


# Monitor and Log with Google Cloud Observability

## 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 quiklabs-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@quiklabs.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. Configure Cloud Monitoring

1. Your first task is to enable Cloud Monitoring for your project.  
A basic Cloud Monitoring dashboard, called **Media\_Dashboard**, will be made available to you automatically, but you have to enable Cloud Monitoring in your project before you will be able to access this dashboard.

```
METRIC_NAME=large_input_size
```

```
ZONE=$(gcloud compute instances list --filter="name=video-queue-monitor" --format "get(zone)" |  
awk -F/ '{print $NF}')
```

```
REGION=${ZONE::-2}
```

```
ID=$(gcloud compute instances describe video-queue-monitor --zone $ZONE --format="value(id)")
```

```
cat > my-start-up-script.sh <<EOF
```

```
#!/bin/bash
```

```
REGION=us-east1
```

```
ZONE=us-east1-d
```

```
PROJECT_ID=
```

## Install Video queue Go source code

```
cd /work/go
```

```
mkdir video
```

```
gsutil cp gs://splis/gsp338/video_queue/main.go /work/go/video/main.go
```

## Task 2. Configure a Compute Instance to generate Custom Cloud Monitoring metrics

1. Your next task is to confirm that the monitoring service that checks the length of the video processing queue is working correctly.

The monitoring service creates a custom

metric, `opencensus/my.videoservice.org/measure/input_queue_size`, that allows you to monitor the state of the Jooli Inc.'s video processing queue. This custom metric is created and written to by a Go application that runs on a Compute Instance called **video-queue-monitor**.

The **video-queue-monitor** Compute Instance has been deployed for you and uses a startup script to install and launch the input queue monitoring Go application. This application was tested fully in a development environment but the configuration in your Compute Instance has not been finalized. The Go application will not write custom metric data until the application is correctly configured by the startup script.

2. You must modify the startup script for the **video-queue-monitor** Compute Instance so that the queue monitoring application (the Go application) can create and write to custom metrics. Once you have updated the startup script you will need to restart the instance.

The Go application is installed in the `/work/go` directory in the Compute Instance by the startup script.

3. You can confirm that the application is working by searching for the metric `input_queue_size` in the Metrics Explorer in Cloud Monitoring.

#---CLICK DASHBOARD LINK > MEDIA DASHBOARD > ADD WIDGET > LINE

#---SELECT METRIC > CUSTOM > VM INSTANCE > CUSTOM METRICS > OpenCensus..... > APPLY

#---FILTER > INSTANCE ID = <AUTO\_POPULATED\_NUMBER> > APPLY

#--ADD WIDGET > LINE > SELECT METRIC > UNTICK ACTIVE > LOGGING > VM INSTANCE > LOG-BASED-METRIC > {YOUR\_METRIC\_NAME} > APPLY > APPLY

#--ALERTING FROM LEFT HAND MENU > CREATE POLICY > SELECT METRIC > UNTICK ACTIVE > LOGGING > VM INSTANCE > LOG-BASED-METRIC > <{YOUR\_METRIC\_NAME}> > APPLY >

#---NEXT > {THRESHOLD VALUE FROM LAB INSTRUCTIONS} > NEXT > TOGGLE OFF NOTIFICATIONS CHANNEL > ALERTING POLICY NAME : ABC > CREATE POLICY

## Task 3. Create a custom metric using Cloud Operations logging events

1. Examine the Cloud Operations logs and create a custom metric that tracks the total volume of uploaded media files to your Cloud Function. The video upload Cloud Function creates a Cloud Operations Logging event that includes metadata about the type of video file the video processing system handles.
2. You have been asked to configure a custom log based metric called `Custom Metric Name` that will monitor the rate at which high resolution video files, those recorded at either 4K or 8K resolution, are uploaded.

The Cloud Function is already processing this data, and if you search the Cloud Operations logs using the advanced filter mode you will find log entries that contain the string "file\_format: 4K" or "file\_format: 8K" in the `textPayload` field whenever the `video_processing` Cloud Function receives a request to process a high resolution video. You can use that filter to create your custom metric.

# Get Cloud Monitoring (stackdriver) modules

go get go.opencensus.io

```
go get contrib.go.opencensus.io/exporter/stackdriver
```

```
# Configure env vars for the Video Queue processing application
```

```
export MY_PROJECT_ID=$DEVSHIELD_PROJECT_ID
```

```
export MY_GCE_INSTANCE_ID=$ID
```

```
export MY_GCE_INSTANCE_ZONE=$ZONE
```

```
# Initialize and run the Go application
```

```
cd /work
```

```
go mod init go/video/main
```

```
go mod tidy
```

```
go run /work/go/video/main.go
```

```
EOF
```

## Task 4. Add custom metrics to the Media Dashboard in Cloud Operations Monitoring

You must now add two charts to the Media Dashboard:

1. Add a chart for the video input queue length custom metric that is generated by the Go application running on the **video-queue-monitor** Compute Instance.
2. Add a chart for the high resolution video upload rate custom log based metric to the **Media\_Dashboard** custom dashbo

```
gcloud compute instances remove-metadata video-queue-monitor --keys=startup-script --zone=$ZONE
```

```
gcloud compute instances add-metadata video-queue-monitor --metadata-from-file=startup-script=$(readlink -f my-start-up-script.sh) --zone=$ZONE
```

```
gcloud logging metrics create $METRIC_NAME --description="custome metric" --log-  
filter='textPayload=~"file_format\: ([4,8]K).*"'
```

## Task 5. Create a Cloud Operations alert based on the rate of high resolution video file uploads

- Create a custom alert using the high resolution video upload metric that triggers when the upload rate for large videos exceeds a count of `Threshold` per second.

```
gcloud compute instances stop video-queue-monitor --zone=$ZONE  
gcloud compute instances start video-queue-monitor --zone=$ZONE  
  
echo "  
"  
echo "-----"  
  
echo "DASHBOARD LINK :  
https://console.cloud.google.com/monitoring/dashboards?project=\$DEVSHIELD\_PROJECT\_ID  
&pageState=\(%22dashboards%22:\(%22t%22:%22All%22\)\)"  
echo "-----"  
echo "  
"
```