Pub/Sub Lite: Qwik Start

GSP832

Google Cloud Self-Paced Labs

Overview

Complementing Pub/Sub, Pub/Sub Lite is a zonal service for messaging systems with predictable traffic patterns. If you publish 1 MiB-1 GiB of messages per second, Pub/Sub Lite is a low cost option for high-volume event ingestion.
Publishers send messages to Lite topics and subscribers receive messages from Lite subscriptions. Lite topics and Lite subscriptions are zonal resources that must be in the same Cloud project and zone. For a list of zones that Pub/Sub Lite supports, see Pub/Sub Lite locations.
In this lab you will learn how to:

- Create Lite topics and Lite subscriptions using the Google Cloud Console.
- Send and receive messages using the Pub/Sub Lite client library for Python.

Setup

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

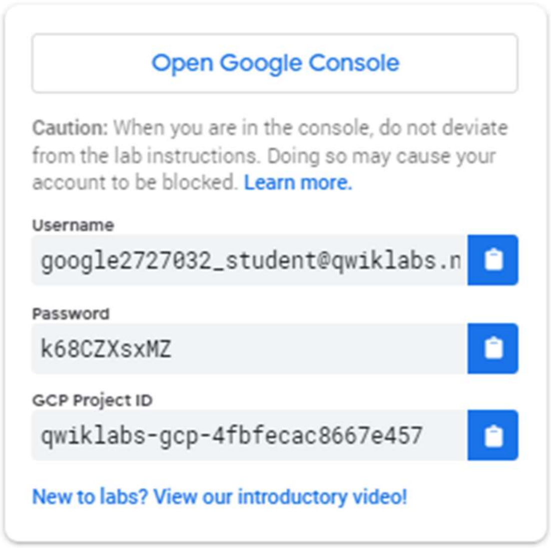
To complete this lab, you need:

- 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.



Open Google Console

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.

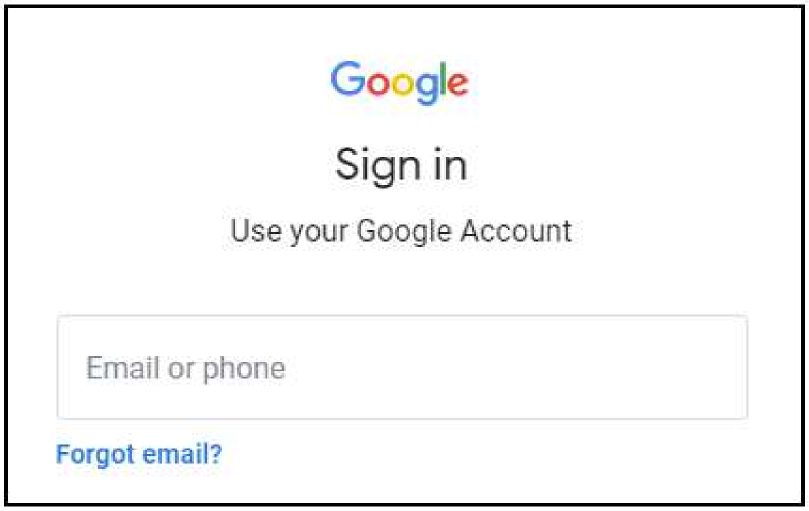
Username
google2727032_student@qwiklabs.n

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

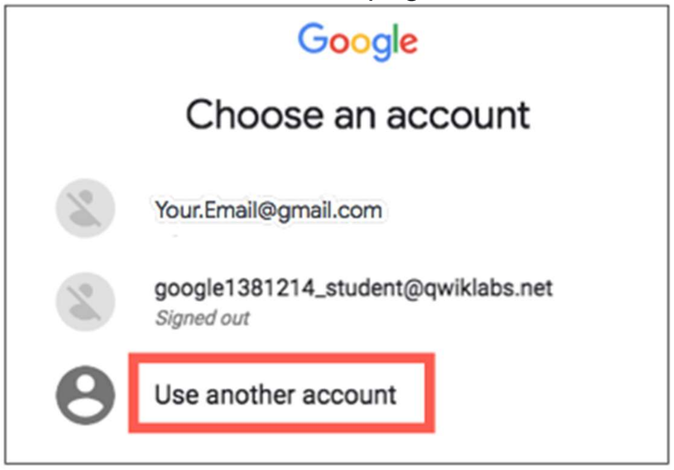
Use your Google Account

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.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

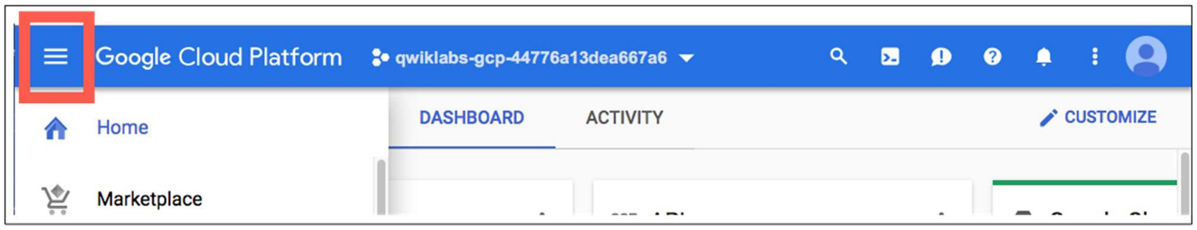
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

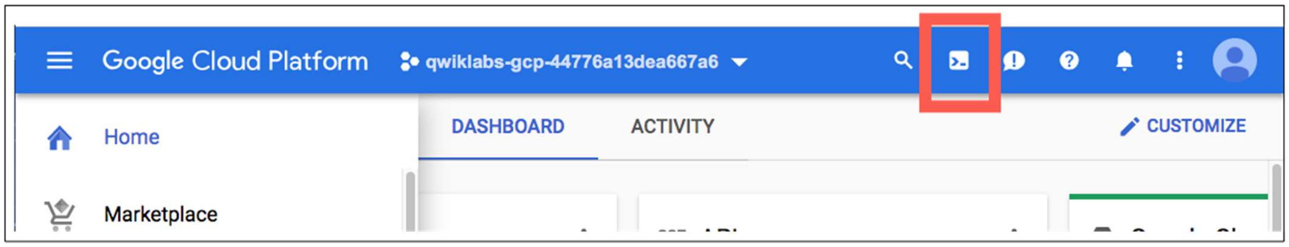
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



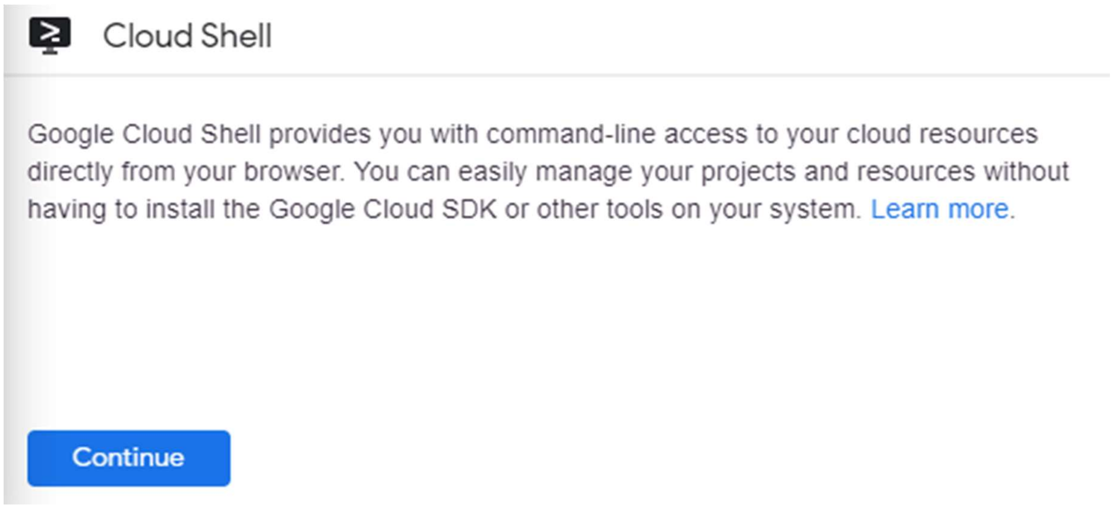
Activate Cloud Shell

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.

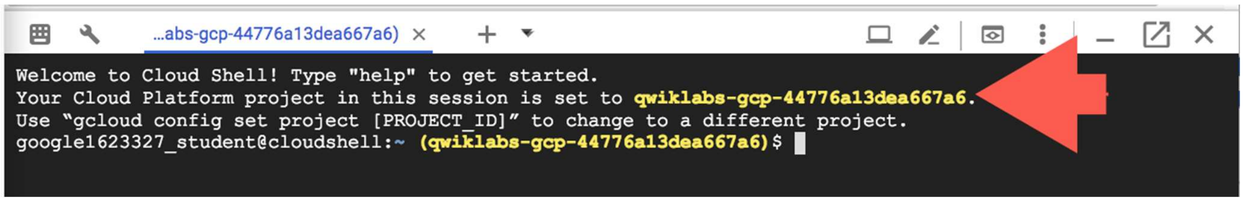


Click **Continue**.



Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

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:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

`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@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

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

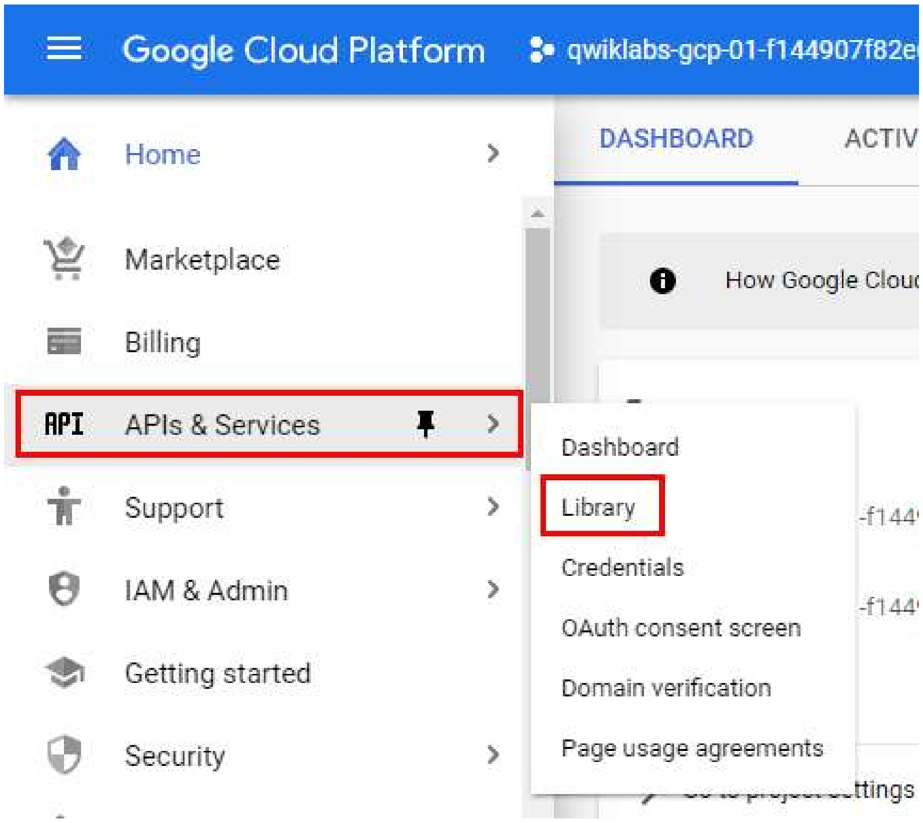
(Example output)

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

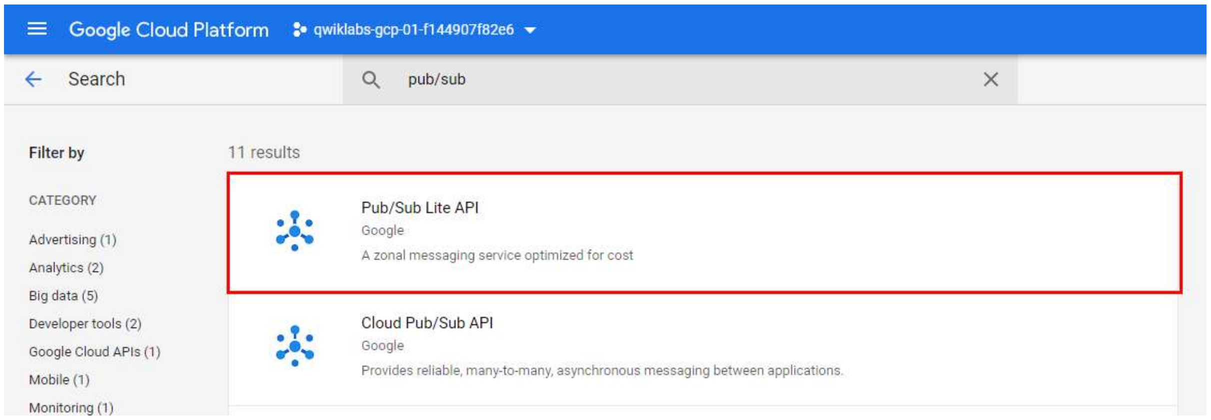
For full documentation of `gcloud` see the gcloud command-line tool overview.

Enable the Pub/Sub Lite API

In the Cloud Console, navigate to **APIs & Services** > **Library**:



Start typing "pub/sub" in the Search bar, then select the **Pub/Sub Lite** tile:



Pub/Sub Lite API
Google
A zonal messaging service optimized for cost

Cloud Pub/Sub API
Google
Provides reliable, many-to-many, asynchronous messaging between applications.

Now click the **Enable** button.

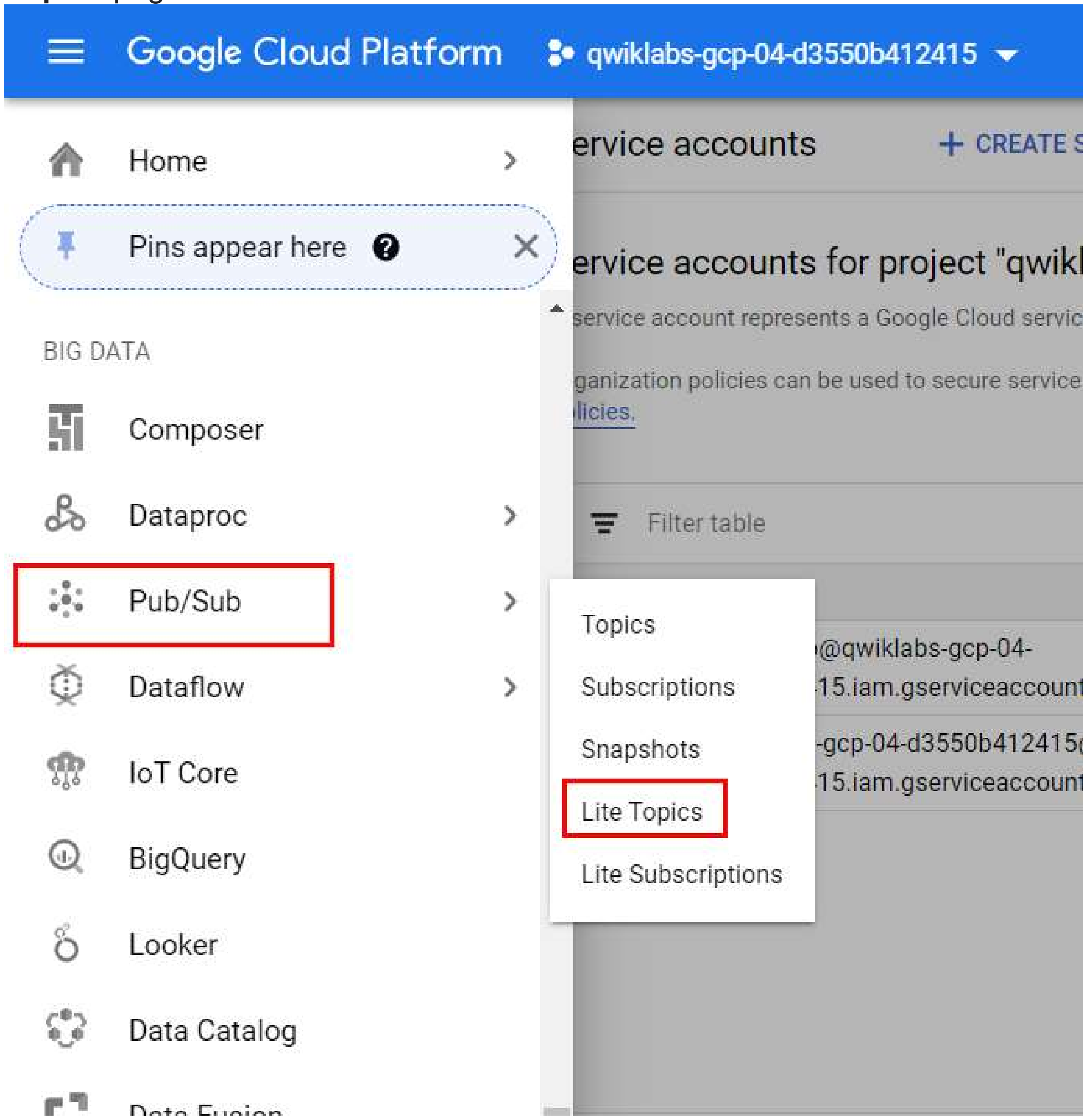
In Cloud Shel, run the following to install the the Python client library:

```
pip3 install --upgrade google-cloud-pubsublite
```

Create a Lite Topic

To create a Lite topic with the Cloud Console, follow these steps:

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



2. Click **Create Lite topic**.
3. Select a region and a zone in the region (this lab uses `us-central1-a`), then click **Continue**.
4. In the **Name** section, enter my-lite-topic as the Lite topic ID. The Lite topic name includes the Lite topic ID, the zone, and the project number. Click **Continue**.
5. Click **Create**.

Click *Check my progress* to verify the objective.

Create a Lite Topic

Check my progress

Create a Lite Subscription

To create a Lite subscription with Cloud Console, follow these steps:

1. Click on **Lite Subscriptions** in the left menu.
2. Click **Create Lite subscription**.
3. In the **Lite subscription ID** field, enter my-lite-subscription.
4. Select a Lite topic to receive messages from - you created this in the last step.
5. In the **Delivery requirement** section, select **Deliver messages after stored**.
6. Click **Create**.

The Lite subscription is in the same zone as the Lite topic.

Click *Check my progress* to verify the objective.

Send messages

Send messages to the Lite topic using the following publisher application. Using the Cloud Shell Editor or the editor of your choice, create a file called `send_messages.py` and add the following script, then:

- **Uncomment** all steps in the TODO section.
- Replace `project_number` with the project ID you can see in the Lite topic and subscription name in the Console.
- If you used a different `cloud_region` and `zone_id`, update with your information.

```
from google.cloud.pubsublite.cloudpubsub import PublisherClient
from google.cloud.pubsublite.types import (
    CloudRegion,
    CloudZone,
    PublishMetadata,
    TopicPath,
)

# TODO(developer):
# project_number = 1122334455
# cloud_region = "us-central1"
# zone_id = "a"
# topic_id = "my-lite-topic"
# num_messages = 100

location = CloudZone(CloudRegion(cloud_region), zone_id)
topic_path = TopicPath(project_number, location, topic_id)

# PublisherClient() must be used in a `with` block or have __enter__() called before
use.
with PublisherClient() as publisher_client:
    data = "Hello world!"
    api_future = publisher_client.publish(topic_path, data.encode("utf-8"))
    # result() blocks. To resolve API futures asynchronously, use add_done_callback().
    message_id = api_future.result()
    publish_metadata = PublishMetadata.decode(message_id)
    print(
        f"Published a message to partition {publish_metadata.partition.value} and
offset {publish_metadata.cursor.offset}."
    )
```

The publisher sends 100 messages to a Lite topic and prints the number of messages that the Pub/Sub Lite service receives.

Receive messages

Receive messages from the Lite subscription using the following subscriber application. Using the Cloud Shell Editor or the editor of your choice, create a file called `receive_messages.py` and add the following script, then

- **Uncomment** all steps in the TODO section.
- Replace `project_number` with the project ID you can see in the Lite topic and subscription name in the Console.
- If you used a different `cloud_region` and `zone_id`, update with your information.

```
from concurrent.futures._base import TimeoutError
from google.cloud.pubsublite.cloudpubsub import SubscriberClient
from google.cloud.pubsublite.types import (
    CloudRegion,
    CloudZone,
    FlowControlSettings,
    SubscriptionPath,
)

# TODO(developer):
# project_number = 1122334455
# cloud_region = "us-central1"
# zone_id = "a"
# subscription_id = "my-lite-subscription"
# timeout = 90

location = CloudZone(CloudRegion(cloud_region), zone_id)
subscription_path = SubscriptionPath(project_number, location, subscription_id)
# Configure when to pause the message stream for more incoming messages based on the
# maximum size or number of messages that a single-partition subscriber has received,
# whichever condition is met first.
per_partition_flow_control_settings = FlowControlSettings(
    # 1,000 outstanding messages. Must be >0.
    messages_outstanding=1000,
    # 10 MiB. Must be greater than the allowed size of the largest message (1 MiB).
    bytes_outstanding=10 * 1024 * 1024,
)

def callback(message):
    message_data = message.data.decode("utf-8")
    print(f"Received {message_data} of ordering key {message.ordering_key}.")
    message.ack()

# SubscriberClient() must be used in a `with` block or have __enter__() called before
use.
with SubscriberClient() as subscriber_client:

    streaming_pull_future = subscriber_client.subscribe(
        subscription_path,
        callback=callback,
        per_partition_flow_control_settings=per_partition_flow_control_settings,
    )

    print(f"Listening for messages on {str(subscription_path)}...")

    try:
        streaming_pull_future.result(timeout=timeout)
    except TimeoutError or KeyboardInterrupt:
        streaming_pull_future.cancel()
        assert streaming_pull_future.done()
```

After the subscriber receives a message, the subscriber prints the message ID and the message data.

Now run both scripts:

```
python3 send_messages.py
python3 receive_messages.py
```

You will see output in Cloud Shell that looks like this:

```
Received Hello world! of ordering key .
```

You will also be able to monitor activity on the Lite Topics and Lite Subscriptions pages in the Cloud Console.

Clean up

Although all resources will be destroyed when you end this lab, it's good practice to clean up resources you don't need to avoid charges.

1. In the Cloud Console, go to the **Lite Topics** page.
2. Click **my-lite-topic**.
3. On the **Lite topic details** page, click **Delete**.
4. In the field that appears, type delete to confirm that you want to delete the Lite topic.
5. Click **Delete**.

Congratulations!

Now you can use Pub/Sub Lite to create topics and subscriptions, and send and receive messages.

Next Steps / Learn More

- Choosing Pub/Sub or Pub/Sub Lite
- Pub/Sub Lite documentation

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 January 19, 2021
Lab Last Tested January 19, 2021