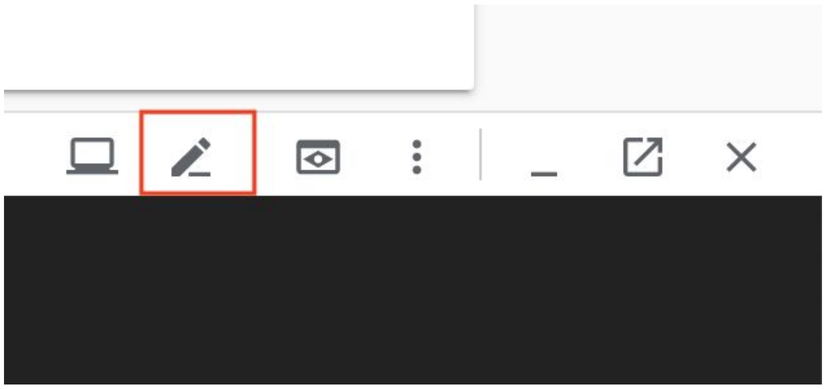
Cloud Logging on Kubernetes Engine

- **Clone demo**

In Cloud Shell, open the code editor by clicking the icon in the top ribbon:



Run the following command to set your GCP project ID, replacing `<YOUR_PROJECT_ID>` with your Qwiklabs Project ID.

```
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

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

1- 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.

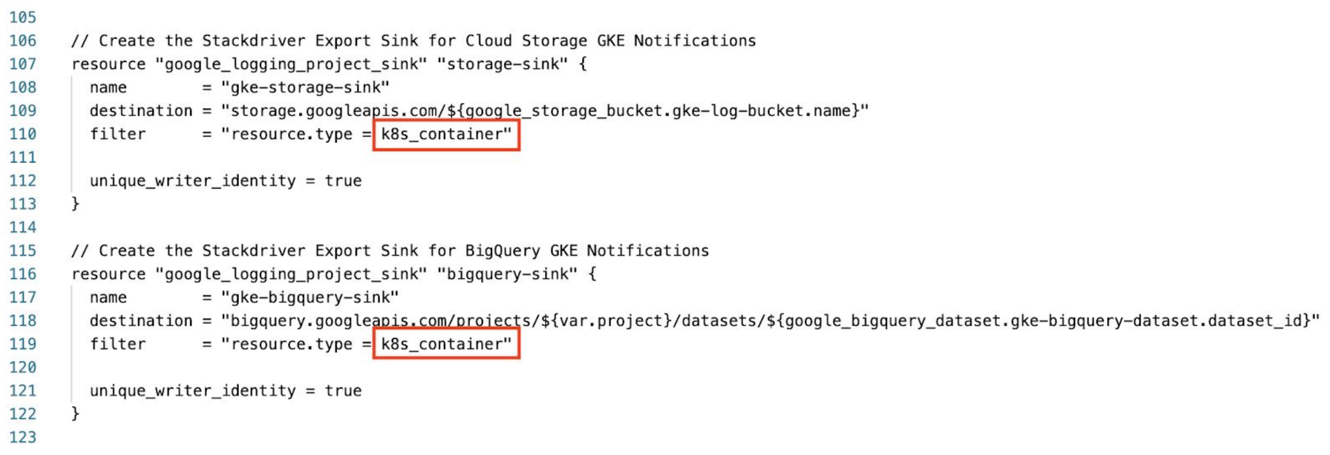
- **Deploying the cluster**

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.

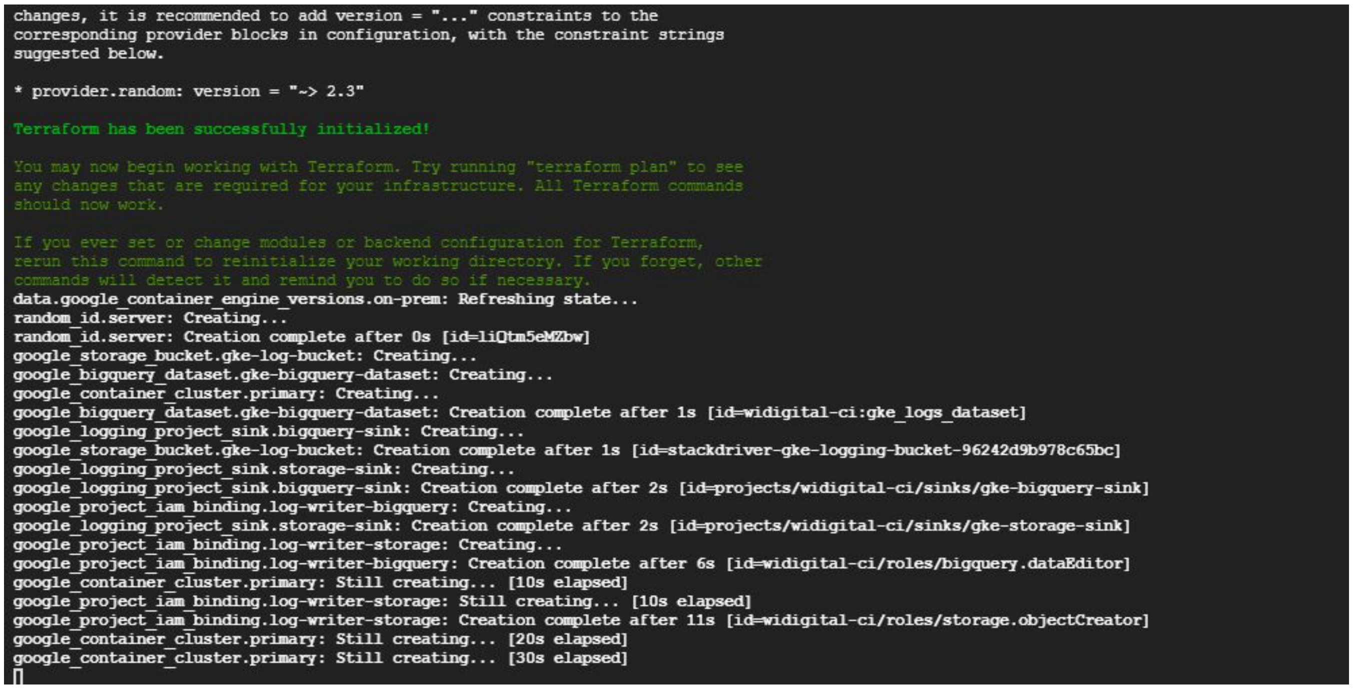
Change the filter's `resource.type` from container to k8s_container.
Do the same for the bigquery-sink on line 116. Ensure that these two export sync sections look like the following before moving on:



```
// Create the Stackdriver Export Sink for Cloud Storage GKE Notifications
resource "google_logging_project_sink" "storage-sink" {
  name        = "gke-storage-sink"
  destination = "storage.googleapis.com/${google_storage_bucket.gke-log-bucket.name}"
  filter      = "resource.type = k8s_container"

  unique_writer_identity = true
}

// Create the Stackdriver Export Sink for BigQuery GKE Notifications
resource "google_logging_project_sink" "bigquery-sink" {
  name        = "gke-bigquery-sink"
  destination = "bigquery.googleapis.com/projects/${var.project}/datasets/${google_bigquery_dataset.gke-bigquery-dataset.dataset_id}"
  filter      = "resource.type = k8s_container"

  unique_writer_identity = true
}
```

Save and close the file.

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



```
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.random: version = "~> 2.3"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
data.google_container_engine_versions.on-prem: Refreshing state...
random_id.server: Creating...
random_id.server: Creation complete after 0s [id=liQtm5eMZbw]
google_storage_bucket.gke-log-bucket: Creating...
google_bigquery_dataset.gke-bigquery-dataset: Creating...
google_container_cluster.primary: Creating...
google_bigquery_dataset.gke-bigquery-dataset: Creation complete after 1s [id=widigital-ci:gke_logs_dataset]
google_logging_project_sink.bigquery-sink: Creating...
google_storage_bucket.gke-log-bucket: Creation complete after 1s [id=stackdriver-gke-logging-bucket-96242d9b978c65bc]
google_logging_project_sink.storage-sink: Creating...
google_logging_project_sink.bigquery-sink: Creation complete after 2s [id=projects/widigital-ci/sinks/gke-bigquery-sink]
google_project_iam_binding.log-writer-bigquery: Creating...
google_logging_project_sink.storage-sink: Creation complete after 2s [id=projects/widigital-ci/sinks/gke-storage-sink]
google_project_iam_binding.log-writer-storage: Creating...
google_project_iam_binding.log-writer-bigquery: Creation complete after 6s [id=widigital-ci/roles/bigquery.dataEditor]
google_container_cluster.primary: Still creating... [10s elapsed]
google_project_iam_binding.log-writer-storage: Still creating... [10s elapsed]
google_project_iam_binding.log-writer-storage: Creation complete after 11s [id=widigital-ci/roles/storage.objectCreator]
google_container_cluster.primary: Still creating... [20s elapsed]
google_container_cluster.primary: Still creating... [30s elapsed]
```

Validation:

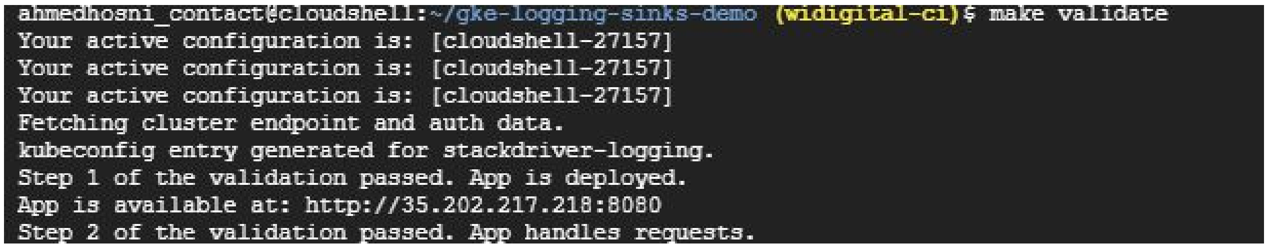
If no errors are displayed during deployment, after a few minutes you should see your Kubernetes Engine cluster in the Cloud Console.

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:



```
ahmedhosni_contact@cloudshell:~/gke-logging-sinks-demo (widigital-ci)$ make validate
Your active configuration is: [cloudshell-27157]
Your active configuration is: [cloudshell-27157]
Your active configuration is: [cloudshell-27157]
Fetching cluster endpoint and auth data.
kubeconfig entry generated for stackdriver-logging.
Step 1 of the validation passed. App is deployed.
App is available at: http://35.202.217.218:8080
Step 2 of the validation passed. App handles requests.
```

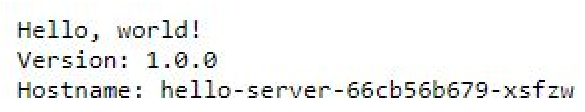
Now that the application is deployed to Kubernetes Engine you can generate log data and use Cloud Logging and other tools to view it.

2- 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:



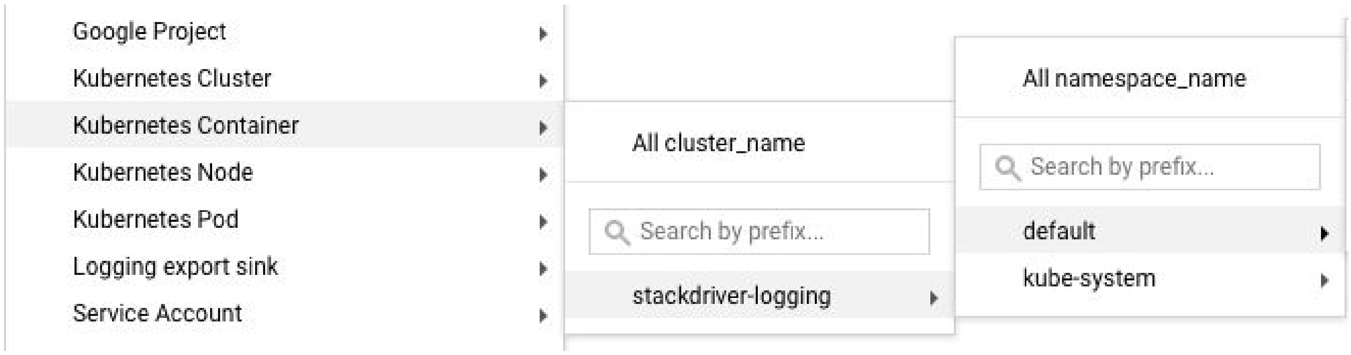
```
Hello, world!
Version: 1.0.0
Hostname: hello-server-66cb56b679-xsfzw
```

3- 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.

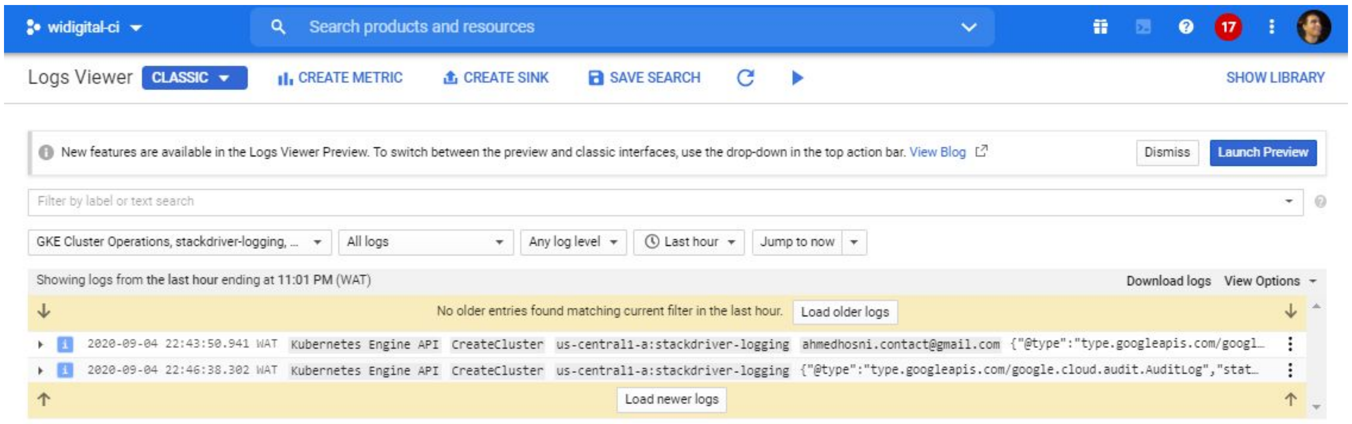
To access the cloud Logging console perform the following steps:

1. In the Cloud Console, from the **Navigation menu**, in the Operations section, click on **Logging**.
2. On this page change the Audited Resource filter to **Kubernetes Container** > **stackdriver-logging** > **default** (**stackdriver-logging** is the cluster and **default** is the namespace).



3. On this screen, you can expand the bulleted log items to view more details about a log entry.

On the Logging console, you can perform any type of text search, or try out the various filters by log type, log level, timeframe, etc.



New features are available in the Logs Viewer Preview. To switch between the preview and classic interfaces, use the drop-down in the top action bar. View Blog

Showing logs from the last hour ending at 11:01 PM (WAT)

No older entries found matching current filter in the last hour.

```
2020-09-04 22:43:50.941 WAT  Kubernetes Engine API  CreateCluster  us-central1-a:stackdriver-logging  ahmedhosni.contact@gmail.com  {"@type":"type.googleapis.com/googl…
2020-09-04 22:46:38.302 WAT  Kubernetes Engine API  CreateCluster  us-central1-a:stackdriver-logging  {"@type":"type.googleapis.com/google.cloud.audit.AuditLog","stat…
```

4- 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 two 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.

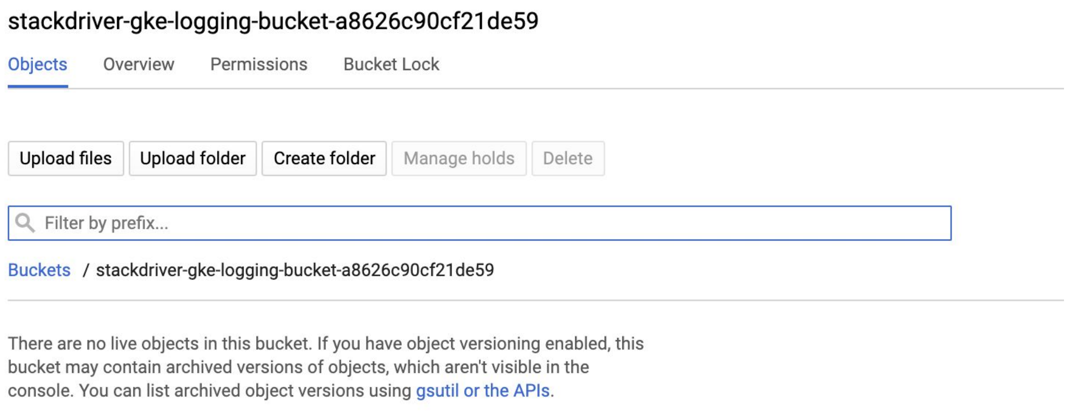
5- Logs in Cloud Storage

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.

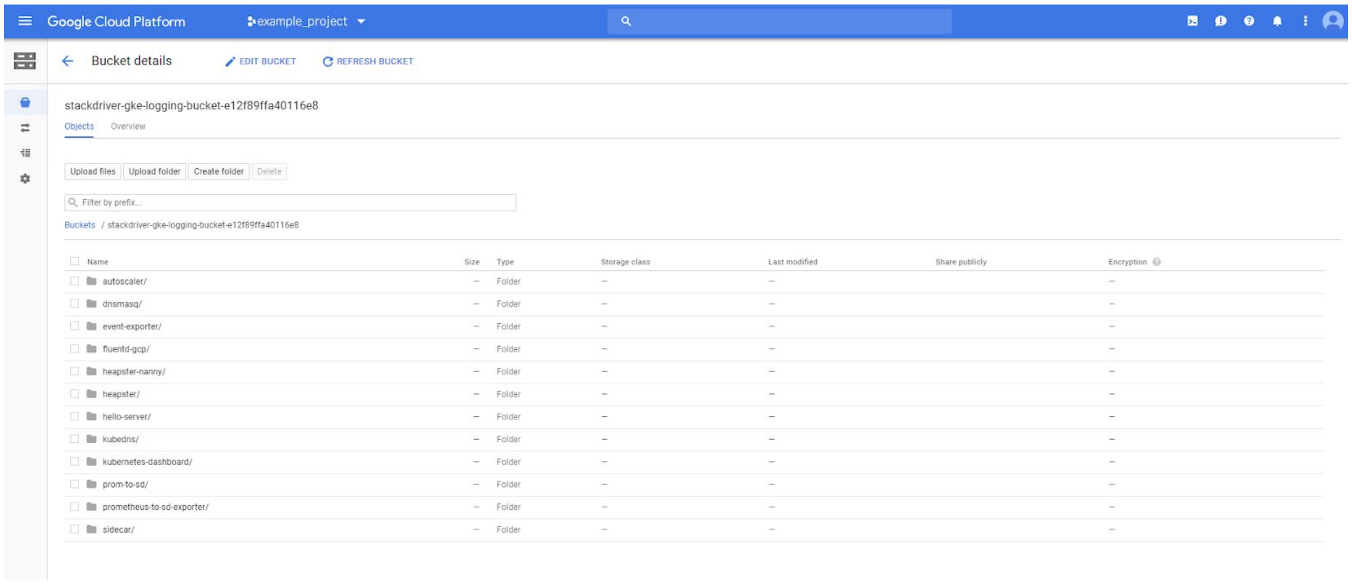
To access the logs in Cloud Storage perform the following steps:

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:


stackdriver-gke-logging-bucket-a8626c90cf21de59

There are no live objects in this bucket. If you have object versioning enabled, this bucket may contain archived versions of objects, which aren't visible in the console. You can list archived object versions using gsutil or the APIs.

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.).



stackdriver-gke-logging-bucket-e12f89ffa40116e8

Buckets / stackdriver-gke-logging-bucket-e12f89ffa40116e8

Name	Size	Type	Storage class	Last modified	Share publicly	Encryption
autoscaler/	–	Folder	–	–		–
dnsmasq/	–	Folder	–	–		–
event-exporter/	–	Folder	–	–		–
fluentd-gcp/	–	Folder	–	–		–
heapster-nanny/	–	Folder	–	–		–
heapster/	–	Folder	–	–		–
hello-server/	–	Folder	–	–		–
kubedns/	–	Folder	–	–		–
kubernetes-dashboard/	–	Folder	–	–		–
prom-to-sd/	–	Folder	–	–		–
prometheus-to-sd-exporter/	–	Folder	–	–		–
sidecar/	–	Folder	–	–		–

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

6- 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.

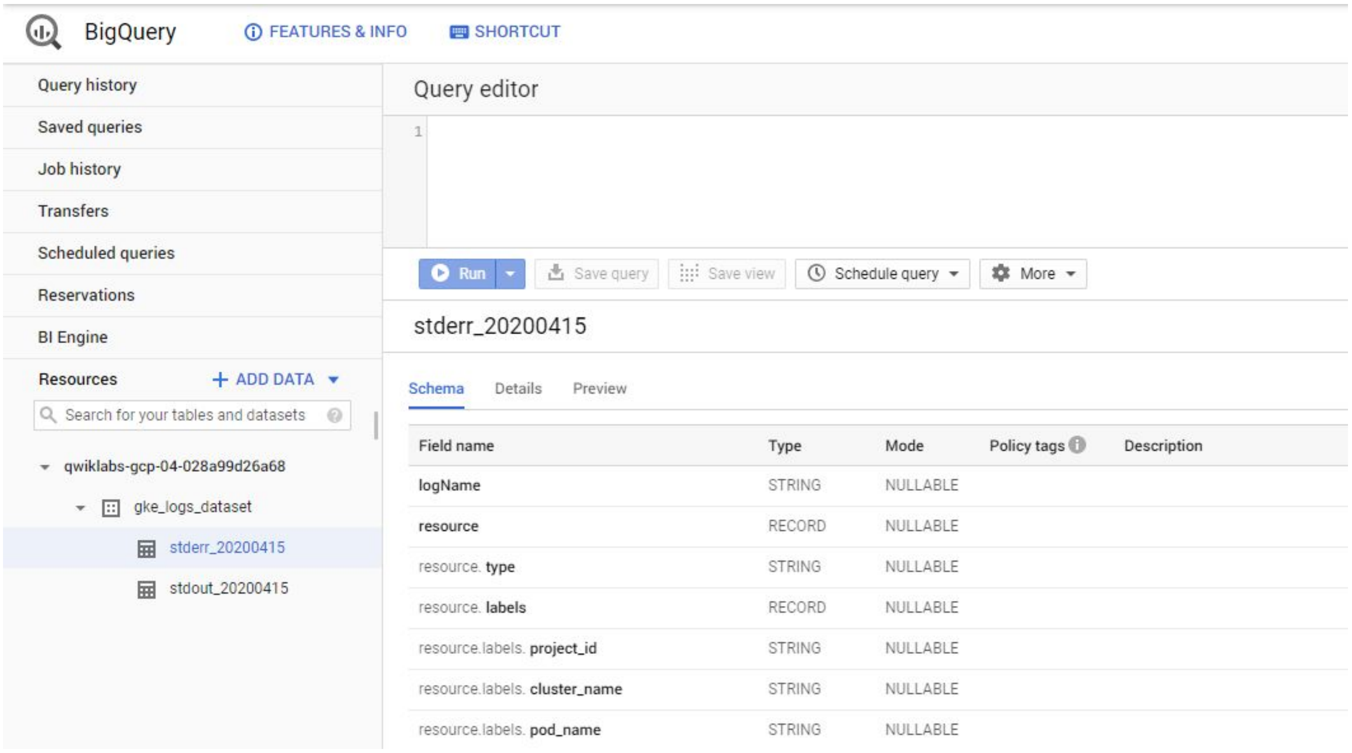
The Terraform configuration will create a BigQuery DataSet named `gke_logs_dataset`. This dataset will be set up to include all Kubernetes Engine related logs for the last hour (by setting a Default Table Expiration for the dataset). Kubernetes Engine container logs will be pushed to the dataset.

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.



Field name	Type	Mode	Policy tags	Description
logName	STRING	NULLABLE		
resource	RECORD	NULLABLE		
resource. type	STRING	NULLABLE		
resource. labels	RECORD	NULLABLE		
resource.labels. project_id	STRING	NULLABLE		
resource.labels. cluster_name	STRING	NULLABLE		
resource.labels. pod_name	STRING	NULLABLE		

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.

7- 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.