


Pub/Sub: Qwik Start - Command Line

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](#).

Pub/Sub basics

As stated earlier, 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 Cloud Pub/Sub topic.
- Subscribers make a "`subscription`" to a topic where they will either pull messages from the subscription or configure webhooks for push subscriptions. Every subscriber must acknowledge each message within a configurable window of time.

To sum it up, a producer publishes messages to a topic and a consumer creates a subscription to a topic to receive messages from it.


Task 1. Pub/Sub topics

Pub/Sub comes preinstalled in Cloud Shell, so there are no installations or configurations required to get started with this service.

1. Run the following command to create a topic called `myTopic`:
`gcloud pubsub topics create myTopic`

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](#).

Pub/Sub basics

As stated earlier, 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 Cloud Pub/Sub topic.
- Subscribers make a "subscription" to a topic where they will either pull messages from the subscription or configure webhooks for push subscriptions. Every subscriber must acknowledge each message within a configurable window of time.

To sum it up, a producer publishes messages to a topic and a consumer creates a subscription to a topic to receive messages from it.

Task 1. Pub/Sub topics

Pub/Sub comes preinstalled in Cloud Shell, so there are no installations or configurations required to get started with this service.

1. Run the following command to create a topic called `myTopic`:

```
gcloud pubsub topics create myTopic  
gcloud pubsub subscriptions create --topic myTopic mySubscription
```

2. Add another two subscriptions to myTopic. Run the following commands to make Test1 and Test2 subscriptions:

```
gcloud pubsub subscriptions create --topic myTopic Test1
```

Copied!

content_copy

```
gcloud pubsub subscriptions create --topic myTopic Test2
```

Copied!

content_copy

3. Run the following command to list the subscriptions to myTopic:

```
gcloud pubsub topics list-subscriptions myTopic
```

Copied!

content_copy

Your output should resemble the following:

```
---
projects/qwiklabs-gcp-3450558d2b043890/subscriptions/Test2
---
projects/qwiklabs-gcp-3450558d2b043890/subscriptions/Test1
---
projects/qwiklabs-gcp-3450558d2b043890/subscriptions/mySubscription
```

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.

4. Now delete the Test1 and Test2 subscriptions. Run the following commands:

```
gcloud pubsub subscriptions delete Test1
```

Copied!

content_copy

```
gcloud pubsub subscriptions delete Test2
```

Copied!

content_copy

5. See if the Test1 and Test2 subscriptions were deleted. Run the list-subscriptions command one more time:

```
gcloud pubsub topics list-subscriptions myTopic
```

Copied!

content_copy

You should get the following output:

```
---
projects/qwiklabs-gcp-3450558d2b043890/subscriptions/mySubscription
```

Task 3. Pub/Sub publishing and pulling a single message

Next you'll learn how to publish a message to a Pub/Sub topic.

1. Run the following command to publish the message "hello" to the topic you created previously (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

Next, use the `pull` command to get the messages from your topic. The `pull` command is subscription based, meaning it should work because earlier you set up the subscription mySubscription to the topic myTopic.

3. Use the following command to pull the messages you just published from the Pub/Sub topic:

```
gcloud pubsub subscriptions pull mySubscription --auto-ack
```

Copied!

content_copy

Your output should resemble the following:

```
DATA: Publisher likes to eat Chicken  
MESSAGE_ID: 7881270261639526  
ORDERING_KEY:  
ATTRIBUTES:  
DELIVERY_ATTEMPT:  
ACK_STATUS: SUCCESS
```

What's going on here? You published 4 messages to your topic, but only 1 was outputted.

Now is an important time to mention a couple features of the `pull` command that often trip developers up:

- **Using the pull command without any flags will output only one message, even if you are subscribed to a topic that has more held in it.**
- **Once an individual message has been outputted from a particular subscription-based pull command, you cannot access that message again with the pull command.**
 4. To see what the second bullet is talking about, run the last command three more times. You will see that it will output the other messages you published before.
 5. Now, run the command a 4th time. You'll get the following output (since there were none left to return):

```
gcpstaging20394_student@cloudshell:~ (qwiklabs-gcp-3450558d2b043890) $  
gcloud pubsub subscriptions pull mySubscription --auto-ack  
Listed 0 items.
```

In the last section, you will learn how to pull multiple messages from a topic with a flag.

Task 4. Pub/Sub pulling all messages from subscriptions

Since you pulled all of the messages from your topic in the last example, populate `myTopic` with a few more messages.

1. Run the following commands:

```
gcloud pubsub topics publish myTopic --message "Publisher is starting to get the hang of Pub/Sub"
```

Copied!

content_copy

```
gcloud pubsub topics publish myTopic --message "Publisher wonders if all messages will be pulled"
```

Copied!

content_copy

```
gcloud pubsub topics publish myTopic --message "Publisher will have to test to find out"
```

Copied!

content_copy

2. Add a flag to your command so you can output all three messages in one request. You may have not noticed, but you have actually been using a flag this entire time: the `--auto-ack` part of the `pull` command is a flag that has been formatting your messages into the neat boxes that you see your pulled messages in.

`limit` is another flag that sets an upper limit on the number of messages to pull.

3. Wait a minute to let the topics get created. Run the pull command with the `limit` flag:

```
gcloud pubsub subscriptions pull mySubscription --limit=3
```

Copied!

content_copy

Your output should match the following:


```
DATA: Publisher is starting to get the hang of Pub/Sub  
MESSAGE_ID: 7881308064928233  
ORDERING_KEY:  
ATTRIBUTES:  
DELIVERY_ATTEMPT:  
ACK_STATUS: SUCCESS
```

```
DATA: Publisher wonders if all messages will be pulled  
MESSAGE_ID: 7881510968966871  
ORDERING_KEY:  
ATTRIBUTES:  
DELIVERY_ATTEMPT:  
ACK_STATUS: SUCCESS
```

```
DATA: Publisher will have to test to find out  
MESSAGE_ID: 7881307576501852  
ORDERING_KEY:  
ATTRIBUTES:  
DELIVERY_ATTEMPT:  
ACK_STATUS: SUCCESS
```

Now you know how to add flags to a Pub/Sub command to output a larger pool of messages. You are well on your way to becoming a Pub/Sub master.