Using Cloud Logging with IoT Core Devices

GSP258

Google Cloud Self-Paced Labs

Overview

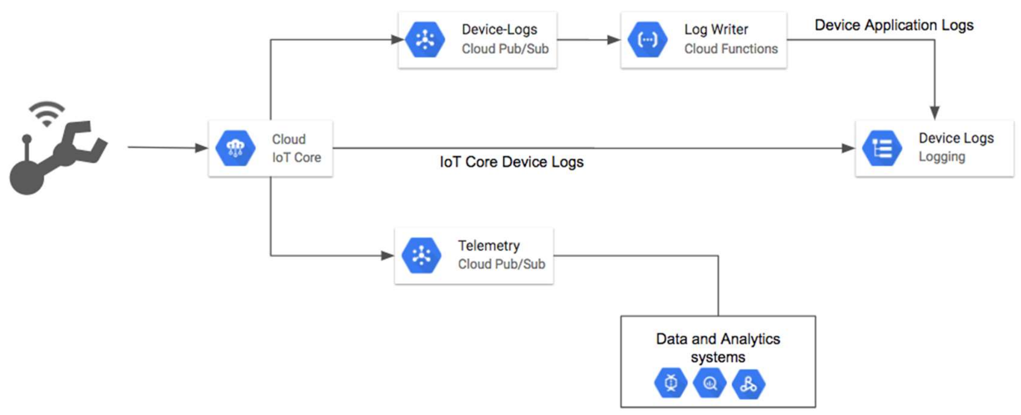
In this hands-on lab you will learn how to configure Cloud Functions to send IoT Core device application logs to Cloud Logging.

What you'll learn

- How to send application logs from simulated IoT devices to IoT Core
- How to configure a Cloud Function to write logs Cloud Logging
- How to view device logs in Cloud Logging

Introduction

Cloud IoT Core is a fully managed service that allows you to easily and securely connect, manage, and ingest data from millions of globally dispersed devices. Cloud IoT Core, in combination with other services on Google Cloud, provides a complete solution for collecting, processing, analyzing, and visualizing IoT data in real time to support improved operational efficiency.
Cloud Functions is a lightweight compute solution for developers to create single-purpose, stand-alone functions that respond to Cloud events without the need to manage a server or runtime environment.
Cloud Logging allows you to store, search, analyze, monitor, and alert on log data and events from Google Cloud and Amazon Web Services (AWS). It is a fully managed service that performs at scale and can ingest application and system log data from thousands of VMs.
In this lab you will simulate an IOT device using command line that sends device application logs to Cloud IOT Core and is eventually sent to Cloud Logging for further analysis.



Setup and Requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

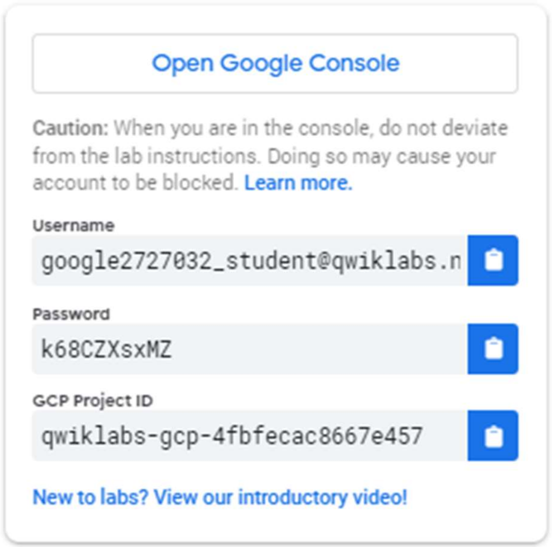
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.
  Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

Your lab will take a few minutes to build a environment with the required resources. When you see the green Lab Running light you can begin your lab. Feel free to log into the Console wile this is happening.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

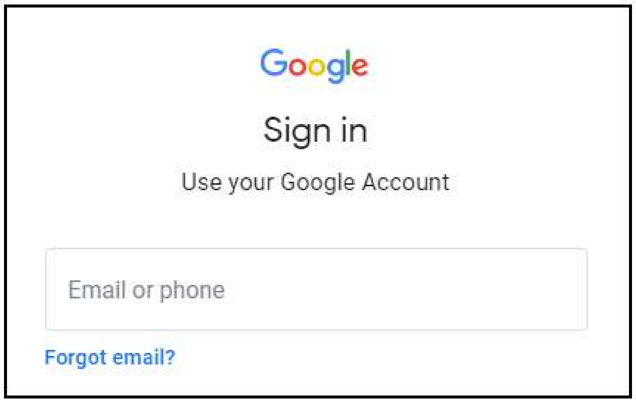
Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

New to labs? View our introductory video!

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Google

Sign in

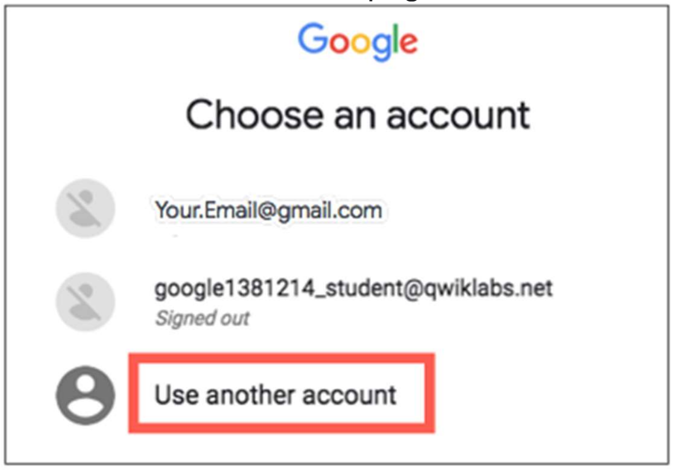
Use your Google Account

Email or phone

Forgot email?

Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**.



Google

Choose an account

Your.Email@gmail.com

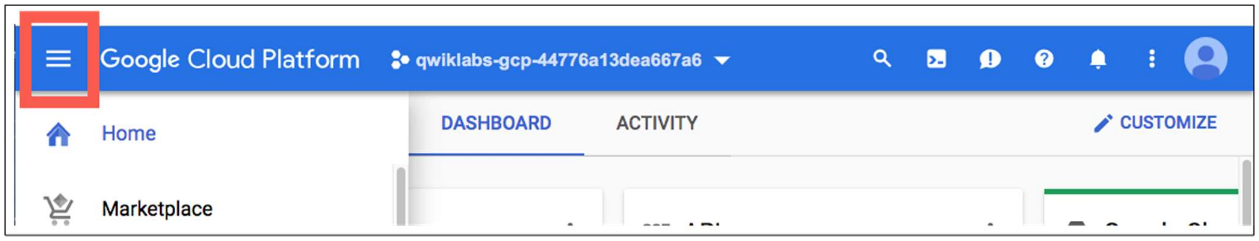
google1381214_student@qwiklabs.net
Signed out

Use another account

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).
4. Click through the subsequent pages:
    - Accept the terms and conditions.
    - Do not add recovery options or two-factor authentication (because this is a temporary account).
    - Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-
left.



Set up the environment

A compute VM called **labvm** has been created as part of the lab setup. You will be SSHing into this VM and configuring it to function as an IoT device simulator as well as execute the relevant commands to interact with the different services required in this lab.

SSH into labvm

1. From the **Navigation menu**, select **Compute Engine**.

2. In the VM instances tab, click the **SSH** button next to the **labvm** instance.

You're now remotely connected to the **labvm** via a SSH connection.

Install latest version of gcloud SDK

You will need to use gcloud SDK to interact with the different services used in this lab. The following steps will take you through the process of ensuring the latest gcloud SDK is installed.

1. In your SSH session, enter the following command to remove the default Google Cloud SDK installation:

```
sudo apt-get remove google-cloud-sdk -y
```

2. Install the latest version of the Google Cloud SDK:

```
curl https://sdk.cloud.google.com | bash
```

3. At the installation director prompt, press **ENTER** to accept all defaults.

4. And answer "Y" to continue (twice)

5. When asked to enter a path to an rc file to update, press **Enter**.

6. End your SSH session.

```
exit
```

Configure student credentials

You will now configure the `gcloud` CLI tool with the student credentials that were supplied to you as part of the lab.

1. Start another SSH session on your VM instance.

2. Initialize the gcloud SDK

```
gcloud init
```

3. When you are asked whether to authenticate with an @developer.gserviceaccount.com account or to log in with a new account, enter the number to **log in with a new account**.

4. Type in "Y" to confirm that you want to continue.

5. Click on the URL in the output to open a new browser window that displays a verification code.

6. Choose the lab credentials you signed into this lab with and click **Allow**.

7. Copy the verification code and paste it as the response to "Enter verification code:" and press **Enter**.

8. In response to "Pick cloud project to use", enter the number for your Google Cloud project the lab created for you.

9. Now enter this command to make sure that the components of the SDK are up to date:

```
gcloud components update
```

10. Now install the beta components:

```
gcloud components install beta
```

11. Enter "Y" confirming you want to continue.

Create PubSub topic

Next you will create a PubSub topic. The IoT Core service will publish messages (device logs) that it receives into this topic.

First you'll configure a few environment variables, then create the PubSub topic.

1. Set the name of the Cloud PubSub topic using an environment variable:

```
export TOPIC_NAME=device-logs
```

2. Set the following values specific to your lab environment

Replace `region_name` with one of the following regions currently supporting IoT registry:

- us-central1
- europe-west1
- asia-east1

```
export MY_REGION=region_name
```

For example:

```
export MY_REGION=us-central1
```

In the following command, replace [Google Cloud_PROJECT_ID] with your lab's Project ID:

```
export PROJECT_ID=[GCP_PROJECT_ID]
```

3. Create a PubSub topic you will use for device logs:

```
gcloud pubsub topics create $TOPIC_NAME
```

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Create a Cloud Pub/Sub Topic.

Check my progress

4. Provide the IoT service account with the Pub/Sub Publisher role on the created topic:

```
gcloud beta pubsub topics add-iam-policy-binding $TOPIC_NAME --
member=serviceAccount:cloud-iot@system.gserviceaccount.com --
role=roles/pubsub.publisher
```

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Create a Cloud Function

You will now create a Cloud Function that reads a message from IoT Core, extracts the log payload and device information, then writes a structured log entry to Cloud Logging.

This function is configured to be invoked on every message published to the PubSub topic.

1. Create the `functions` directory and change into that directory:

```
mkdir functions
cd functions
```

2. Create a file called `index.js` using the following code:

```
nano index.js
```

3. Enter the following code into the `nano` editor:

```
const loggingClient = require('@google-cloud/logging');

// start cloud function

exports.deviceLog = (data, context) => {

    const projectId=data.attributes.projectId;
    const logging = new loggingClient({
        projectId: process.env.projectId,
    });
    const log = logging.log('device-logs');
    const metadata = {
      // Set the Cloud IoT Device you are writing a log for
      // you extract the required device info from the PubSub attributes
      resource: {
        type: 'cloudiot_device',
        labels: {
          project_id: data.attributes.projectId,
          device_num_id: data.attributes.deviceNumId,
          device_registry_id: data.attributes.deviceRegistryId,
          location: data.attributes.deviceRegistrylocation,
        }
      },
      labels: {
        // note device_id is not part of the monitored resource, but you can
        // include it as another log label
        device_id: data.attributes.deviceId,
      }
    };
    const logData = JSON.parse(Buffer.from(data.data, 'base64').toString());

    const temperature = 'temperature:'+logData.temperature;
    const deviceId = 'deviceId:'+logData.device;

    // write the log entry to Stackdriver Logging
    const entry = log.entry(metadata, deviceId+','+temperature);
   log
  .write(entry)
  .then(() => {
    console.log(`Logged: ${temperature}`);
  })
  .catch(err => {
    console.error('ERROR:', err);
  });
 };
```

4. **Ctrl+ o** to save and then **Ctrl+x** to exit the editor.

5. Create the dependency packages file to instruct CloudFunction about the dependency on the google-cloud/logging package:

```
nano package.json
```

6. Copy the following into the `nano` editor:

```
{
  "dependencies": {
    "@google-cloud/logging": "^2.0.0"
  }
}
```

7. **Ctrl+ o** to save and then **Ctrl+x** to exit the editor.

8. Deploy the function `deviceLog` onto CloudFunction:

```
gcloud functions deploy deviceLog --allow-unauthenticated --runtime nodejs8 --trigger-
resource $TOPIC_NAME --trigger-event google.pubsub.topic.publish
```

The function can take 2 minutes to deploy.

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Clone IoT simulator code

1. Clone the below repository that contains the simulator code needed for this lab:

```
cd $HOME
git clone http://github.com/GoogleCloudPlatform/training-data-analyst
```

Execute the following command to download and update the packages list.

```
sudo apt-get update
```

Python virtual environments are used to isolate package installation from the system.

```
sudo apt-get install virtualenv
```

If prompted [Y/n], press `Y` and then `Enter`.

```
virtualenv -p python3 venv
```

Activate the virtual environment.

```
source venv/bin/activate
```

2. Use the `pip` command to add needed Python components which will be used to run the scripts in this lab:

```
pip install pyjwt paho-mqtt cryptography
```

Create IoT registry and devices

To register devices, you must create a registry for the devices. The registry is a point of control for devices.

1. Navigate to the lab code directory:

```
cd $HOME/training-data-analyst/quests/iotlab/
```

2. Create the IoT registry:

```
export REGISTRY_NAME=iot-stackdriver

gcloud iot registries create $REGISTRY_NAME \
    --project=$PROJECT_ID \
    --region=$MY_REGION \
    --event-notification-config=topic=projects/$PROJECT_ID/topics/$TOPIC_NAME
```

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Create a Registry for IoT Devices.

Check my progress

3. To allow IoT devices to connect securely to Cloud IoT Core, you must create a cryptographic key pair. Use the `openssl` command to create an RSA cryptographic key pair, and then write it to a file called `rsa_private.pem`:

```
openssl req -x509 -newkey rsa:2048 -keyout rsa_private.pem \
    -nodes -out rsa_cert.pem -subj "/CN=unused"
```

4. Now add simulated devices to the registry:

```
export DEVICE_NAME_1=iot-device-1
export DEVICE_NAME_2=iot-device-2
```

5. Create the first simulated device:

```
gcloud iot devices create $DEVICE_NAME_1 \
  --project=$PROJECT_ID \
  --region=$MY_REGION \
  --registry=$REGISTRY_NAME \
  --public-key path=rsa_cert.pem,type=rs256
```

6. Create the second simulated device:

```
gcloud iot devices create $DEVICE_NAME_2 \
  --project=$PROJECT_ID \
  --region=$MY_REGION \
  --registry=$REGISTRY_NAME \
  --public-key path=rsa_cert.pem,type=rs256
```

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Add Simulated Devices to the Registry.

Check my progress

Run simulated devices

You will run the simulator to publish messages to the IOT core in this section. You will run it twice with the two different device names to simulate separate devices interacting with IoT Core.

1. Enter these commands to download the CA root certificates from `pki.google.com` to the appropriate directory:

```
cd $HOME/training-data-analyst/quests/iotlab/

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

2. Run the first simulated device:

```
python cloudiot_mqtt_example_json.py \
  --project_id=$PROJECT_ID \
  --registry_id=$REGISTRY_NAME \
  --device_id=$DEVICE_NAME_1 \
  --private_key_file=rsa_private.pem \
  --cloud_region=$MY_REGION \
  --num_messages=400 \
  --message_type=event \
  --algorithm=RS256  > devicelog1.txt 2>&1 &
```

3. Run the second simulated device:

```
python cloudiot_mqtt_example_json.py \
  --project_id=$PROJECT_ID \
  --registry_id=$REGISTRY_NAME \
  --device_id=$DEVICE_NAME_2 \
  --private_key_file=rsa_private.pem \
  --cloud_region=$MY_REGION \
  --num_messages=400 \
  --message_type=event \
  --algorithm=RS256  > devicelog2.txt 2>&1 &
```

NOTE: Wait for few seconds for the logs to get generated in the file.

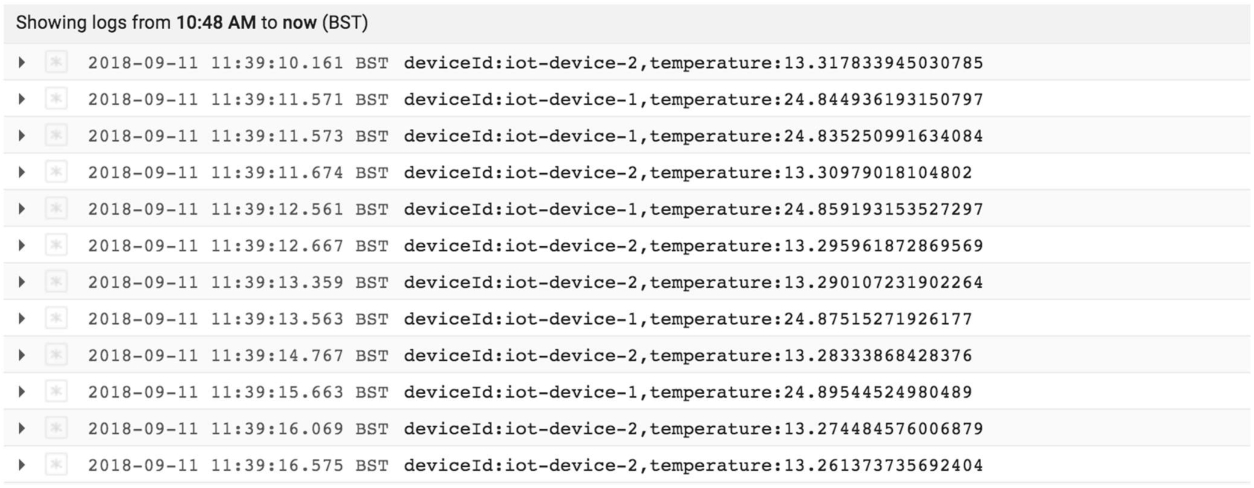
Explore device logs on Stackdriver

Using Cloud Logging, you can inspect both the system as well as the application (device) logs being written by the deployed Cloud Function.

1. In the **Navigation menu**, in the Stackdriver section select **Logging**.

2. In the filter drop-down, select **Cloud IoT Device**.

You should be able to see the logs related to different temperature readings being reported by the simulated IoT devices.



Showing logs from **10:48 AM** to **now** (BST)

```
2018-09-11 11:39:10.161 BST deviceId:iot-device-2,temperature:13.317833945030785
2018-09-11 11:39:11.571 BST deviceId:iot-device-1,temperature:24.844936193150797
2018-09-11 11:39:11.573 BST deviceId:iot-device-1,temperature:24.835250991634084
2018-09-11 11:39:11.674 BST deviceId:iot-device-2,temperature:13.30979018104802
2018-09-11 11:39:12.561 BST deviceId:iot-device-1,temperature:24.859193153527297
2018-09-11 11:39:12.667 BST deviceId:iot-device-2,temperature:13.295961872869569
2018-09-11 11:39:13.359 BST deviceId:iot-device-2,temperature:13.290107231902264
2018-09-11 11:39:13.563 BST deviceId:iot-device-1,temperature:24.87515271926177
2018-09-11 11:39:14.767 BST deviceId:iot-device-2,temperature:13.28333868428376
2018-09-11 11:39:15.663 BST deviceId:iot-device-1,temperature:24.89544524980489
2018-09-11 11:39:16.069 BST deviceId:iot-device-2,temperature:13.274484576006879
2018-09-11 11:39:16.575 BST deviceId:iot-device-2,temperature:13.261373735692404
```

Update IoT Device config

Using IoT Core, you can also update the config of individual devices based on certain conditions in your environment. For example, by inspecting the device logs on Stackdriver, you might want the device to turn its fan on if the temperature goes over a certain degree.

Try it out! To notify the `iot-device-1`to update its device config (turn on its fan) if it crosses a temperature threshold, run the following command in the SSH session:

```
gcloud iot devices configs update --device=$DEVICE_NAME_1 \
    --region=$MY_REGION \
    --registry=$REGISTRY_NAME \
    --project=$PROJECT_ID \
    --config-data="turn-on-the-fan"
```

In these cases, the device would need to be configured to respond to such a config update message. In this lab, since you are using a simulator, you will not notice any change in the application logs, but you can inspect Cloud Logging for the system log related to the update config request.

1. Go back to Cloud Logging tab.

2. Refresh the Logs page and inspect the update device state log being displayed in Cloud Logging Console.

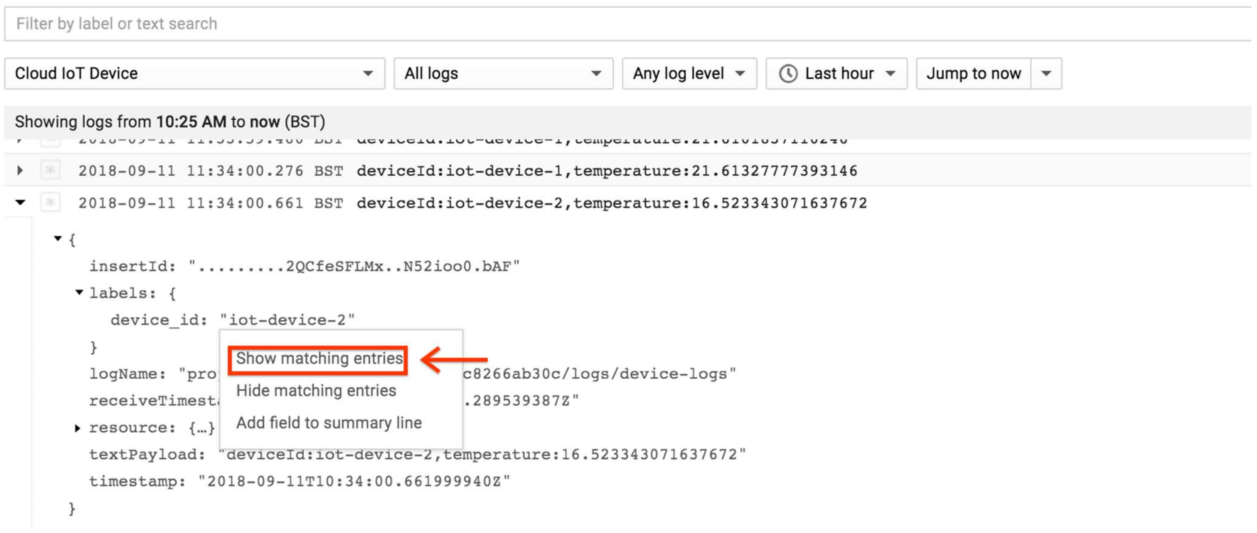
This shows how you can view both system logs from IoT Core and device application logs in one place.

Filter logs

So far you have been looking at a log that contains entries for all devices. Given that these are structured log entries, you can use the device ID of the resource to limit your view to only one device, which is a more realistic scenario when multiple devices are writing log events.

1. Still in Cloud Logging in the Google Cloud console, open any individual log entry by clicking an arrow.

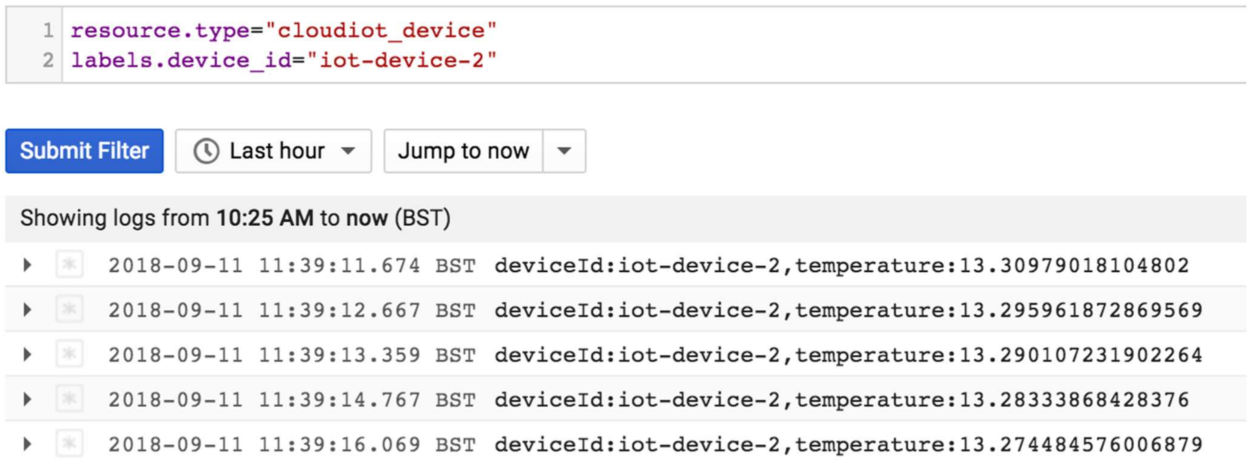
2. Click on the labels arrow to open the topic, then click on **device_id** and choose **Show matching entries** to find logs pertaining only to that device.



Showing logs from **10:25 AM** to **now** (BST)

```
2018-09-11 11:34:00.276 BST deviceId:iot-device-1,temperature:21.61327777393146
2018-09-11 11:34:00.661 BST deviceId:iot-device-2,temperature:16.523343071637672
{
  insertId: "........2QCfeSFLMx..N52ioo0.bAF"
  labels: {
    device_id: "iot-device-2"
  }
  logName: "pro                  c8266ab30c/logs/device-logs"
  receiveTimest                 .289539387Z"
  resource: {…}
  textPayload: "deviceId:iot-device-2,temperature:16.523343071637672"
  timestamp: "2018-09-11T10:34:00.661999940Z"
}
```

This creates a filter of the log like :



```
resource.type="cloudiot_device"
labels.device_id="iot-device-2"
```

Showing logs from **10:25 AM** to **now** (BST)

```
2018-09-11 11:39:11.674 BST  deviceId:iot-device-2,temperature:13.30979018104802
2018-09-11 11:39:12.667 BST  deviceId:iot-device-2,temperature:13.295961872869569
2018-09-11 11:39:13.359 BST  deviceId:iot-device-2,temperature:13.290107231902264
2018-09-11 11:39:14.767 BST  deviceId:iot-device-2,temperature:13.28333868428376
2018-09-11 11:39:16.069 BST  deviceId:iot-device-2,temperature:13.274484576006879
```

Test your Understanding

Below are multiple-choice questions to reinforce your understanding of this lab's concepts. Answer them to the best of your abilities.

Cloud Logging allows you to store, search, analyze, monitor, and alert on log data and events from Google Cloud.
True

Congratulations!

You learned how to configure CloudFunctions to send IoT Core device application logs to Cloud Logging.

Finish Your Quest

This self-paced lab is part of the Qwiklabs IoT in the Google Cloud Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

Take your Next Lab

Continue your quest with A Tour of Cloud IoT Core, or try these suggestions:

- Building an IoT Analytics Pipeline on Google Cloud
- Streaming IoT Core to Cloud Storage

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.
Manual Last Updated June 3, 2020
Lab Last Tested June 3, 2020