

Responding to Cloud Logging Messages with Cloud Functions | Google Cloud Skills Boost

Qwiklabs : 16-20 minutes

GSP447



Google Cloud Self-Paced Labs

Overview

In this lab, you will learn how to use Cloud Functions to perform lightweight processing of Cloud Logging messages.

Use case: You want to track creation of Compute Engine virtual machine (VM) instances and ensure that each instance is tagged with the user that created it. Here are the high-level steps involved:

- VM instance creation generates log messages
- Cloud Logging sends the log messages to Cloud Pub/Sub as a sink destination
- Cloud Pub/Sub topic triggers a Cloud Function
- Cloud Function updates the VM metadata to include the user who created the VM

What you'll learn

- How to filter and export Cloud Logging messages to Cloud Pub/Sub
- How to trigger Cloud Functions from Pub/Sub
- How to write a Cloud Function to do simple processing
- How to update a VM's metadata

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 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.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

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 the **Lab Details** panel with the following:
 - The **Open Google Console** button
 - Time remaining
 - The temporary credentials that you must use for this lab
 - Other information, if needed, to step through this lab
2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

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

Note: If you see the **Choose an account** dialog, click **Use Another Account**.

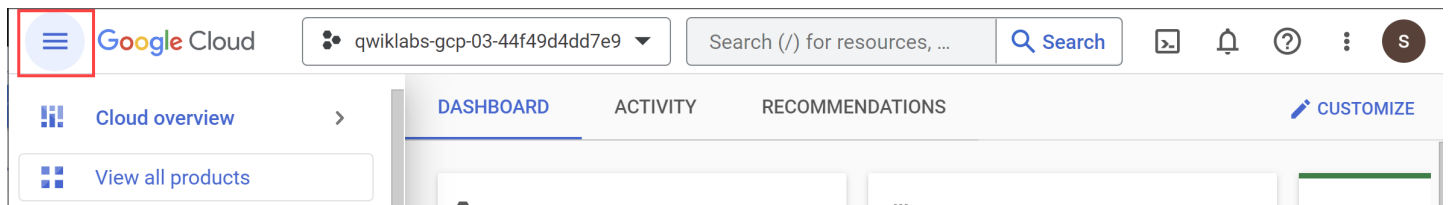
3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.

5. 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.

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

When you are connected, you are already authenticated, and the project is set to your **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 YOUR_PROJECT_ID

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

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

```
gcloud auth list
```

3. Click **Authorize**.

4. Your output should now look like this:

Output:

```
ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net 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
```

Output:

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

Example output:

```
[core] project = qwiklabs-gcp-44776a13dea667a6 Note: For full documentation of gcloud, in Google Cloud,
refer to the gcloud CLI overview guide.
```

Task 1. Create a Cloud Pub/Sub topic

Start by creating a Pub/Sub topic that will hold the log messages exported from Cloud Logging.

- In Cloud Shell, run the following to create a Cloud Pub/Sub topic named `vm-audit-logs`:

```
gcloud pubsub topics create vm-audit-logs
```

Click *Check my progress* to verify the objective. Create a Cloud Pub/Sub topic

Task 2. Generate some Compute Engine log messages

The use case is to respond to log messages generated by Compute Engine VM creation operations. Before you can respond to the log messages, you need to know what those messages look like. Now you will perform some simple Compute Engine operations to generate some sample log messages.

1. Set an environment variable for the Compute Engine zone:

```
export ZONE=us-central1-c
```

2. Now create a virtual machine instance in your selected zone. This machine is created only to generate log messages, you can use the defaults.

```
gcloud compute instances create --zone $ZONE instance-1
```

After a minute or so you will receive confirmation on the command line that the VM creation operation has completed.

3. In order to have log messages for different VM operations, which will make the log filtering more interesting, you can now stop the instance:

```
gcloud compute instances stop --zone $ZONE instance-1
```

Click *Check my progress* to verify the objective. Generate some Compute Engine log messages

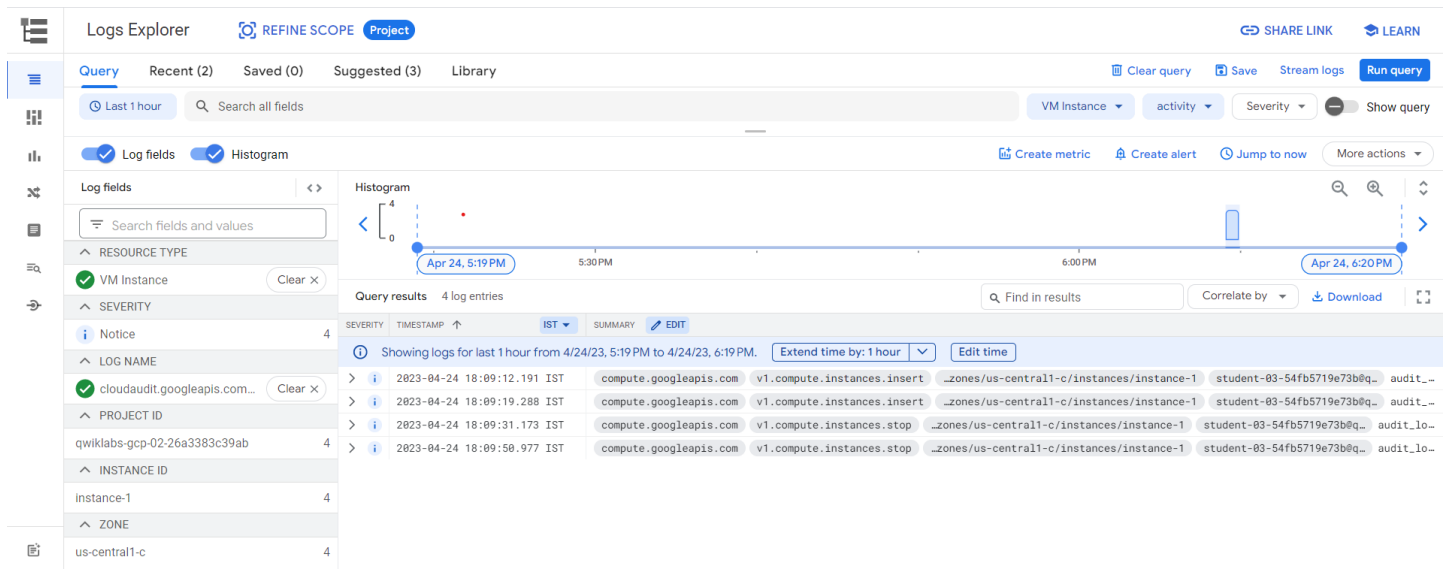
Task 3. Configure filters

Cloud Logging can export logs to Cloud Storage, BigQuery, and Cloud Pub/Sub. You can include or exclude log messages from the export using filters.

Next, you will define a filter that captures audit log messages that indicate a VM has been created.

1. In the Console, navigate to **Navigation menu > Logging > Logs Explorer**:
2. First, do some coarse-grained filtering of the log messages so you only see important messages related to Compute Engine VM instances. You will define a **Basic Filter** in the Console:
 - In the **Resource** drop-down, select **VM Instance > All instance_id** and click **Apply**.
 - In the **Log name** drop-down, search for `cloudaudit.googleapis.com` and select **activity**. Click **Apply**.

You should now see a much smaller number of messages.



The view pane is now displaying all audit activity logs for Compute Engine. You should see log messages relating to both the "insert" (create) and "stop" operations you performed in the previous step.

3. The use case also requires that only log messages for a successful "insert" operation be captured. For that you need to define an Advanced Filter.

- Expand an "insert" log message to view its full JSON. (You expand the message by clicking on the triangle-shaped icon on the left hand side of each log entry in the list)
- Expand the protoPayload element and examine the contents
- Click on the methodName: "v1.compute.instances.insert" line and then click **Show matching entries**.
- On the top of your screen the Basic Filter drop-downs have been replaced with the Advanced Filter textbox. The advanced filter conditions include the basic filter elements you set via the drop-down boxes
- Now only "insert" log messages are displayed in the view pane

4. For the use case, you only want to take action when the VM insert operation has finished. You need to add additional conditions to further filter the messages:

- Expand the last log message in the pane
- Expand the operation element
- Click on the last: true line and select "Show matching entries"
- The advanced filter conditions have been updated to include the operation.

You should now see only a single insert log message in the view pane. Your Advanced Filter should look something like this (Enable the **Show query** toggle if required):

Task 4. Configure log export

In the previous step, you defined a filter that captures audit log messages that indicate a VM has been created. Next, you will create a Cloud Logging export that sends the captured messages to the Cloud Pub/Sub topic you created earlier.

With your Advanced Filter still defined:

1. Click on **More Actions > Create Sink**.
2. **Sink Name:** instance-insert-sink, click **Next**.
3. **Select Sink Service:** Cloud Pub/Sub topic.
4. **Select a Cloud Pub/Sub topic:** vm-audit-logs (the Cloud Pub/Sub topic you created earlier as the sink destination).
5. Click **Create sink**.

Click *Check my progress* to verify the objective. Configure Stackdriver log export

Task 5. Create a simple Cloud Function

Now that you have configured certain log messages to be sent to a Cloud Pub/Sub topic, the next step is to consume those messages and take appropriate action.

Cloud Functions provide a serverless, lightweight way to respond to events including Cloud Pub/Sub messages.

Create a very simple Cloud Function initially.

1. In the Console, navigate to the **Navigation menu > Cloud Functions** page.
2. Click **Create Function**.
 - **Function Name:** addVmCreatorMetadata
 - **Trigger Type:** Cloud Pub/Sub
 - **Cloud Pub/Sub Topic:** select the topic you created earlier: vm-audit-logs, click **Save** and then click **Next**.
 - **Runtime:** Python 3.7

Note: The UI has added some default Python code to handle a Cloud Pub/Sub message.

3. Click **Deploy**. This can take a little while to complete.

Click *Check my progress* to verify the objective. Create a simple Cloud Function

Task 6. Update Cloud Function logic to add VM metadata

In the previous step, you created a simple Cloud Function that executes in response to messages on a particular Cloud Pub/Sub topic. Now you will update the Cloud Function code so that it parses some information from the Pub/Sub message and then updates the VM metadata.

This time you will use `gcloud` rather than the UI to update the Cloud Function.

1. In Cloud Shell make a new directory to hold your Cloud Function code:

```
mkdir ~/gcf-vm-metadata
```

2. Execute the below command to create a new Python file called `main.py` with the below code in the new directory; then read through the code to understand its operation:

```
cat > ~/gcf-vm-metadata/main.py <<'EOF' import base64 import json import googleapiclient.discovery def
tag_with_creator(event, context): """Adds a custom metadata entry for a new virtual machine. Triggered by
a Cloud Pub/Sub message containing a Compute Engine audit activity Stackdriver log message """
pubsub_message = base64.b64decode(event['data']).decode('utf-8') msg_json =
json.loads(pubsub_message) proto_payload = msg_json['protoPayload'] resource_name =
proto_payload['resourceName'] email = proto_payload['authenticationInfo']['principalEmail'] # compute
engine API compute = googleapiclient.discovery.build( 'compute', 'v1', cache_discovery=False) # full name
is of the form # projects/$PROJ_NAME/zones/$ZONE/instances/$INST_NAME name_tokens =
resource_name.split('/') project = name_tokens[1] zone = name_tokens[3] instance_name =
name_tokens[5] # need to get current vm metadata before we can update it vm_details =
compute.instances().get( project=project, zone=zone, instance=instance_name).execute() vm_metadata =
vm_details['metadata'] # add/replace metadata item _update_metadata(vm_metadata, 'creator', email)
response = compute.instances().setMetadata( project=project, zone=zone, instance=instance_name,
body=vm_metadata).execute() print('Updated metadata for resource %s' % resource_name) def
_update_metadata(vm_meta, key, value): """Update existing vm metadata with the supplied key/value pair.
If the key already exists, value is overwritten. """ if 'items' not in vm_meta: vm_meta['items'] = [] for item in
vm_meta['items']: if item['key'] == key: item['value'] = value return vm_meta['items'].append({'key': key,
'value': value }) EOF Note: The code interacts with the Compute Engine API, so you need to include the
appropriate client library in the set of dependencies.
```

3. Execute the following to create a new `requirements.txt` file with the required Python dependency:

```
cat > ~/gcf-vm-metadata/requirements.txt <<'EOF' google-api-python-client==1.7.4 EOF
```

4. Execute the below command to update the Cloud Function you created earlier. Note that the `--source` parameter references the directory that contains the `main.py` source file and `requirements.txt` dependency file:

```
gcloud functions deploy addVmCreatorMetadata --source ~/gcf-vm-metadata --entry-point tag_with_creator
```

Note: You are updating the `--entry-point` such that the `tag_with_creator` function will be called when the Cloud Function executes.

5. **Refresh** the Cloud Functions page in the Console and verify that the **Executed function** field has been updated to the new `tag_with_creator` value.

Cloud Functions

Functions

CREATE FUNCTION

REFRESH

LEARN

RELEASE NOTES

Filter

Filter functions

Environment

Name

↑

Last deployed

Region

Recommendation

Trigger

Runtime

Memory allocated

Executed function

Actions

✓

1st gen

[addVmCreatorMetadata](#)

Apr 24, 2023, 6:58:37 PM

us-central1

Topic: [vm-audit-logs](#)

Python 3.7

256 MB

tag_with_creator

Click *Check my progress* to verify the objective. Update Cloud Function logic to add VM metadata

Task 7. Trigger the Cloud Function by creating a VM

Earlier in the lab you created a Cloud Logging export that runs anytime a Compute Engine VM is created, and the corresponding audit log messages are published to a Cloud Pub/Sub topic. In the previous step, you added a Cloud Function to consume these messages and then update the metadata of the newly created VM.

Time to test it end-to-end!

1. In Cloud Shell, restart the virtual machine you created earlier:

```
gcloud compute instances start --zone $ZONE instance-1
```

2. Once the VM has completed starting, navigate to **Navigation menu > Logging > Logs Explorer**.
3. In the **Resource** drop-down, select **Cloud Function > addVmCreatorMetadata** and click **Apply**. You should see some log messages relating to the creation and update of the function performed in the previous step.

Look at the log messages more closely. You should **not** see any log messages indicating that the addVmCreatorMetadata Cloud Function has executed.

Remember your export only sends *insert* (VM creation) messages to the Pub/Sub topic. So restarting an existing VM will not generate a message into the topic and hence the Cloud Function does not trigger.

4. Create a new VM in Cloud Shell, which should trigger the Cloud Function:

```
gcloud compute instances create --zone $ZONE instance-2
```

5. Once the VM creation has completed, go back to refresh the Logging view pane with the existing filter intact. You should now see some log messages, indicating that the Cloud Function executed.

- Examine the log messages and cross-reference with the function code to gain an understanding of the different steps.
- Navigate to **Compute Engine > VM Instances** and click on the **instance-2** machine. Verify that the metadata for the newly created VM has been updated to include the email of the account that created it.
- Scroll down through the details and verify that the Custom Metadata section has the **creator** key and expected email address added by the Cloud Function:

Custom metadata

Key	Value
creator	student-03-54fb5719e73b@qwiklabs.net

Click **Check my progress** to verify the objective. Trigger the Cloud Function by creating a VM

Congratulations

You have now completed the Responding to Cloud Logging messages with Cloud Functions lab!

What you learned

- How to define Cloud Logging filters and export certain log messages to Cloud Pub/Sub
- How to create a Cloud Function that triggers from Pub/Sub messages
- How to update a Cloud Function with Python code that parses Cloud Logging log messages and calls the Compute Engine API to update a VM instance's metadata

Next steps / Learn more

- Learn more about [Cloud Functions](#)
- Learn more about [Google Cloud's operations suite](#)
- Learn more about [Serverless](#) on Google Cloud

End your lab

When you have completed your lab, click **End Lab**. Your account and the resources you've used are removed from the lab platform.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Manual Last Updated: April 25, 2023

Lab Last Tested: April 25, 2023

Copyright 2023 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.