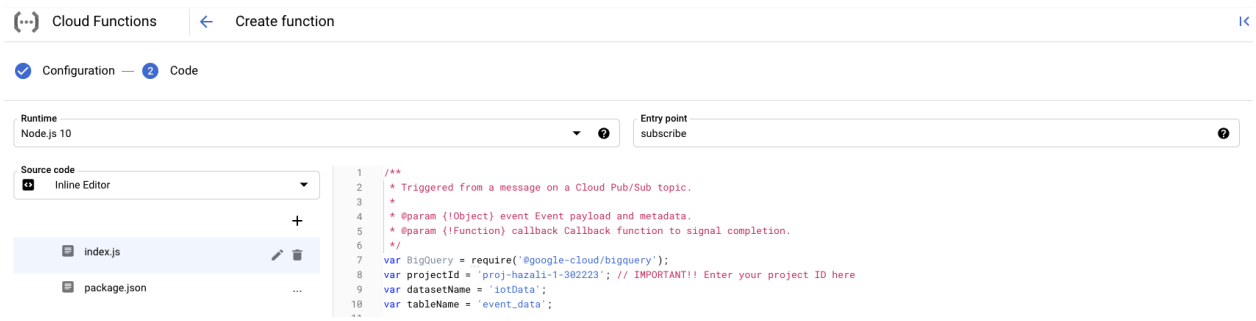
4. In the **Name** field, type **function-eventsPubSubToBQ**
5. For **Trigger** select **Cloud Pub/Sub**
6. In the **Topic** dropdown select “**iotlab**”



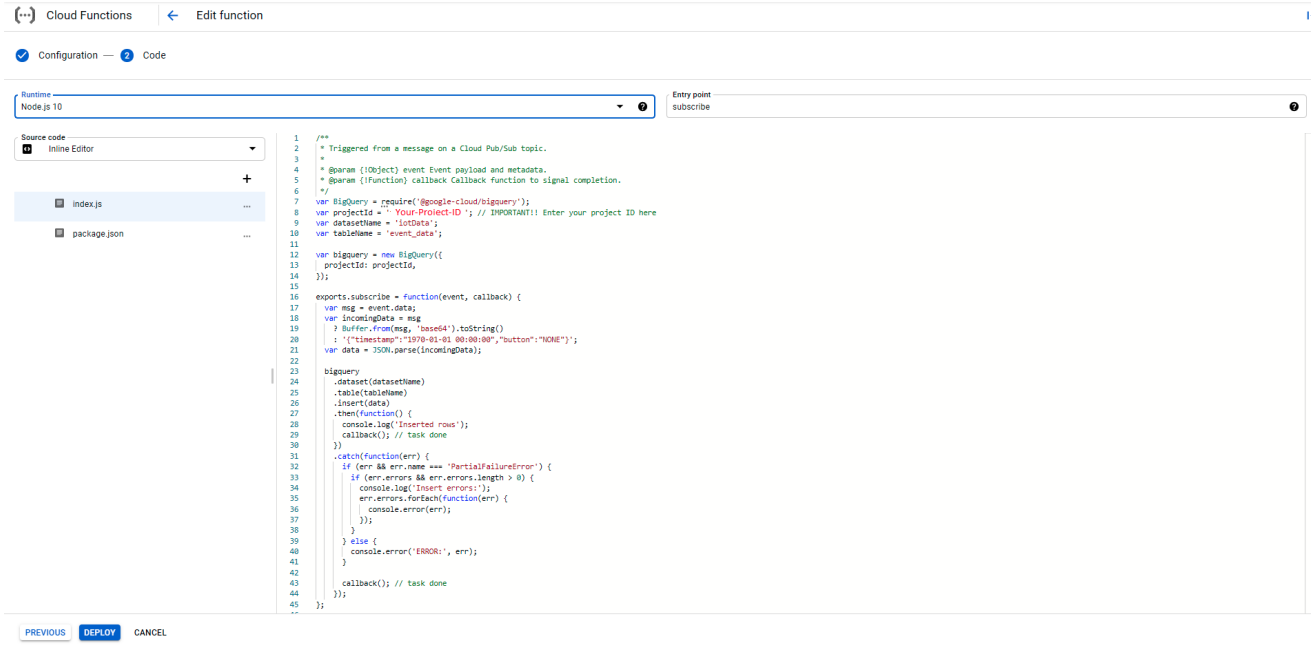
7. In **Variables, Networking and Advanced Settings,** under **Advanced** tab Leave the default value for **Memory allocated** field (256MB).



8. Enable **Cloud Build API** if not enabled.
9. For **Source code**, select **Inline editor**
10. Select “**Node.js 10**” in the “**Runtime**” dropdown
11. In the **Entry Point** field, type "**subscribe**".



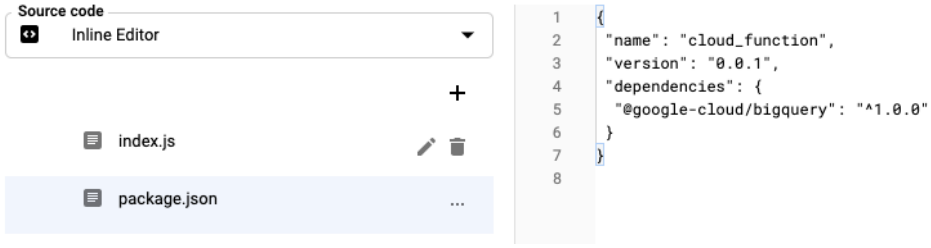
12. In the **index.js tab**, paste the following code over what is there to start with. **Make sure to change the constants for projectId, datasetId and tableId to fit your environment.** If you need to find your project ID, it is at the Google Cloud Platform dashboard




```
/**
 * Triggered from a message on a Cloud Pub/Sub topic.
 *
 * @param {!Object} event Event payload and metadata.
 * @param {!Function} callback Callback function to signal completion.
 */
var BigQuery = require('@google-cloud/bigquery');
var projectId = 'YOUR-PROJECT-ID-HERE'; // IMPORTANT!! Enter your project ID here
var datasetName = 'iotData';
var tableName = 'event_data';

var bigquery = new BigQuery({
  projectId: projectId,
});

exports.subscribe = function(event, callback) {
  var msg = event.data;
  var incomingData = msg
    ? Buffer.from(msg, 'base64').toString()
    : '{"timestamp":"1970-01-01 00:00:00","button":"NONE"}';
  var data = JSON.parse(incomingData);

  bigquery
    .dataset(datasetName)
    .table(tableName)
```

```
    .insert(data)
    .then(function() {
      console.log('Inserted rows');
      callback(); // task done
    })
    .catch(function(err) {
      if (err && err.name === 'PartialFailureError') {
        if (err.errors && err.errors.length > 0) {
          console.log('Insert errors:');
          err.errors.forEach(function(err) {
            console.error(err);
          });
        }
      } else {
        console.error('ERROR:', err);
      }

      callback(); // task done
    });
};
```

13. In the **package.json** tab, paste the following code over the placeholder code that is there

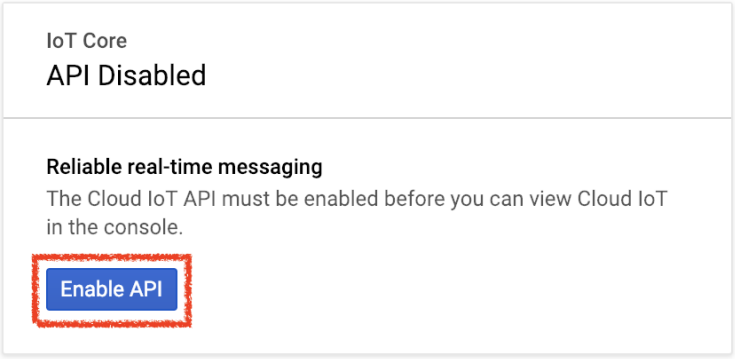



```
{
 "name": "cloud_function",
 "version": "0.0.1",
 "dependencies": {
  "@google-cloud/bigquery": "^1.0.0"
 }
}
```

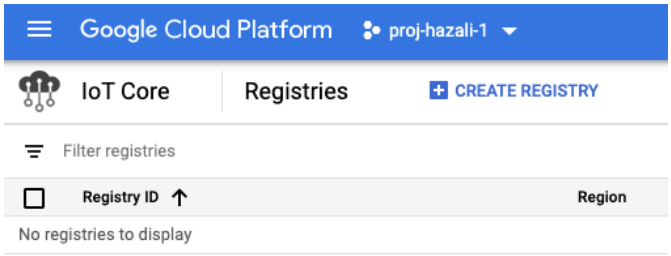
14. Click the **Deploy** button

Create an IoT Core registry

1. On the **Navigation menu**, click **IoT Core** menu. If the IoT Core API is disable, you should enable it before use, by clicking **Enable API**



2. Click **Create Registry**



3. On the **Create a registry** page, specify the following, and leave the remaining settings as their defaults:

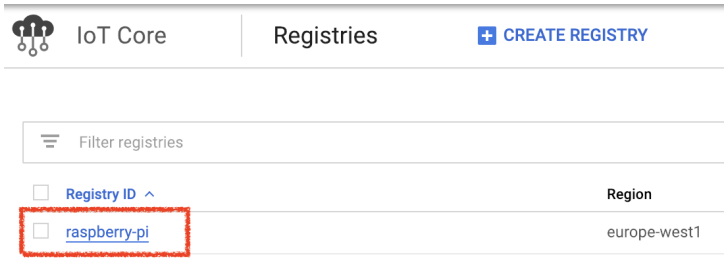
Property	**Value (type or select)**
Registry ID	raspberry-pi
Region	europe-west1
Select a Cloud Pub/Sub topic	projects/**<project_id>**/topics/iotlab

4. Click Create

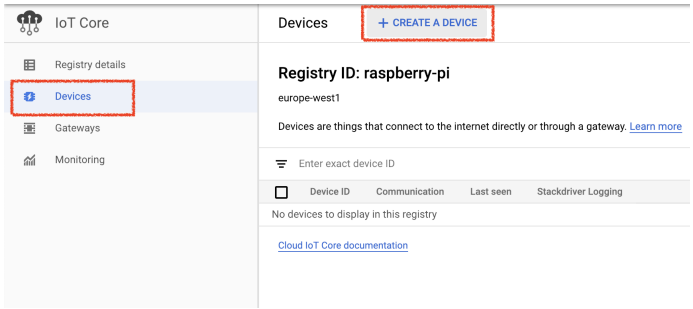
Add the device to the IoT Core Registry

In this step you will add a device to your device registry and later we will authorize this device to publish data on our Pub/Sub topic using a digital key.

1. On the **Navigation menu**, click **IoT Core** menu
2. Click on the device registry name you have created on the previous step (**raspberry-pi**)



3. On the left panel, click on **Devices** and then click **Create a device**



4. Fill the following information and leave the other information as default. You will return to this page later, because you should authorize your device (the Raspberry Pi) to publish data to the Pub/Sub Topic. Therefore, later you will add a digital key to authenticate the device.

Property	**Value (type or select)**
Device ID	rasp1
Device communication	Allow
Authentication	Enter manually (PS: You will add a digital key file later)

5. Click **Create**

Congratulations, you have finished the Google Cloud configuration. Now it is time to move to our Raspberry Pi, install the required software, configure the system, create the digital keys to authenticate

the Raspberry with the Google Cloud, write some code and collect data from sensors.

Add the digital keys to your device

To add the public key for the Raspberry Pi into IoT Core Registry we need a way to get it from the Raspberry Pi. If you can access the Raspberry pi using ssh the key entry can be done as in the Qwicklab “Streaming IoT Data to Google Cloud Storage” you did in Lab 2.

But if you do not have ssh access, you can access the IoT Core directly from the Raspberry Pi system, as described below.

Note that the required software should be already installed in your Raspberry Pi. Please see section Set up the Raspberry Pi and the SenseHat if you have not set up your Raspberry Pi system yet.

In this step you will create a digital key to authenticate your Raspberry Pi in the Google Cloud Platform, and you will add the generated digital key information to your device representation. The Google Cloud Platform will allow your Raspberry Pi to send data to your Pub/Sub Topic based in this authentication.

Create the digital keys

The first step is to create a certificate file in your Raspberry Pi. To do that, follow the nexts steps **in your Raspberry Pi** system:

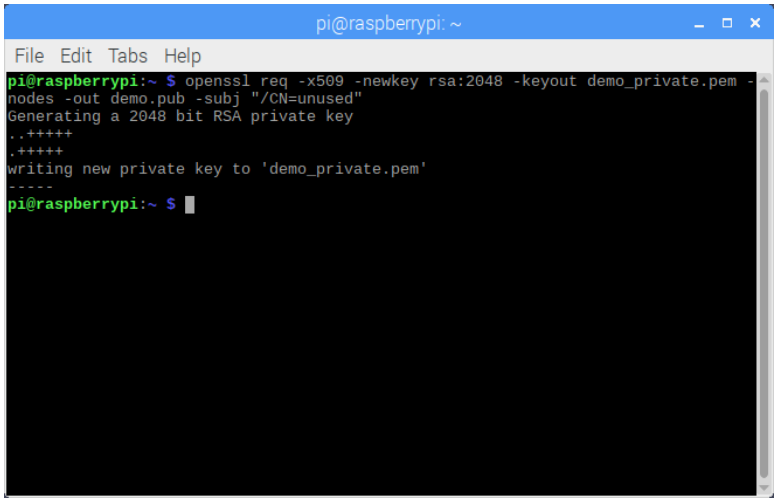
1. Click the **Terminal** icon to open a terminal window
2. Just to be sure that you are at you “home” directory (i.e., the default user directory), type this command (and press Enter):

```
cd ~
```

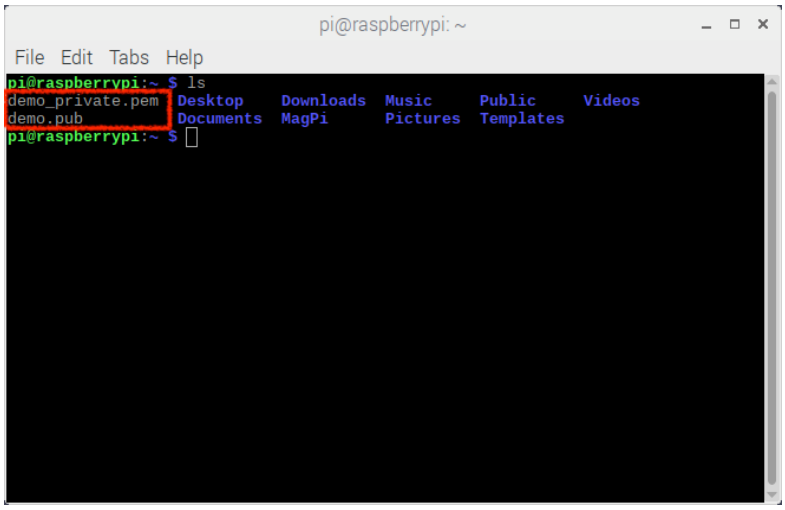
3. Now, run the command to create the digital key:

```
openssl req -x509 -newkey rsa:2048 -keyout demo_private.pem -nodes -out
demo.pub -subj "/CN=unused"
```

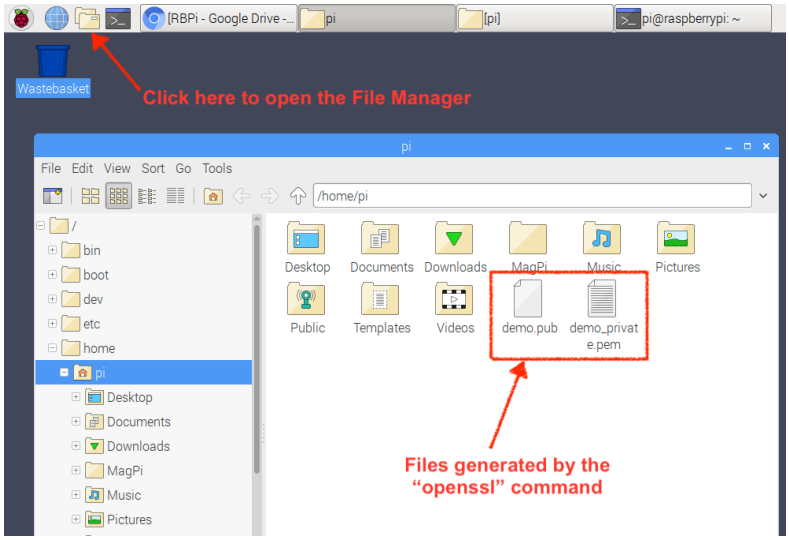
4. The command output should look like this (DO NOT COPY THE FOLLOWING LINES):



5. Let's check the files generated by the command in the previous step: In the **Terminal** window type "ls" (lowercase "LS") and press enter. This command will list the content of the current directory (/home/pi). You should see something similar this:



6. You can also see the files using the File Manager (from you Raspberry Pi):



7. The last step is to get the Google **roots.pem** file, so your device knows it's talking to Google in a secure connection. Again, in the **Terminal**, run these two commands:

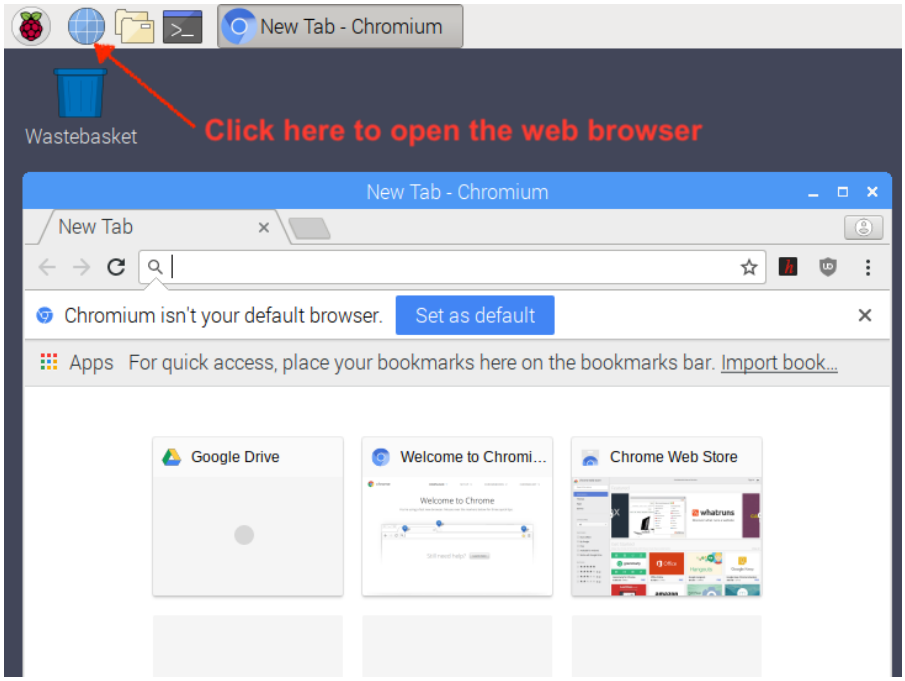
```
cd ~

wget https://pki.google.com/roots.pem
```

8. Now you can close the **Terminal** window.

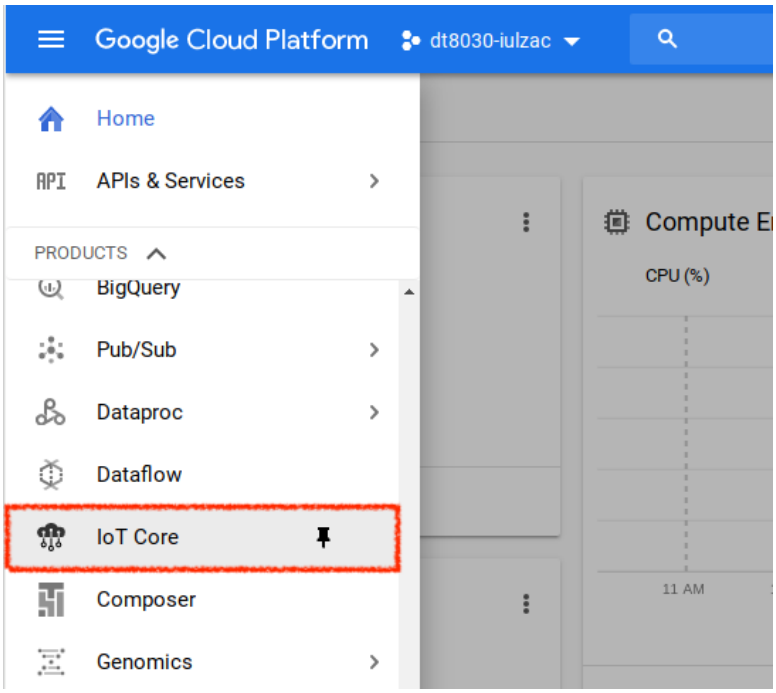
Add the Certificates to the device

In this step, you will add the generated keys in the Google Cloud Platform. You should perform these steps **from your Raspberry Pi** system to be able to upload the keys to the Google Cloud Platform console. Hopefully, there is a web browser installed in your Raspberry Pi by default.

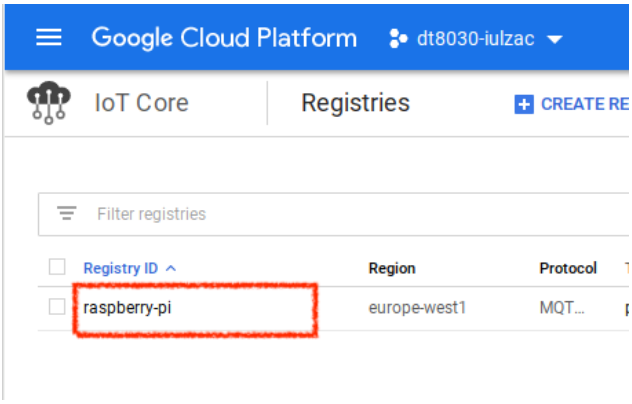
1. Open the web browser (called Chromium) in your Raspberry Pi:



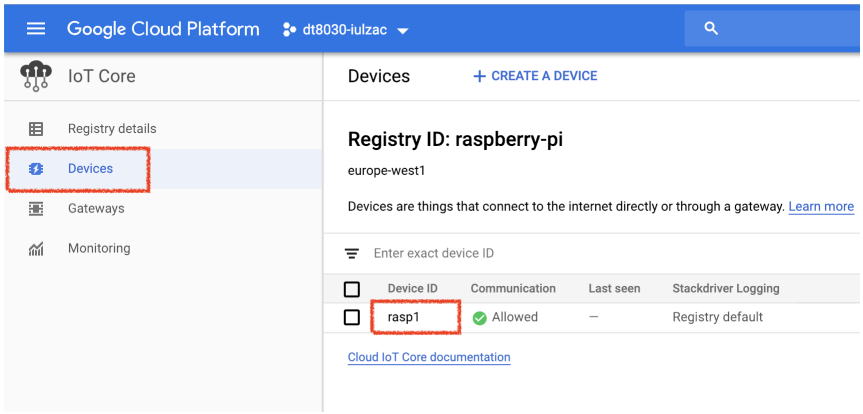
2. Go to the Google Cloud Platform website (https://console.cloud.google.com)
3. Login with your Google Cloud Platform account (the same account used in the first part of this laboratory: Create a new project at Google Cloud Platform)
4. Once you have logged in, click **IoT Core** menu on the Navigation menu



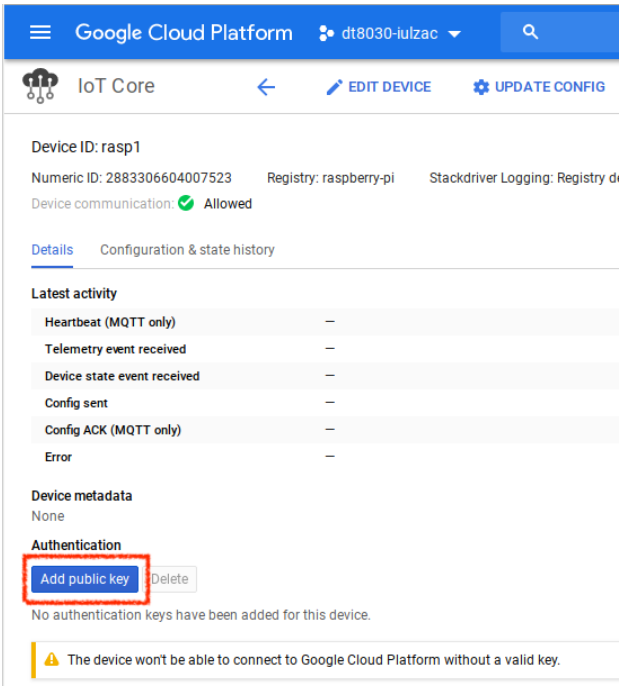
5. Click on the device register created previously (**raspberry-pi**)



6. On the left panel, click on Devices and then click on the device called **rasp1**

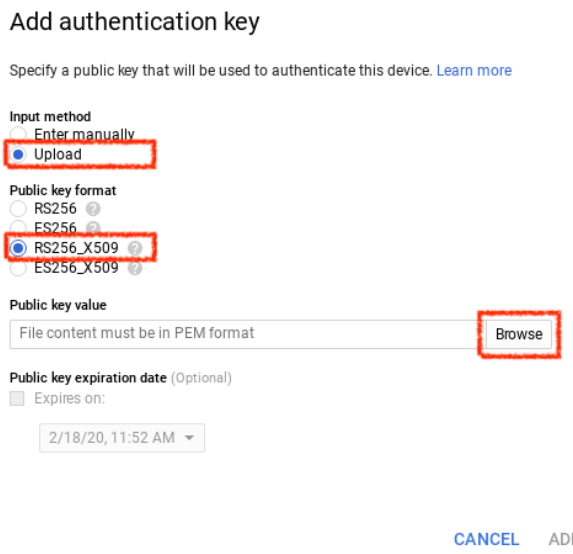


7. Go to the **AUTHENTICATION** tab then Click on the **Add public key** button

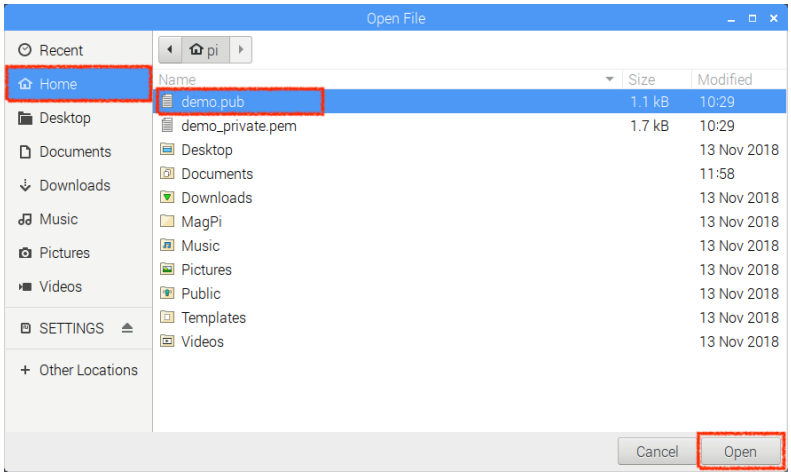


8. Under the **Input method**, select **Upload**
9. Select **RS256_X509** for **Public key format**

10. Click the **Browse** button, on the right of the field **Public key value**



11. A new window will open. Go to your home directory by clicking in the **Home** button, in the left panel of the new window
12. On the right panel, select the **demo.pub** file and click **Open**



13. Click **ADD** at the bottom right corner of the windows **Add authentication key**

Congratulations! **Your Raspberry Pi device is now authorized** to communicate with your project on the Google Cloud Platform.

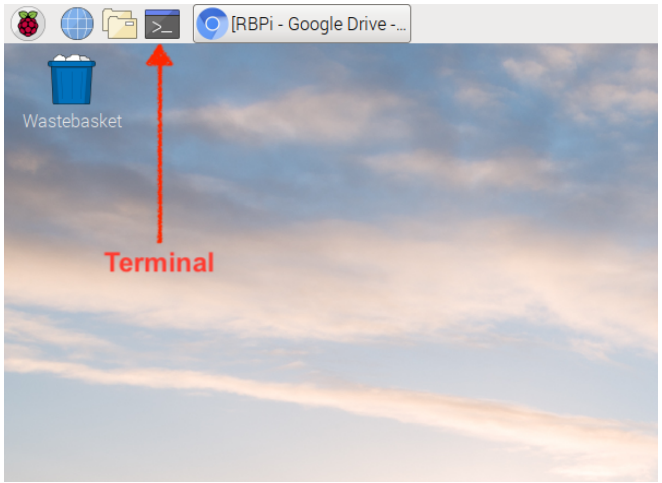
Start sensing and streaming data from Raspberry Pi

Now it is time to start streaming data from the Raspberry Pi to the Cloud. Using the code provided in this example you will stream simple events to the cloud. In other words, the Raspberry Pi board will transmit data to the cloud when it detects a button press (we will stream events from the SenseHat “joystick” buttons).

The code will set up a connection to the google cloud and will wait for some button event. Once a button event happen, the board will transmit the event to the cloud. The code was written using the Python Language and it is hardly based on the examples provided in the official Google Cloud Platform documentation and official examples. You just need to configure the parameters regarding your project and the Pub/Sub topic.

You will use (again) the **Terminal** tool on your Raspberry Pi. Therefore, you should begin by opening a Terminal Window.

1. Click the **Terminal** icon to open a Terminal window



2. The code is stored in a GitHub repository, however we will use the **Git** tool to get the code. Run the following command to make sure you have the Git tool installed in your Raspberry Pi:

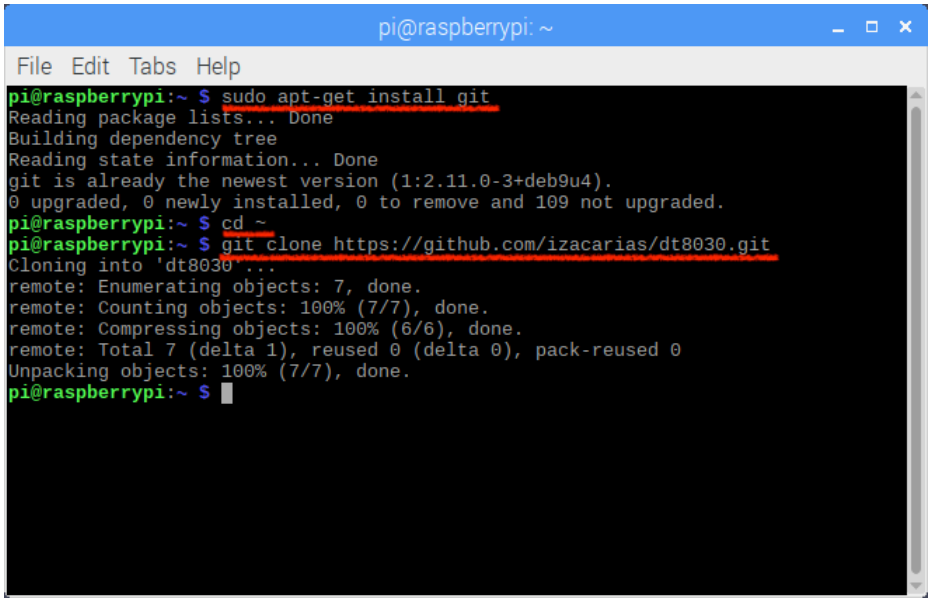
```
sudo apt-get install git
```

3. Go to your “home” directory using the **cd** command:

```
cd ~
```

4. And grab the code running the git command:

```
git clone https://github.com/izacarias/dt8030.git
```



5. The git tool will download the code in a new directory called **dt8030**. Inside this directory there is a file called **dt8030-buttons.py**. You should edit the code to send data to your project. Run the following command to edit the code:

```
geany dt8030/dt8030-buttons.py
```

6. In the Geany Editor, go to line 41. You should see this fragment of code:

```
40
41  ssl_private_key_filepath = ''    # /home/pi/demo_private.pem
42  ssl_algorithm = ''               # RS256
43  root_cert_filepath = ''          # /home/pi/roots.pem
44  project_id = ''                  # your project ID
45  gcp_location = ''                # europe-west1
46  registry_id = ''                 # the registry name (raspberry-pi)
47  device_id = ''                   # the device ID (rasp1)
48  ###################################################################
```

[Added 2019-02-25]:
Line 170 Please also change the sampling rate (`DATA_INTERVAL`) to a sample every 10 seconds, in order to get more data to analyse.

7. Provide the required information by setting the variables to the values used in your project. Se the next table for a description of the values:

Variable	**Value**
ssl_private_key_filepath	The full path to the private key we created using the **openssl** command: /home/pi/demo_private.pem

ssl_algorithm	Either RS256 or ES256. It depends on how do you had generated the digital key. Type **RS256**
root_cert_filepath	The full path to the **roots.pem** we downloaded using wget command above (**/home/pi/roots.pem**)
project_id	Your project ID. You can see the project id info in your Google Cloud Platform dashboard.
gcp_location	The region you picked when creating the IoT device registry (**europe-west1**)
registry_id	The Registry ID you typed when creating the IoT device registry (**raspberry-pi**)
device_id	The Device ID you choose when creating the IoT device (**rasp1**)

8. Save the file using the save icon

Start the data pipeline

1. Run the code with this command:

```
python ./dt8030/dt8030-buttons.py
```

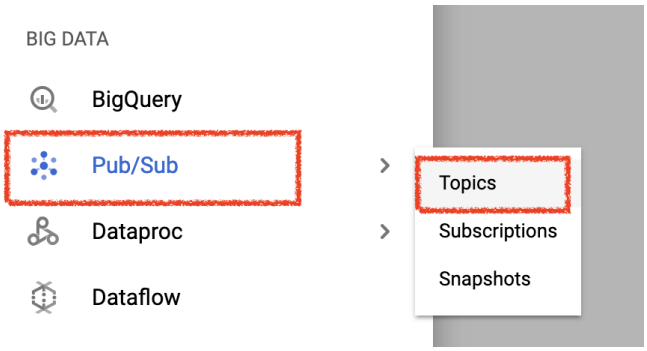
2. Move the “joystick” buttons on your SenseHat board

Verify that data is arriving into the cloud

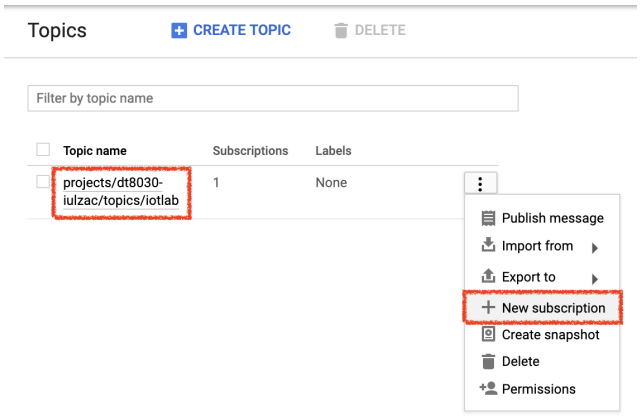
In this step, you will use the Google Cloud Shell to check if the data is arriving the the Google Cloud Platform. Google Cloud Shell provides command-line access to your GCP resources from the command line, and allows you to perform operations in Google Cloud Pub/Sub topics. You also need to create a Pub/Sub Subscription to access the Pub/Sub Topic.

Create a Pub/Sub Subscription

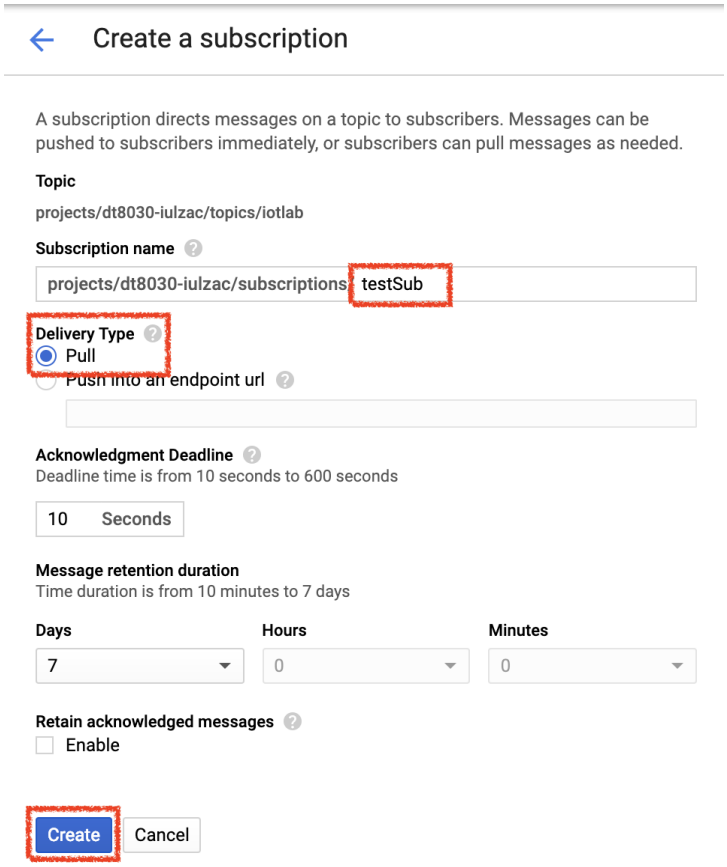
1. Click **Navigation menu** > **Pub/Sub** > **Topics**



2. Locate the topic you have created (**"iotlab"**), click the **three dot icon** > **Create subscription**

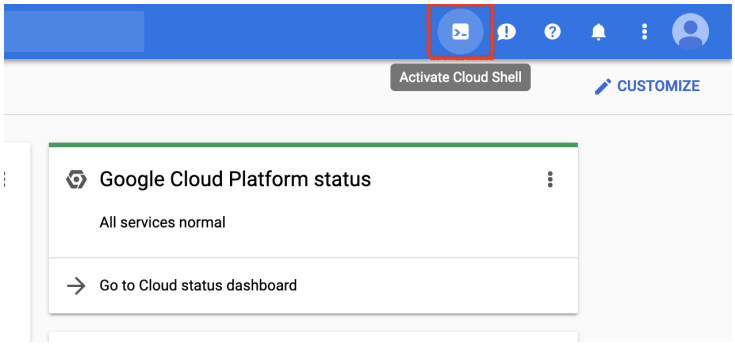


3. Type the name for the subscription: **testSub**
4. Set the **Delivery Type** to **Pull**, then click Create



Access the Pub/Sub Topic using the Cloud Shell

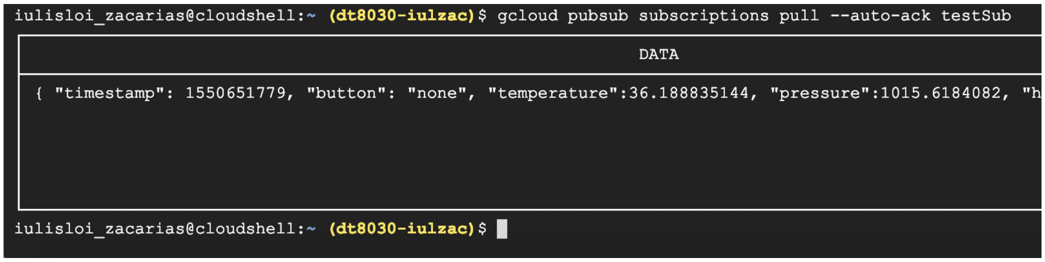
1. In GCP console, on the top right toolbar, click the **Open Cloud Shell** button



2. If a dialog box opens, click **START CLOUD SHELL**
3. To view the message you'll use the **testSub** subscription to **pull** the messages from the Pub/Sub topic (**iotlab**). Enter the following command in the Google Cloud Shell command line:

```
gcloud pubsub subscriptions pull --auto-ack testSub
```

4. The message appears in the DATA field of the command output, and should look like the next picture

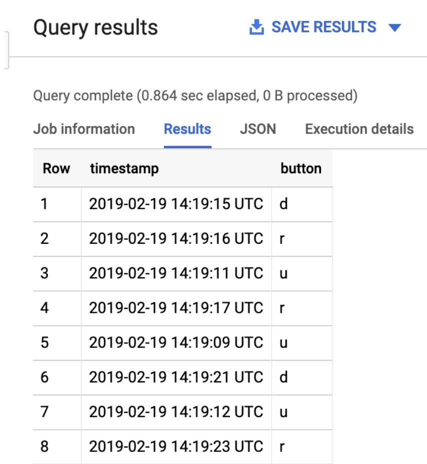


Examine the stored data

1. On the **Navigation menu**, click **BigQuery**
2. In the left-hand side of the browser window, click on **Compose New Query**
3. Enter the following query. Remember to replace **<project_id>** with the ID of your Google Cloud Platform project:

```
SELECT
  *
FROM
  `<project_id>.iotData.event_data`
LIMIT
  1000
```

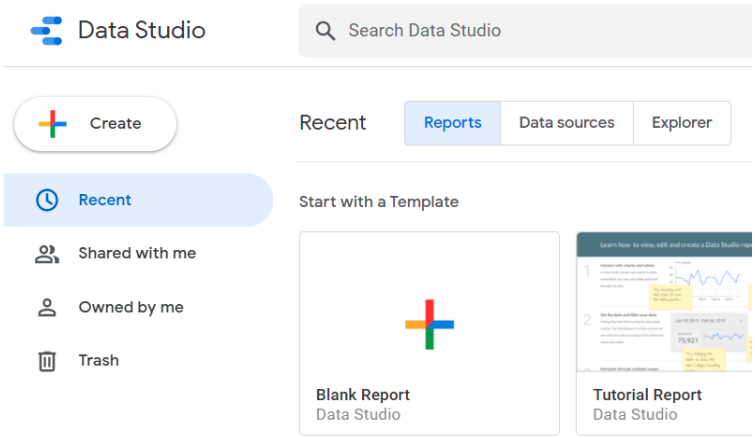
4. Browse the data using the previous-page and next-page links provided



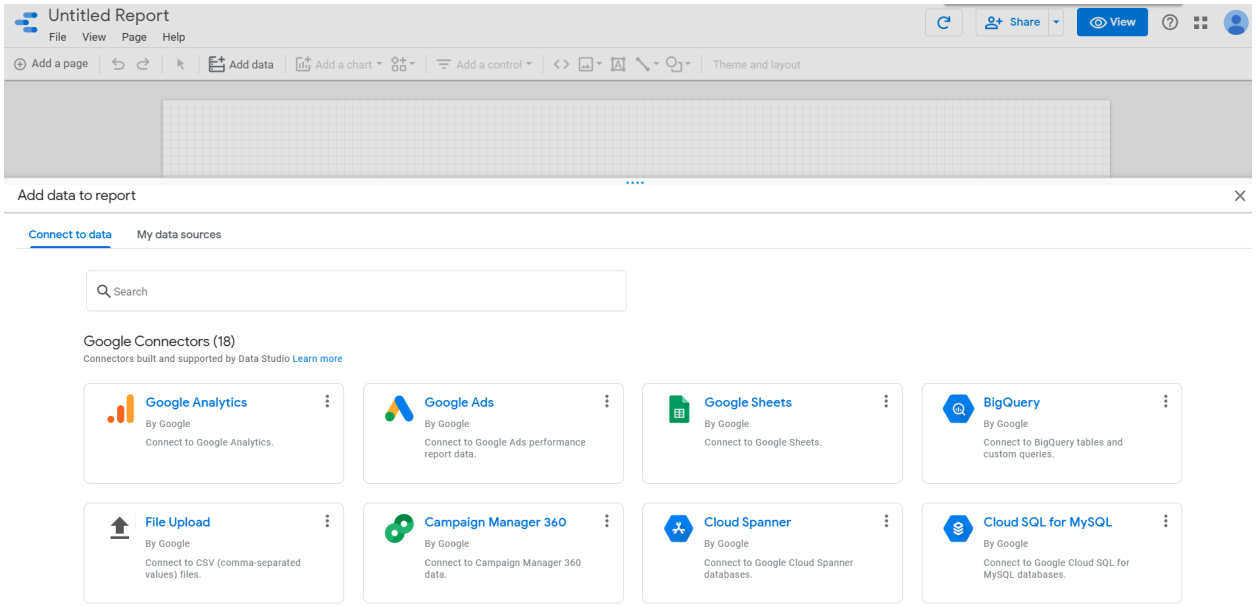
Create a Data Studio dashboard to visualise the data

Google Data Studio turns your data into informative dashboards and reports that are easy to read, easy to share, and fully customizable.

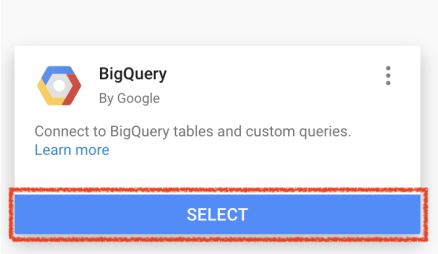
1. From your web browser, go to https://datastudio.google.com
2. Under “**Reports**” and **“Start with a Template”** , click on **Blank Report**.



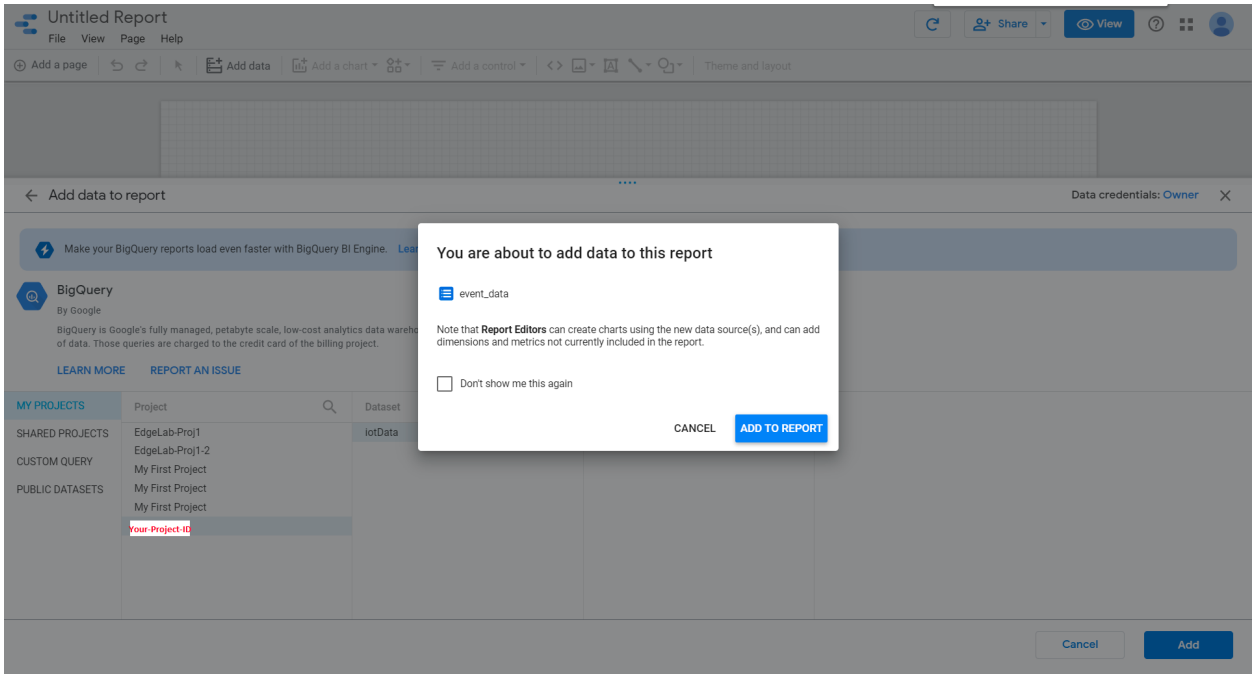
3. Click the checkbox to accept the terms, click the Next button, select which emails you are interested in receiving and click on the Done button. Once again, under "Start a new report", click on Blank
4. Under **Add data to report** and **Connect to data**.



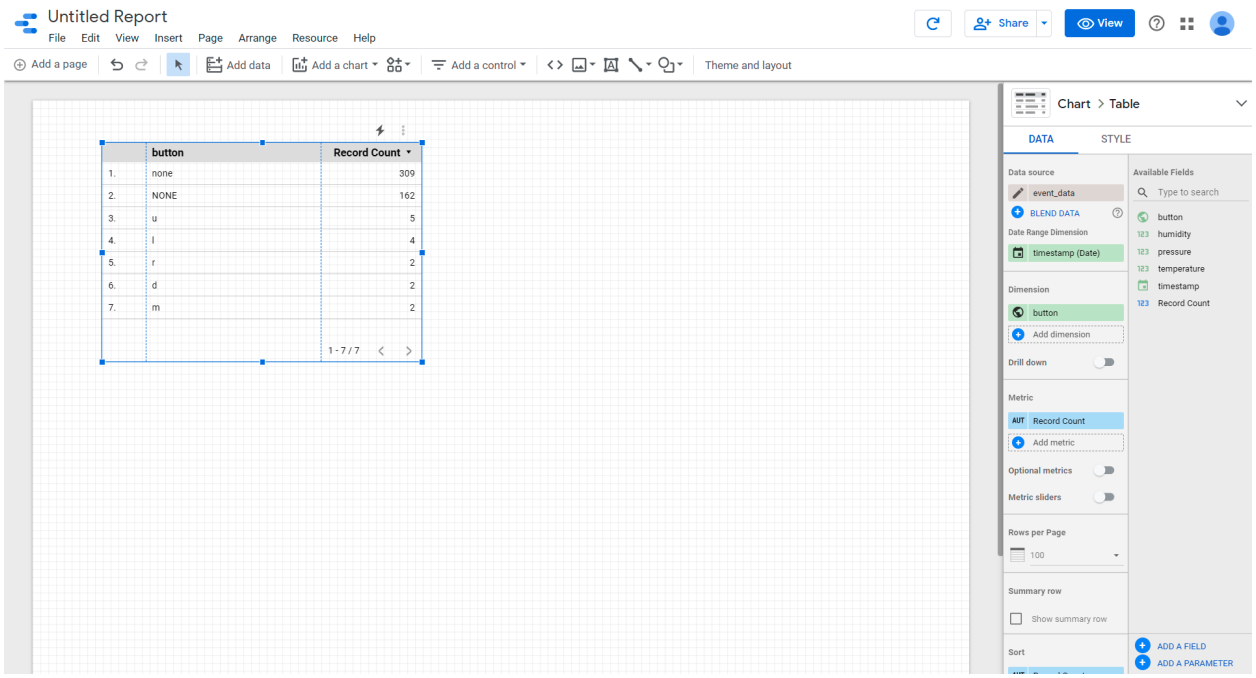
5. Search for **BigQuery,** click **Select**, then on the Authorize button and then choose the Google account you wish to use with Data Studio (it should be the same one that you have been using in the Google Cloud Platform project)



6. Click on the **Allow** button
7. Select your project name (your project ID in the Google Cloud Platform), dataset (**iotData**) and table (**event_data**)
8. Click the **Add** button, then, click **ADD TO REPORT**

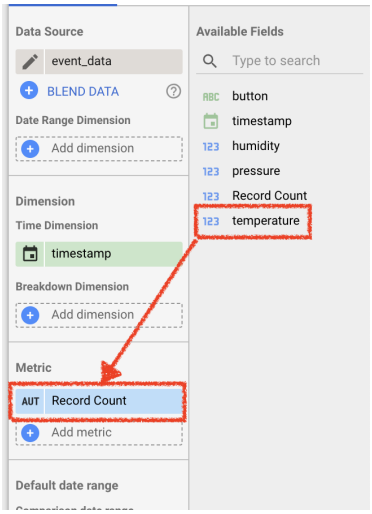


9. After this you will get the following screen.

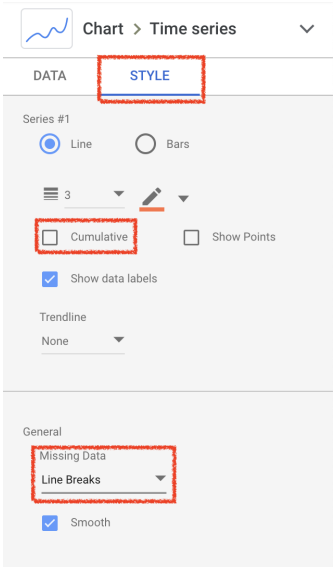


10. On the right hand side click on **Chart** and select **Time series chart.**

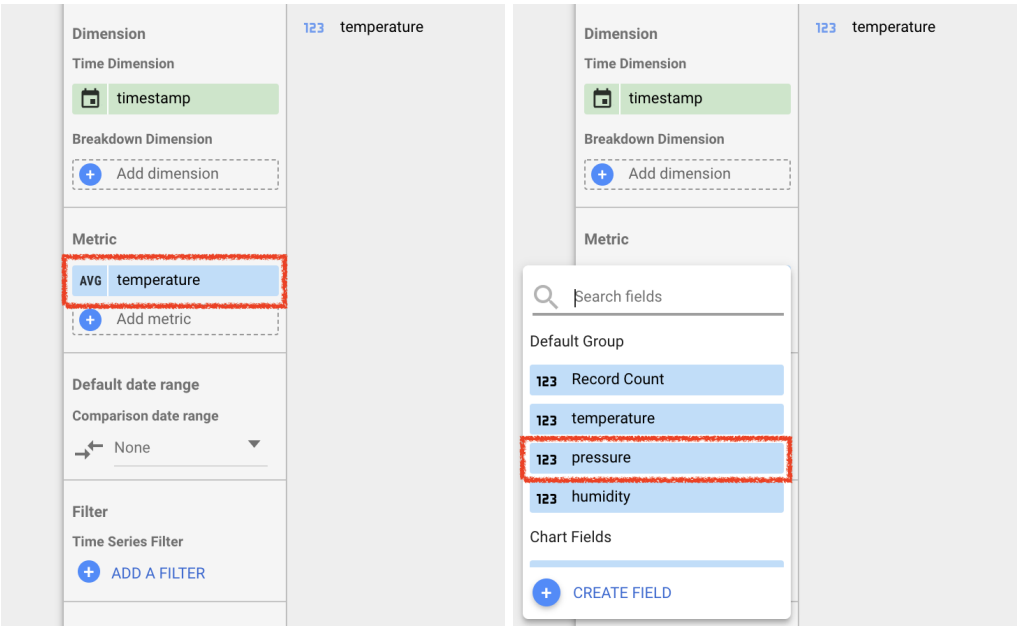
11. On the left panel, under the **Available Fields**, click the **temperature** field and drag it over the **Record Count** field



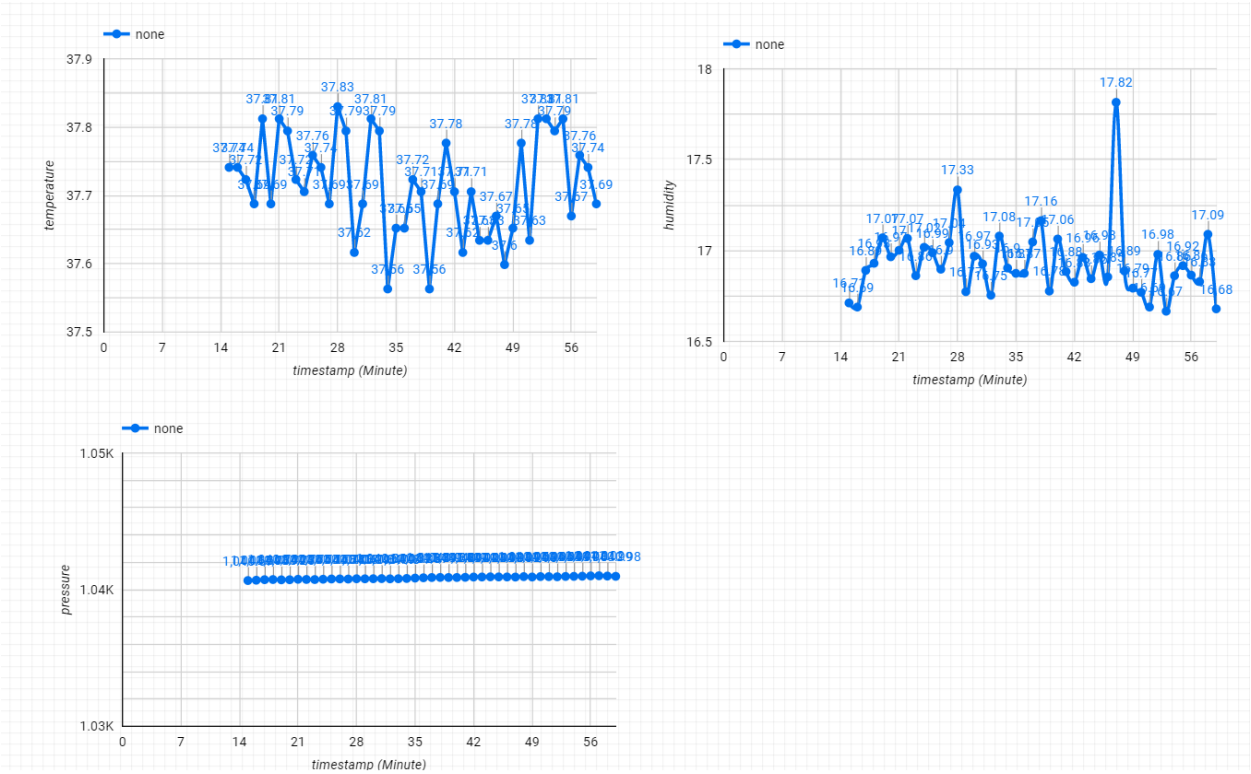
12. On the left panel, under the **Available Fields**, click the **temperature** field and drag it over the **Record Count** field
13. The amount of data that you see in the resulting charts will depend on how much data has been collected. If you built out the IoT sensor and have just started collecting data, you may only see one point.
14. On the right hand side of the window, select the **Style** tab
15. Uncheck the “**Cumulative**” option
16. Change Missing Data from "**Line To Zero**" to "**Line Breaks**"



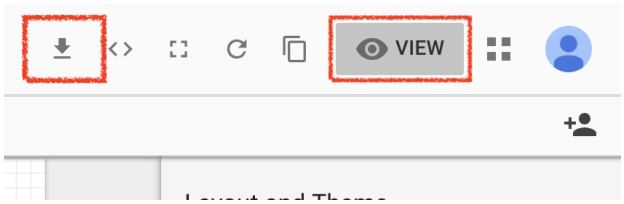
17. Click the graph on the sheet and copy/paste (Ctrl-C/Ctrl-V) it 2 times. Align the graphs so that each has ¼ of the layout
18. Click on each graph and under the **Metric** section (on the right panel) click on the existing metric (**temperature**), choose a different metric to be displayed until all tree weather readings (temperature, humidity and pressure) have their own graph



19. You now have a basic dashboard!



20. When you have finished your report, click on the **View** button to see the final version. You can make a PDF copy of your report by clicking in the **Download report button**



Data Studio has many capabilities not covered here. Feel free to explore with adding titles, line colors and other elements that may interest you.

You can also run the Qwiklab course: “**Visualizing Data with Google Data Studio (7 Credits)**” https://www.qwiklabs.com/focuses/1158?parent=catalog

Streaming accelerometer and magnetometer data to the cloud

In this section, you stream data from accelerometer and magnetometer sensors to the cloud in a high data rate. The setup is similar to the setup you have for streaming the button events and the weather data, and you can run both programs to collect data simultaneously. However, in the Google Cloud Platform an IoT Device (our real Raspberry Pi) is linked to an IoT Registry, and the IoT Registry, in its turn, is associated with a Pub/Sub Topic. The Pub/Sub Topic triggers the Cloud Function which inserts the data received by the Pub/Sub Topic in a BigQuery Table. Due to the characteristics of this architecture, it is necessary to instantiate (set up) a new "stack" of components in the Google Cloud Platform. In other words, you will create new instances of Pub/Sub Topic, IoT Core Registry, New Device in the IoT Core Registry (and add the key to authorize the Raspberry Pi to connect to this new Device / Registry), a new BigQuery Table to store the data, and a new Cloud Function. The steps to create all these components are described in the previous sections. Therefore, in this section, you only find the specification to create each component.

Pub / Sub Topic

Create a new Pub Sub Topic (Pub / Sub → Topics):

Property	Value (type or select)
Name	position

IoT Registry

Create a new IoT Core Registry (IoT Core → Create Registry)

Property	Value (type or select)
Registry ID	raspberry-pi-2
Region	europe-west1
Protocol	MQTT and HTTP
Cloud Pub/Sub topics	projects/**<project_id>**/topics/position

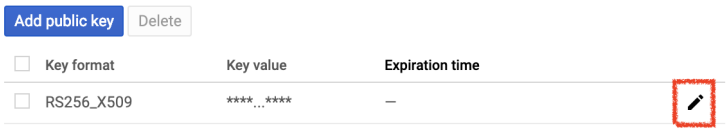
Add device to the IoT Registry

Create a new device (IoT Core → respberry-pi-2 → Devices)

Property	Value (type or select)
Device ID	rasp2
Device communication	Allow
Authentication	Enter manually (PS: You will add a digital key file later)

Authorize the device (add the digital key)

Add the digital key to authorize the device. You can just copy the key from device “rasp1”. From the **Navigation** menu, go to IoT Core, click on the “raspberry-pi” Registry, click Devices, locate the device “rasp1” and click on it. On the device details page, locate the RS256_X509 key and clink on the pencil icon at the end of line:



Copy the text box content under “Public key value”.

From the **Navigation** menu, click IoT Core, then click on the “raspberry-pi2” Registry name, click Devices, locate the device “rasp2” and click on it. Click **Add public key** button:

Property	Value (type or select)
Input method	Enter manually
Public key format	RS256_X509
Public key value	<< Paste the digital key content here >>

Big Query Table

Create a new BigQuery Table (BigQuery → **<project_id>** → iotData → Create table)

Property	Value (type or select)
Create table from	Empty table
Project name	**<project_id>**
Dataset name	iotData

Table name	position_data

Add the following fields to the table:

Field name	**Type**	**Mode**
timestamp	TIMESTAMP	NULLABLE
device_id	STRING	NULLABLE
accel_x	FLOAT	NULLABLE
accel_y	FLOAT	NULLABLE
accel_z	FLOAT	NULLABLE
direction	FLOAT	NULLABLE

Cloud Function

Add a Cloud Function to add data from the Pub/Sub Topic to the BigQuery Table (Cloud Functions → Create Function)

Property	**Value (type or select)**
Name	function-positionPubSubToBQ
Memory allocated	128MB
Trigger	Cloud Pub/Sub
Topic	position
Source code	Inline Editor
Entry point	subscribe

In the **index.js tab**, paste the following code over the code that is there. **Make sure to change the constant projectId:**

```
/**
 * Triggered from a message on a Cloud Pub/Sub topic.
 *
 * @param {!Object} event Event payload and metadata.
 * @param {!Function} callback Callback function to signal completion.
 */
var BigQuery = require('@google-cloud/bigquery');
```

```
var projectId = 'YOUR-PROJECT-ID-HERE'; // IMPORTANT!! Enter your project ID here
var datasetName = 'iotData';
var tableName = 'position_data';

var bigquery = new BigQuery({
  projectId: projectId,
});

exports.subscribe = function(event, callback) {
  var msg = event.data;
  var incomingData = msg
    ? Buffer.from(msg, 'base64').toString()
    : '{"timestamp":"1970-01-01 00:00:00","device_id":"", "accel_x":"0", "accel_y":"0",
"accel_z":"0", "direction": "0"}';
  var data = JSON.parse(incomingData);

  bigquery
    .dataset(datasetName)
    .table(tableName)
    .insert(data)
    .then(function() {
      console.log('Inserted rows');
      callback(); // task done
    })
    .catch(function(err) {
      if (err && err.name === 'PartialFailureError') {
        if (err.errors && err.errors.length > 0) {
          console.log('Insert errors:');
          err.errors.forEach(function(err) {
            console.error(err);
          });
        }
      } else {
        console.error('ERROR:', err);
      }

      callback(); // task done
    });
};
```

In the **package.json** tab, paste the following code over the placeholder code that is there:

```
{
 "name": "cloud_function",
 "version": "0.0.1",
 "dependencies": {
  "@google-cloud/bigquery": "^1.0.0"
 }
}
```

Start streaming accelerometer and magnetometer data

Do not forget to update the project and device configuration before running the script. To do so, use a text editor (for example, the Geany Editor), open the file dt8030-position.py, and set the values of the following variables:

```
41
42    ssl_private_key_filepath = ''    # /home/pi/demo_private.pem
43    ssl_algorithm = ''               # RS256
44    root_cert_filepath = ''          # /home/pi/roots.pem
45    project_id = ''                  # your project ID
46    gcp_location = ''                # europe-west1
47    registry_id = ''                 # the registry name (raspberry-pi-2)
48    device_id = ''                   # the device ID (rasp2)
49    ###############################################################################
```

If you are following this document and using the suggested locations and names for resources and files, you can use the values from the next table:

Variable	Value
ssl_private_key_filepath	/home/pi/demo_private.pem
ssl_algorithm	RS256
root_cert_filepath	/home/pi/roots.pem
root_cert_filepath	**YOUR-PROJECT-ID**
gcp_location	europe-west1
registry_id	raspberry-pi-2
device_id	rasp2

Start streaming accelerometer and magnetometer data from your Raspberry system. Using a **Terminal** window, run the python script “dt8030-position.py” on the “dt8030” directory:

```
python ./dt8030/dt8030-position.py
```

Verify that you receive data into the cloud by one of the methods you already used earlier.

Can you see movement of the Raspberry Pi in the data?

Outcomes from this lab

At the end of this lab, you should understand the employed architecture and how components interact with each other (Pub/Sub Topics, IoT Registry, IoT Device, Cloud Functions, BigQuery tables). You also should be able to set up a basic using a Raspberry Pi Device to stream sensor data to the cloud.

As a result of this lab, you should provide a short report of the developed activities with a maximum of 3 pages, in PDF format, including some graphs generated with Data Studio. Please include a short description of how data flows from the sensor device (the Raspberry Pi in our lab) to the data storage location (Big Query).

Also, answer the following questions:

- Imagine you want to include additional sensors to your system, and collect data from them (for example a GPS module which collects the latitude, longitude and altitude data). Which components should you modify to end with the data in your BigQuery Table? To simplify, let's suppose that the SenseHat provide these data using the methods "sense.get_latitude", "sense.get_longitude" and "sense.get_altitude").
- Describe the steps to include an additional IoT sensor to this system (i.e., another Raspberry Pi collecting data).

- How often can our program dt8030-position.py sample the acceleration (and send it to the cloud)?
- Estimate how much data is flowing in this and the earlier scenarios.
- Elaborate on the possibility to reliable detect movement of the Raspberry Pi.

The report should be submitted in the Blackboard under the assignment "Lab 3 Submission".

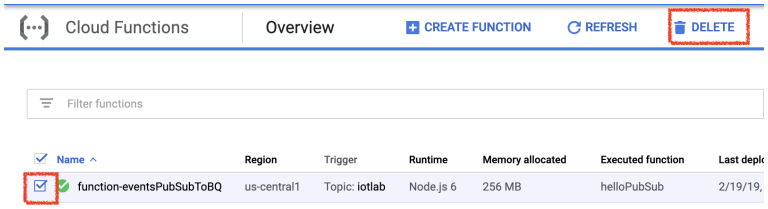
Cleaning up

Once you are done experimenting with the data, you can remove the running resources.

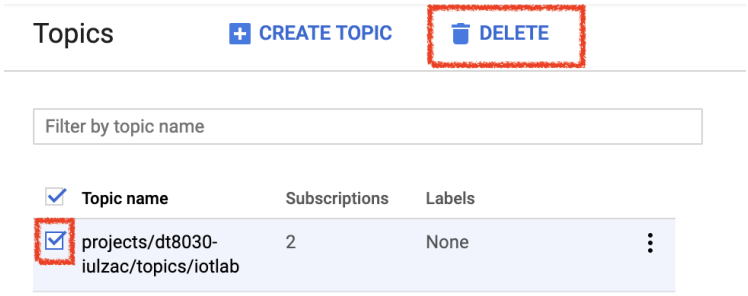
If the python script still running in your Raspberry Pi, hit Ctrl-C in the terminal window to stop the script and then type the following to power down the Raspberry Pi

```
sudo shutdown -h now
```

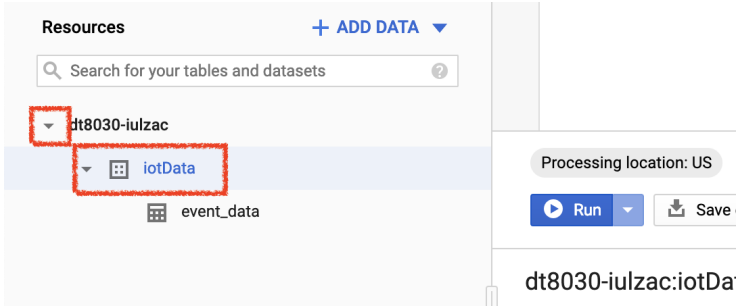
Go to Cloud Functions (on Google Cloud Platform), click on the checkbox next to **function-eventsPubSubToBQ** and **function-positionPubSubToBQ** and then click on **Delete**

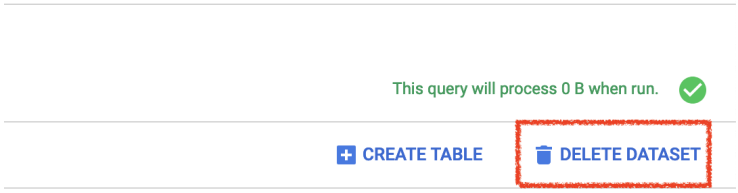


Go to **Pub/Sub** (on Google Cloud Platform), click on the checkboxes next to the **iotlab** and **positions** topics and then click on **Delete**



Go to **BigQuery**, click the down arrow next to your project name, click the down arrow to the right of the iotData dataset, then click on **Delete dataset**.





When prompted, type in the dataset ID (iotData) in order to finish deleting the data.

Go to **IoT Core** (on Google Cloud Platform), click on the Registry ID (**raspberry-pi**) then click on “**Devices**” on the left panel. Click on the device name (**rasp1**), then click “**DELETE**”. Confirm deletion by typing the device ID (**rasp1**). Again, on the left panel click “**Registry details**” and click “**Delete Registry**”**.** When prompted, type in the Registry ID (**raspberry-pi**) in order to finish deleting the registry. Repeat the same steps to delete the **rasp2** IoT Device and the **raspberry-pi-2** registry.

References

“Setting up your Raspberry Pi”. Online:
https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

“Getting started with the Sense HAT”. Online:
https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat

“Streaming IoT Data to Google Cloud Storage”
https://www.qwiklabs.com/focuses/2765?parent=catalog

“Cloud IoT step-by-step: Connecting Raspberry PI + Python”
https://medium.com/google-cloud/cloud-iot-step-by-step-connecting-raspberry-pi-python-2f27a2893ab5

“Using IoT Core to Stream Heart Rate Data”
https://codelabs.developers.google.com/codelabs/iotcore-heartrate/index.html

“Visualizing Data with Google Data Studio”
https://www.qwiklabs.com/focuses/1158?parent=catalog