Pub/Sub: Qwik Start - Python

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.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

2. Click through the following windows:

   - Continue through the Cloud Shell information window.
   - Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `PROJECT_ID`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

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

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

Copied!

content_copy

4. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Copied!

content_copy

Output:

```
[core]
project = "PROJECT_ID"
```

Note: For full documentation of `gcloud`, in Google Cloud, refer to the gcloud CLI overview guide.

Task 1. Create a virtual environment

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

1. Install the `virtualenv` environment:

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

Copied!

content_copy

2. Build the virtual environment:

```
python3 -m venv venv
```

Copied!

content_copy

3. Activate the virtual environment:

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

Copied!

content_copy

Task 2. Install the client library

1. Run the following to install the client library:

```
pip install --upgrade google-cloud-pubsub
```

Copied!

content_copy

2. Get the sample code by cloning a GitHub repository:

```
git clone https://github.com/googleapis/python-pubsub.git
```

Copied!

content_copy

3. Navigate to the directory:

```
cd python-pubsub/samples/snippets
```

Copied!

content_copy

Task 3. Pub/Sub - the Basics

Pub/Sub is an asynchronous global messaging service. There are three terms in Pub/Sub that appear often: *topics*, *publishing*, and *subscribing*.

A topic is a shared string that allows applications to connect with one another through a common thread.

Publishers push (or publish) a message to a Pub/Sub topic. Subscribers will then make a *subscription* to that thread, where they will either pull messages from the topic or configure webhooks for push subscriptions. Every subscriber must acknowledge each message within a configurable window of time.

In sum, a publisher creates and sends messages to a topic and a subscriber creates a subscription to a topic to receive messages from it.

Pub/Sub in Google CLoud

Pub/Sub comes preinstalled in Cloud Shell, so there are no installations or configurations required to get started with this service. In this lab you use Python to create the topic, subscriber, and then view the message. You use a gcloud command to publish the message to the topic.

Task 4. Create a topic

To publish data to Pub/Sub you create a topic and then configure a publisher to the topic.

1. In Cloud Shell, your Project ID should automatically be stored in the environment variable `GOOGLE_CLOUD_PROJECT`:

```
echo $GOOGLE_CLOUD_PROJECT
```

Copied!

content_copy

2. Ensure the output is the same as the Project ID in your CONNECTION DETAILS.

`publisher.py` is a script that demonstrates how to perform basic operations on topics with the Cloud Pub/Sub API. View the content of publisher script:

```
cat publisher.py
```

Copied!

content_copy

Note: Alternatively, you can use the shell editors that are installed on Cloud Shell, such as nano or vim or use the Cloud Shell code editor to view `python-pubsub/samples/snippets/publisher.py`.

3. For information about the publisher script:

```
python publisher.py -h
```

Copied!

content_copy

Example output:

```
usage: publisher.py [-h]
                    project
                    {list,create,delete,publish,publish-with-custom-
attributes,publish-with-futures,publish-with-error-handler,publish-
with-batch-settings}
                    ...

This application demonstrates how to perform basic operations on topics
with the Cloud Pub/Sub API.

For more information, see the README.md under /pubsub and the
documentation
at https://cloud.google.com/pubsub/docs.

positional arguments:
  project               Your Google Cloud project ID
  {list,create,delete,publish,publish-with-custom-attributes,publish-
with-futures,publish-with-error-handler,publish-with-batch-settings}
    list                Lists all Pub/Sub topics in the given project.
    create              Create a new Pub/Sub topic.
    delete              Deletes an existing Pub/Sub topic.
    publish             Publishes multiple messages to a Pub/Sub topic.
    publish-with-custom-attributes
                        Publishes multiple messages with custom
attributes to
                        a Pub/Sub topic.
    publish-with-futures
                        Publishes multiple messages to a Pub/Sub topic
and
                        prints their message IDs.
    publish-with-error-handler
                        Publishes multiple messages to a Pub/Sub topic
with an
                        error handler.
    publish-with-batch-settings
                        Publishes multiple messages to a Pub/Sub topic
with
                        batch settings.

optional arguments:
  -h, --help            show this help message and exit
```

4. Run the publisher script to create Pub/Sub Topic:

```
python publisher.py $GOOGLE_CLOUD_PROJECT create MyTopic
```

Copied!

content_copy

Example output:

```
Topic created: name: "projects/qwiklabs-gcp-
fe27729bc161fb22/topics/MyTopic"
```

5. This command returns a list of all Pub/Sub topics in a given project:

```
python publisher.py $GOOGLE_CLOUD_PROJECT list
```

Copied!

content_copy

Example output:

```
name: "projects/qwiklabs-gcp-fe27729bc161fb22/topics/MyTopic"
```

You can also view the topic you just made in the Cloud Console.

6. Navigate to **Navigation menu** > **Pub/Sub** > **Topics**.

You should see `MyTopic`.

Task 5. Create a subscription

1. Create a Pub/Sub subscription for topic with `subscriber.py` script:

```
python subscriber.py $GOOGLE_CLOUD_PROJECT create MyTopic MySub
```

2. This command returns a list of subscribers in given project:

```
python subscriber.py $GOOGLE_CLOUD_PROJECT list-in-project
```

Copied!

content_copy

You'll see only one subscription because you've made only one subscription.

Example output:

```
projects/qwiklabs-gcp-7877af129f04d8b3/subscriptions/MySub
```

3. Check out the subscription you just made in the console. In the left pane, click **Subscriptions**. You should see the subscription name and other details.

4. For information about the `subscriber` script:

```
python subscriber.py -h
```

Copied!

content_copy

Output:

```
usage: subscriber.py [-h]
                     project
                     {list_in_topic,list_in_project,create,create-
push,delete,update,receive,receive-custom-attributes,receive-flow-
control,receive-synchronously,listen_for_errors}
                     ...

This application demonstrates how to perform basic operations on
subscriptions with the Cloud Pub/Sub API.

For more information, see the README.md under /pubsub and the
documentation
at https://cloud.google.com/pubsub/docs.

positional arguments:
  project               Your Google Cloud project ID
  {list_in_topic,list_in_project,create,create-
push,delete,update,receive,receive-custom-attributes,receive-flow-
control,receive-synchronously,listen_for_errors}
    list_in_topic       Lists all subscriptions for a given topic.
    list_in_project     Lists all subscriptions in the current project.
    create              Create a new pull subscription on the given
topic.
    create-push         Create a new push subscription on the given
topic.
    delete              Deletes an existing Pub/Sub topic.
    update              Updates an existing Pub/Sub subscription's push
                        endpoint URL. Note that certain properties of a
                        subscription, such as its topic, are not
modifiable.
    receive             Receives messages from a pull subscription.
    receive-custom-attributes
                        Receives messages from a pull subscription.
    receive-flow-control
                        Receives messages from a pull subscription with
flow
                        control.
    receive-synchronously
                        Pulling messages synchronously.
    listen_for_errors   Receives messages and catches errors from a
pull
                        subscription.

optional arguments:
  -h, --help            show this help message and exit
```

Task 6. Publish messages

Now that you've set up `MyTopic` (the topic) and a subscription to `MyTopic` (`MySub`), use `gcloud` commands to publish a message to `MyTopic`.

1. Publish the message "Hello" to `MyTopic`:

```
gcloud pubsub topics publish MyTopic --message "Hello"
```

Copied!

content_copy

2. Publish a few more messages to `MyTopic`—run the following commands (replacing <YOUR NAME> with your name and <FOOD> with a food you like to eat):

```
gcloud pubsub topics publish MyTopic --message "Publisher's name is
<YOUR NAME>"
```

Copied!

content_copy

```
gcloud pubsub topics publish MyTopic --message "Publisher likes to eat
<FOOD>"
```

Copied!

content_copy

```
gcloud pubsub topics publish MyTopic --message "Publisher thinks
Pub/Sub is awesome"
```

Copied!

content_copy

Task 7. View messages

Now that you've published messages to MyTopic, pull and view the messages using MySub.

1. Use MySub to pull the message from MyTopic:

```
python subscriber.py $GOOGLE_CLOUD_PROJECT receive MySub
```

Copied!

content_copy

Example output:

```
Listening for messages on projects/qwiklabs-gcp-
7877af129f04d8b3/subscriptions/MySub
Received message: Message {
  data: 'Publisher thinks Pub/Sub is awesome'
  attributes: {}
}
Received message: Message {
  data: 'Hello'
  attributes: {}
}
Received message: Message {
  data: "Publisher's name is Harry"
  attributes: {}
}
Received message: Message {
  data: 'Publisher likes to eat cheese'
  attributes: {}
}
```

2. Click **Ctrl**+**c** to stop listening.