

APIs Explorer: PubSub and IoT

GSP284



Overview

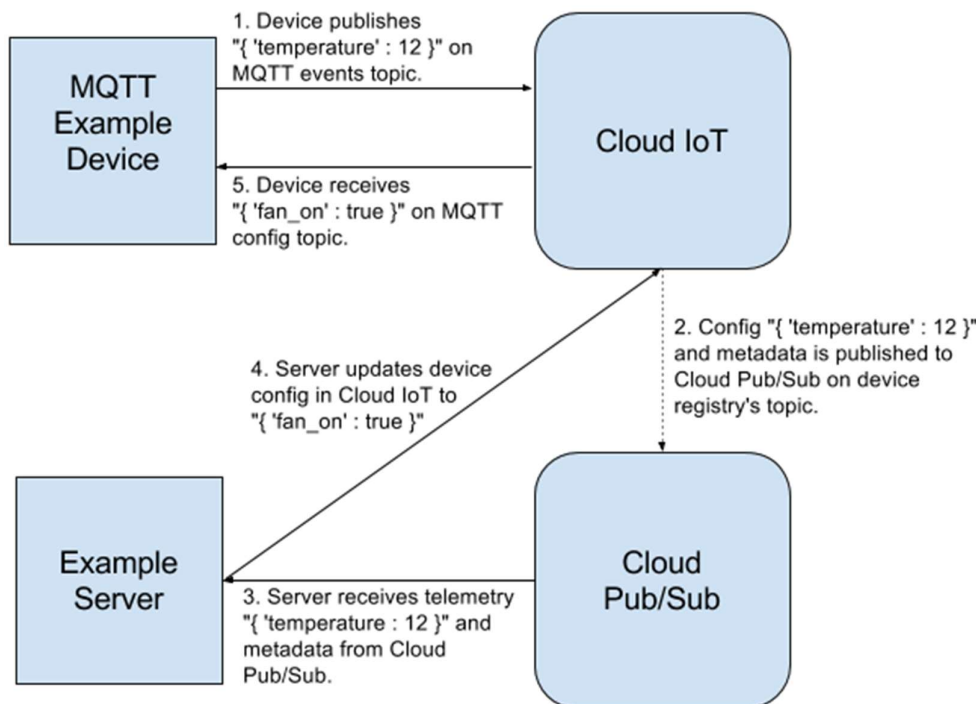
The [Google APIs Explorer](#) is a tool that helps you explore various Google APIs interactively. With the APIs Explorer, you can:

- Browse quickly through available APIs and versions.
- See methods available for each API and what parameters they support along with inline documentation.
- Execute requests for any method and see responses in real time.
- Make authenticated and authorized API calls.
- Search across all services, methods, and your recent requests to quickly find what you are looking for.

[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. In this lab you'll build a simple but complete IoT system using Cloud IoT Core and [Pub/Sub](#).

The devices in this system publish temperature data to their telemetry feeds, and a server consumes the telemetry data from a Cloud Pub/Sub topic. The server then decides whether to turn on or off the individual devices' fans, via a Cloud IoT Core configuration update. The device will respond to configuration changes from a server based on real-time data.

To accomplish this, you will create Pub/Sub topics and subscriptions as well as IoT registries and devices. You will configure and deploy these services with the Google APIs Explorer. The following model illustrates how the system's components are interconnected:



Objectives

In this lab, you will:

- Create a Pub/Sub topic and subscription with the APIs Explorer.
- Create an IoT registry and add a device to it.
- Provision a device and transmit telemetry data from it.
- Control a device using a server based on a telemetry stream.

Prerequisites

This is an **advanced level** lab. You should be familiar with the basic functioning and architecture of APIs. Experience with Cloud Shell and command line interface tools is recommended.

Familiarity with the the APIs Explorer tool, Cloud IoT, and Cloud Pub/Sub is recommended, so please at a minimum take the following labs before attempting this one:

- [APIs Explorer: Qwik Start](#)
- [Internet of Things: Qwik Start](#)
- [Google Cloud Pub/Sub: Qwik Start - Console](#)

Once you're ready, scroll down and follow the steps below to get your lab environment set up.

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:

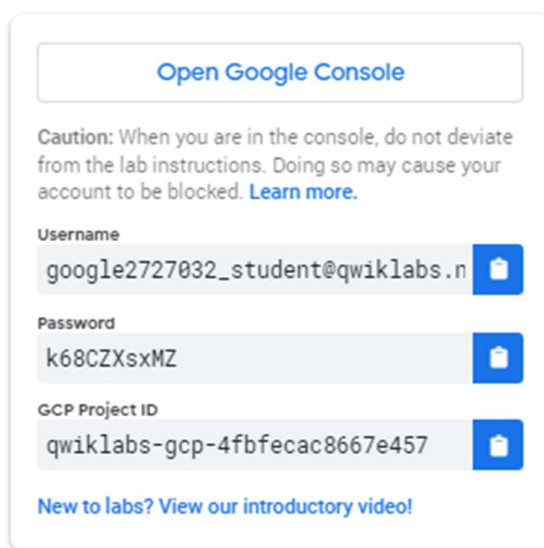
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

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.

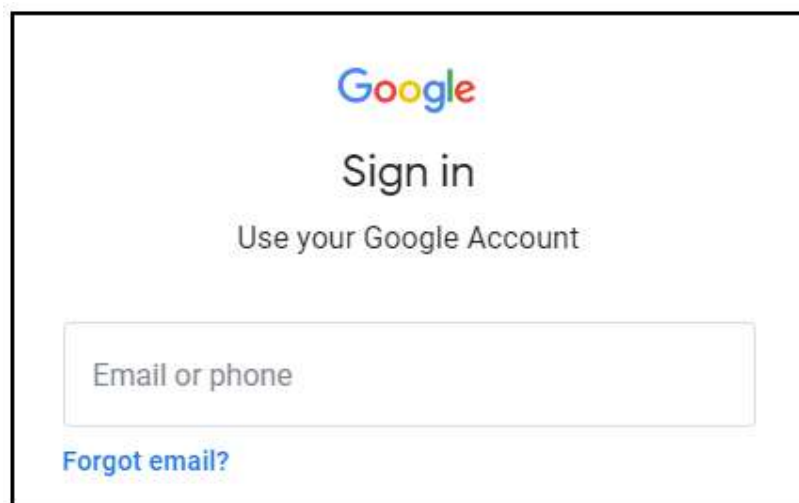
How to start your lab and sign in to the Google Cloud Console

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.



A screenshot of a web interface for starting a lab. At the top is a button labeled "Open Google Console". Below it is a caution message: "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.](#)". There are three input fields, each with a copy icon to its right: "Username" with the value "google2727032_student@qwiklabs.n", "Password" with the value "k68CZXsxMZ", and "GCP Project ID" with the value "qwiklabs-gcp-4fbfecac8667e457". At the bottom is a link: "New to labs? View our introductory video!"

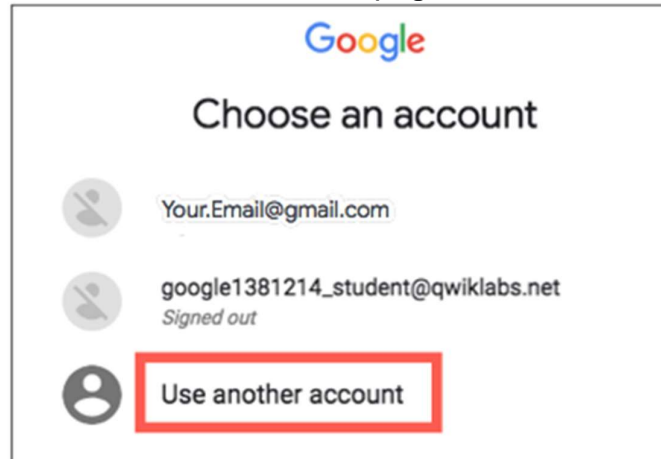
2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



A screenshot of the Google Sign in page. It features the Google logo at the top, followed by the text "Sign in" and "Use your Google Account". Below this is a large input field labeled "Email or phone". At the bottom left is a link: "Forgot email?"

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

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

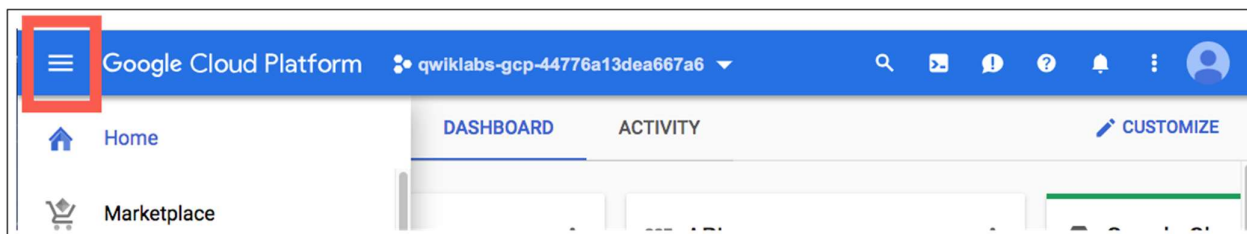
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

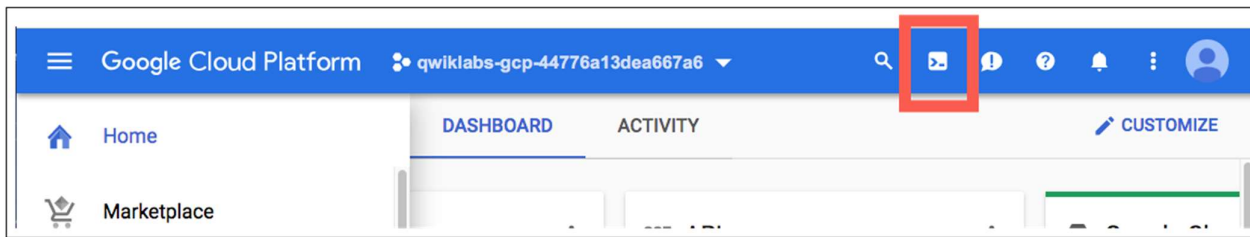
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



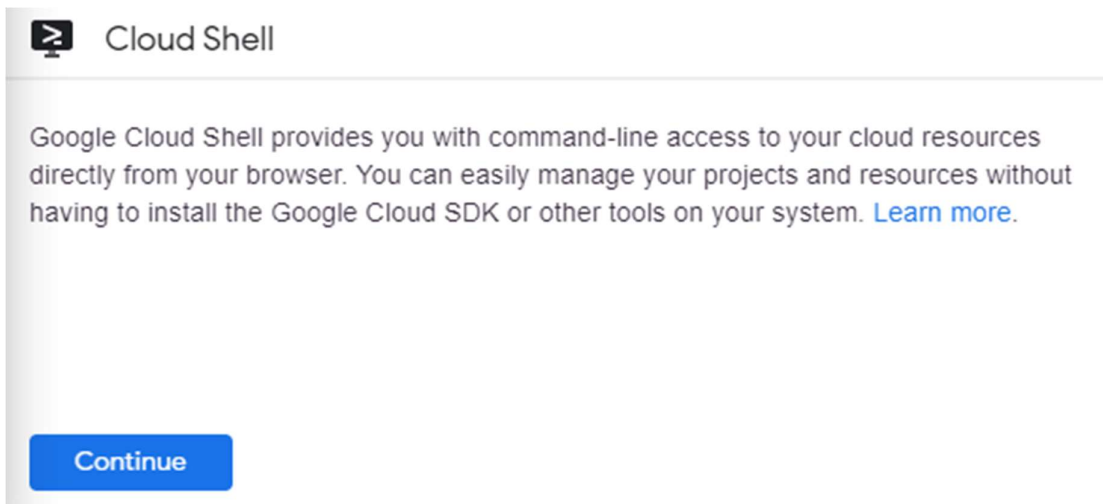
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

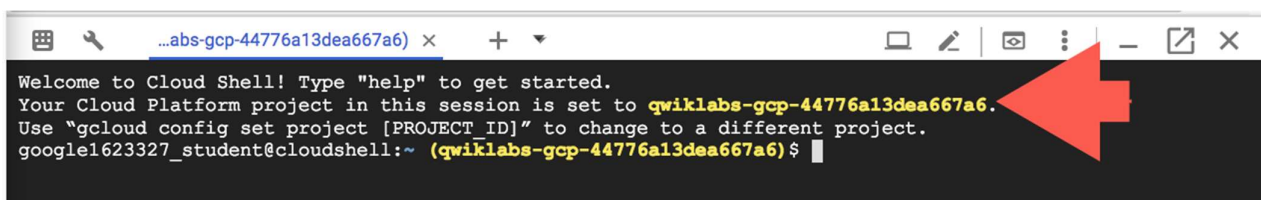
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
- <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:  
- google1623327 student@gwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]  
project = <project_ID>
```

(Example output)

```
[core]  
project = gwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the [gcloud command-line tool overview](#).

Download the Sample Code

In Cloud Shell run the following command to clone the codebase that contains the telemetry server:

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples
```

Enter the following command to `cd` into the sample folder:

```
cd python-docs-samples/iot/api-client/end_to_end_example
```

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
```

Install the sample dependencies:

```
pip install -r requirements.txt
```

Check that you have installed the Python dependencies correctly by running all the Python scripts without passing any parameters:

```
python clouddiot_pubsub_example_mqtt_device.py
python clouddiot_pubsub_example_server.py
```

If the dependencies installed successfully, the programs will print their respective usage messages. This is an example of a successful installation:

```
gcpstaging20065_student@qwiklabs-gcp-4f438a9a5e0b68c9:~/python-docs-samples/iot/api-
client/end_to_end_example$ python clouddiot_pubsub_example_mqtt_device.py
usage: clouddiot_pubsub_example_mqtt_device.py [-h] --project id PROJECT_ID
                                              --registry id REGISTRY_ID
                                              --device id DEVICE_ID
                                              --private_key_file
                                              PRIVATE_KEY_FILE --algorithm
                                              {RS256,ES256}
                                              [--cloud_region CLOUD_REGION]
                                              [--ca_certs CA_CERTS]
                                              [--num_messages NUM_MESSAGES]
                                              [--mqtt_bridge_hostname
MQTT_BRIDGE_HOSTNAME]
                                              [--mqtt_bridge_port MQTT_BRIDGE_PORT]
                                              [--message_type {event,state}]
clouddiot_pubsub_example_mqtt_device.py: error: the following arguments are required: --
project id, --registry id, --device id, --private key file, --algorithm
(env) gcpstaging20065_student@qwiklabs-gcp-4f438a9a5e0b68c9:~/python-docs-
samples/iot/api-client/end_to_end_example$ python clouddiot_pubsub_example_server.py
usage: clouddiot_pubsub_example_server.py [-h] --project id PROJECT_ID
                                           --pubsub_subscription
                                           PUBSUB_SUBSCRIPTION
                                           [--service_account_json SERVICE_ACCOUNT_JSON]
clouddiot_pubsub_example_server.py: error: the following arguments are required: --
project id, --pubsub_subscription
```

You can safely ignore any errors that resemble `clouddiot_pubsub_example_mqtt_device.py: error: argument --project id is required`

Create a Pub/Sub Topic

To access the Pub/Sub APIs Explorer tool to use the `projects.topics.create` method, open [this link](#) in a new tab. You will see a "Try this API" panel on the right of the screen.

Now fill out the method so that the **name** field matches the following, replacing `<YOUR-QWIKLABS-PROJECT>` with your Project ID for this lab:

```
projects/<YOUR-QWIKLABS-PROJECT>/topics/pubsub-topic
```


Your method should now resemble the following:

Try this API

Call this method on live data to see the API request and response. Need help with the API Explorer? Check the [support page](#).

Request parameters

name

projects/qwiklabs-gcp-00-d6bc3e616896/topics/pubsub-topic

Show standard parameters ▾

Request body

{

⊕ Add request body parameters

}

For suggestions, press control+space or click one of the blue "add" circles.

Credentials ⓘ

☒ Google OAuth 2.0

OAuth 2.0 provides authenticated access to an API. [Show scopes ▾](#)

☒ API key

An API key is a unique string that lets you access an API.

EXECUTE

By clicking above, I agree that my use of the API Explorer is governed by the [Terms](#) and [Privacy Policy](#).

Make sure that there are no trailing spaces in the name field. Click the **Execute** button, choose your Qwiklabs student account and click **Allow**.

Your response should resemble the following:

```
200
- Show headers -
{
  "name": "projects/qwiklabs-gcp-a50ec278400a3be5/topics/pubsub-topic"
}
```

The devices in this system publish temperature data to their telemetry feeds, and the server will consume telemetry data from the Cloud Pub/Sub topic you just created.

Test Completed Task

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

Close this browser tab once you have verified the above output.

Create a Pub/Sub Subscription

To access the Pub/Sub APIs Explorer tool to use the `projects.subscriptions.create` method, open [this link](#) in a new tab.

You will see a "Try this API" panel on the right side of the screen.

Now fill out the method so that the **name** field matches the following, replacing `<YOUR-QWIKLABS-PROJECT>` with your Qwiklabs project ID:

```
projects/<YOUR-QWIKLABS-PROJECT>/subscriptions/iot-sub
```

Click inside the curly braces of the request body and select **topic** from the dropdown menu. Then click inside the quotations for `topic` and add the following, replacing `<YOUR-PROJECT-ID>` with your Google Cloud project ID:

```
projects/<YOUR-PROJECT-ID>/topics/pubsub-topic
```

Your method should now resemble the following:

Try this API

Call this method on live data to see the API request and response. Need help with the API Explorer? Check the [support page](#).

Request parameters

name

projects/qwiklabs-gcp-00-2a912aa77c89/subscriptions/iot-sub

[Show standard parameters](#) ▾

Request body

{

"topic": "projects/qwiklabs-gcp-00-2a912aa77c89/topics/pubsub-topi

}

For suggestions, press control+space or click one of the blue "add" circles.

Credentials ?

Make sure that there are no trailing spaces in the name field. Click the **Execute** button. Your response should resemble the following:

```
200
- Show headers -
{
  "name": "projects/qwiklabs-gcp-00-2a912aa77c89/subscriptions/iot-sub",
  "topic": "projects/qwiklabs-gcp-00-2a912aa77c89/topics/pubsub-topic",
  "pushConfig": {},
  "ackDeadlineSeconds": 10,
  "messageRetentionDuration": "604800s",
  "expirationPolicy": {
    "ttl": "2678400s"
  }
}
```

This subscription is a direct channel to the Pub/Sub topic where the telemetry data is published and it allows your device to pull messages directly from `pubsub-topic`.

Test Completed Task

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

You can now close the Pub/Sub APIs Explorer tab.

View the topic and subscription in the Console

Return to the Cloud Console and from the **Navigation menu** under the Big Data header click **Pub/Sub > Topics**. You should see the `pubsub-topic` you created:

Topics			
		+ CREATE TOPIC	🗑 DELETE
☰ Filter table			
<input type="checkbox"/>	Topic ID ↑	Encryption	Topic name
<input type="checkbox"/>	pubsub-topic	Google-managed	projects/qwiklabs-gcp-00-2a912aa77c89/topics/pubsub-topic 📄

From the left-hand menu, select **Subscriptions**. You should see the `iot-sub` subscription that you created:

Subscriptions

CREATE SUBSCRIPTION

DELETE

Filter table

<div><input type="checkbox"/></div> <div>Subscription ID <div>↑</div></div>	Delivery type	Topic name	Subscription name
<div><input type="checkbox"/></div> <div>iot-sub</div>	Pull	projects/qwiklabs-gcp-00-2a...	projects/qwiklabs-gcp-00-2a91...

Now that you have your Pub/Sub topic and subscription created, you will build a registry to contain your IoT device.

Create an IoT Registry

In your Cloud Shell session, run the following command to create an IoT registry named `iot-registry`:

```
gcloud iot registries create iot-registry \
  --region=us-centrall --event-notification-config=topic=pubsub-topic
```

From the **Navigation menu** under the Big Data header click **IoT Core**. This should take you to the registries page, and you should see the `iot-registry` you just created:

IoT Core		Registries	+ CREATE REGISTRY
Filter registries			
<input type="checkbox"/>	Registry ID ^		
<input type="checkbox"/>	iot-registry		

Test Completed Task

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

Generate RSA keys and create a device

Now generate RSA public and private keys that will be used for authenticating your virtual device when it connects.

In Cloud Shell, verify that you are still in the `/python-docs-samples/iot/api-client/end_to_end_example` directory.

Run the following command to generate RSA keys:

```
openssl req -x509 -newkey rsa:2048 -days 3650 -keyout rsa_private.pem \
  -nodes -out rsa_public.pem -subj "/CN=unused"
```

Run the following command to view the contents of the key:

```
nano rsa_public.pem
```

You should get a similar output:

```
-----BEGIN CERTIFICATE-----
MIIC+DCCAeCgAwIBAgIJAP1GA1sC2yFmMA0GCSqGSIb3DQEBCwUAMBExDzANBgNV
BAMMBnVudXNlZDAeFw0xODEwMjUxNzQ3NDRaFw0yODEwMjUxNzQ3NDRaMBExDzAN
BgNVBAMMBnVudXNlZDCCASiwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKSQ
RUUwhJAxFSpkz2V4/6eR3XIwi9eXlc/qWlZnxlOWdCOuyoMXQZZs4DzloIlfCbKg
opa3zUfoz9QdlShJP9e4gXggWlrxDlJdPT4ixsaeWk3E9oE22smpGCea8ZvhcZF/
NFow9i3D2j+DihvELuYPROxBP2keg3wZiEBo8U5imxf81x00+sTPQ0v5ArqvTW3P
GBU2DlZ6MhwmgilmKGc4Q6pUSQzWYYbsmkt5/VM01cTVSYKJN7G/zQRwbMffAslI
R4bvC8jdibnw2oIQoAjPS+I7gaeGLSFMkYGkg9BNB5RashOsZOMedvqu8QjYcpgO
VXNB8JJv8e48wFygmfMCAwEAAaNTMFEwHQYDVR0OBBYEFF+XBeH7OF7E65fcfj00
Hg0rNFGEMB8GA1UdIwQYMBaAFF+XBeH7OF7E65fcfj00Hg0rNFGEMA8GA1UdEwEB
/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAI/sijZDeIOJ7+ffs4IBgBVTekCN
nU3NZ9H6bxTItb8dOIzUKBGW80p3VJtjqjixaJQl2JLfit4kCU89IPq1glPY9oab
GaoikrvF3kgxJRC7H+vtVGKmu3bH12iTAqVgh9hjuGIyB+CWGJExfumTs7dcWF1+
GT5NeQKdmak4mzx/POVwqE/NY+UVoRt9rOduv6UZyBc+X0516hOcI5omrPKh/7c6
hknHidsKzTK1oxSPUD2OCQjuKz55ugYVpEUu/TvvUEx26YbWFkIjrlGWyyzaVjEs
UL4HuPT51u4rdNC2TSqcTyQW/m6i3wuVzfqr9jhGFFB7amDIdRX4bJ8qx5Q=
-----END CERTIFICATE-----
```

Then exit the `nano` editor by holding **CTRL + X**.

<!-->

Now run the following command to create a device called `my-device`:

```
gcloud iot devices create my-device --region=us-central1 \
  --registry=iot-registry \
  --public-key path=rsa_public.pem,type=rs256
```

</!-->

Now in the Console click on `iot-registry`. In the left panel click **Devices** then click **Create a Device**.

Device ID = `my-device`

Click **Create**.

Click the **Devices** in the left panel again and you should see your newly created device `my-device`.

Test Completed Task

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

Start the Virtual Server

Now that you have created all of the Cloud resources, connect your device and communicate with it via Pub/Sub, simulating a device and server.

Run the following command to open the `cloudiot_pubsub_example_server.py` file with the `nano` text editor:

```
nano cloudiot_pubsub_example_server.py
```

Scroll down and replace the code used to generate service account credentials from a provided JSON file to instead use the built-in credentials for Compute Engine, which is running under the hood of the Cloud Shell.

Replace the following code:

```
def __init__(self, service_account_json):
    credentials = ServiceAccountCredentials.from_json_keyfile_name(
        service_account_json, API_SCOPES)
```

With this:

```
def __init__(self, service_account_json):
    from google.auth import compute_engine
    credentials = compute_engine.Credentials()
```

Then press **CTRL+X** → **Y** → **ENTER** to save the file and exit the text editor.

Now that you have changed the server to use the Compute Engine credentials, start the server by running the following command, replacing `<YOUR-PROJECT-ID>` with your Project ID:

```
python clouddiot_pubsub_example_server.py \
  --project_id "<YOUR-PROJECT-ID>" \
  --pubsub_subscription=iot-sub
```

When the server starts, you will see the message `Listening for messages on projects/your-project-id/subscriptions/iot-sub`, which indicates the server is running.

Start the Virtual Device and Observe

Add a new tab to your Cloud Shell by clicking the **+** icon on the Cloud Shell ribbon.

In the new Cloud Shell tab, run the following to navigate to the device sample folder and initialize your virtual environment:

```
cd python-docs-samples/iot/api-client/end_to_end_example
source venv/bin/activate
```

Retrieve the latest root certificate from Google:

```
wget pki.goog/roots.pem
```

Now, connect the virtual device using the private key, registry ID, device ID, and so on by running the following command, replacing `<YOUR-PROJECT-ID>` with your Project ID:

```
python clouddiot_pubsub_example_mqtt_device.py \
  --registry_id iot-registry \
  --device_id my-device \
  --project_id "<YOUR-PROJECT-ID>" \
  --private_key_file rsa_private.pem \
  --mqtt_bridge_port 443 \
  --algorithm RS256 \
  --ca certs/roots.pem \
  --cloud_region us-central1
```

When you connect the device, it will show and report its temperature, which increases when the fan is turned off. If the fan is enabled, the virtual device's temperature will decrease. Because the device is controlled from the server, which is analyzing the stream of incoming sensor data and making this decision for it, the device does not need to be aware of the conditions for enabling or disabling its fan.

The following section shows the output of the server that is subscribed to the telemetry events from the device:

```
Published message acked.
('Publishing payload', '{"temperature": 0}')
Published message acked.
```

```
('Publishing payload', '{"temperature": -1}')  
Published message acked.  
Received message '{"fan_on": false}' on topic '/devices/my-device/config' with Qos 1  
Fan turned off.  
('Publishing payload', '{"temperature": 0}')  
Published message acked.  
('Publishing payload', '{"temperature": 1}')  
Published message acked.  
('Publishing payload', '{"temperature": 2}')
```

Test your Understanding

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

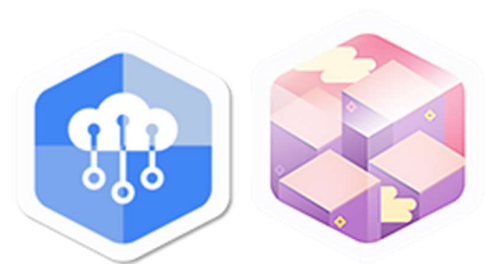
Google Cloud Pub/Sub service allows applications to exchange messages reliably, quickly, and asynchronously.
True

_____ provides a way to manage millions of IoT devices across the globe.
Cloud IoT Core

Congratulations!

You have now set up a virtual device and are successfully transmitting telemetry data and receiving configuration changes. At this point you have a solid understanding of Google Cloud IoT Core, Pub/Sub, and how you can provision their services through the APIs Explorer.

Finish Your Quest



This self-paced lab is part of the Qwiklabs [IoT in the Google Cloud](#) and [Exploring APIs](#) Quests. A Quest is a series of related labs that form a learning path. Completing a 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 a Quest and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](#).

Next Steps / Learn More

Be sure to check out the following labs for more practice with APIs:

- [Using Firestore with Cloud IoT Core for Device Configuration](#)
- [Streaming IoT Kafka to Google Cloud](#)

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 February 10, 2021

Lab Last Tested February 10, 2021

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.