Monitoring and Logging for Cloud Run Functions

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**, `qwiklabs-gcp-03-771ef8d99f04`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to qwiklabs-gcp-03-771ef8d99f04
```

`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: student-01-76290d69260b@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
```
Copied!

content_copy

Output:

```
[core]
project = qwiklabs-gcp-03-771ef8d99f04
```

Task 1. Viewing Cloud Run function logs & metrics in Cloud Monitoring

Before you collect logs and alerts, you need something to monitor. In this section, you create a Hello World Cloud Run function to monitor.

1. In the Cloud console, select **Navigation menu** (☰) > **Cloud Run**, and then click **Write a Function**.
2. Set the following:
    - **Service Name:** `helloworld`
    - **Region:** `us-east4`
    - **Runtime:** Node.js 22
    - **Authentication:** select radio button next to **Allow unauthenticated invocations**
3. Expand **Container(s), Volumes, Networking, Security** and set the following:
    - **Execution environment:** select **second generation**.
    - **Revision scaling**, set the **Maximum number of instances** to **5**.
4. Leave the rest of the fields as default. Click **Create**.

Note: A helpful popup may appear to validate the required APIs are enabled in the project. Click the **Enable** button when requested.

5. Click **Save and Redeploy**.

6. Open a new Cloud Shell window by clicking the **Activate Cloud Shell** at the top of the Google Cloud console.

7. In Cloud Shell, run the following to get a tool called **vegeta** that will let you send some test traffic to your Cloud Run function:

```
curl -LO
'https://github.com/tsenart/vegeta/releases/download/v12.12.0/vegeta_12
.12.0_linux_386.tar.gz'
```
Copied!

content_copy

8. Unpack the **vegeta** tool by running the following:

```
tar -xvzf vegeta_12.12.0_linux_386.tar.gz
```
Copied!

content_copy

9. On the Cloud Run page, click your function name, then click the helloworld **URL** for your function.

If you see `Hello World!` in the new browser tab that opens, you're up and running!

10. Set and verify the Cloud Run URL link:

```
CLOUD_RUN_URL=$(gcloud run services describe helloworld --region=us-
east4 --format='value(status.url)')
echo $CLOUD_RUN_URL
```
Copied!

content_copy

11. Now send traffic to your Cloud Run function:

```
echo "GET $CLOUD_RUN_URL" | ./vegeta attack -duration=300s -rate=200 >
results.bin
```
Copied!

content_copy

12. Keep this running in the background while you complete the next sections.

Task 2. Create a logs-based metric

Now you'll create a Distribution type logs based metric to extract the value of latency from the log entries `httpRequest.latency` field.

1. In the console, select **Navigation menu** > **View All Products** > **Observability** > **Logging** > **Logs Explorer**. The Cloud Logging opens in the console.

2. To look at just the logs from your Cloud Run function, in the **All resources** dropdown, select **Cloud Run Revision** > **helloWorld** then click **Apply**.

3. Click **Run query**.

4. In Actions dropdown click **Create metric**.

5. In the Create log-based metric form:

- Change the Metric Type to **Distribution**.
- In Log-based metric name enter **CloudRunFunctionLatency-Logs**.
- Enter `httpRequest.latency` for Field name.
  The log-based metric should look like this:

6. Click **Create metric**.

Now you'll see your user-defined metric added to your Logs-based Metrics page.

Task 3. Metrics Explorer

Next, you'll use Metrics Explorer to look at the data for your Cloud Run function.

Create a Monitoring Metrics Scope

Set up a Monitoring Metrics Scope that's tied to your Google Cloud Project. The following steps create a new account that has a free trial of Monitoring.

- In the Cloud Console, click **Navigation menu** (☰) > View All Products > Observability > **Monitoring**.
  When the Monitoring **Overview** page opens, your metrics scope project is ready.

1. In the left menu, click **Metrics explorer**.

2. Click on **Select a metric** dropdown. Deselect the **Active** button.

3. Type `CloudRunFunctionLatency-Logs` in filter by resource or metric name and then select **Cloud Run Revision > Logs-based metric > Logging/user/CloudRunFunctionLatency-Logs** from the suggested metrics and click **Apply**.

4. On the top right corner change the **widget type** to **Stacked bar chart** using the dropdown menu.

5. Explore other graph options and try a different metric. For example, click your current **Cloud Run Revision** metric to open the dropdown, select **Cloud Run Revision > Request Count**, and change the widget type to **Stacked area chart**.

6. Continue to explore and experiment. For example, go back to the **Cloud Run Revision** metric and change the `Aggregation` to the **95th percentile**. Select the widget type **Line chart**.

Task 4. Create charts on the Monitoring Overview window

Creating charts on the Monitoring Overview window is a great way to track metrics that are important to you. In this section, you set up the same charts you created in the previous section, but now they'll be saved into the Monitoring Overview window.

1. In the left menu, click **Dashboards**.

2. Click **Create Custom Dashboard**.

3. Click on **Add widget**.

4. Under Visualization, select **Stacked bar**.

5. Under **Select a metric** dropdown select the default **Cloud Run Revision > Request Count** metric to open the dropdown and change the metric.
   Click **Apply** on the top right.

6. Click **Add widget**. Under Visualization, select **Heatmap**.

7. Start typing `Cloud Run Revision` into the **Select a metric** dropdown, and then select **Cloud Run Revision > Logs-based metric > Logging/user/CloudRunFunctionLatency-Logs** from the suggested metrics and click **Apply**.

Note: If CloudRunFunctionLatency-Logs metric is not visible in the dropdown, uncheck `Active`.

8. Click **Add widget**. Under Visualization, select **Line**.

9. Start typing `Cloud Run Revision` into the **Select a metric** dropdown, and then select **Cloud Run Revision** > **Request_latency** from the suggested metrics. Set the **Aggregation** to **Mean** and click **Apply**.

10. Click **Add widget**. Under Visualization, select **Stacked bar**.

11. Start typing `Cloud Run Revision` into the **Select a metric** dropdown, and then select **Cloud Run Revision** > **Container** > **Container CPU Allocation** from the suggested metrics. Click **Apply**.

By default, the charts name themselves after the metric you're using, but you can rename them.

12. Click the Dashboard name at the top it should start with `New Dashboard` `-` and rename it to `Cloud Run Function Custom Dashboard`.

For a quick reference, to see these charts click **Dashboards** in the left panel of the Monitoring page.