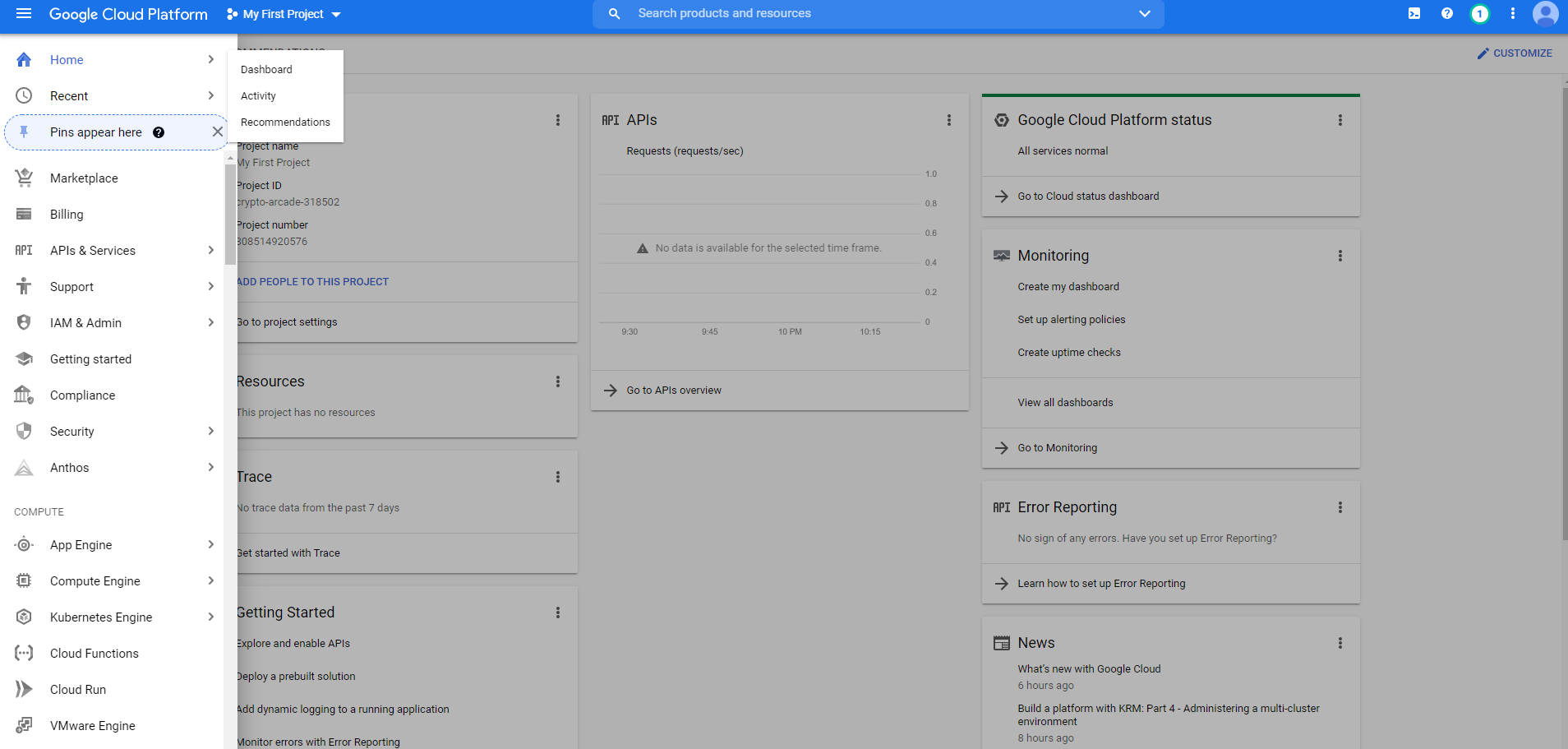
**Raspberry Pi to Google Cloud BigQuery**

Inspired by: <https://codelabs.developers.google.com/codelabs/iot-data-pipeline#0>

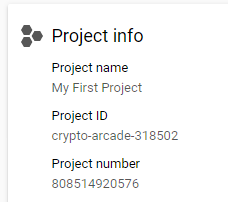
**First Step: Starting a Google Cloud Account**

First go to console.cloud.google.com and create an account using their free trial, this will give you access for 90 days and $300 of credit.



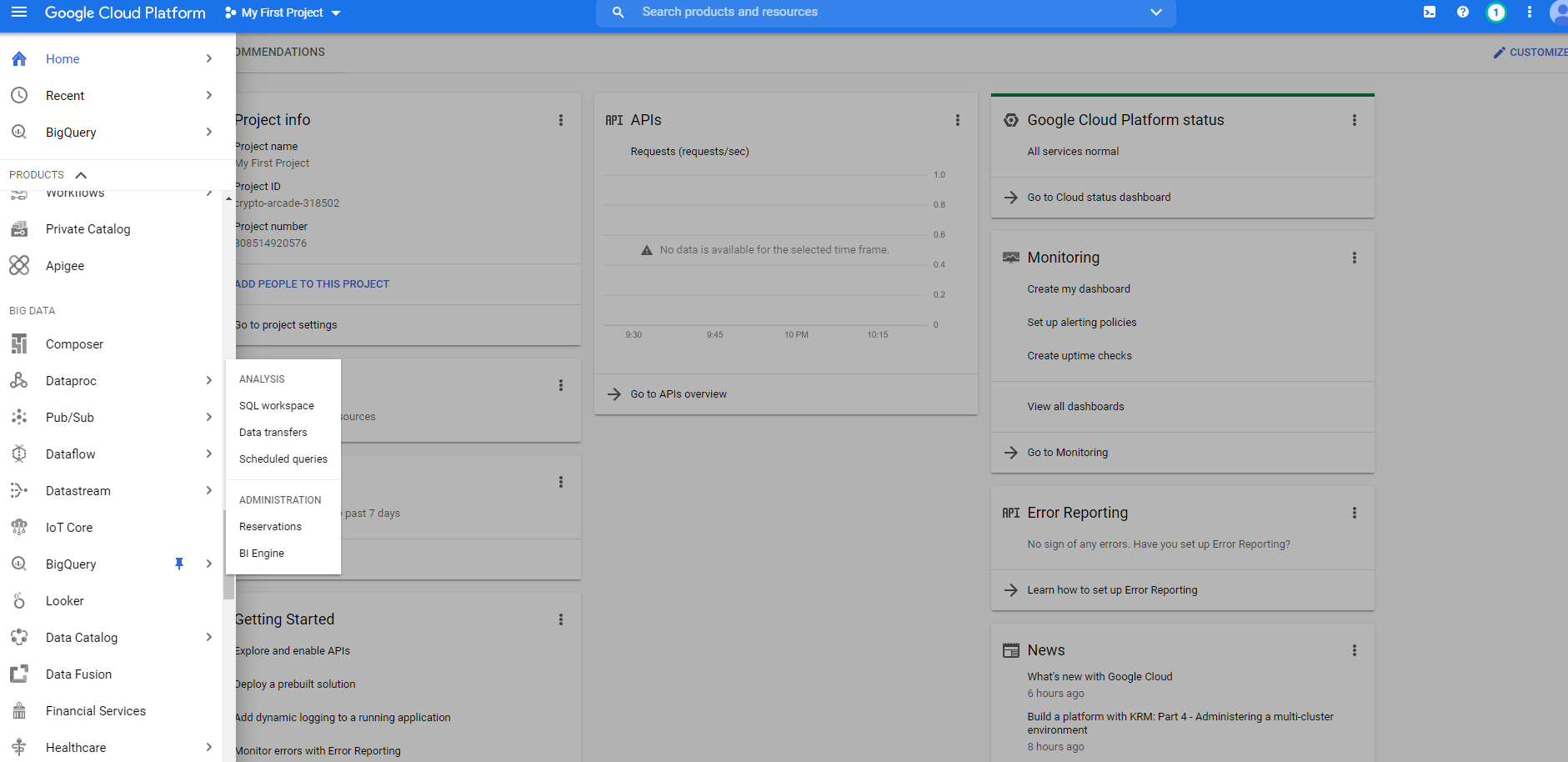
Once your account is created click on the **hamburger** icon on the top right to access the navigation menu, then click on **home** and go to **dashboard.**

This will show you your First Project make sure to take note of the Project ID in a notebook, you can always refer back to it by coming back here, but it is easier to just write it down somewhere for easier access.

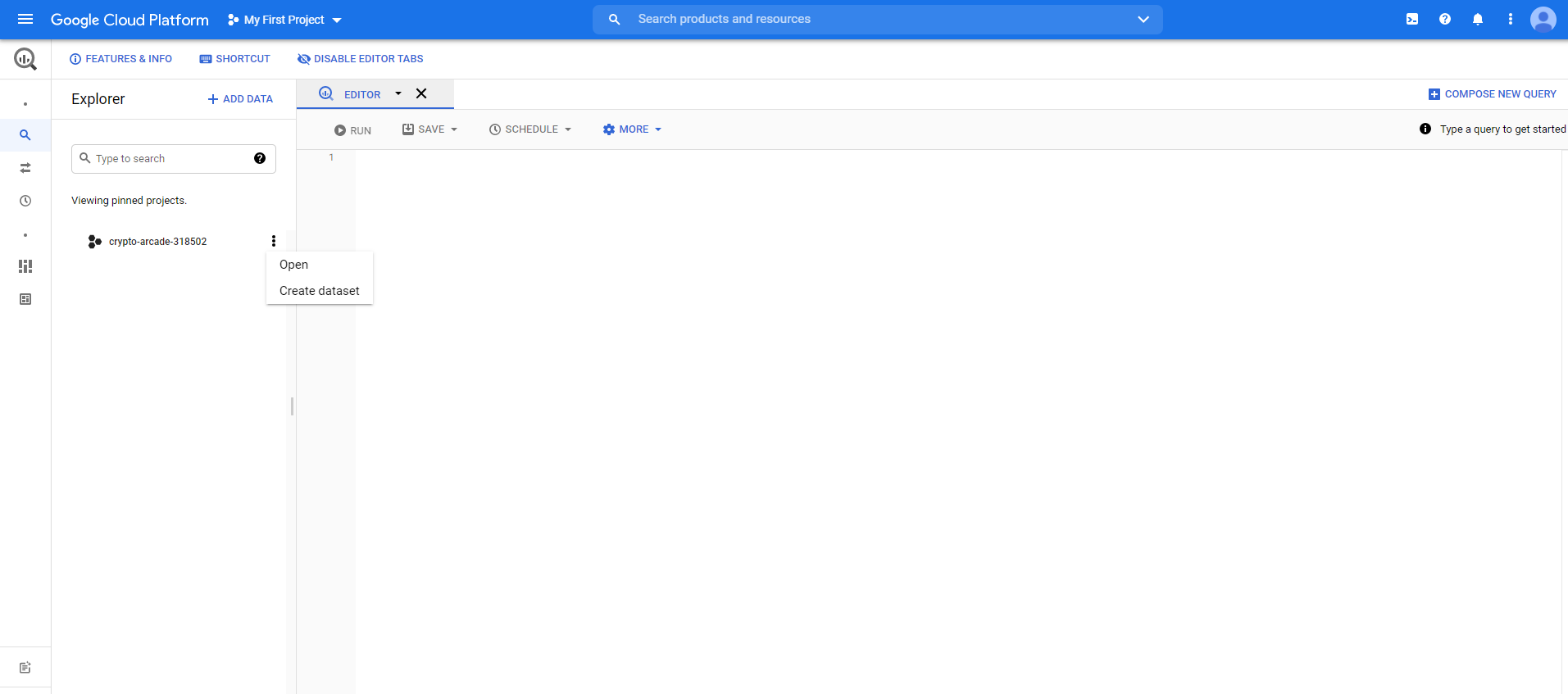


**Second Step: Making a BigQuery Table**

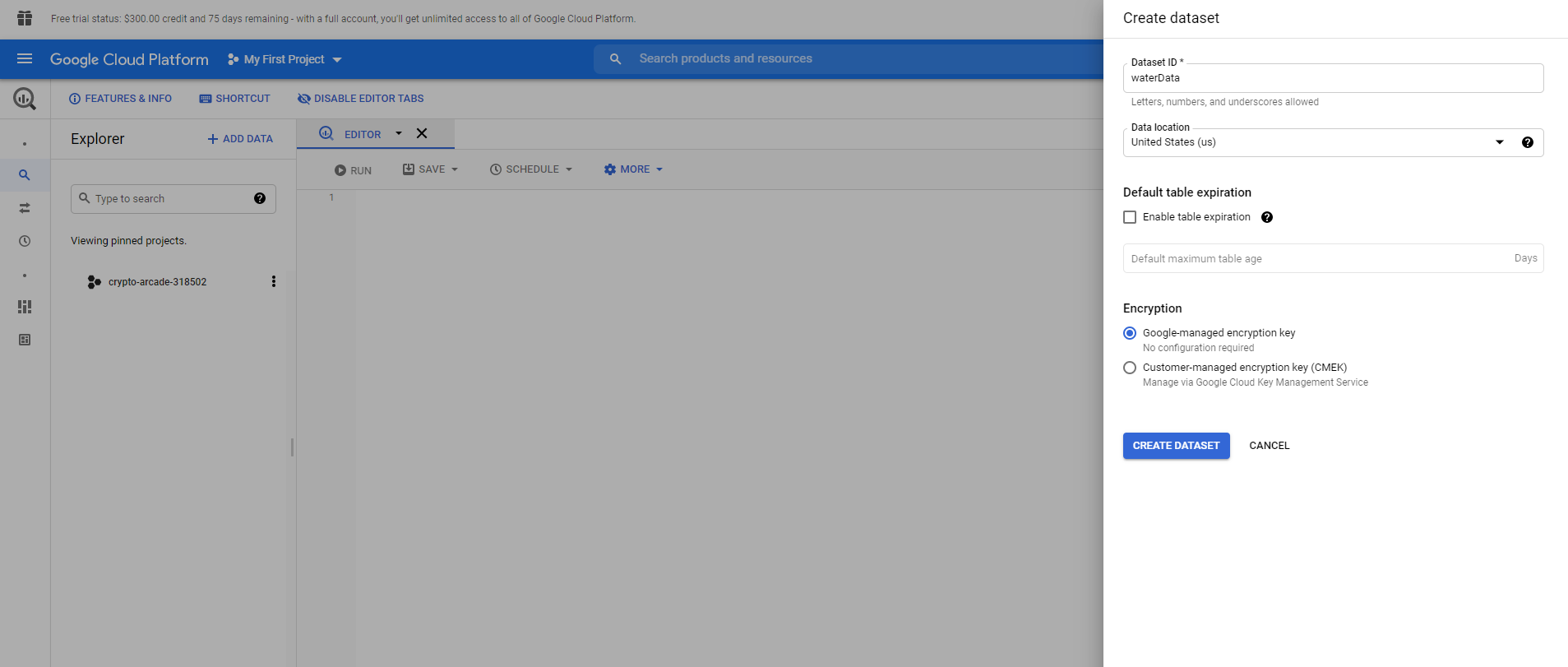
Now you will click on the **hamburger icon** again to access the navigation menu then you will scroll down to the **Big Data** section and select **BigQuery**, I highly recommend pinning the category so it will show up at the top of the navigation menu from now on.



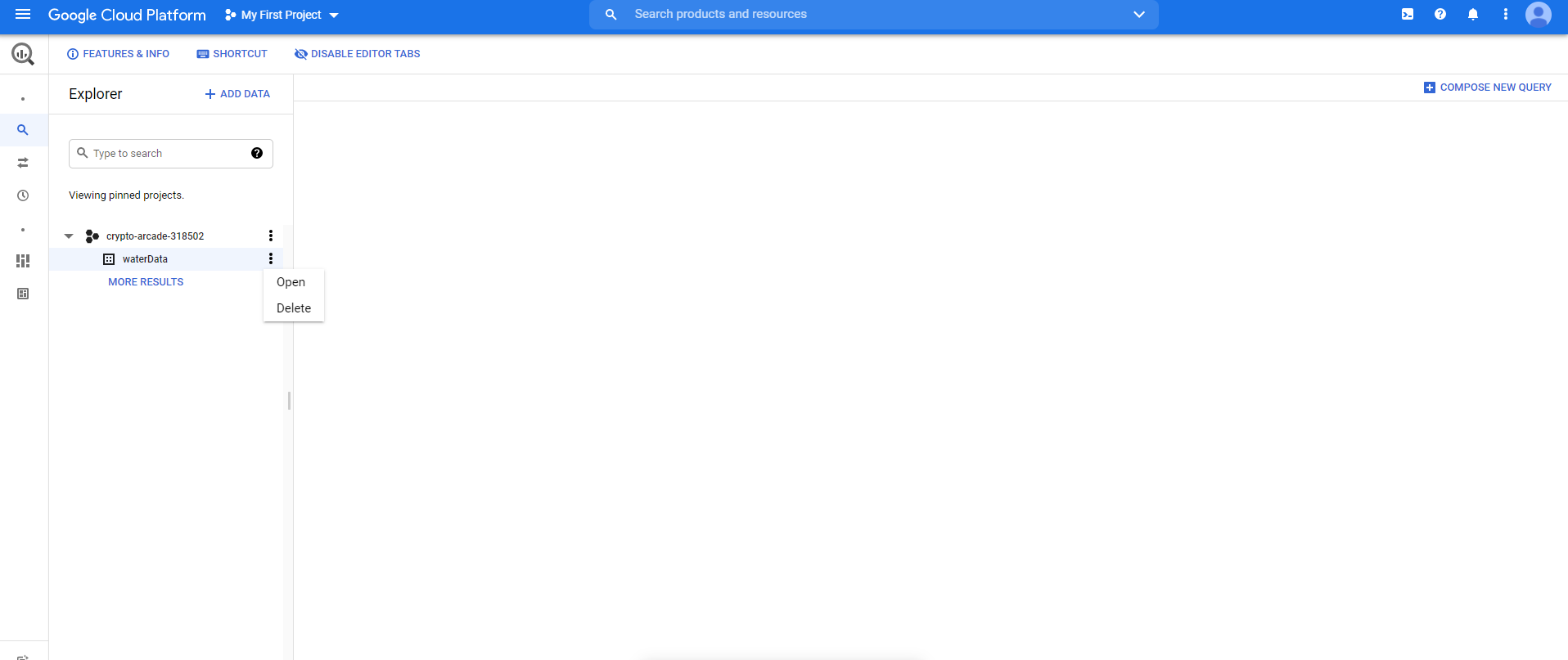
Once in the **BigQuery** section, navigate to pinned projects, which will be under the **Explorer**. Once you’ve found your project, click the three vertical dots and click **Create Dataset**.



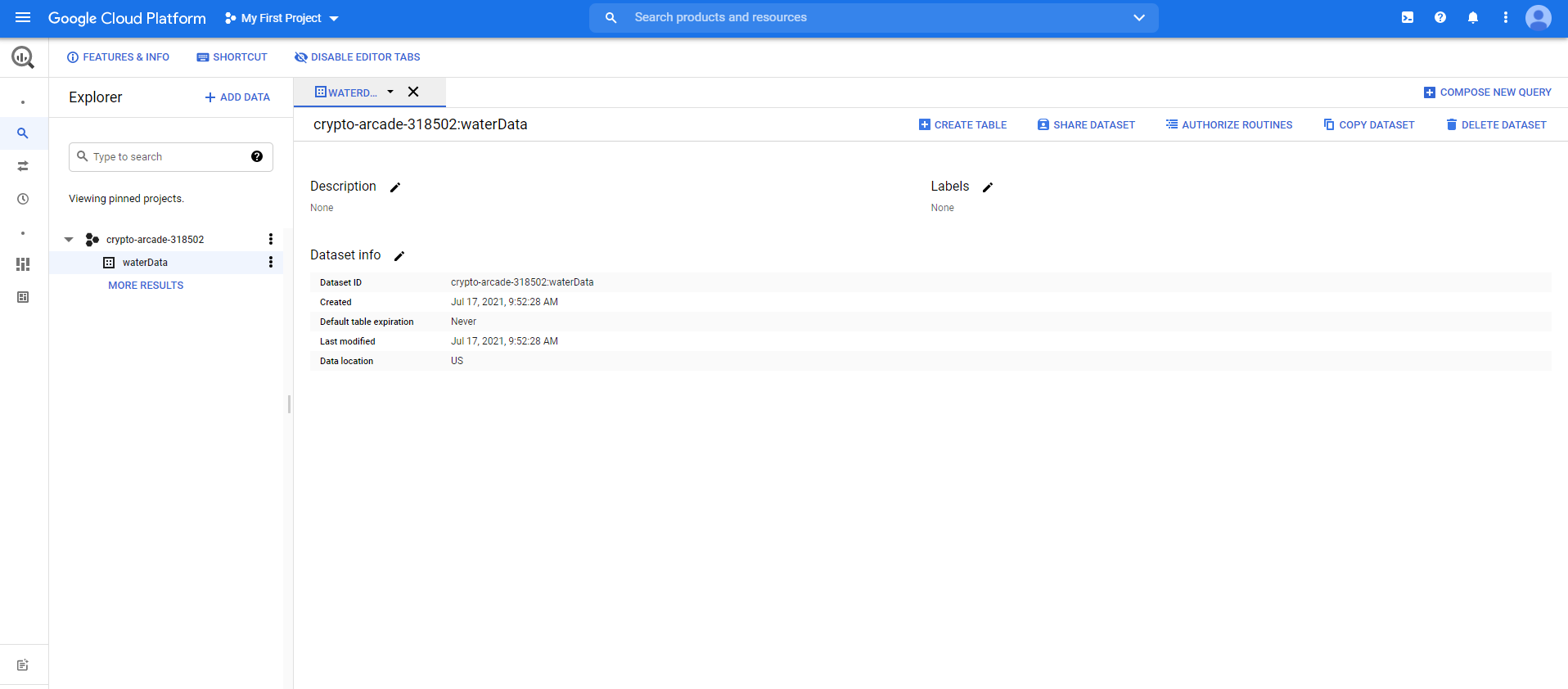
Once you have clicked **Create Dataset** this will open a pop-up on the right side, there are two fields that must be filled out **Dataset ID** and **Data Location**, the **Dataset ID** will be something like waterData and **Data Location** will be what region you are located in.



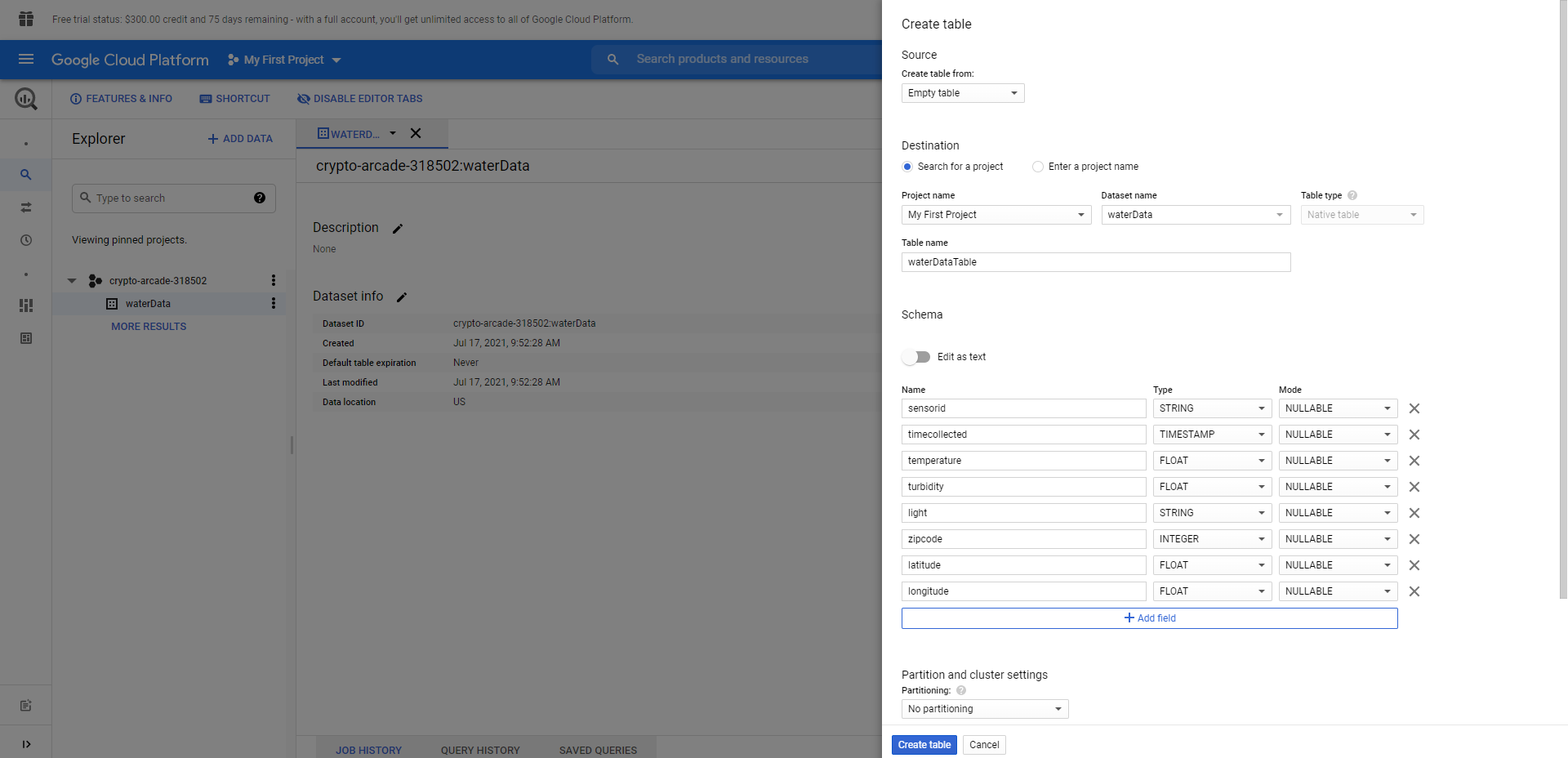
Once these fields are filled click the blue button at the bottom that says **Create Dataset**. This will create the ability to dropdown a menu where it says your project name, this will show the dataset’s name. You will then click on the three dots on the dataset’s row. This will give you two options, **Open** and **Delete**. You will click on **Open**.



You will then be greeted with a page like this, from here you will click **Create Table**.



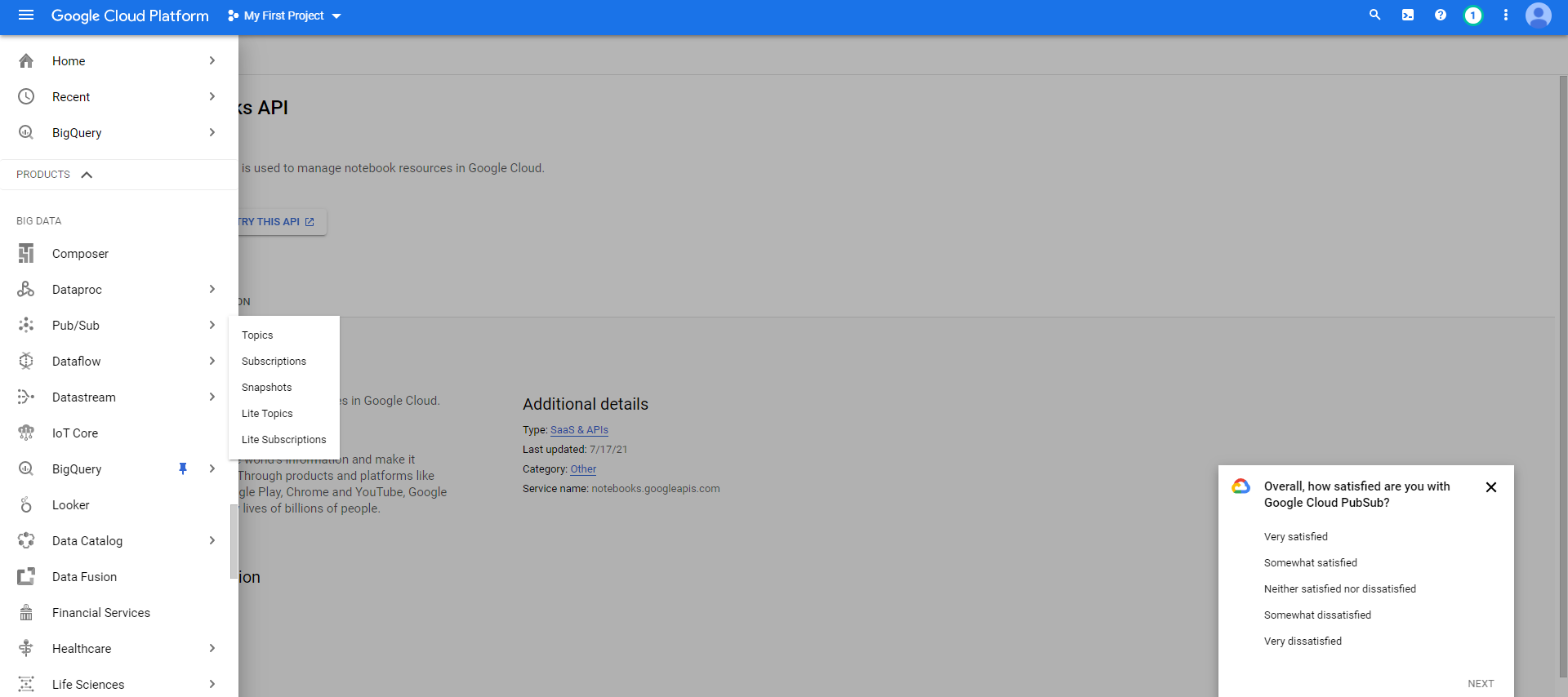
Once you have clicked that, another window will pop-up from the right side, this is where we will enter our schema and the name of the table.



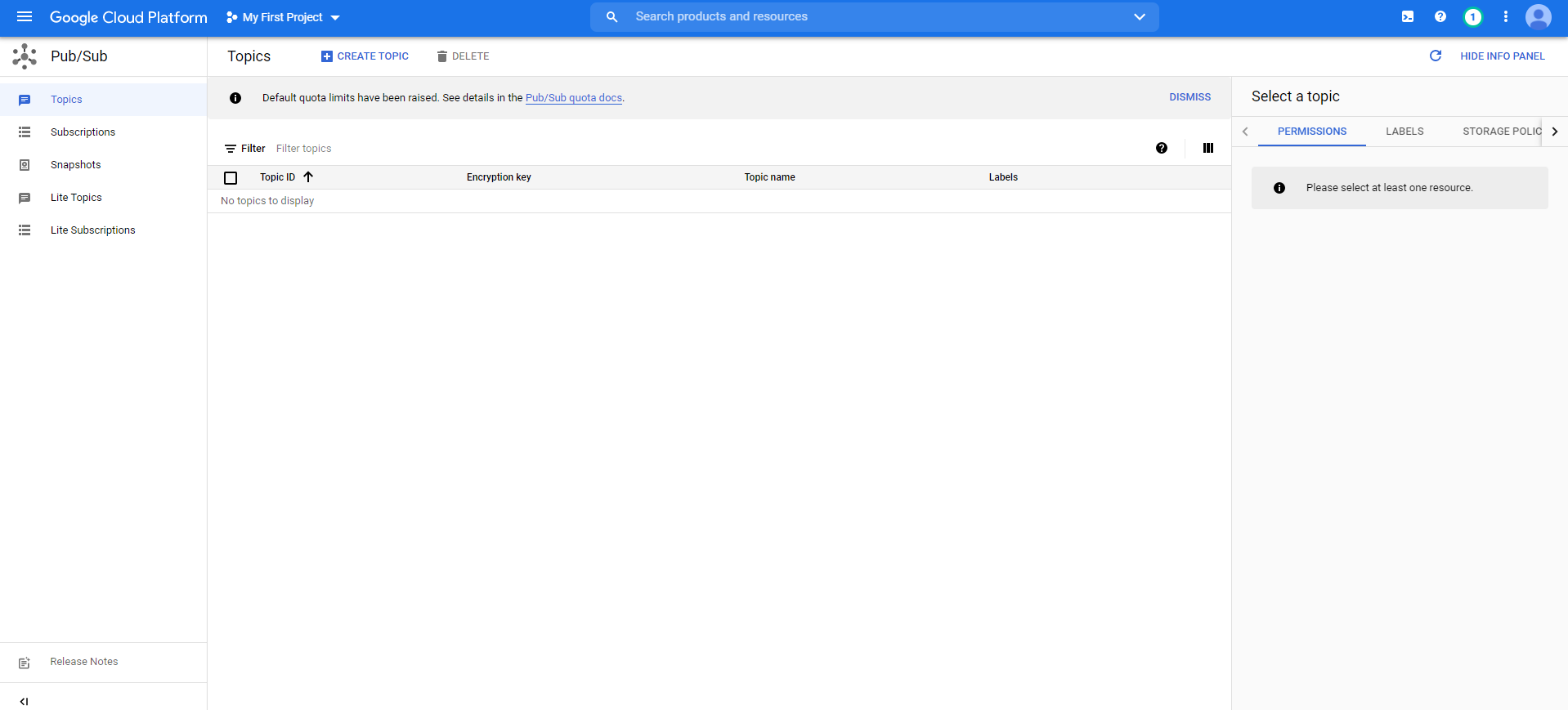
You will want to set the **Table Name** to something like waterDataTable and for the schema you will want to follow this exact order and naming convention. Your first field will be sensorid and the type is STRING, second timecollected and the type is TIMESTAMP, third temperature and the type is float, fourth turbidity and the type is float, fifth light and the type is STRING, sixth is zipcode and the type is INTEGER, seventh is latitude and the type is FLOAT, and lastly longitude and the type is FLOAT. The Mode of all the fields will be NULLABLE. Then you will click **Create Table** when it is fully filled out. Once this is done, we will move on to creating a **Pub/Sub Topic**.

**Third Step: Making a Pub/Sub Topic**

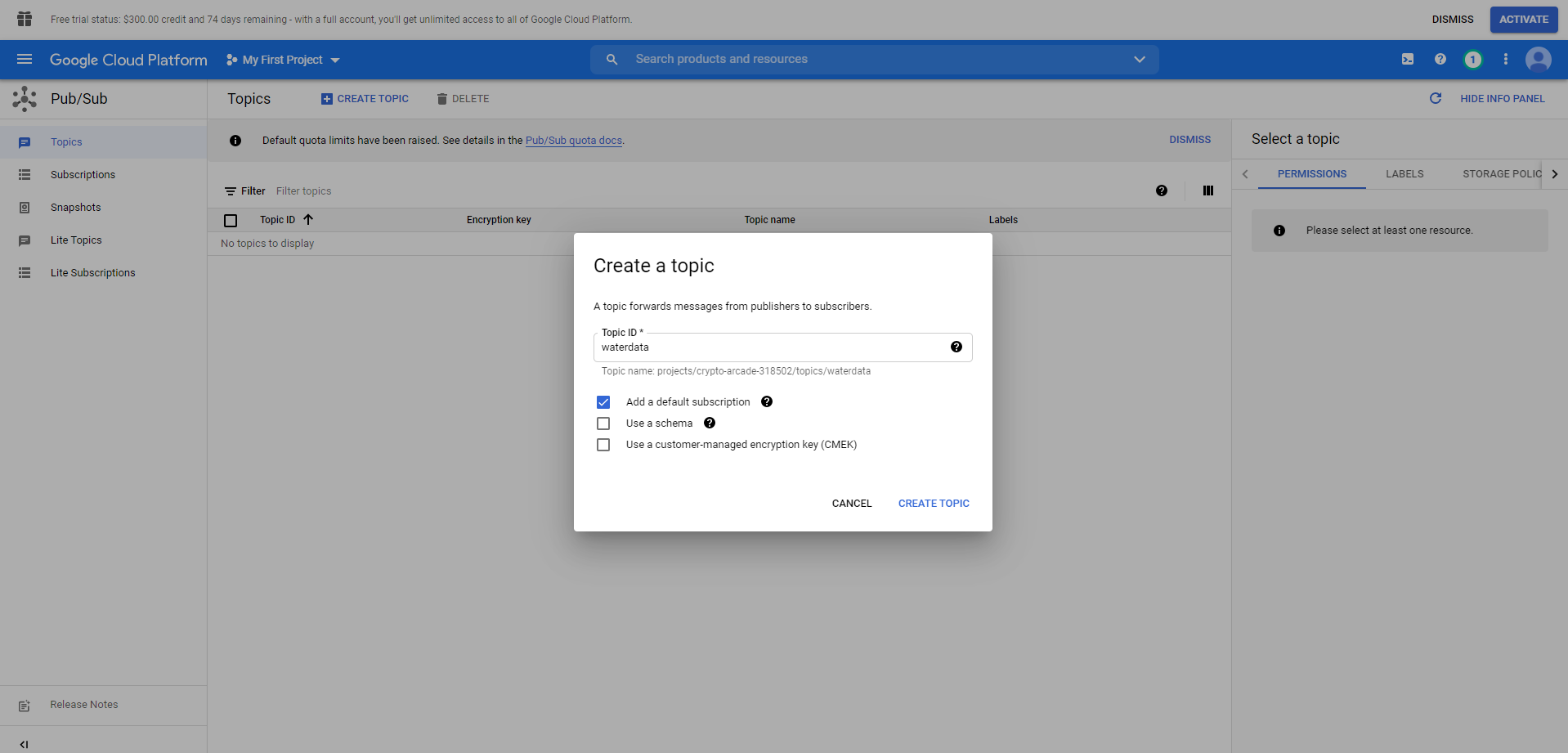
Now you will click the **hamburger icon** in the top right corner again and navigate to **Big Data** and under that you should be able to find **Pub/Sub**, then you will click on **Topics** in that pop-up menu from **Pub/Sub**. Also again I recommend that you pin the category.



This should bring you to a page where you can **Create a Topic**, but if it for some reason shows you an **Enable API** button, first click that and then redirect back to **Pub/Sub > Topics**.



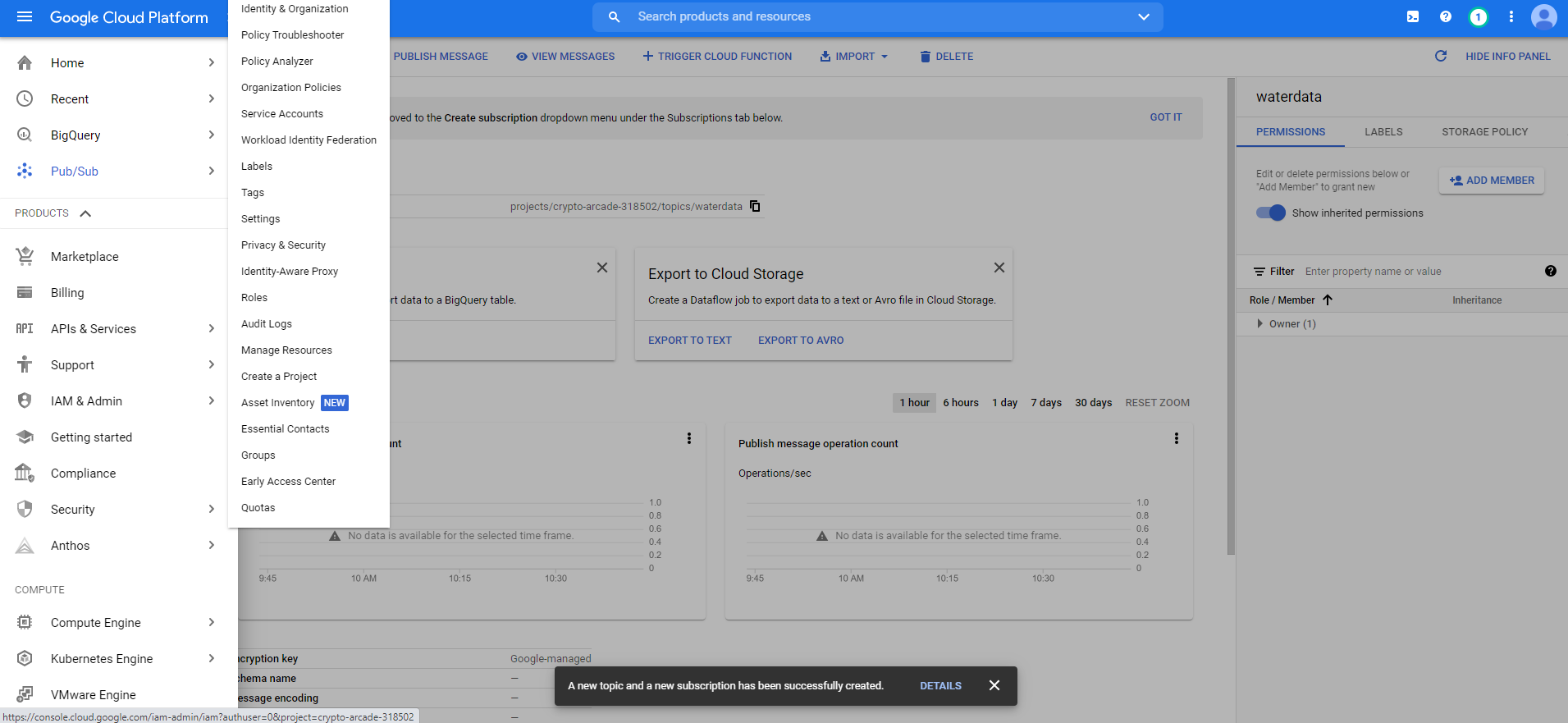
Then you will click **Create Topic** at the very top just under the blue header. This will show a pop-up menu with a header that says ‘Create a topic,’ and the only field you need to fill out is **Topic ID**. For this field you will want to name it something like waterdata.



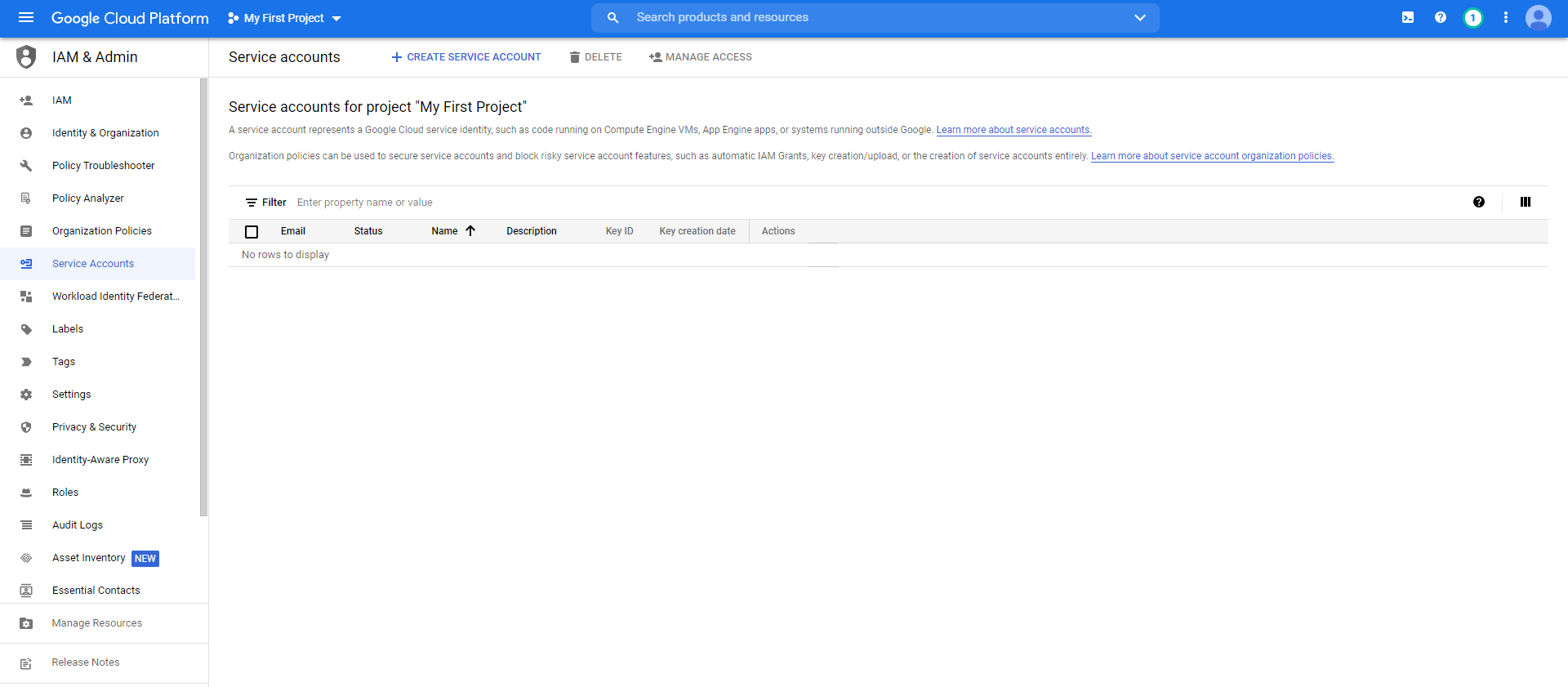
Once the fields are filled out click **Create Topic**, after that is done we will want to make sure our communications from the IoT device to the publishing topic is secure.

**Fourth Step: Secure Publishing**

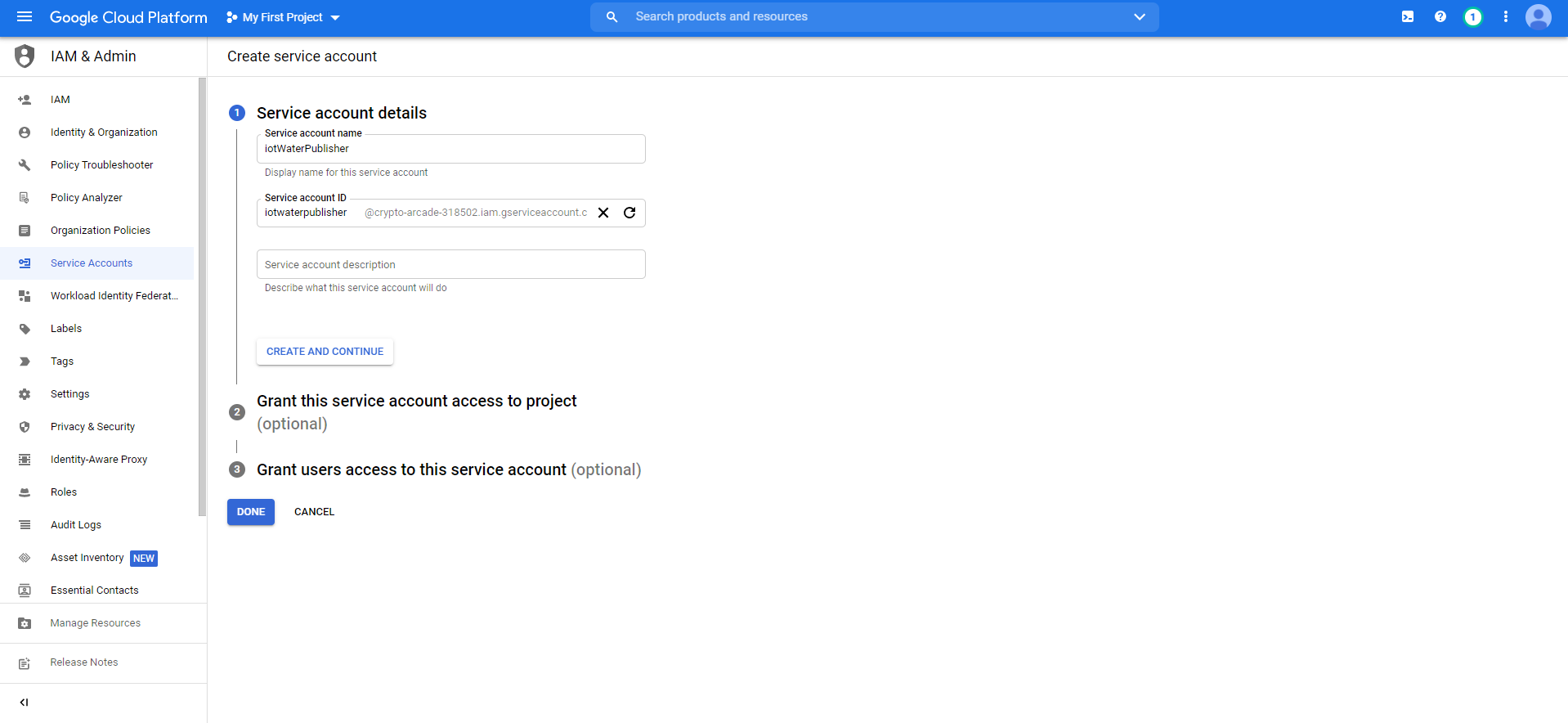
Now we will redirect to the **hamburger icon** again and move down to **IAM & Admin > Service Accounts**.



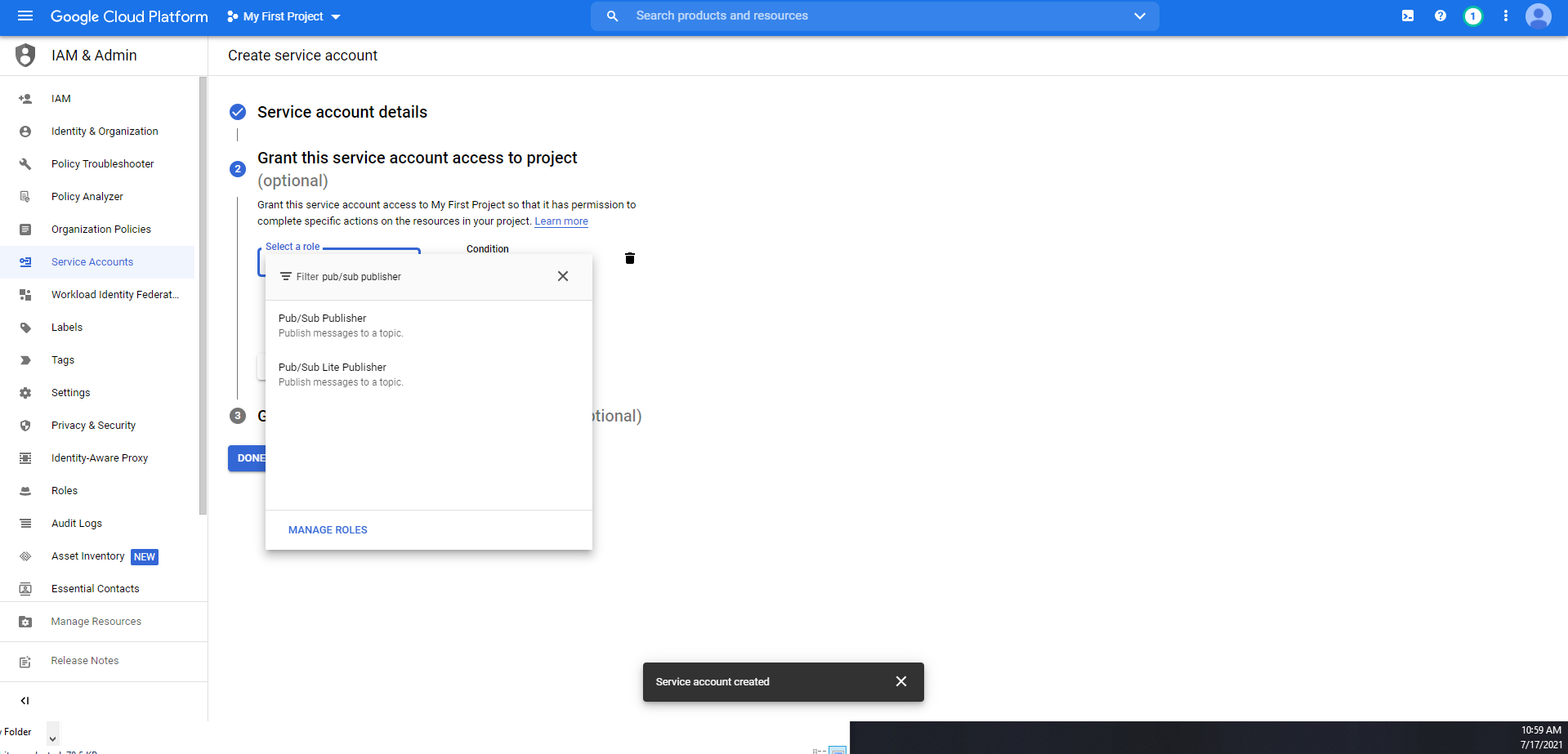
Once there you will click on **Create Service Account**. Which is located at the very top under the blue banner.



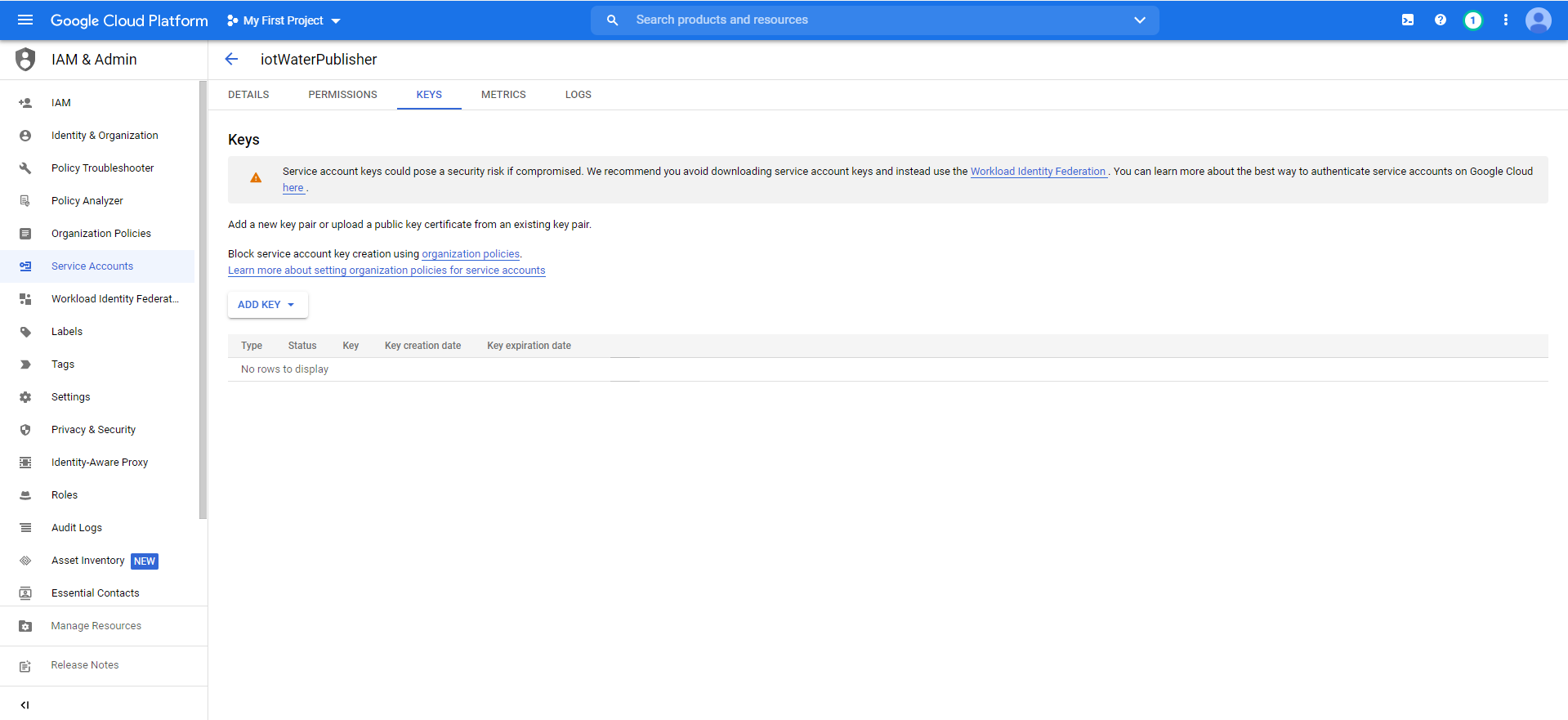
Once you have clicked on **Create Service Account** it should redirect you to a new page where you are to enter information regarding the account. The first page will ask for an account name, for this you will want to make it something like iotWaterPublisher. Then you will click the **Create And Continue** button.



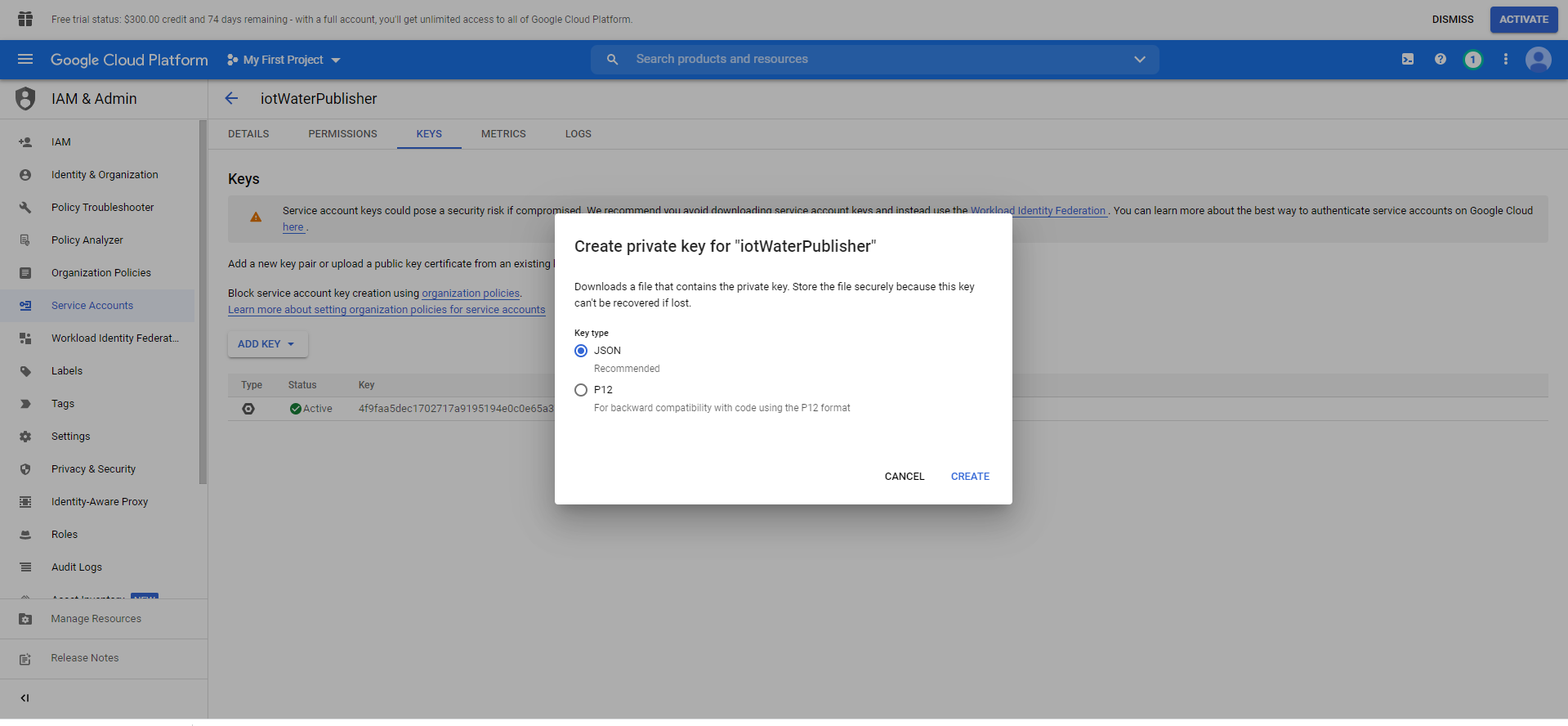
Once you have clicked that button it will bring you to the next step which is to enter in the role, you must click the dropdown menu on **Select a role** and in the search bar enter: ‘pub/sub publisher’ then select the top result. Then you will click **Done**.



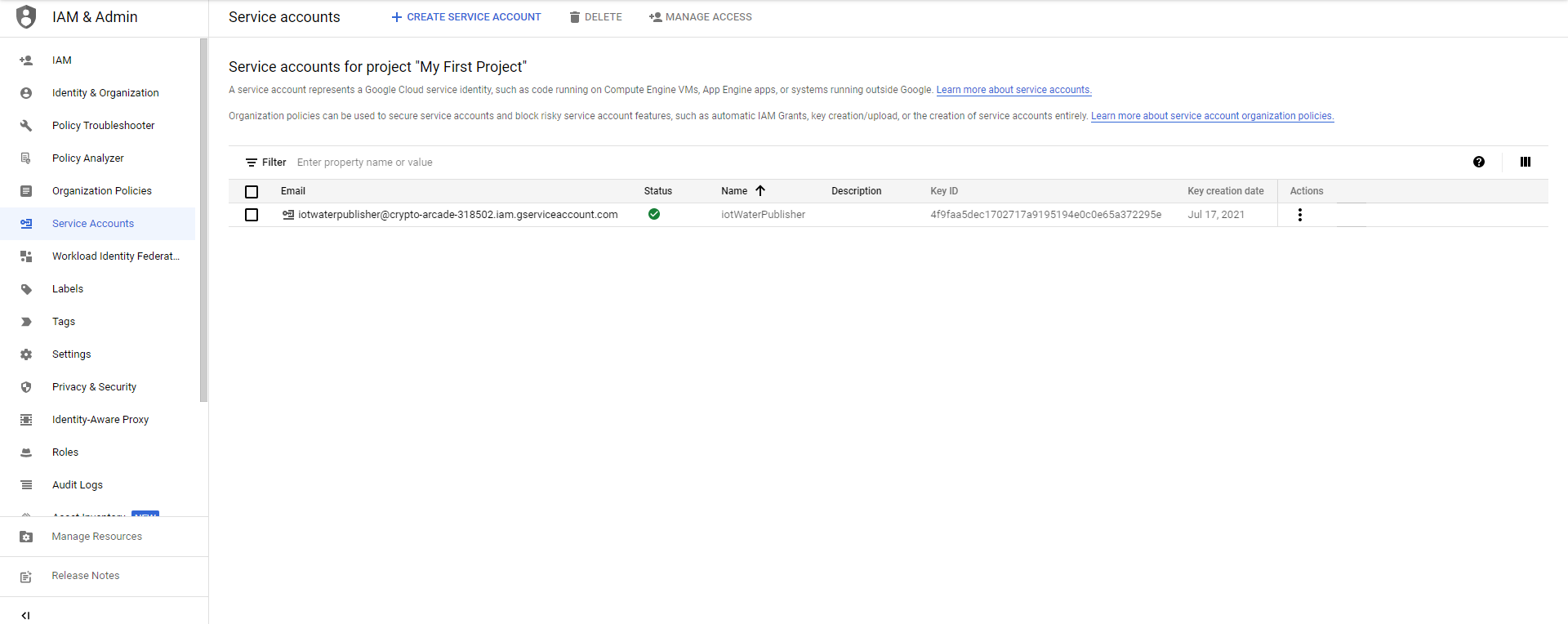
Once you have clicked the **Done** button you will be brought back to the home page of Service Accounts. You should see one entry now with the name of iotWaterPublisher, from here navigate to **Actions** in the same row as that name and click the **three vertical dots > Manage keys**.



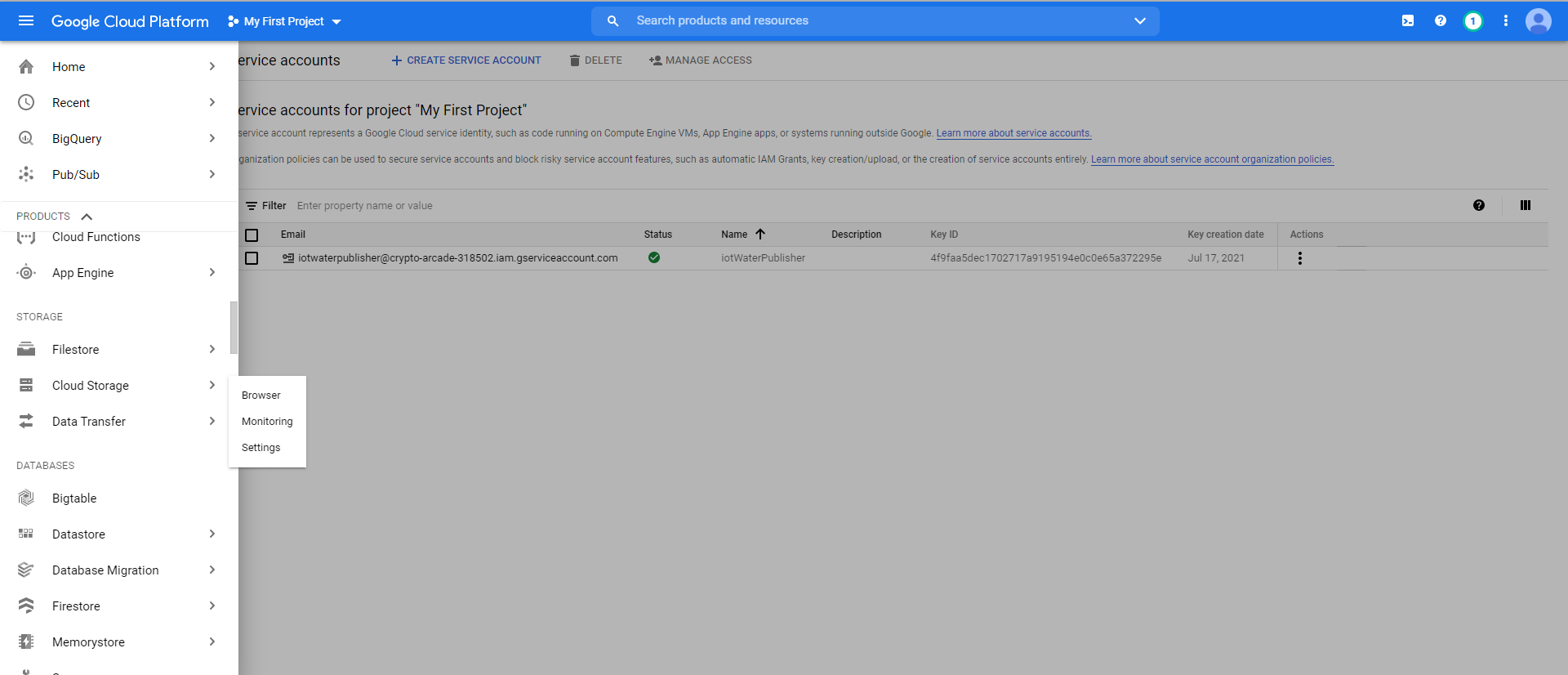
Once here, you will want to click on the **ADD KEY** dropdown menu and select **Create New Key**. This will open a pop-up menu that gives two options **JSON** or **P12**, you will want to select **JSON** and click **Create**. You will notice that downloads a .json file to your computer put this file in a safe place and make note of where it is stored.



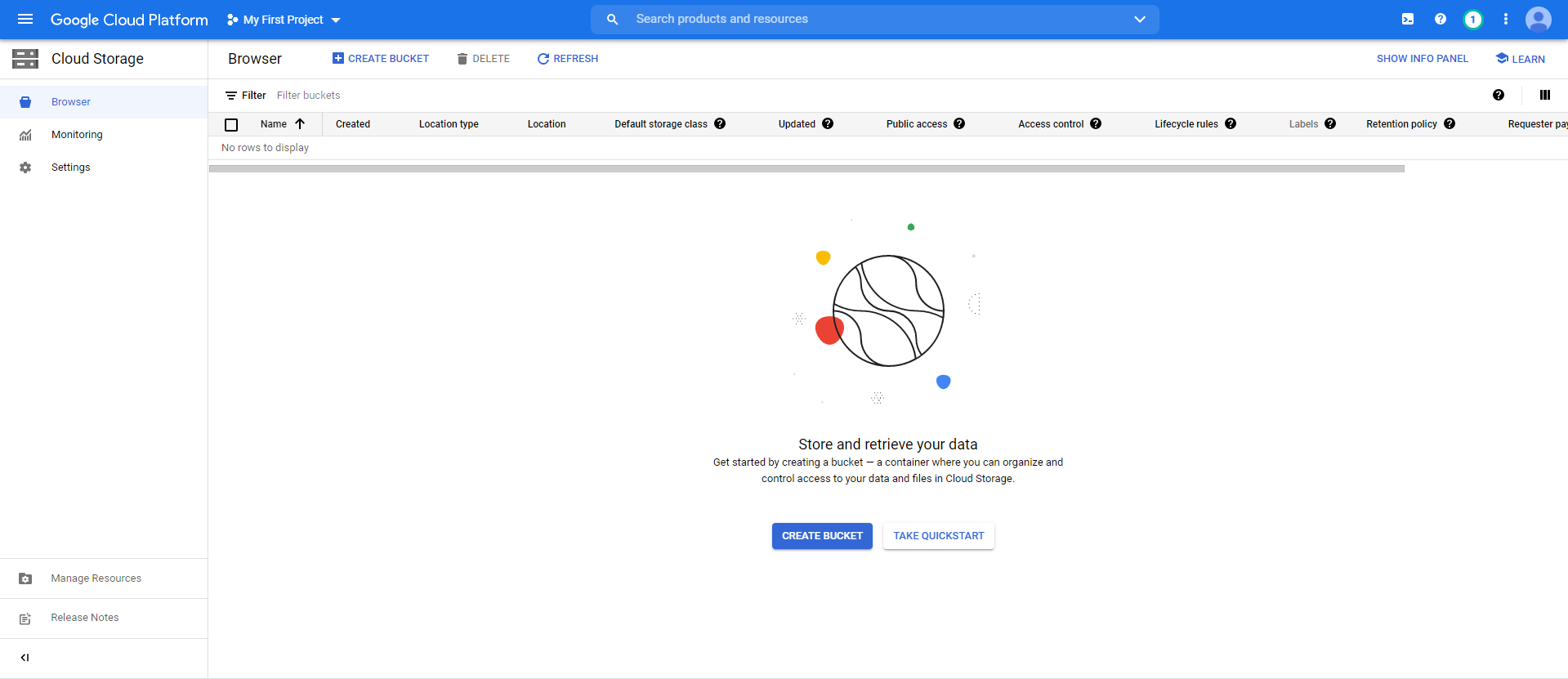
Once this is done, navigate back to the main page of **Service Accounts**, this can be accomplished by click on **Service Accounts**, which is highlighted in the image above, it will be on the right side of the screen in the scroll down menu.



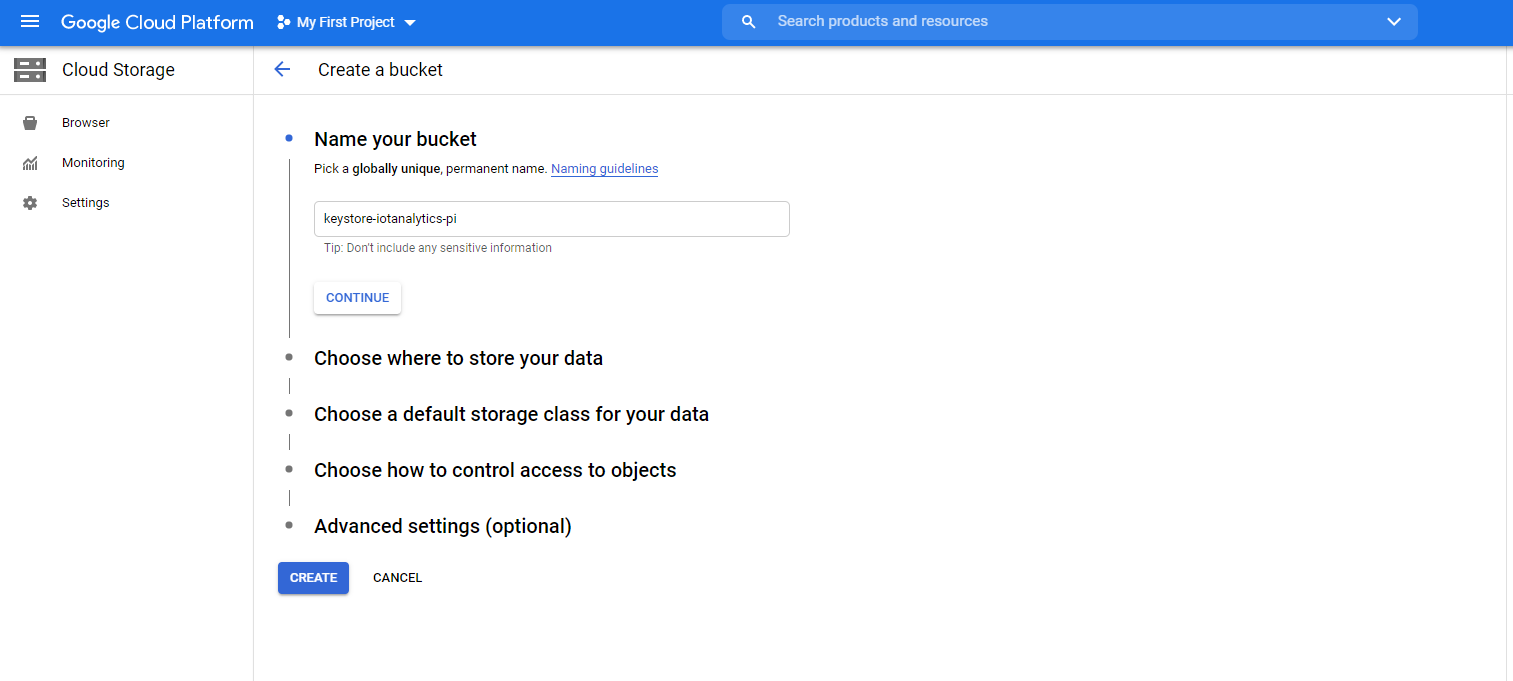
Once here you should be able to see that in the row with your key name it should now also have a **Key ID**. The next part will be done, but only as a backup function. To start you will want to go to the **hamburger menu** make your way **Storage > Cloud Storage > Browser**.



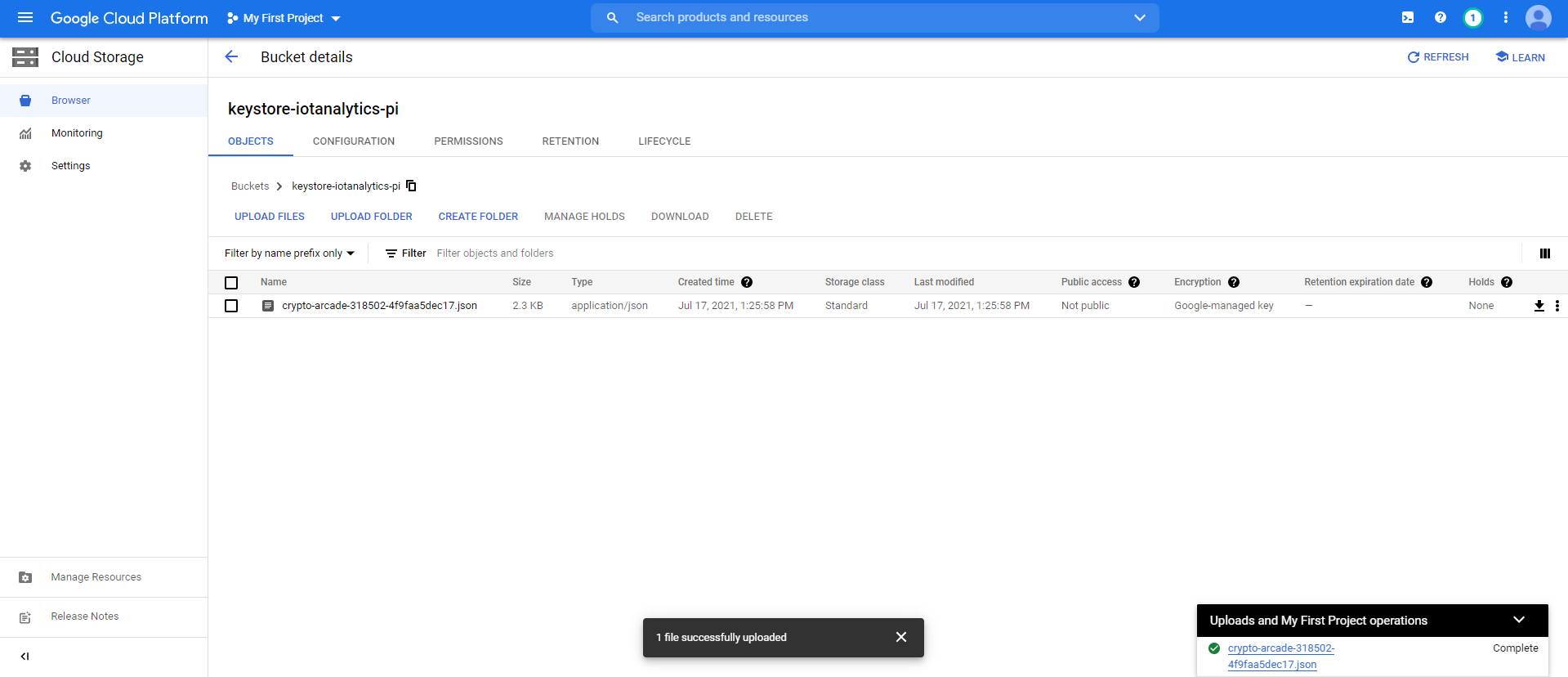
Once here you will press the button **Create Bucket**.



Once you have clicked **Create Bucket**, you will be taken to another screen where you will give the bucket a name. I suggest naming it something like keystore-iotanalytics-pi, but each name has to be unique so you will not be able to use this one. Then click **Create**.



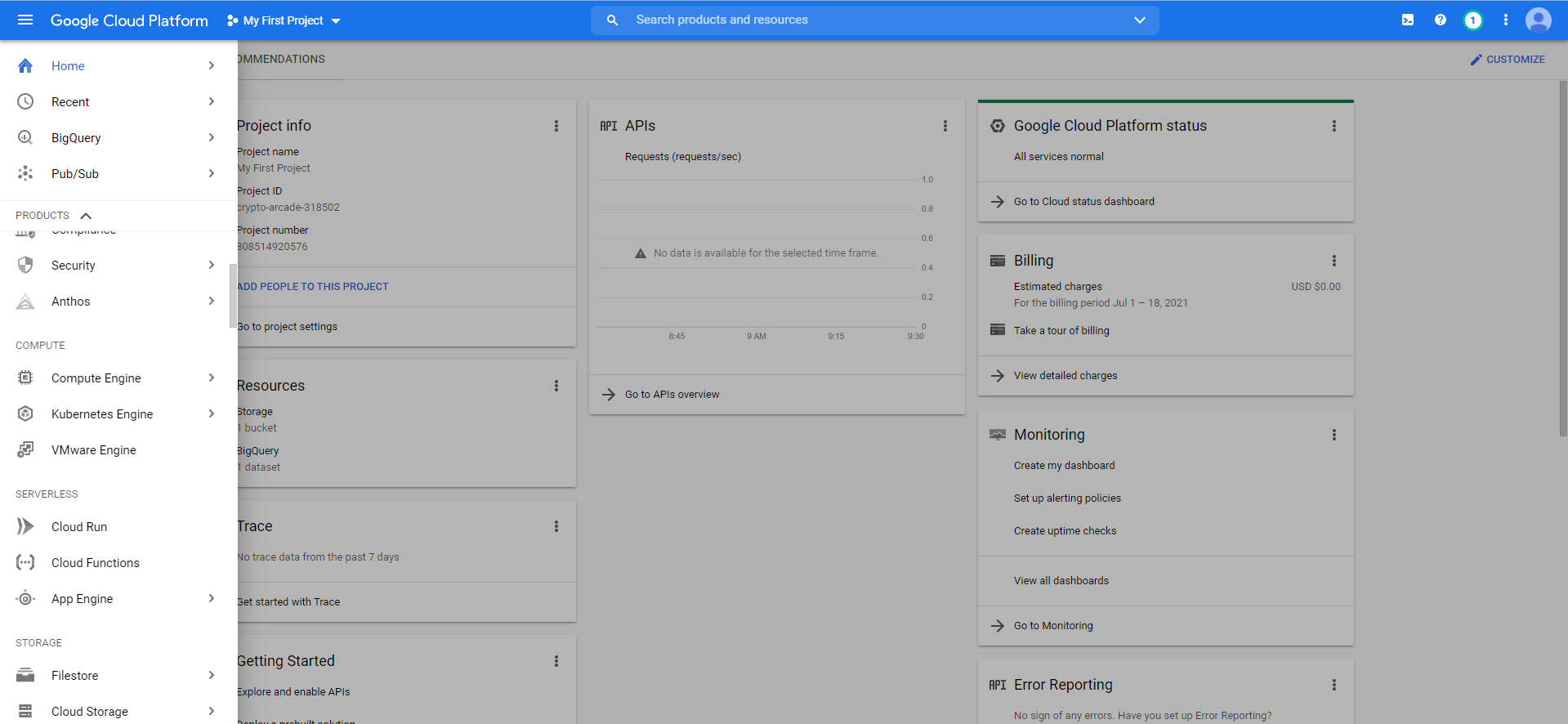
Once you have created your bucket you will be brought back to the homepage of the **Cloud Storage**, then just drag and drop your .json key file into the bucket.



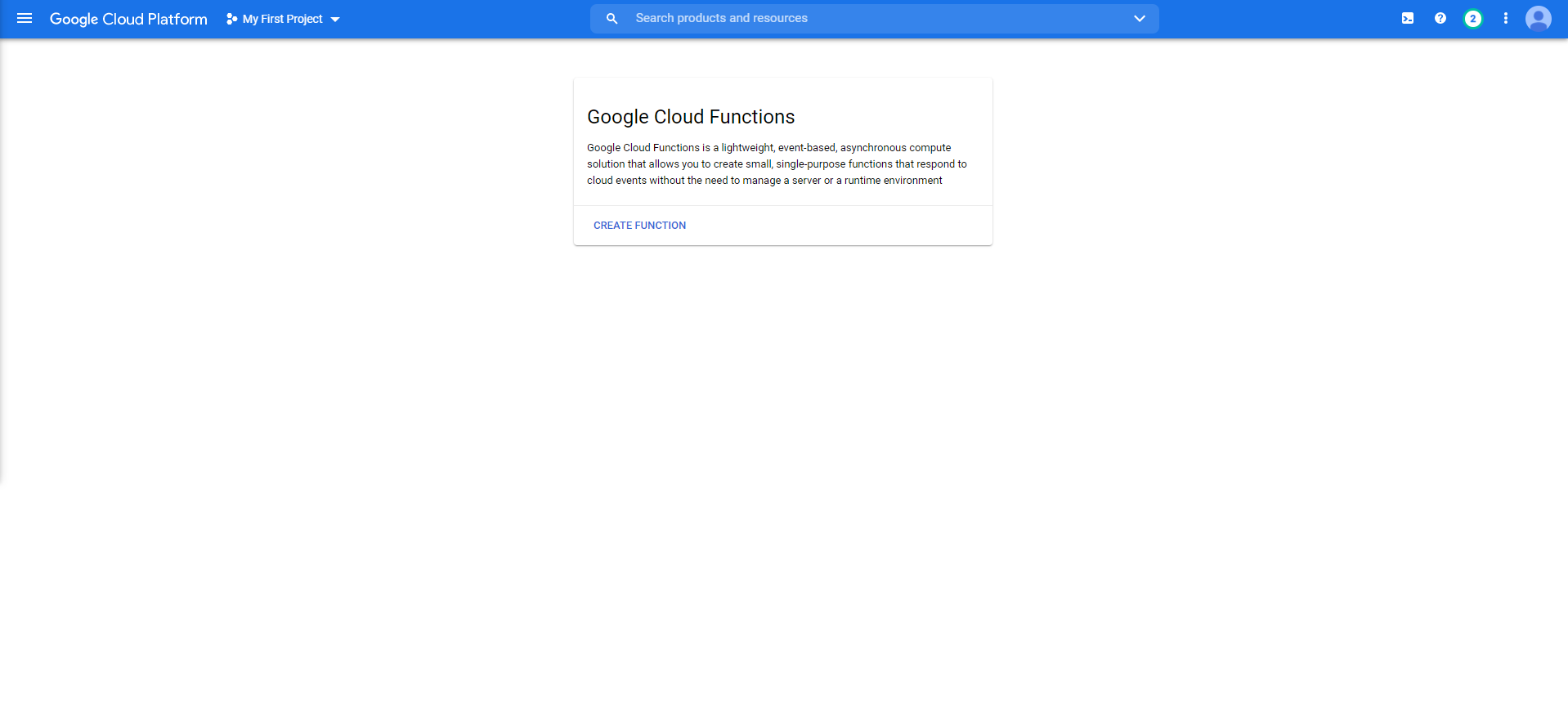
It should look something like this.

**Fifth Step: Creating a Cloud Function**

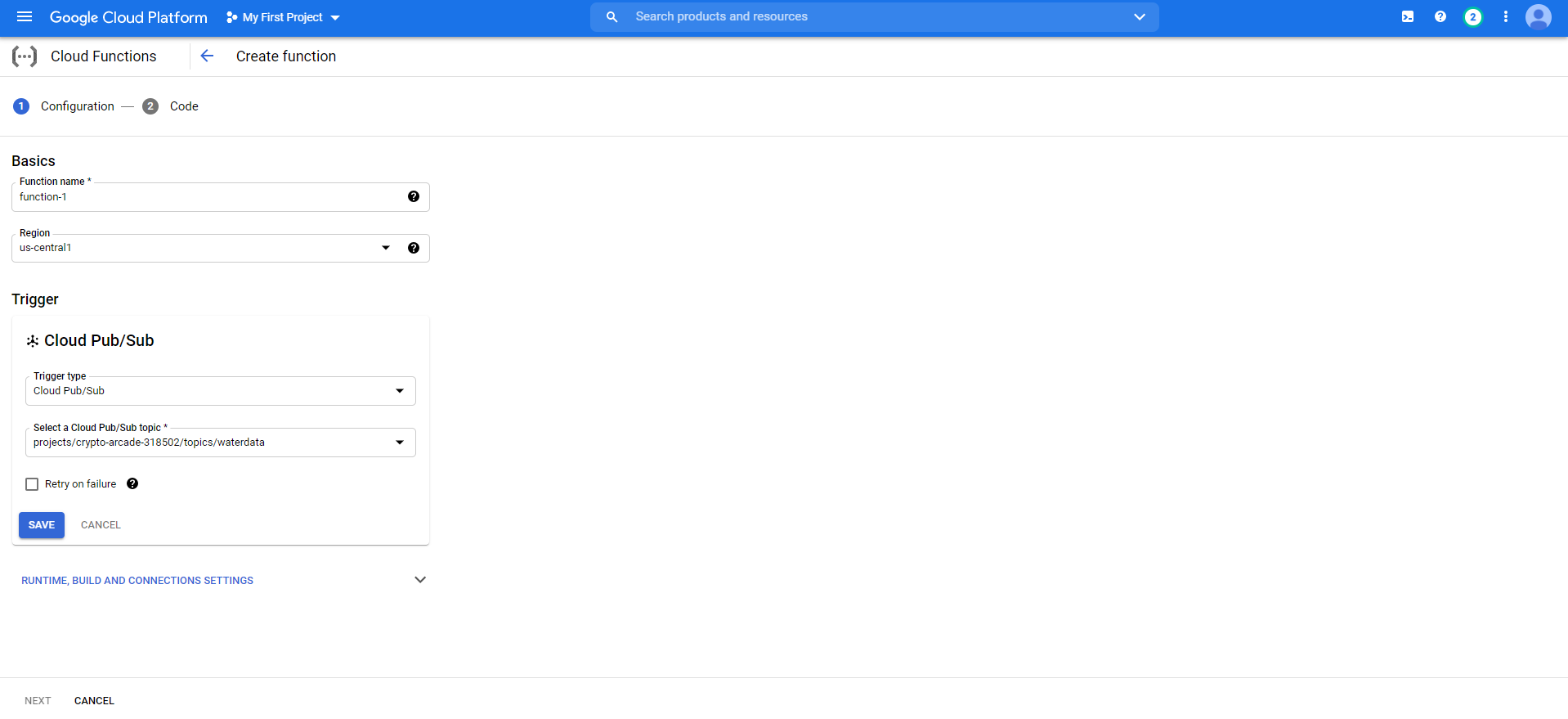
Next step you will need to navigate to **Cloud Functions** to do this navigate to the **hamburger menu** then to **Serverless > Cloud Functions**.



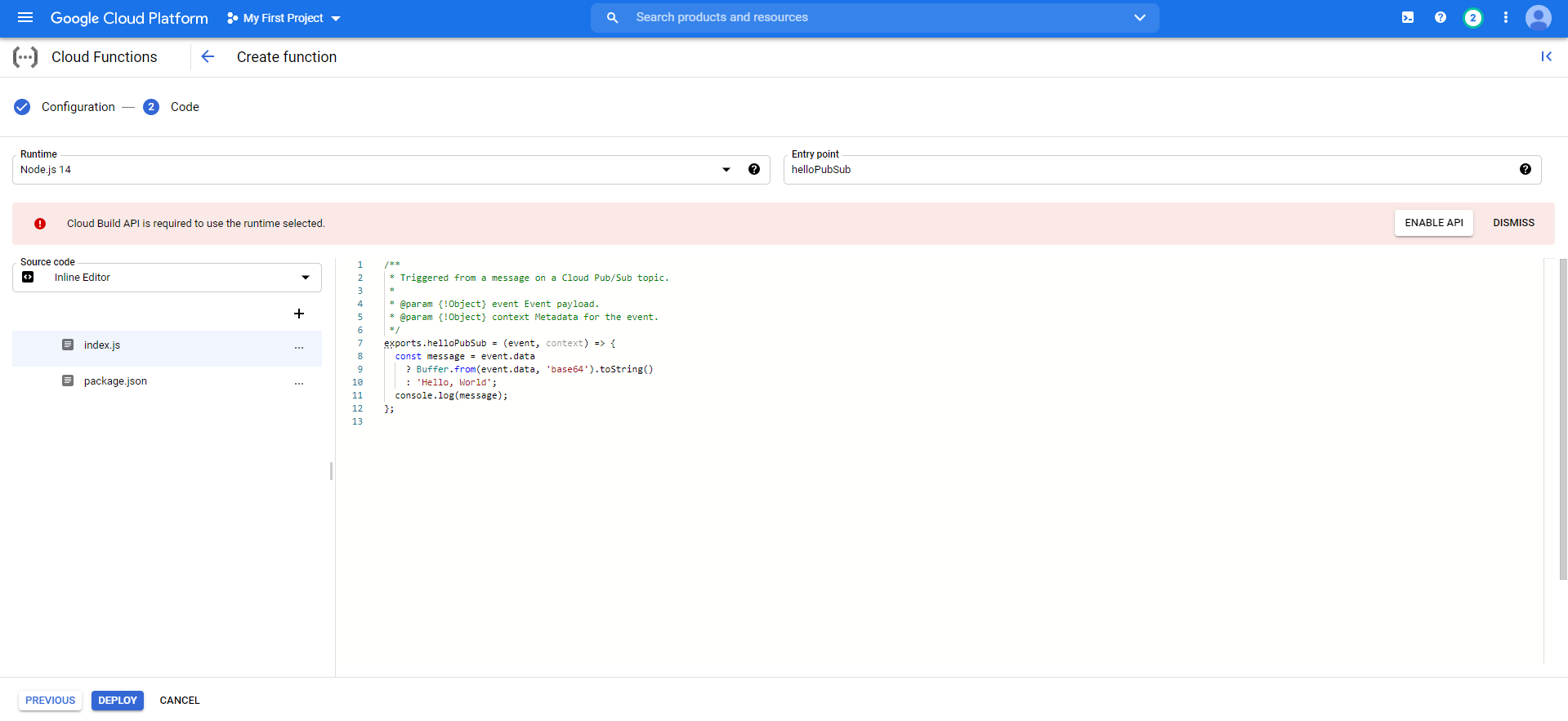
Once here you will be greeted with a screen that simply says **Create Function**, just press this button and that will be bringing you to the next step.



Once to the next screen you will have two fields that need to be filled out, **Trigger Type**, and **Select a Cloud Pub/Sub Topic**. Click on the **Trigger Type** dropdown menu and select **Cloud Pub/Sub**. Once you have selected this it will give you the second option of **Select a Cloud Pub/Sub Topic**. For this then select your topic you created earlier. Once this is done click **Save** then click **Next**.



Once there you will be brought to a new screen where you can input code, but you will notice there is a warning telling you to enable some API. Just click **Enable API**, and the warning will go away. Next find the dropdown menu called **Runtime**, and select **Python 3.7**. Then copy paste the below code or design your own to place into the main.py file. Second click on **Requirements.txt** and under what is already there type: google-cloud-bigquery==2.3.1



**NOTE:** you will need to change tableId to whatever your tableId is, mine will be below but will have a comment to point out that it must be changed.

import base64, json

from google.cloud import bigquery

bigqueryClient = bigquery.Client()

tableId = "crypto-aracade-318502.waterData.waterDataTable" #fixme update to your info

def hello\_pubsub(event, context):

"""Triggered from a message on a Cloud Pub/Sub topic.

Args:

event (dict): Event payload.

context (google.cloud.functions.Context): Metadata for the event.

"""

pubsub\_message = base64.b64decode(event['data']).decode('utf-8')

pubsub\_message = json.loads(pubsub\_message)

sensorid = pubsub\_message['sensorid']

timecollected = pubsub\_message['timecollected']

temperature = pubsub\_message['temperature']

turbidity = pubsub\_message['turbidity']

light = pubsub\_message['light']

zipcode = pubsub\_message['zipcode']

latitude = pubsub\_message['latitude']

longitude = pubsub\_message['longitude']

rows = [

{

"sensorid": sensorid,

"timecollected": timecollected,

"temperature": temperature,

"turbidity": turbidity,

"light": light,

"zipcode": zipcode,

"latitude": latitude,

"longitude": longitude

}

]

errors = bigqueryClient.insert\_rows\_json(tableId, rows)

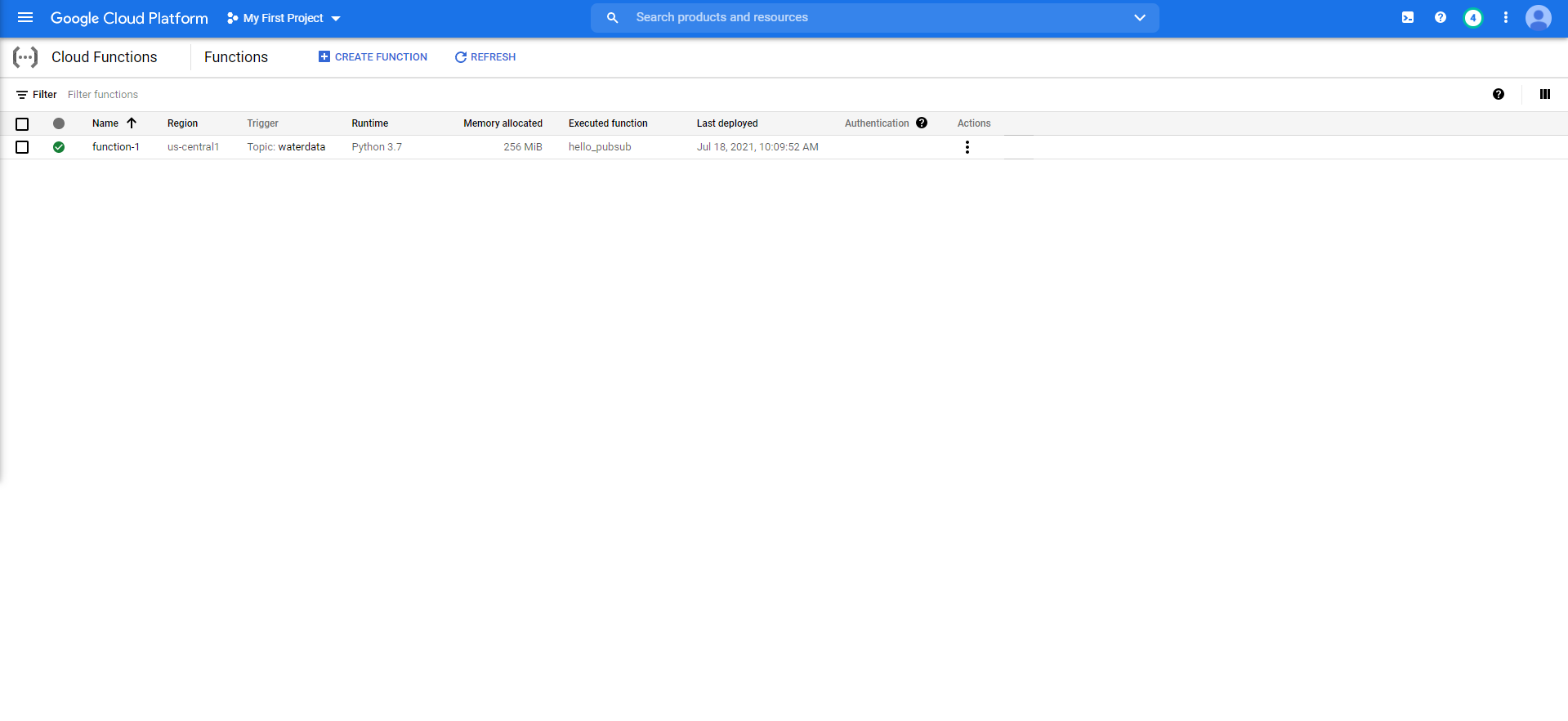
if errors == []:

raise RuntimeError("new rows have been added")

else:

print("errors {}".format(errors))

Once both of these files have been edited simply click **Deploy**.This may take a while you’ll know it worked if the loading icon next to the function name turns green.



**Sixth Step: Configuring your Raspberry Pi to Communicate**

Next you will want to access your Raspberry Pi’s terminal, this can be done by manually logging into the device or by using SSH to access it, however you cannot SSH into it without previously enabling it. First I would recommend installing Vim on your Raspberry Pi, you can do this by typing **sudo apt install vim**. Next we will create a directory to hold our files, I would name it something like iotDataPipeline using the command **mkdir iotDataPipeline**, then change your current directory to that new directory using **cd iotDataPipeline**. Next we will set an environment variable to help install the Google Cloud SDK. To do this we will use the command **export CLOUD\_SDK\_REPO=”cloud-sdk”** then type the command **echo "deb http://packages.cloud.google.com/apt $CLOUD\_SDK\_REPO main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list**. Then type the command **curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -** then type the command **sudo apt-get update && sudo apt-get install google-cloud-sdk**. Next we are gonna install things to use for our Python script. To do this we need to use the command **pip3 install tendo**, then **sudo pip3 install --upgrade google-cloud-pubsub**, then **sudo pip3 install --upgrade oauth2client**. Then we will type the command **gcloud init --console-only**. This will prompt you to follow a link simply paste that into any browser and follow the instructions. Then you will get a verification code to paste in your terminal. You will then be prompted to choose which project you want to use, simply pick the option that we have been working on for this tutorial. Next we will install some software that we will be picking and choosing some bits of code to use. Type the command: **git clone https://github.com/googlecodelabs/iot-data-pipeline**. Once that is done simply change your current directory to **iot-data-pipeline** using the command **cd iot-data-pipeline**. Next we will want to replace their version of **checkWeather.py**, to do this I have included a segment of code to use that will make the program work. **NOTE:** You will need to change a few variables they have been marked with a fixme comment, you will need to adhere to those rules in order for the program to work.

Code:

#! /usr/bin/env python3

import simplejson as json

import os, serial, time

from google.cloud import pubsub

from oauth2client.client import GoogleCredentials

from tendo import singleton

from picamera import PiCamera

import RPi.GPIO as GPIO

from datetime import datetime, timedelta

me = singleton.SingleInstance() # will sys.exit(-1) if other instance is running

SEND\_INTERVAL = 60#seconds

credentials = GoogleCredentials.get\_application\_default()

project = "" #enter in your own project id

topic = "waterdata"#should be correct if you have followed along exactly otherwise #change to topic you are publishing to

sensorID = "s-Googleplex"

sensorZipCode = "49401"

sensorLat = "42.9675"

sensorLong = "85.9509"

ser = serial.Serial('/dev/ttyACM0', 9600) #for turbidity/arduino sensors

camera = PiCamera()

futures = dict()

GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.IN)

def sensor():

for i in os.listdir('/sys/bus/w1/devices'):

if i != 'w1\_bus\_master1':

ds18b20 = i

return ds18b20

def capture(): #file path is listed below replace if necessary

timestamp = datetime.now().isoformat()

camera.capture('/home/pi/Desktop/Software/Pictures/%s.jpg' % timestamp)

def read(ds18b20):

location = '/sys/bus/w1/devices/' + ds18b20 + '/w1\_slave'

tfile = open(location)

text = tfile.read()

tfile.close()

secondline = text.split("\n")[1]

temperaturedata = secondline.split(" ")[9]

temperature = float(temperaturedata[2:])

celsius = temperature / 1000

farenheit = (celsius \* 1.8) + 32

return farenheit

def createJSON(senseid, timestamp, temp, turbid, light, zipco, lat, longi):

data = {

'sensorid' : senseid,

'timecollected' : timestamp,

'temperature' : temp,

'turbidity' : turbid,

'light' : light,

'zipcode' : zipco,

'latitude' : lat,

'longitude' : longi

}

json\_str = json.dumps(data)

return json\_str

def internalSystem():

turbid = "Booting up"

if(ser.in\_waiting > 0):

turbid = ser.readline()

serialNum = sensor()

finalTemperature = read(serialNum)

if GPIO.input(17) == 1:

light = "Dark"

else:

light = "Light"

if turbid != "":

finalTurbid = turbid

finalTime = datetime.now().strftime('%y-%m-%d %H:%M:%S')

return (finalTemperature, light, finalTurbid, finalTime)

def formatString(string):

newString = string.replace("\\r\\n", "")

return newString

def main():

publisher = pubsub.PublisherClient()

topicName = publisher.topic\_path(project, topic)

last\_checked = 0

while True:

if time.time() - last\_checked > SEND\_INTERVAL and time.time() < time.time() + SEND\_INTERVAL:

wTemp, wLight, wTurbid, wTime = internalSystem()

s = ", "

waterJSON = createJSON(sensorID, wTime, wTemp, wTurbid, wLight, sensorZipCode, sensorLat, sensorLong)

waterJSON = formatString(waterJSON)

waterJSON = bytes(waterJSON, 'utf-8')

try:

publisher.publish(topicName, waterJSON)

print(waterJSON)

except:

print("Error publishing data")

time.sleep(5)

if \_\_name\_\_ == '\_\_main\_\_':

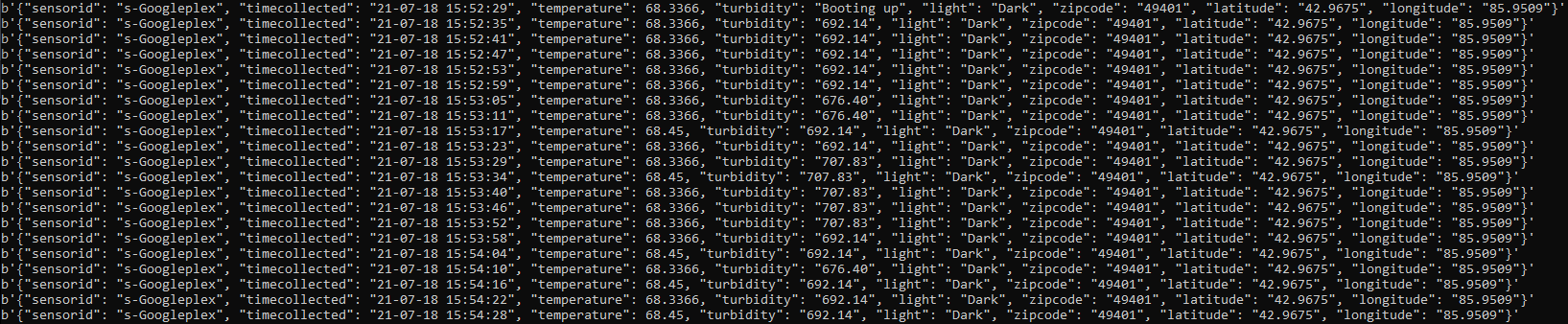
main()

Next we are going to want to edit one of the files that came along with the git that we cloned which is **BigQuery\_Table\_Fields.txt**, run the command **vim BigQuery\_Table\_Fields.txt**, and we are going to change the line to:

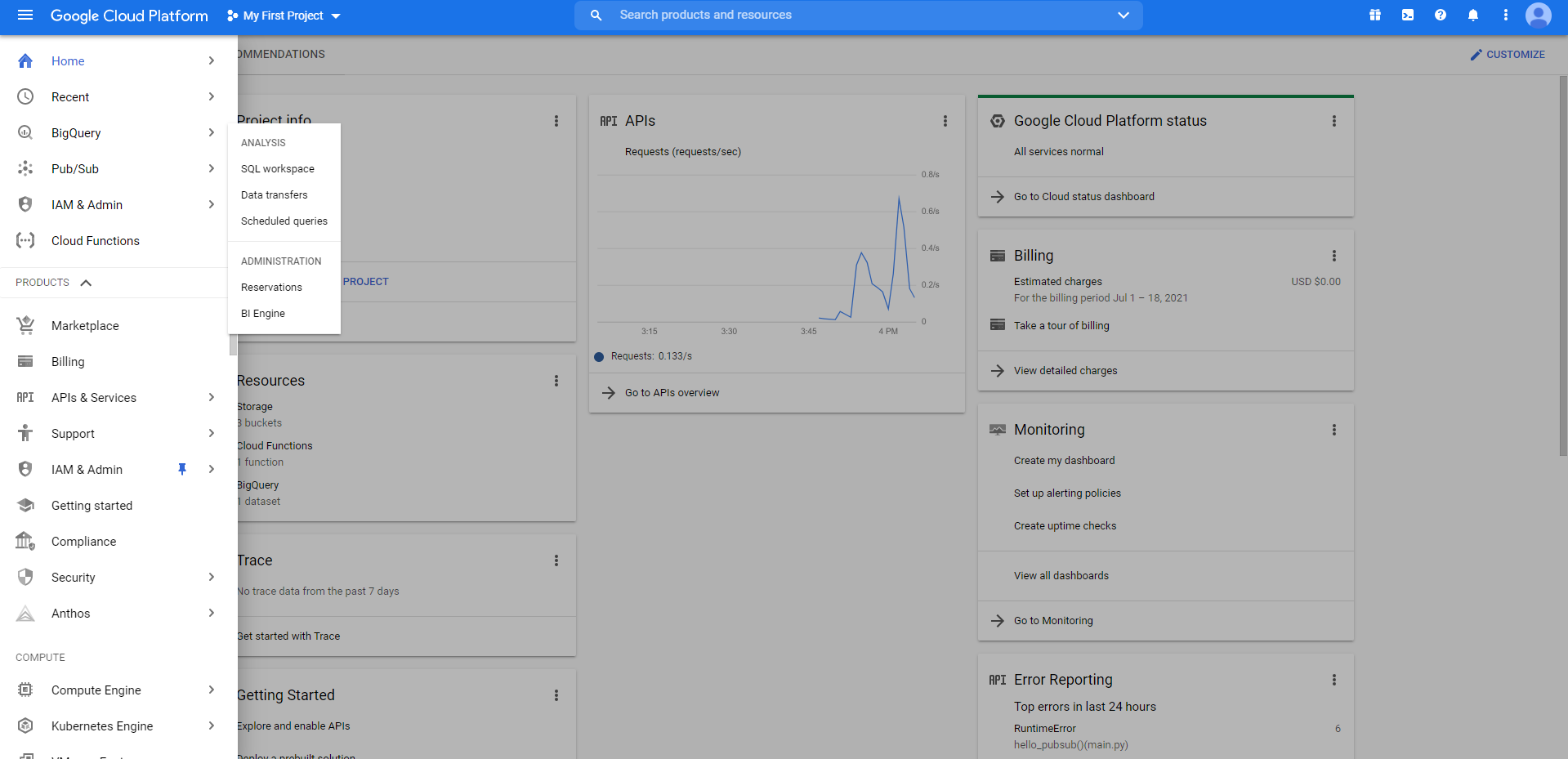
**sensorid:STRING,timecollected:TIMESTAMP,temperature:FLOAT,turbidity:FLOAT,light:STRING,zipcode:INTEGER,latitude:FLOAT,longitude:FLOAT**. Once that is done, we are going to get our security key there are two ways we can do this, one we can **scp** it from your own computer or we can **gsutil** it from our **Google Cloud Bucket**. I will be showing you how to do it using the **Google Cloud Bucket**. To do this we will run this command **gsutil cp gs://nameOfYourBucket/yourSecurityKeyFilename.json .**, don’t forget that dot at the end, as that is what moves the key to your current directory. As you may recall we created our bucket up at the end of the fourth step. The name of your .json file can be found wherever your stored it after it downloaded during the fourth step. For me my command will be **gsutil cp gs://keystore-iotanalytics-pi/crypto-arcade-318502-4f9faa5dec17.json .**. If you see CommandException: No URLs matched, you likely misspelled something in your command. Next we are going to set another environment variable to the name of the .json file so we can use it as our credentials in our program. To do this use the command **export GOOGLE\_APPLICATION\_CREDENTIALS=/home/pi/iot-data-pipeline/yourSecurityKeyFilename.json**, however you will need to change the last part to whatever the name is for me that will be **crypto-arcade-318502-4f9faa5dec17.json**. Now your device should work!

**Seventh Step: Testing Your Output**

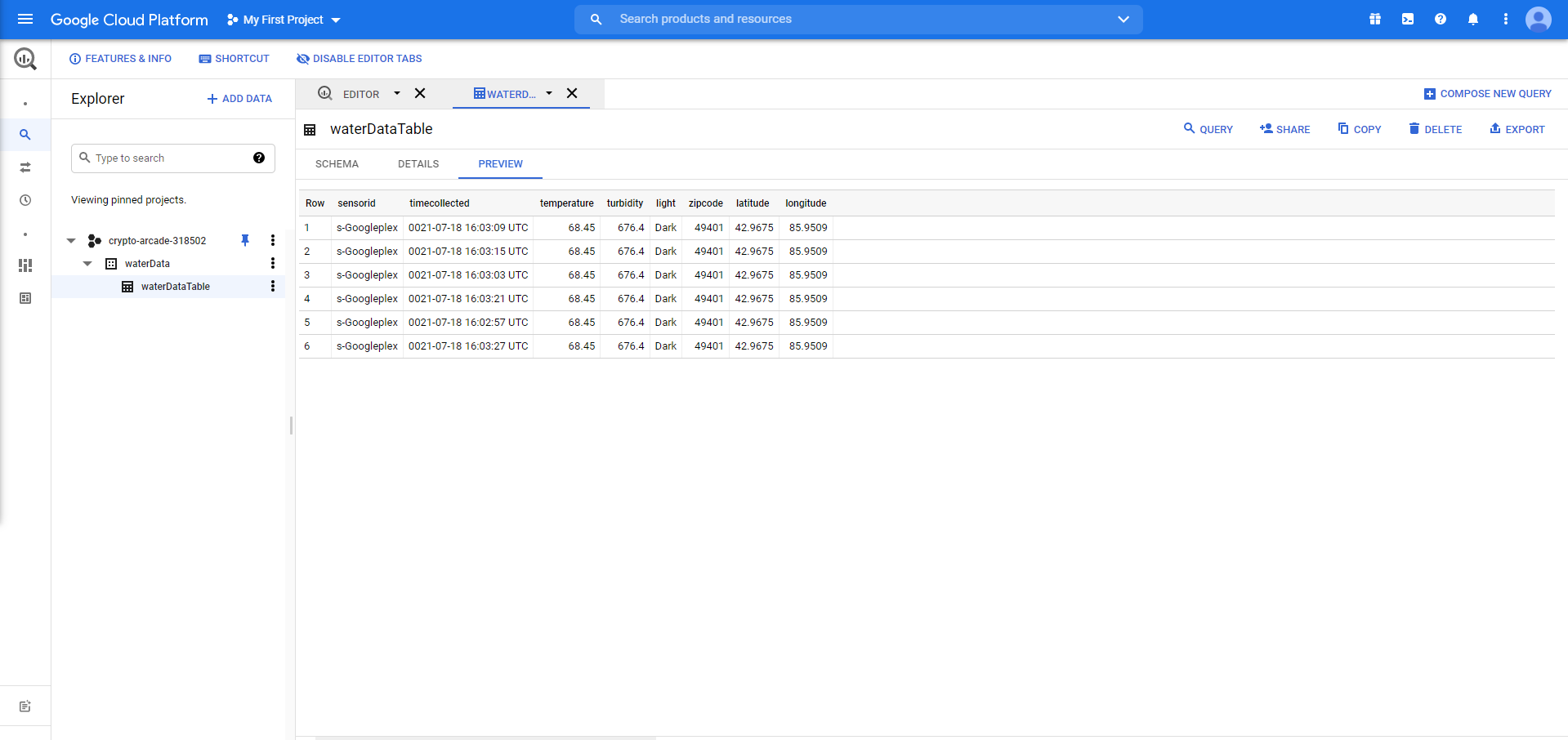
To test your device to make sure it is publishing correctly simply type the command **python3 yourprogramnamehere.py**, by default your program name will probably be called **checkWeather.py**, as this tutorial was adapted from another tutorial aiming to create a weather sensor. Otherwise substitute the second part of the command with whatever you named it. After this is done you should see outputs being printed to your screen, here is an example of them. **Important: You can cancel your program by using SHIFT + C**.



Secondly we are going to want to make sure that our messages are actually being published we are going to check our **BigQuery Table**. To do this simply launch or reopen **Google Cloud**, then navigate to the **hamburger icon** and scroll to **BigQuery**.



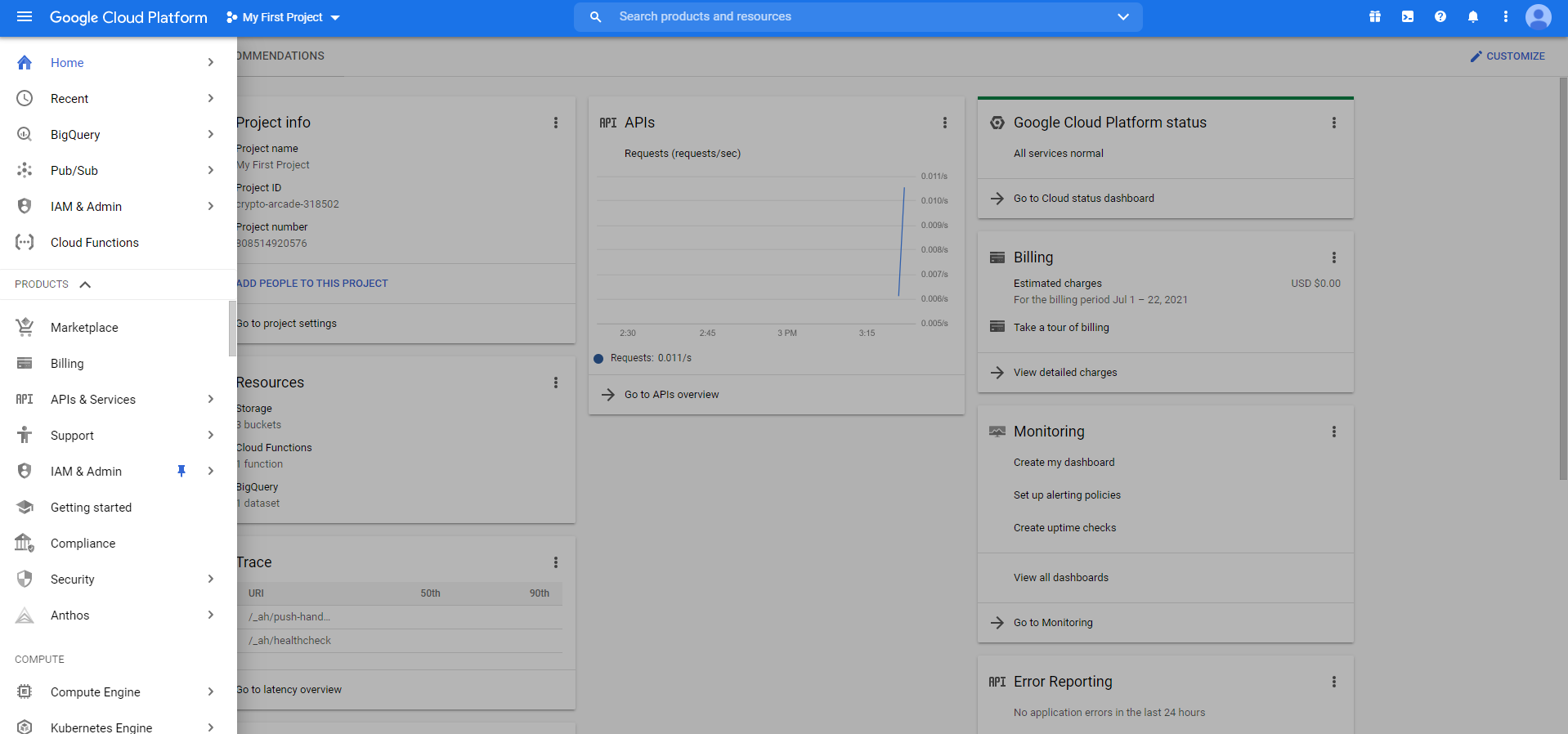
Once you have clicked on **BigQuery** you will be brought to the homepage simply click the dropdown menu on your **Project Name** and then your **Dataset Name** until you find your **Table Name**. Then click on the **Table Name** and navigate to the **Preview** button which is under the name of the table. Something like this.



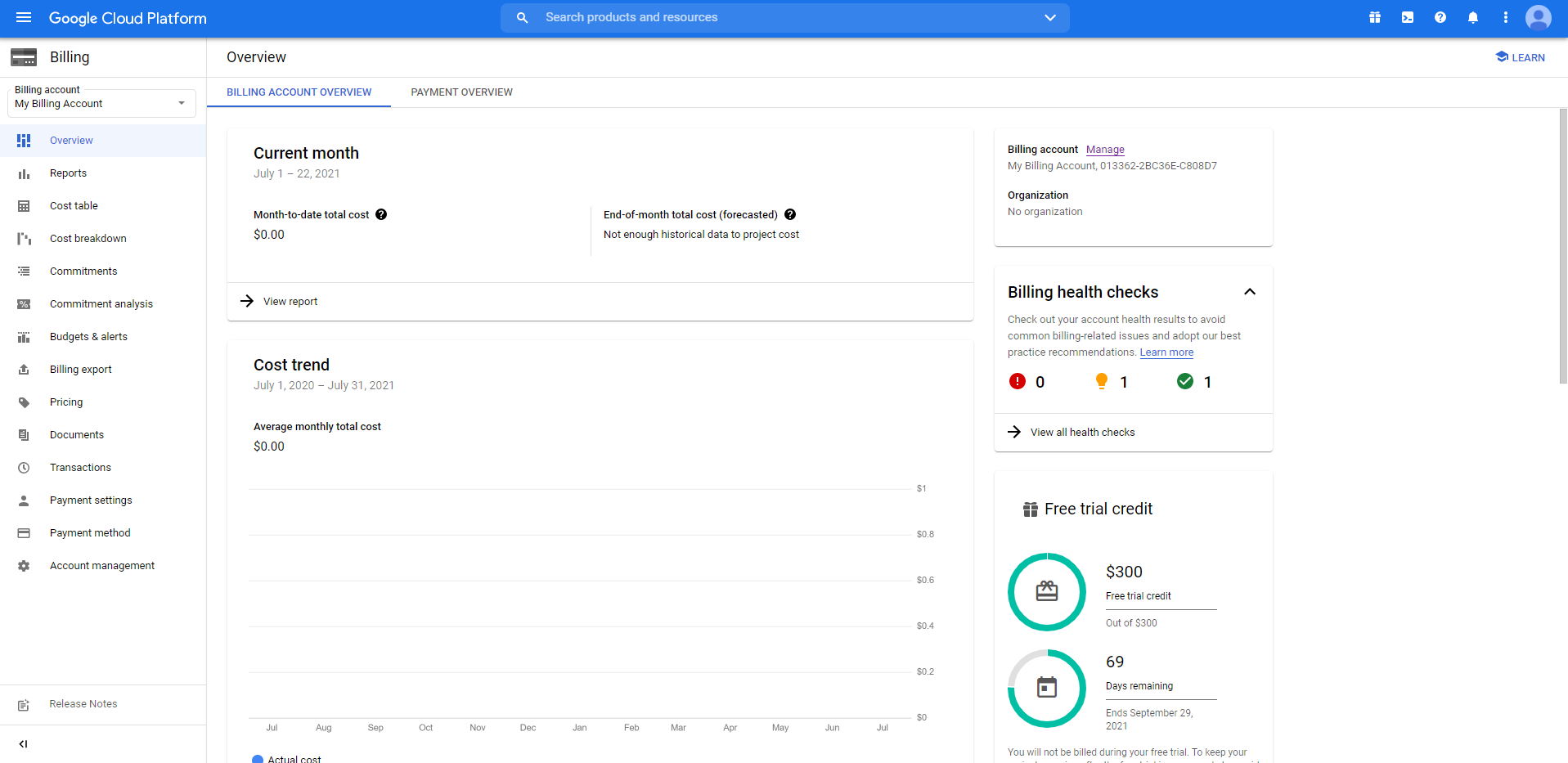
As you can see my table has six different entries, this is what yours should look like too. If you run into any problems along the way make sure to go back and review the tutorial completely.

**Eighth Step: Gracefully Shutting Down**

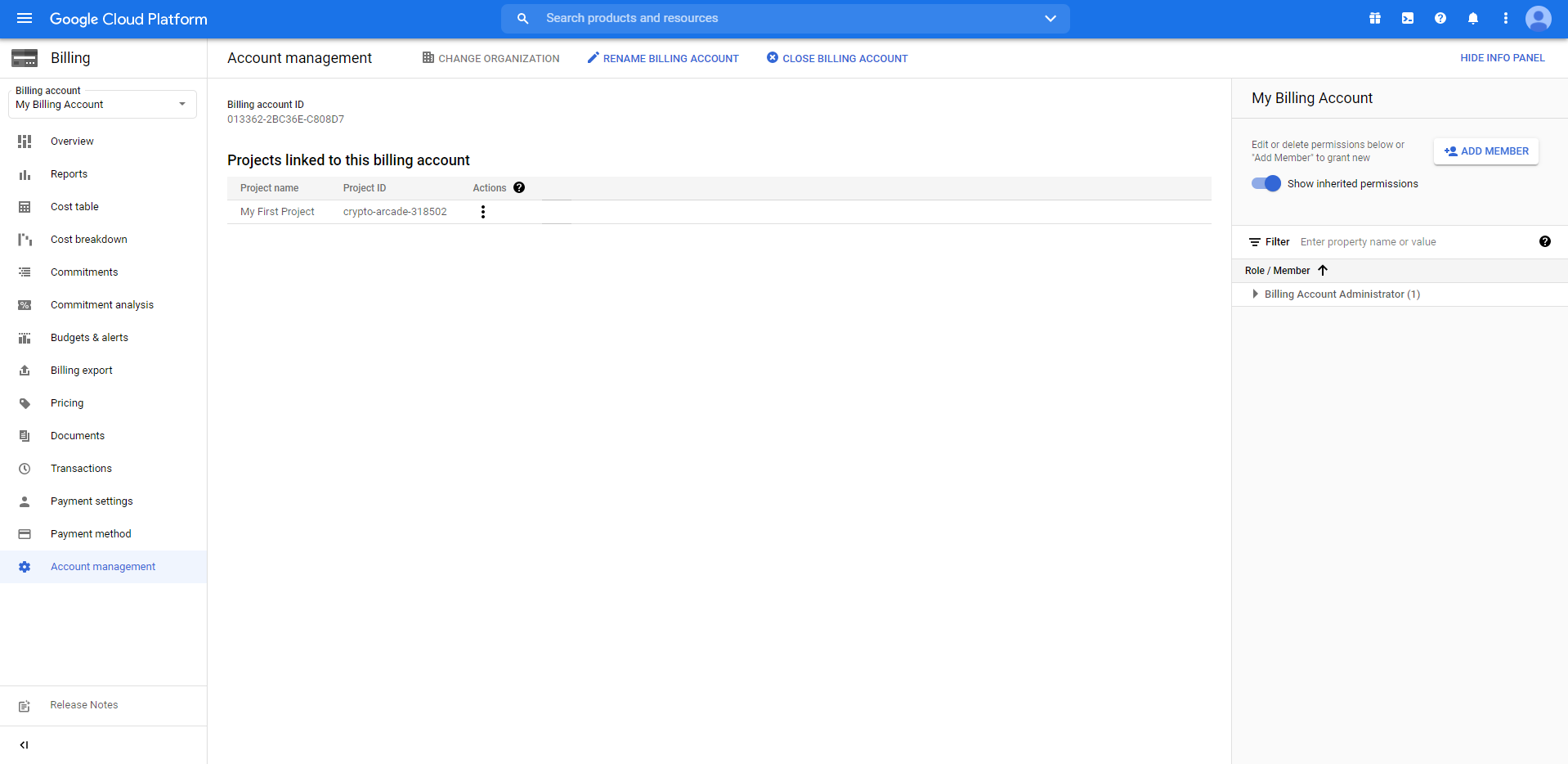
Once you have completed the tutorial and are ready to shut the service off to prevent any sort of billing charges to your account we will be deleting your account. First navigate back to the home dashboard page. Click the **hamburger icon**, go down to **Products > Billing**



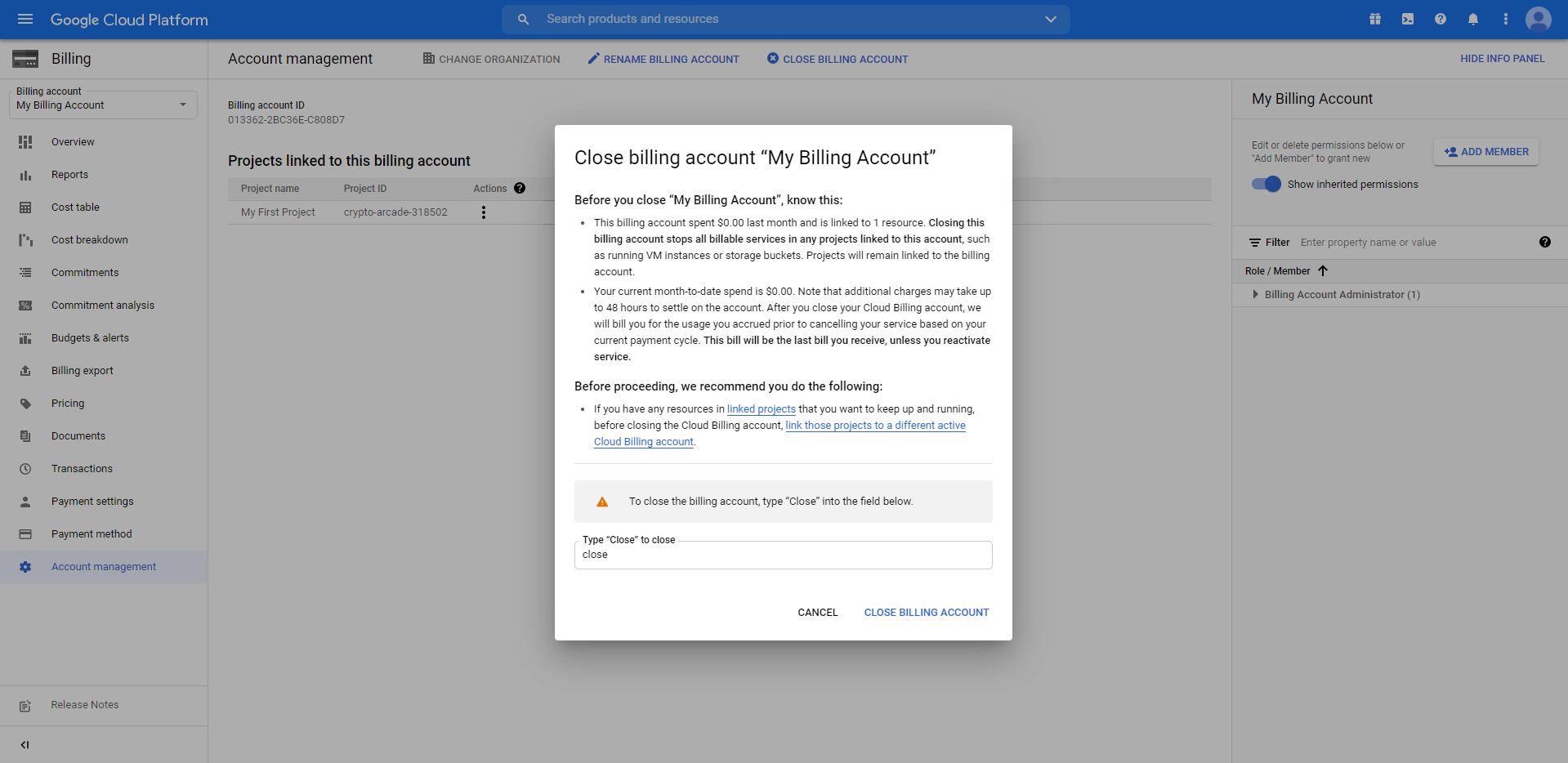
Once you have clicked the **Billing** button it will bring you to this screen.



Make your way down the menu on the far left to **Account Management**, and click on it.



At the very top of the screen you will see a button that says **Close Billing Account**, simply click that and type **Close** in the prompt as seen below.



Once you have answered the prompt simply click **Close Billing Account**, located on the bottom right.