Lab : Streaming Data Pipelines

objectives

In this lab, you will perform the following tasks:
•Launch Dataflow and run a Dataflow job
•Understand how data elements flow through the transformations of a Dataflow pipeline
•Connect Dataflow to Pub/Sub and BigQuery
•Observe and understand how Dataflow autoscaling adjusts compute resources to process input data optimally
•Learn where to find logging information created by Dataflow
•Explore metrics and create alerts and dashboards with Cloud Monitoring

Preparation

- ✓ **Compute Engine** > **VM instances > training-vm > Connect**
  In vm terminal write ls /training to Verify initialization
- ✓ download a code repository
  git clone https://github.com/GoogleCloudPlatform/training-data-analyst
- ✓ sets the DEVSHELL_PROJECT_ID and BUCKET environment variables
  source /training/project_env.sh
- ✓ **Create dataset Demo in bigquery**
- ✓ Cloud Storage Bucket
  Cloud Storage > **Browser**. With the following properties
  Name : project_id
  Default storage class : Regional
  Location : us-central1

Simulate traffic sensor data into Pub/Sub

- ✓ In the **training-vm** SSH terminal, start the sensor simulator. The script reads sample data from a CSV file and publishes it to Pub/Sub.

  /training/sensor_magic.sh
- ✓ Open a second SSH terminal and connect to the training VM

Launch Dataflow Pipeline

✓ **Enable Dataflow API**

second **training-vm** SSH terminal.

cd ~/training-data-analyst/courses/streaming/process/sandiego

6.Identify the script that creates and runs the Dataflow pipeline.

cat run_oncloud.sh

7.Copy-and-paste the following URL into a new browser tab to view the source code on Github

https://github.com/Googlestreaming/process/sandiego/run_oncloud.sh

8.The **script requires** three arguments: **project id**, **bucket name**, **classname**

A 4th optional argument is **options**. The **options** argument discussed later in this lab.

CloudPlatform/training-data-analyst/blob/master/courses/

project id	<your Project ID>
bucket name	<your Bucket Name>
classname	<java file that runs aggregations>
options	<options>

There are 4 java files that you can choose from for **classname**. Each reads the traffic data from Pub/Sub and runs different aggregations/computations

Launch Dataflow Pipeline

9.Go into the java directory. Identify the source file **AverageSpeeds.java**.

cd ~/training-data-analyst/courses/streaming/process/sandiego/src/main/java/com/google/cloud/training/dataanalyst/sandiego

cat AverageSpeeds.java

10.Copy-and-paste the following URL into a browser tab to view the source code on Github.

https://github.com/GoogleCloudPlatform/training-data-analyst/blob/master/courses/streaming/process/sandiego/src/main/java/com/google/cloud/training/dataanalyst/sandiego/AverageSpeeds.java

11.Return to the **training-vm** SSH terminal. Run the Dataflow pipeline to read from PubSub and write into BigQuery.

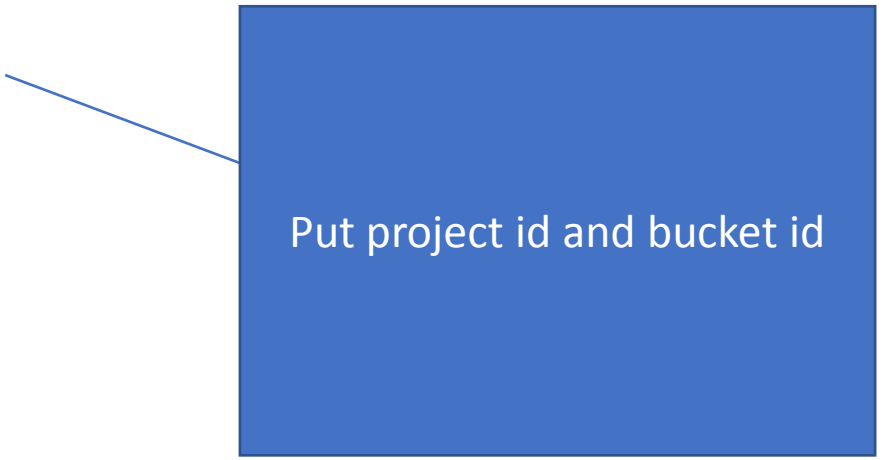
cd ~/training-data-analyst/courses/streaming/process/sandiego

./run_oncloud.sh $DEVSHELL_PROJECT_ID $BUCKET AverageSpeeds

This script uses maven to build a Dataflow streaming pipeline in Java.

Example successful completion:

```
INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 45.542 s
[INFO] Finished at: 2018-06-08T16:51:30+00:00
[INFO] Final Memory: 56M/216M
[INFO] ------------------------------------------------------------------------
```



Explore the pipeline

- ✓ click **Dataflow** and click on your job to monitor progress.
- ✓ **Pub/Sub** > **Topics** **examine** topic **sandiego**
- ✓ **Dataflow >** graph pipeline > **GetMessages step**
  - which corresponds to Pub/Sub messages that have been read.
  - Do you see a subscription created?
  - How does the code pull messages from Pub/Sub?
- ✓ **Dataflow >** graph pipeline > **Time Window step**
  - What is the window interval?
  - How often is a new window created?
- ✓ **BySensor ,AvgBySensor , ToBQRow and BigQueryIO.Write**
- ✓ Go back to bigquery **demos** you find **average_speeds** table

Find correspond code To this steps AverageSpeeds.java

Determine throughput rates

Dataflow >> GetMessages , Time Window

- **System Lag** is an important metric for streaming pipelines. It represents the amount of time data elements are waiting to be processed since they "arrived" in the input of the transformation step.
- **Elements Added** metric under output collections tells you how many data elements exited this step (for the **Read PubSub Msg** step of the pipeline it also represents the number of Pub/Sub messages read from the topic by the Pub/Sub IO connector).

Bigquery

```
SELECT *
FROM `demos.average_speeds`
ORDER BY timestamp DESC
LIMIT 100
```

Find the last update to the table by running the following SQL.

```
SELECT
MAX(timestamp)
FROM
`demos.average_speeds`
```

Observe and understand autoscaling

Observe how Dataflow scales the number of workers to process the backlog of incoming Pub/Sub messages.

Dataflow > JOB METRICS > **Autoscaling > More history** (monitor workers) > **Worker pool (status)**

Refresh the sensor data simulation script

✓ **Interrupt training-vm** SSH first terminal **CRTL+C**

```
cd ~/training-data-analyst/courses/streaming/publish
./send_sensor_data.py --speedFactor=60 --project $DEVSHELL_PROJECT_ID
```

✓ **open third vm terminal and write**

- source /training/project_env.sh # create environment variables
- cd ~/training-data-analyst/courses/streaming/publish
  ./send_sensor_data.py --speedFactor=60 --project $DEVSHELL_PROJECT_ID

Cloud Monitoring integration

Cloud Monitoring integration with Dataflow allows users to access Dataflow job metrics such as System Lag (for streaming jobs), Job Status (Failed, Successful), Element Counts, and User Counters from within Cloud Monitoring.
Integration features of Cloud Monitoring
•**Explore Dataflow Metrics**: Browse through available Dataflow pipeline metrics and visualize them in charts.
Some common Dataflow metrics.

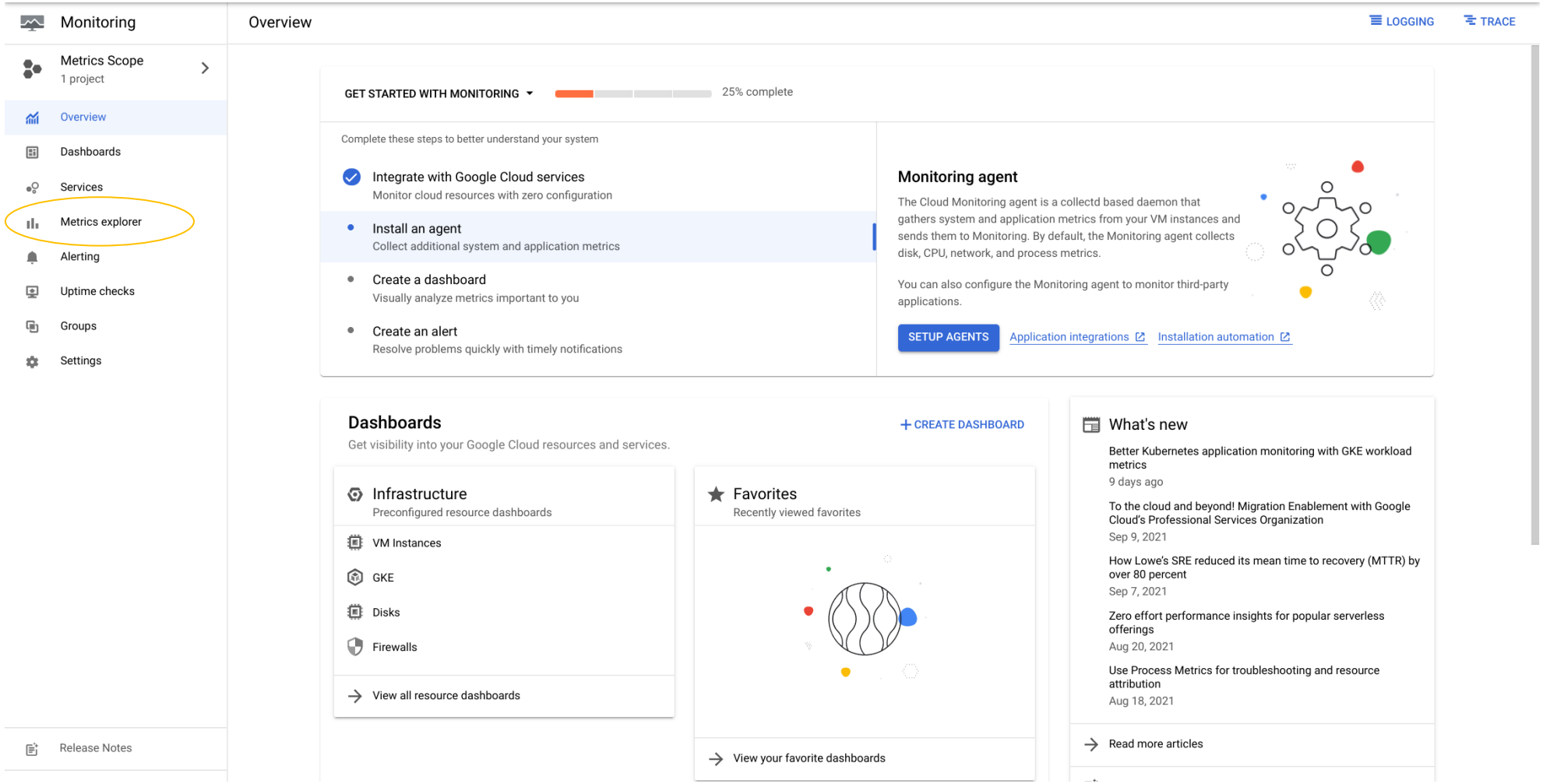
Metrics	Features
Job status	Job status (Failed, Successful), reported as an enum every 30 secs and on update.
Elapsed time	Job elapsed time (measured in seconds), reported every 30 secs.
System lag	Max lag across the entire pipeline, reported in seconds.
Current vCPU count	Current # of virtual CPUs used by job and updated on value change.
Estimated byte count	Number of bytes processed per PCollection

Cloud Monitoring integration

•**Chart Dataflow metrics in Monitoring Dashboards**: Create Dashboards and chart time series of Dataflow metrics.
•**Configure Alerts**: Define thresholds on job or resource group-level metrics and alert when these metrics reach specified values. Monitoring alert can notify on a variety of conditions such as long streaming system lag or failed jobs.
•**Monitor User-Defined Metrics**: In addition to Dataflow metrics, Dataflow exposes user-defined metrics (SDK Aggregators) as Monitoring custom counters in the Monitoring UI, available for charting and alerting. Any Aggregator defined in a Dataflow pipeline will be reported to Monitoring as a custom metric. Dataflow will define a new custom metric on behalf of the user and report incremental updates to Monitoring approximately every 30 seconds.

Explore metrics

Navigation menu > Monitoring.



Explore metrics

Metrics Explorer > under **Resource & Metric** click on **SELECT A METRIC**

5.Select **Dataflow Job > Job** You should see a list of available Dataflow-related metrics. Select **Data watermark lag** and click **Apply**.

6.Under metric, click on the **Reset** to remove the **Data watermark lag** metric. Select a new dataflow metric **System lag**.

.

Note: the metrics that Dataflow provides to Monitoring are listed here. You can search on the page for Dataflow. The metrics you have viewed are useful indicators of pipeline performance.

Data watermark lag: The age (time since event timestamp) of the most recent item of data that has been fully processed by the pipeline.

System lag: The current maximum duration that an item of data has been awaiting processing, in seconds

Create alerts

If you want to be notified when a certain metric crosses a specified threshold (for example, when System Lag of our lab streaming pipeline increases above a predefined value), you could use the Alerting mechanisms of Monitoring to accomplish that.
Create an alert
1.On the Cloud Monitoring, click **Alerting**.
2.Click **+ Create Policy**.
3.Click **Add Condition**.
4.In the **Target** section, set the **RESOURCE TYPE** to **Dataflow Job**.
5.Set the **Metric** to **System Lag**.
6.Under **Configuration** set **CONDITION** to **is above**.
7.Set **THRESHOLD** to **5**.
8.Set **FOR** to **1 minute**.
9.Click **Add**.
10.Click **Next**.

Create alerts

Add a notification

11.Click on drop down arrow next to **Notification Channels**, then click on **Manage Notification Channels**.

A **Notification channels** page will open in new tab.

12.Scroll down the page and click on **ADD NEW** for **Email**.

13.In **Create Email Channel** dialog box, enter the Qwiklabs username as the **Email Address** field and a **Display name**.

14.Note: if you enter your own email address, you might get alerts until all the resources in the project have been deleted.Click **Save**.

15.Go back to the previous **Create alerting policy** tab.

16.Click on **Notification Channels** again, then click on the **Refresh icon** to get the display name you mentioned in the previous step.

17.Now, select your **Display name** and click **OK**.

18.Click **Next**.

19.Set **Alert name** as **MyAlertPolicy**.

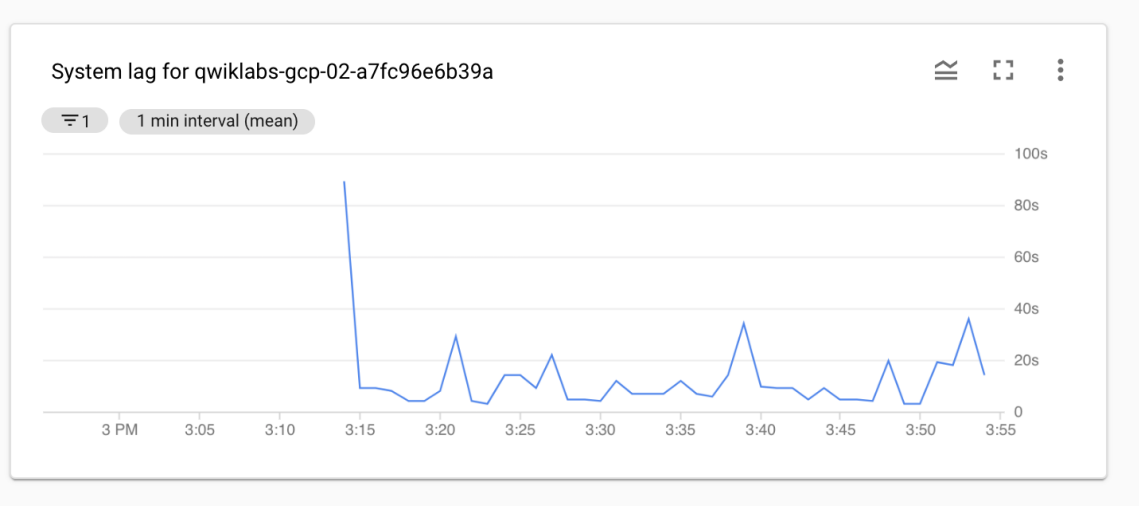
20.Skip the Documentation step.

21.Click **Save**.

Set up dashboards

You can easily build dashboards with the most relevant Dataflow-related charts with Cloud Monitoring Dashboards.

1. In the left pane, click **Dashboards**.
2. Click **+Create Dashboard**.
3. For **New Dashboard Name**, type **My Dashboard**.
4. Click **Line Chart**.
5. Click on the dropdown box under **Resource & Metric**.
6. Select Dataflow > Job > System Lag and click **Apply**.
7. In the **Filters** panel, click **+ Add Filter**.
8. Select **project_id** in Label field, then select or type your *GCP project ID* in the Value field.
9. Click **Done**.



You can add more charts to the dashboard, if you would like, for example, Pub/Sub publish rates on the topic, or subscription backlog (which is a signal to the Dataflow auto-scaler).

Launch another streaming pipeline

- ✓ **Interrupt first training-vm** SSH terminal **CRTL+C**
- ✓ **cd ~/training-data-analyst/courses/streaming/publish**
  ./send_sensor_data.py --speedFactor=60 --project $DEVSHELL_PROJECT_ID
- ✓ **cd ~/training-data-analyst/courses/streaming/process/sandiego/src/main/java/com/google/cloud/training/dataanalyst/sandiego**
  cat CurrentConditions.java
- ✓ **see code** https://github.com/GoogleCloudPlatform/training-data-analyst/blob/master/courses/streaming/process/sandiego/src/main/java/com/google/cloud/training/dataanalyst/sandiego/CurrentConditions.java
- ✓ **In another training-vm** SSH terminal
  cd ~/training-data-analyst/courses/streaming/process/sandiego
  ./run_oncloud.sh $DEVSHELL_PROJECT_ID $BUCKET CurrentConditions
- ✓ go to bigquery appear **current_conditions** table