Cloud Logging on Kubernetes Engine

GSP483

Overview

Cloud Logging can be used aggregate logs from all Google Cloud resources, as well as any custom resources on other platforms, to allow for one centralized store for all logs and metrics. Logs are aggregated and then viewable within the provided Cloud Logging UI. They can also be exported to Sinks to support more specialized of use cases. Currently, Cloud Logging supports exporting to the following sinks:

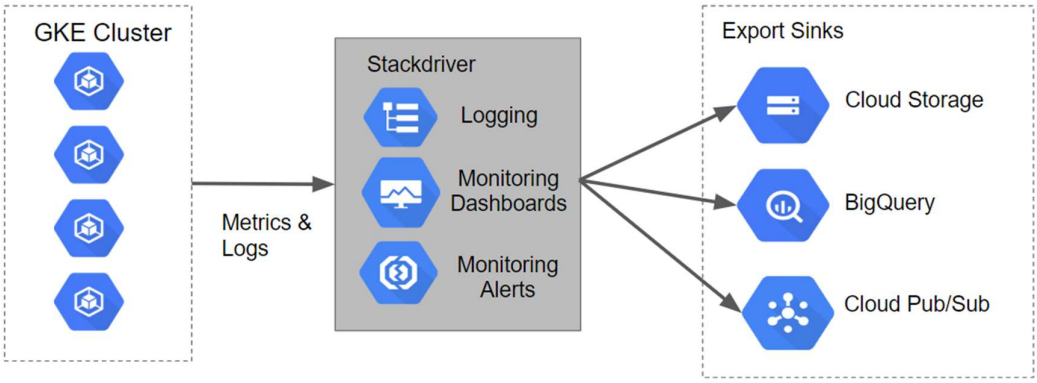
- Cloud Storage
- Pub/Sub
- BigQuery

In this lab you will deploy a sample application to Kubernetes Engine that forwards log events to Cloud Logging using Terraform, a declarative Infrastructure as Code tool that enables configuration files to automate the deployment and evolution of infrastructure in the cloud. The configuration will also create a Cloud Storage bucket and a BigQuery dataset for exporting log data to.

This lab was created by GKE Helmsman engineers to give you a better understanding of Cloud Logging. You can view this demo by running `gsutil cp -r gs://spls/gke-binary-auth/* .` and `cd gke-binary-auth-demo` command in cloud shell. We encourage any and all to contribute to our assets!

Architecture

The Terraform configurations are going to build a Kubernetes Engine cluster that will generate logs and metrics that can be ingested by Stackdriver. The scripts will also build out Logging Export Sinks for Cloud Storage, BigQuery, and Cloud Pub/Sub. The diagram of how this will look along with the data flow can be seen in the following graphic:



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

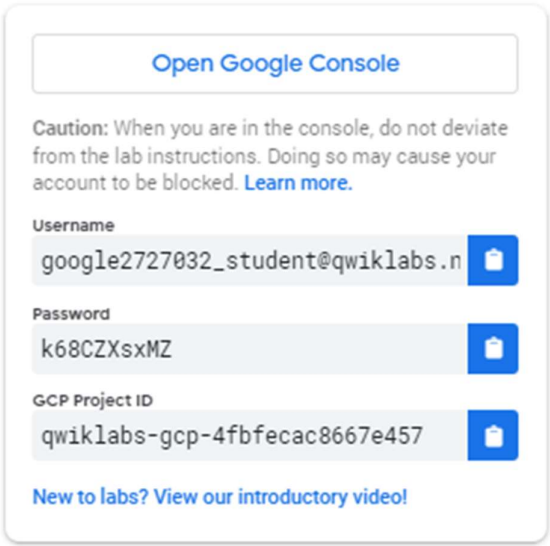
To complete this lab, you need:

- 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.

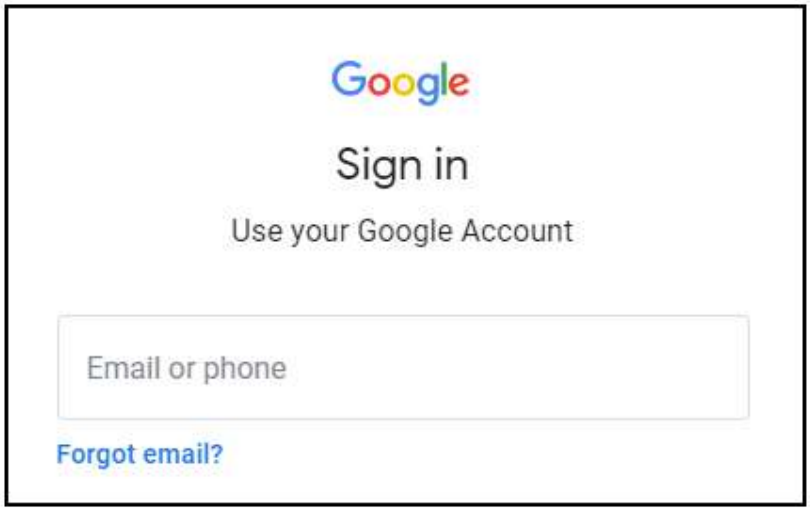
Note: If you are using a Pixelbook, open an Incognito window to run this lab.

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.

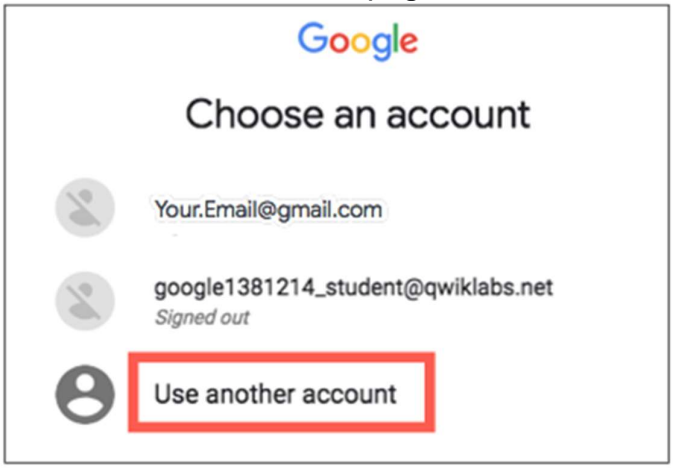


2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



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.

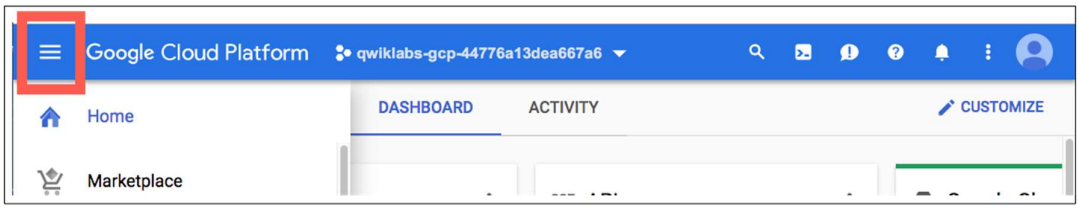
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

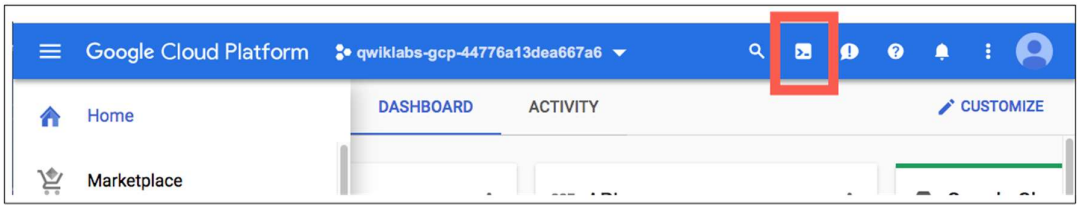
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



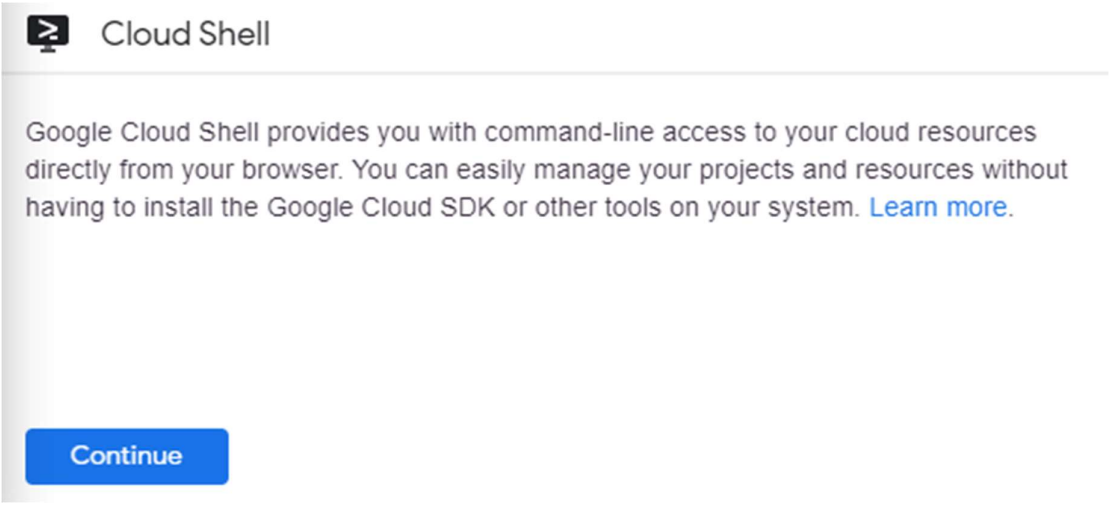
Activate Cloud Shell

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.

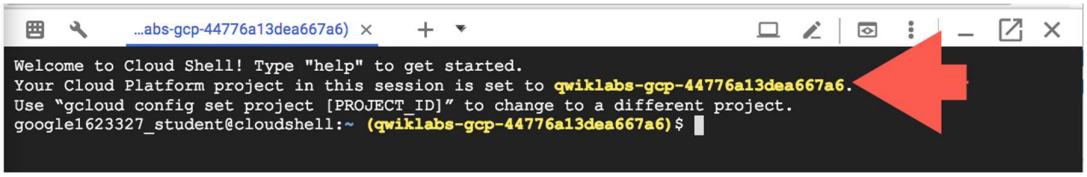
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



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:



`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.

Clone demo

In Cloud Shell, click **Open Editor** in the top ribbon then click **Open in new window**:

Run the following command to set your GCP project ID,
replacing `<YOUR_PROJECT_ID>` with your Qwiklabs Project ID by clicking **Open Terminal** in Cloud Shell.

```
gcloud config set project <YOUR_PROJECT_ID>
```

Now clone the resources needed for this lab:

```
git clone https://github.com/GoogleCloudPlatform/gke-logging-sinks-demo
```

Now change your the directory for this demo:

```
cd gke-logging-sinks-demo
```

Set your region and zone

Certain Compute Engine resources live in regions and zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones.

Learn more about regions and zones and see a complete list in Regions & Zones documentation.
Run the following to set a region and zone for your lab (you can use the region/zone that's best for you):

```
gcloud config set compute/region us-central1
gcloud config set compute/zone us-central1-a
```

Deployment

Following the principles of Infrastructure as Code and Immutable Infrastructure, Terraform supports the writing of declarative descriptions of the desired state of infrastructure. When the descriptor is applied, Terraform uses GCP APIs to provision and update resources to match. Terraform compares the desired state with the current state so incremental changes can be made without deleting everything and starting over. For instance, Terraform can build out GCP projects and compute instances, etc., even set up a Kubernetes Engine cluster and deploy applications to it. When requirements change, the descriptor can be updated and Terraform will adjust the cloud infrastructure accordingly. This lab will start up a Kubernetes Engine cluster and deploy a simple sample application to it. By default, Kubernetes Engine clusters in GCP are provisioned with a pre-configured Fluentd-based collector that forwards logs to Cloud Logging. Interacting with the sample app will produce logs that are visible in the Cloud Logging and other log event sinks.

Update the provider.tf file

Remove the provider version for the Terraform from the `provider.tf` script file.

From the left-hand menu, open the file `/gke-logging-sinks-demo/terraform/provider.tf`.

Set the version to `~> 2.19.0`. After modification your `provider.tf` script file should look like:

```
....
provider "google" {
  project = var.project
  version = "~> 2.19.0"
}
```

Save and close the file.

Deploying the cluster

There are three Terraform files provided with this lab example. The first one, `main.tf`, is the starting point for Terraform. It describes the features that will be used, the resources that will be manipulated, and the outputs that will result. The second file is `provider.tf`, which indicates which cloud provider and version will be the target of the Terraform commands--in this case GCP.

The final file is `variables.tf`, which contains a list of variables that are used as inputs into Terraform. Any variables referenced in the `main.tf` that do not have defaults configured in `variables.tf` will result in prompts to the user at runtime.

You will make one small change to `main.tf`. From the left-hand menu, open the file `/gke-logging-sinks-demo/terraform/main.tf`.

Scroll down to line 106 and find the "Create the Stackdriver Export Sink for Cloud Storage GKE Notifications" section.

Change the filter's `resource.type` from **container** to **k8s_container**.

Do the same for the bigquery-sink on line 116. Ensure that these two export sync sections look like the following before moving on:

```
105
106 // Create the Stackdriver Export Sink for Cloud Storage GKE Notifications
107 resource "google_logging_project_sink" "storage-sink" {
108   name        = "gke-storage-sink"
109   destination = "storage.googleapis.com/${google_storage_bucket.gke-log-bucket.name}"
110   filter      = "resource.type = k8s_container"
111
112   unique_writer_identity = true
113 }
114
115 // Create the Stackdriver Export Sink for BigQuery GKE Notifications
116 resource "google_logging_project_sink" "bigquery-sink" {
117   name        = "gke-bigquery-sink"
118   destination = "bigquery.googleapis.com/projects/${var.project}/datasets/${google_bigquery_dataset.gke-bigquery-dataset.dataset_id}"
119   filter      = "resource.type = k8s_container"
120
121   unique_writer_identity = true
122 }
123
```

Save and close the file.

Now run the following command to build out the executable environment using the `make` command:

```
make create
```

Note: If you get deprecation warnings related to the zone variable, please ignore it and move forward in the lab.

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully deployed necessary infrastructure with Terraform, you will see an assessment score.

Validation

If no errors are displayed during deployment, after a few minutes you should see your Kubernetes Engine cluster in the Cloud Console.

Go to **Navigation menu** > **Kubernetes Engine** > **Clusters** to see the cluster with the sample application deployed.

To validate that the demo deployed correctly, run:

```
make validate
```

Your output will look like this:

```
kubeconfig entry generated for stackdriver-logging.
Step 1 of the validation passed. App is deployed.
App is available at: http://35.224.159.191:8080
Step 2 of the validation passed. App handles requests.
```

Now that the application is deployed to Kubernetes Engine you can generate log data and use Cloud Logging and other tools to view it.

Generating Logs

The sample application that Terraform deployed serves up a simple web page. Each time you open this application in your browser the application will publish log events to Cloud Logging. When you refresh the page a few times to produce several log events.

To get the URL for the application page, perform the following steps:

1. In the Cloud console, from the **Navigation menu**, go to the Networking section and click on **Network services**.
2. On the default **Load balancing** page, click on the name of the TCP load balancer that was set up.
3. On the **Load balancer details** page the top section labeled **Frontend**.
4. In the Frontend, copy the `IP:Port` URL value. Open a new browser and paste the URL. The browser should return a screen that looks similar to the following:

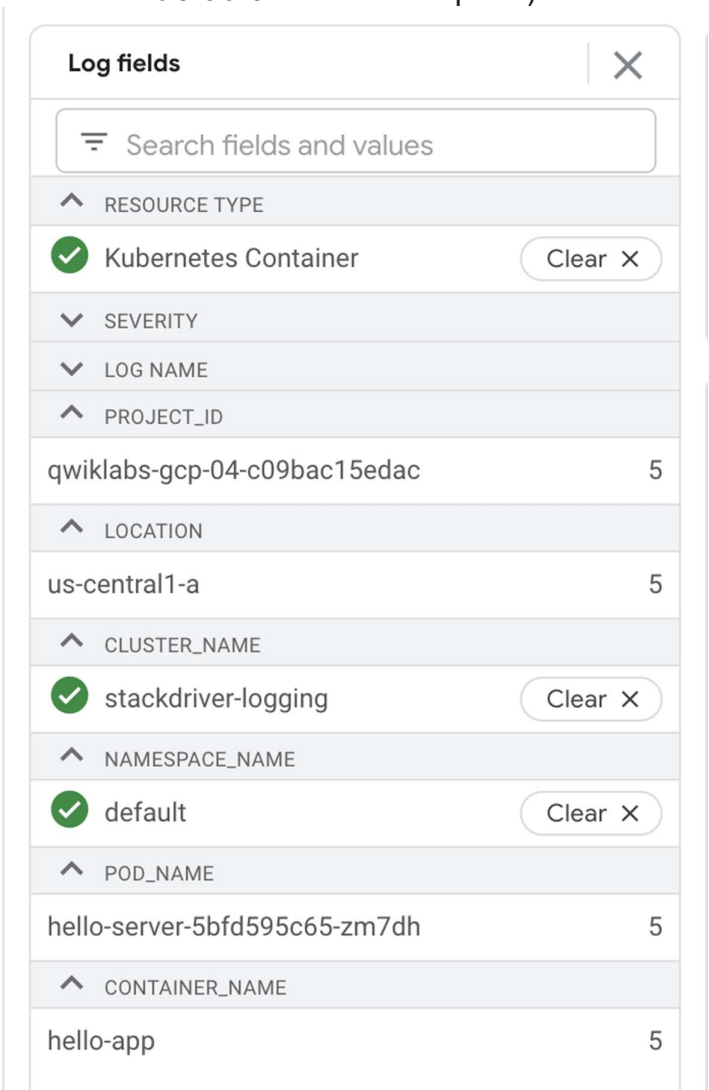
```
Hello, world!
Version: 1.0.0
Hostname: hello-server-66cb56b679-xsfzw
```

Logs in Cloud Logging

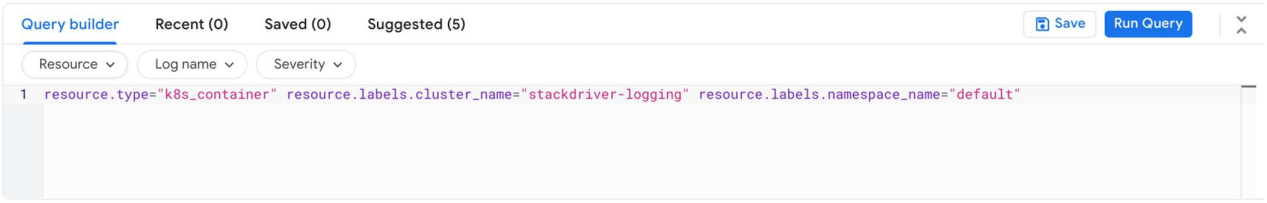
Cloud Logging provides a UI for viewing log events. Basic search and filtering features are provided, which can be useful when debugging system issues. Cloud Logging is best suited to exploring more recent log events. Users requiring longer-term storage of log events should consider some of the tools you'll explore in the following sections.

To access the cloud Logging console perform the following steps:

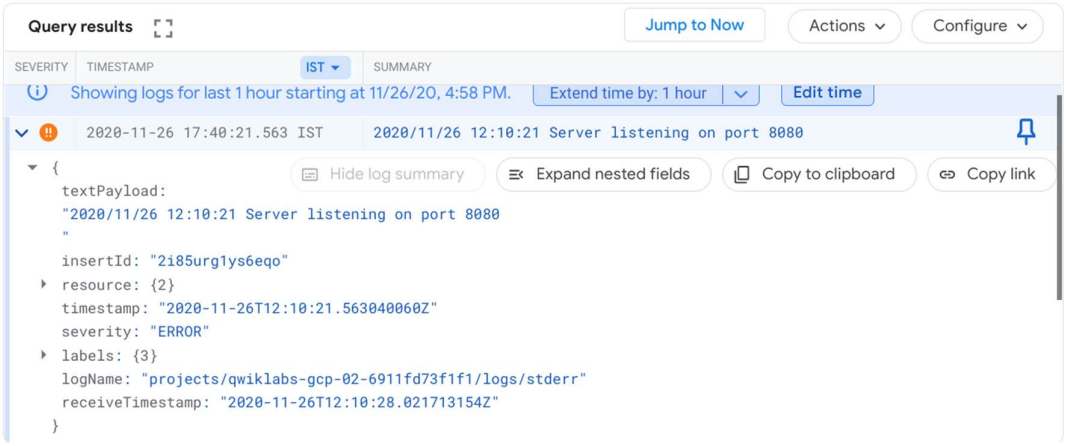
1. In the Cloud Console, from the **Navigation menu**, in the Operations section, click on **Logging**.
2. On this page, select the `Resource type` to **Kubernetes Container**, `cluster_name` to **stackdriver-logging**, `namespace_name` to **Default** (**stackdriver-logging** is the cluster and **default** is the namespace).



3. Now click **Run Query**.



4. In Query results, you can expand the bulleted log items to view more details about a log entry.



On the Logging console, you can build queries using Query builder, or try out various features like log fields, time zone, etc.

Viewing Log Exports

The Terraform configuration built out two Log Export Sinks. To view the sinks perform the following steps:

1. You should still be on the **Logging** page.

2. In the left navigation menu, click on **Logs Router**.

3. You should see four Sinks in the list of log exports.

4. You can edit/view these sinks by clicking on the context menu (three dots) to the right of a sink and selecting the **Edit sink** option.

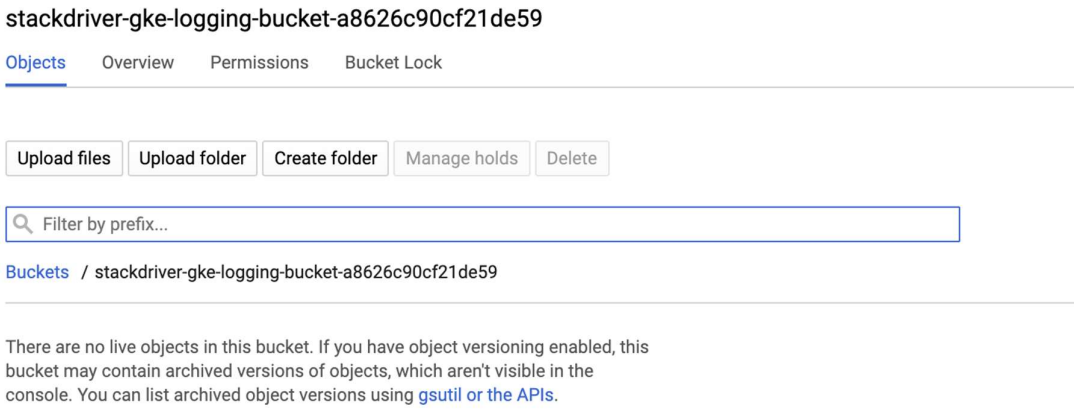
5. Additionally, you could create additional custom export sinks by clicking on the **Create Sink** option in the top of the navigation window.

Logs in Cloud Storage

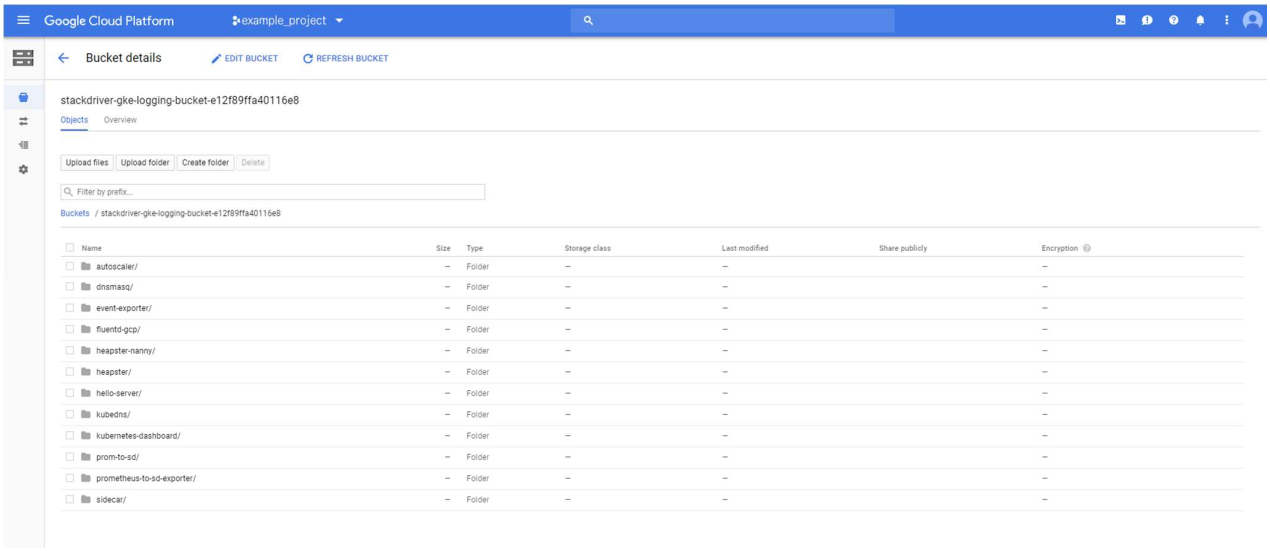
Log events can be stored in Cloud Storage, an object storage system suitable for archiving data. Policies can be configured for Cloud Storage buckets that, for instance, allow aging data to expire and be deleted while more recent data can be stored with a variety of storage classes affecting price and availability.
The Terraform configuration created a Cloud Storage Bucket named stackdriver-gke-logging- to which logs will be exported for medium to long-term archival. In this example, the Storage Class for the bucket is defined as Nearline because the logs should be infrequently accessed in a normal production environment (this will help to manage the costs of medium-term storage). In a production scenario, this bucket may also include a lifecycle policy that moves the content to Coldline storage for cheaper long-term storage of logs.

To access the logs in Cloud Storage perform the following steps:

1. In the Cloud Console from the **Navigation menu**, click **Storage**.
2. Find the Bucket with the name `stackdriver-gke-logging-<random-Id>`, and click on the name.
3. Unfortunately, it takes a while for sinks to propagate to Cloud Storage so you probably will not see any log details in your bucket:



If you come back to the bucket towards the end of your lab you might see folders corresponding to pods running in the cluster (e.g. autoscaler, dnsmasq, etc.).



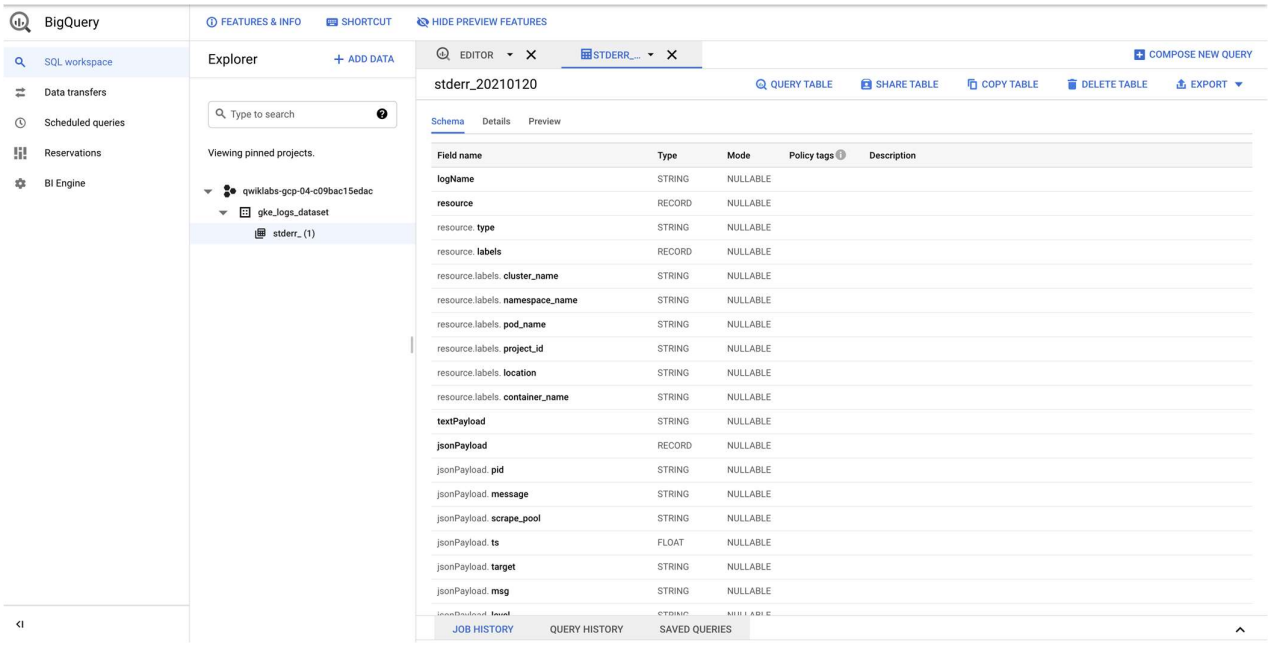
You can click into any of the folders to browse specific log details like heapster, kubedns, sidecar, etc.

Logs in BigQuery

Log events can be configured to be published to BigQuery, a data warehouse tool that supports fast, sophisticated, querying over large data sets.
The Terraform configuration will create a BigQuery DataSet named `gke_logs_dataset`. This dataset will be set up to include all Kubernetes Engine related logs for the last hour (by setting a Default Table Expiration for the dataset). Kubernetes Engine container logs will be pushed to the dataset.
To access the logs in BigQuery, perform the following steps:

Note: The BigQuery Export is not populated immediately. It may take a few moments for logs to appear.

1. From the **Navigation menu**, in the Big Data section, click on **BigQuery**.
2. In the left menu, click on your project name. You should see a dataset named **gke_logs_dataset**. Expand this dataset to view the tables that exist (**Note:** The dataset is created immediately, but the tables are generated as logs are written and new tables are needed).
3. Click on one of the tables to view the table details.
4. Review the schema of the table to note the column names and their data types. This information can be used in the next step when you query the table to look at the data.



5. Click on **Query Table** towards the top right to perform a custom query against the table.
6. This adds a query to the Query Editor, but it has a syntax error.
7. Edit the query to add an asterisk (*) after **Select** to pull in all details from the current table. **Note:** A `Select *` query is generally very expensive and not advised. For this lab the dataset is limited to only the last hour of logs, so the overall dataset is relatively small.
8. Click **Run** to execute the query and return some results from the table.

The results window should display some rows and columns. You can scroll through the various rows of data that are returned. If you want, execute some custom queries that filter for specific data based on the results that were shown in the original query.

Test Completed Task

Click **Check my progress** to verify your performed task. If BigQuery sink written logs in BigQuery dataset, you will see an assessment score.

Teardown

Qwiklabs will take care of shutting down all the resources used for this lab, but here's what you would need to do to clean up your own environment to save on cost and to be a good cloud citizen:

```
make teardown
```

Since Terraform tracks the resources it created, it is able to tear them all down.

Troubleshooting for your production environment

The install script fails with a `Permission denied` when running Terraform. The credentials that Terraform is using do not provide the necessary permissions to create resources in the selected projects. Ensure that the account listed in `gcloud config list` has necessary permissions to create resources. If it does, regenerate the application default credentials using `gcloud auth application-default login`.

Cloud Storage Bucket not populated Once the Terraform configuration is complete the Cloud Storage Bucket will be created, but it is not always populated immediately with log data from the Kubernetes Engine cluster. Give the process some time because it can take up to 2 to 3 hours before the first entries start appearing (https://cloud.google.com/logging/docs/export/using_exported_logs).
No tables created in the BigQuery dataset Once the Terraform configuration is complete the BigQuery Dataset will be created but it will not always have tables created in it by the time you go to review the results. The tables are rarely populated immediately. Give the process some time (minimum of 5 minutes) before determining that something is not working properly.

Congratulations

Finish Your Quest

This self-paced lab is part of the Qwiklabs Google Cloud's Operations Suite on GKE and Google Cloud Logging Quests. A Quest is a series of related labs that form a learning path. Completing a Quest earns you a badge to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in either Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

Next Steps / Learn More

- Kubernetes Engine Logging
- Viewing Logs
- Advanced Logs Filters
- Overview of Logs Exports
- Procesing Logs at Scale Using Cloud Dataflow
- Terraform Google Cloud Provider

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 20, 2021
Lab Last Tested: January 20, 2021