

Tony Nguyen

Dr. Shawn Bowers

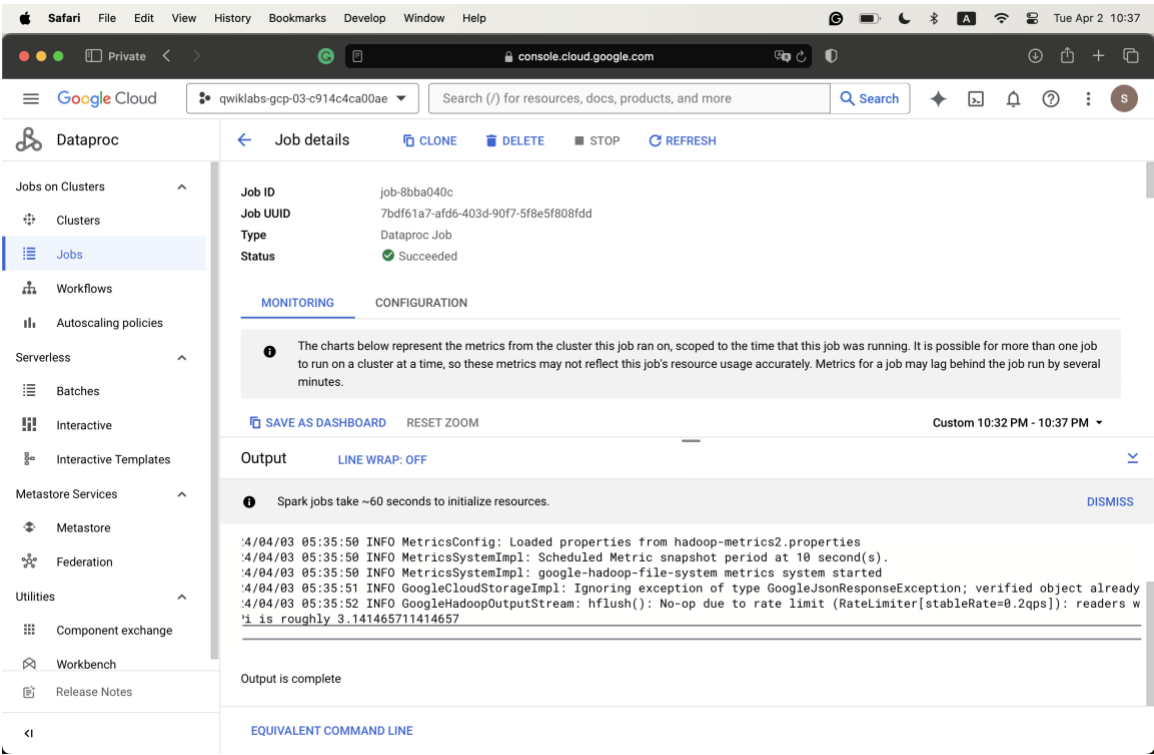
CPSC 324 01

28 March 2024

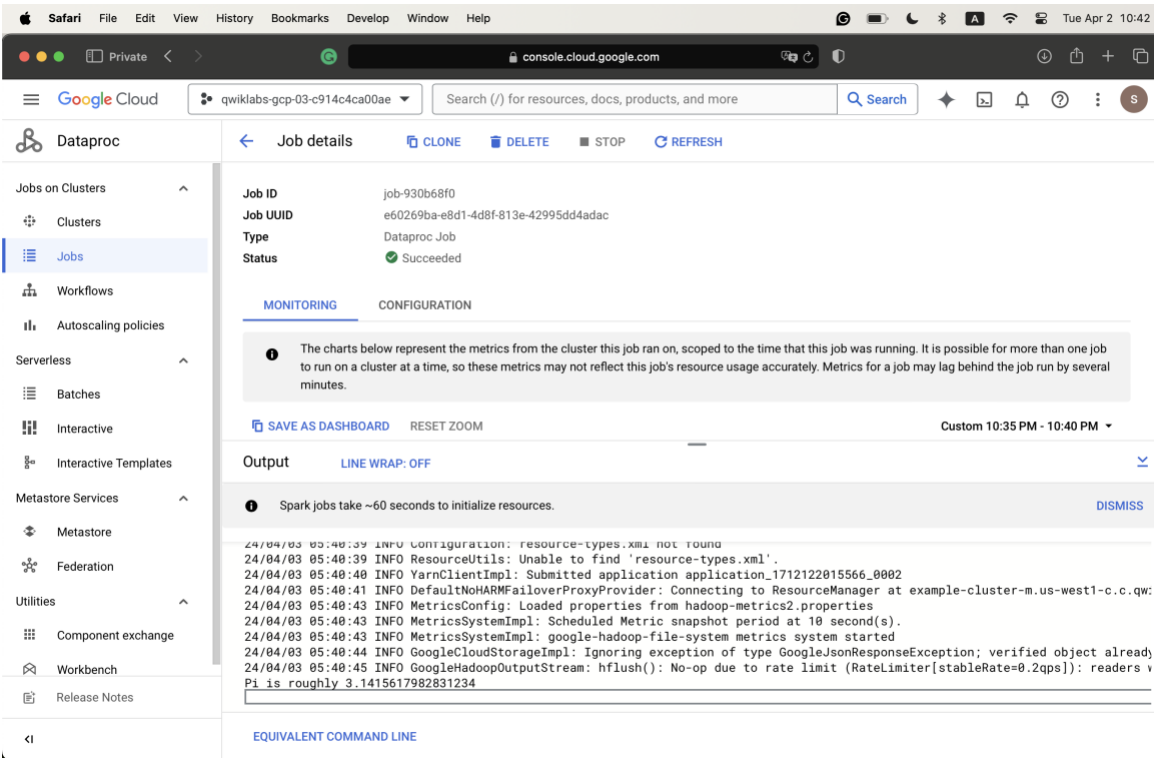
Homework 4

1.

- Steps for creating a cluster and submitting the job
 - Confirm prerequisite
 - Navigation Menu -> API services -> Library -> Cloud Dataproc
 - Assign permission
 - Navigation Menu -> IAM & Admin -> IAM
 - Click on the compute@developer.gserviceaccount.com account
 - Select edit, and add “Storage Admin” role
 - Create a cluster
 - Navigation Menu -> Dataproc -> Cluster -> Cluster on Compute Engine
 - Remember to deselect "Configure all instances to have only internal IP addresses" in the Customer Cluster
 - Submitting a job
 - Under the “Job” tab, hit “Submit Job”
 - Choose the appropriate values
- Screenshots for 1,000



- Screenshot for 5,000



- Screenshot for 10,000

Google Cloud | **Dataproc** | **Job details** | [CLONE](#) | [DELETE](#) | [STOP](#) | [REFRESH](#)

Job ID: job-0e3073b1
Job UUID: 3475294b-653c-4d88-8483-4ce228bc8aae
Type: Dataproc Job
Status: Succeeded

MONITORING | **CONFIGURATION**

The charts below represent the metrics from the cluster this job ran on, scoped to the time that this job was running. It is possible for more than one job to run on a cluster at a time, so these metrics may not reflect this job's resource usage accurately. Metrics for a job may lag behind the job run by several minutes.

[SAVE AS DASHBOARD](#) | [RESET ZOOM](#) | Custom 10:41 PM - 10:46 PM

Output | [LINE WRAP: OFF](#)

Spark jobs take ~60 seconds to initialize resources. [DISMISS](#)

```
24/04/03 05:43:21 INFO ResourceUtils: unable to find resource-types.xml
24/04/03 05:43:22 INFO YarnClientImpl: Submitted application application_1712122015566_0003
24/04/03 05:43:23 INFO DefaultNoHARMAILOverProxyProvider: Connecting to ResourceManager at example-cluster-m.us-west1-c.c.qw:
24/04/03 05:43:25 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/04/03 05:43:25 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/04/03 05:43:25 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/04/03 05:43:27 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already
24/04/03 05:43:28 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers v
P1 is roughly 3.1415722111415723
24/04/03 05:44:38 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers v
```

[EQUIVALENT COMMAND LINE](#)

- Screenshot for 100,000

Google Cloud | **Dataproc** | **Job details** | [CLONE](#) | [DELETE](#) | [STOP](#) | [REFRESH](#)

Job ID: job-04de14fb
Job UUID: d02ef7d1-1928-4eb9-9e33-7f759a47e6c9
Type: Dataproc Job
Status: Succeeded

MONITORING | **CONFIGURATION**

The charts below represent the metrics from the cluster this job ran on, scoped to the time that this job was running. It is possible for more than one job to run on a cluster at a time, so these metrics may not reflect this job's resource usage accurately. Metrics for a job may lag behind the job run by several minutes.

[SAVE AS DASHBOARD](#) | [RESET ZOOM](#) | Custom 10:45 PM - 10:50 PM

Output | [LINE WRAP: OFF](#)

Spark jobs take ~60 seconds to initialize resources. [DISMISS](#)

```
24/04/03 05:46:28 INFO ResourceUtils: unable to find resource-types.xml
24/04/03 05:46:30 INFO YarnClientImpl: Submitted application application_1712122015566_0004
24/04/03 05:46:31 INFO DefaultNoHARMAILOverProxyProvider: Connecting to ResourceManager at example-cluster-m.us-west1-c.c.qw:
24/04/03 05:46:33 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/04/03 05:46:33 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/04/03 05:46:33 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/04/03 05:46:34 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already
24/04/03 05:46:35 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers v
P1 is roughly 3.1415882773153374
24/04/03 05:50:23 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers v
```

[EQUIVALENT COMMAND LINE](#)

- Screenshot for amount of time

The screenshot shows the Google Cloud Dataproc console. The left sidebar contains navigation links: Clusters, Jobs (selected), Workflows, Autoscailing policies, Serverless, Batches, Interactive, Interactive Templates, Metastore Services, Metastore, Federation, Utilities, Component exchange, Workbench, and Release Notes. The main area displays a table of jobs. A warning message at the top states: 'If Dataproc can't decrypt CMEK-enabled job parameters, the job is not listed in the table.' The table has columns: Region, Type, Cluster, Start time, Elapsed time, and Labels. It lists four jobs, all in the 'us-west1' region, using 'Spark' type, and running on 'example-cluster'. The start times are from April 2, 2024, and elapsed times range from 58 seconds to 4 minutes 11 seconds. A notification at the bottom center says 'Job job-04de14fb successfully submitted'.

Region	Type	Cluster	Start time	Elapsed time	Labels
us-west1	Spark	example-cluster	Apr 2, 2024, 10:46:16 PM	4 min 11 sec	None
us-west1	Spark	example-cluster	Apr 2, 2024, 10:43:08 PM	1 min 33 sec	None
us-west1	Spark	example-cluster	Apr 2, 2024, 10:40:26 PM	1 min 10 sec	None
us-west1	Spark	example-cluster	Apr 2, 2024, 10:35:28 PM	58 sec	None

- The more argument values it takes, the longer it is for the jobs to be accomplished.

2.

- Set region
 - `gcloud config set dataproc/region us-west1`
- Get PROJECT_ID and PROJECT_NUMBER variables
 - `echo $PROJECT_ID`
 - `echo $PROJECT_NUMBER`
 - `PROJECT_ID=$(gcloud config get-value project) && \`
 - `gcloud config set project $PROJECT_ID`
 - `PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --format='value(projectNumber)')`
- Set Storage Admin privilege

- gcloud projects add-iam-policy-binding \$PROJECT_ID \

--member=serviceAccount:\$PROJECT_NUMBER-

compute@developer.gserviceaccount.com \

--role=roles/storage.admin
- Create a cluster
 - gcloud dataproc clusters create example-cluster --worker-boot-disk-size 500 --

worker-machine-type=e2-standard-4 --master-machine-type=e2-standard-4
- Run job
 - gcloud dataproc jobs submit spark --cluster example-cluster \

--class org.apache.spark.examples.SparkPi \

--jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000
 - Note that the 1000 value depicts the Argument
- Screenshot of 1,000

```

student 01 15e2f8807c0@cloudshell:~ (qwiklabs-gcp-04-69cb94e46daa) $ gcloud dataproc jobs submit spark --cluster example-cluster \
--class org.apache.spark.examples.SparkPi \
--jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000
Job [5b8370f52756483684e49bdc0bf9d3] submitted.
Waiting for job output...
24/04/03 06:48:47 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
24/04/03 06:48:47 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
24/04/03 06:48:47 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/03 06:48:47 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
24/04/03 06:48:47 INFO org.sparkproject.jetty.util.log: Logging initialized @4449ms to org.sparkproject.jetty.util.log.Slf4jLog
24/04/03 06:48:48 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1
.8.0_402-b06
24/04/03 06:48:48 INFO org.sparkproject.jetty.server.Server: Started 84578ms
24/04/03 06:48:48 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@3ac8cf9b(HTTP/1.1, (http/1.1)) (0.0.0.0:42939)
24/04/03 06:48:49 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8032
24/04/03 06:48:49 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at example-cluster-m/10.138.0.3:10200
24/04/03 06:48:50 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
24/04/03 06:48:50 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
24/04/03 06:48:51 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1712126793334_0001
24/04/03 06:48:52 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8030
24/04/03 06:48:54 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op get file status. latencyMs=311; previousMaxL
atencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history
24/04/03 06:48:55 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException
; verified object already exists with desired state.
24/04/03 06:48:55 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op mkdirs. latencyMs=168; previousMaxLatencyMs=
0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history
Bl is roughly 3.141413471414135
24/04/03 06:49:10 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@3ac8cf9b(HTTP/1.1, (http/1.1)) (0.0.0.0:0)
24/04/03 06:49:11 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op rename. latencyMs=175; previousMaxLatencyMs=
0; operationCount=1; context=rename(gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application_1712126793334_000
1.inprogress -> gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application_1712126793334_0001)
Job [5b8370f52756483684e49bdc0bf9d3] finished successfully.
done: true
driverOutputResourceUri: gs://dataproc-staging-us-west1-337569798961-n0unt6iy/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/5b8370f52756483684e4
9bdc0bf9d3/driveroutput
jobUuid: f0494361-f306-347e-9685-6f9c684d185d
placement:
  clusterName: example-cluster
  clusterUuid: 6ddfecf2-f663-42d9-8e8c-cd03ce67b960
  
```

- Screenshot of 5,000

The screenshot shows the Google Cloud Dataproc console interface. The top navigation bar includes 'Safari', 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Develop', 'Window', and 'Help'. The main header displays 'Google Cloud' and the project 'qwiklabs-gcp-04-69cb94e46daa'. The 'Clusters' tab is active, showing a 'CREATE CLUSTER' button and a 'START' button. Below the header, the 'CLOUD SHELL' terminal window is open, displaying the following command and output:

```
student 01 15e2f88007c0@cloudshell:~ (qwiklabs-gcp-04-69cb94e46daa) $ gcloud dataproc jobs submit spark --cluster example-cluster --class org.apache.spark.examples.SparkPi --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 5000
Job [2bcl4e9c01443efaf1405eab71de6d7f] submitted.
Waiting for job output...
24/04/03 06:53:04 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
24/04/03 06:53:04 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
24/04/03 06:53:04 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/03 06:53:04 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
24/04/03 06:53:04 INFO org.sparkproject.jetty.util.log: Logging initialized @4135ms to org.sparkproject.jetty.util.log.Slf4jLog
24/04/03 06:53:04 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_402-b06
24/04/03 06:53:04 INFO org.sparkproject.jetty.server.Server: Started @4277ms
24/04/03 06:53:07 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@3ac8cf9b(HTTP/1.1, (http/1.1)){0.0.0.0:40567}
24/04/03 06:53:05 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8032
24/04/03 06:53:05 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at example-cluster-m/10.138.0.3:10200
24/04/03 06:53:06 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
24/04/03 06:53:06 INFO org.apache.hadoop.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
24/04/03 06:53:07 INFO org.apache.hadoop.yarn.client.impl.YarnClientImpl: Submitted application application 1712126793334 0002
24/04/03 06:53:08 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8030
24/04/03 06:53:10 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op get file status. latencyMs=247; previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0002; verified object already exists with desired state.
PI is roughly 3.141583646283167
24/04/03 06:53:45 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@3ac8cf9b(HTTP/1.1, (http/1.1)){0.0.0.0:0}
24/04/03 06:53:45 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation stream write close operations. latencyMs=212; p
previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712
126793334 0002; inprogress
24/04/03 06:53:46 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op rename. latencyMs=320; previousMaxLatencyMs=
0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 000
2; inprogress -> gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0002)
Job [2bcl4e9c01443efaf1405eab71de6d7f] finished successfully.
done: true
driverControlFileUri: gs://dataproc-staging-us-west1-337569798961-n0unt6iy/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/2bcl4e9c01443efaf1405
eab71de6d7f/
driverOutputResourceUri: gs://dataproc-staging-us-west1-337569798961-n0unt6iy/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/2bcl4e9c01443efaf14
05eab71de6d7f/driveroutput
jobId: b54a863d-ae72-337e-a5ff-82c1832ebdc0
placement:
  clusterName: example-cluster
  clusterUuid: 6ddfecf2-f663-42d9-8e8c-cd03ce67b960
```

- Screenshot of 10,000

The screenshot shows the Google Cloud Dataproc console interface. The top navigation bar includes 'Safari', 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Develop', 'Window', and 'Help'. The main header displays 'Google Cloud' and the project 'qwiklabs-gcp-04-69cb94e46daa'. The 'Clusters' tab is active, showing a 'CREATE CLUSTER' button and a 'START' button. Below the header, the 'CLOUD SHELL' terminal window is open, displaying the following command and output:

```
student 01 15e2f88007c0@cloudshell:~ (qwiklabs-gcp-04-69cb94e46daa) $ gcloud dataproc jobs submit spark --cluster example-cluster --class org.apache.spark.examples.SparkPi --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 10000
Job [5923fd92fa5e44dd9bec7b902dac46c] submitted.
Waiting for job output...
24/04/03 06:58:09 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
24/04/03 06:58:09 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
24/04/03 06:58:09 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/03 06:58:09 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
24/04/03 06:58:09 INFO org.sparkproject.jetty.util.log: Logging initialized @4043ms to org.sparkproject.jetty.util.log.Slf4jLog
24/04/03 06:58:09 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_402-b06
24/04/03 06:58:09 INFO org.sparkproject.jetty.server.Server: Started @4195ms
24/04/03 06:58:09 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@3ac8cf9b(HTTP/1.1, (http/1.1)){0.0.0.0:40899}
24/04/03 06:58:10 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8032
24/04/03 06:58:10 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at example-cluster-m/10.138.0.3:10200
24/04/03 06:58:11 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
24/04/03 06:58:11 INFO org.apache.hadoop.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
24/04/03 06:58:11 INFO org.apache.hadoop.yarn.client.impl.YarnClientImpl: Submitted application application 1712126793334 0003
24/04/03 06:58:12 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8030
24/04/03 06:58:13 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op get file status. latencyMs=288; previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0003; verified object already exists with desired state.
PI is roughly 3.14152259141552
24/04/03 06:59:25 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@3ac8cf9b(HTTP/1.1, (http/1.1)){0.0.0.0:0}
24/04/03 06:59:26 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation stream write close operations. latencyMs=203; p
previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712
126793334 0003; inprogress
24/04/03 06:59:26 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op rename. latencyMs=197; previousMaxLatencyMs=
0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 000
3; inprogress -> gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0003)
Job [5923fd92fa5e44dd9bec7b902dac46c] finished successfully.
done: true
driverControlFileUri: gs://dataproc-staging-us-west1-337569798961-n0unt6iy/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/5923fd92fa5e44dd9bec
7b902dac46c/
driverOutputResourceUri: gs://dataproc-staging-us-west1-337569798961-n0unt6iy/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/5923fd92fa5e44dd9be
ec7b902dac46c/driveroutput
jobId: e0696950-2ece-3f6e-b54f-daa8c146df2a
placement:
  clusterName: example-cluster
  clusterUuid: 6ddfecf2-f663-42d9-8e8c-cd03ce67b960
```

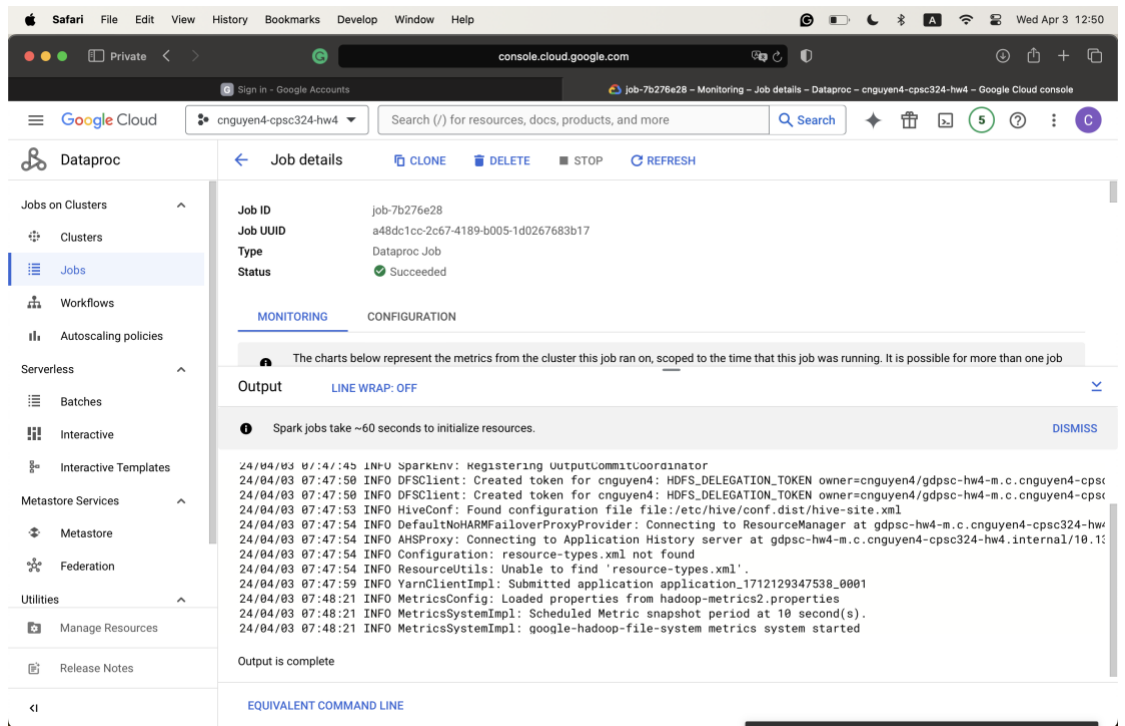

- Screenshot of 100,000

The screenshot shows the Google Cloud console interface. At the top, there's a navigation bar with 'Google Cloud' and a search bar. Below it, the 'Dataproc' section is active, showing a list of clusters. A terminal window is open, displaying the command to submit a Spark job and its subsequent logs. The logs show the job's progress, including the submission of the application, the start of the Spark environment, and the execution of the job. The job is identified by the ID 'd7c5bc092e62461792dd2f2d286871e4'.

```
student 01 15e2f88007c0@cloudshell:~ (qwiklabs-gcp-04-69cb94e46daa)$ gcloud dataproc jobs submit spark --cluster example-cluster --class org.apache.spark.examples.SparkPi --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 100000
Job [d7c5bc092e62461792dd2f2d286871e4] submitted.
Waiting for job output...
24/04/03 07:01:51 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
24/04/03 07:01:51 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
24/04/03 07:01:51 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/03 07:01:51 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
24/04/03 07:01:51 INFO org.sparkproject.jetty.util.log: Logging initialized @4180ms to org.sparkproject.jetty.util.log.Slf4jLog
24/04/03 07:01:51 INFO org.sparkproject.jetty.server.Server: jetty-9.4.0.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_402-b06
24/04/03 07:01:51 INFO org.sparkproject.jetty.server.Server: Started @4325ms
24/04/03 07:01:52 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector$3ac8cf9b[HTTP/1.1, (http/1.1)]{0.0.0.0:45787}
24/04/03 07:01:52 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8032
24/04/03 07:01:53 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at example-cluster-m/10.138.0.3:10200
24/04/03 07:01:53 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
24/04/03 07:01:53 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
24/04/03 07:01:54 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application 1712126793334 0004
24/04/03 07:01:55 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at example-cluster-m/10.138.0.3:8030
24/04/03 07:01:57 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op get file status. latencyMs=238; previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history
24/04/03 07:01:57 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/04/03 07:04:21 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark$3ac8cf9b[HTTP/1.1, (http/1.1)]{0.0.0.0:0}
24/04/03 07:04:21 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation stream write operations. latencyMs=542; previousMaxLatencyMs=0; operationCount=0; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0004.inprogress
24/04/03 07:04:21 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation stream write close operations. latencyMs=293; previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0004.inprogress
24/04/03 07:04:21 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op rename. latencyMs=173; previousMaxLatencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0004.inprogress -> gs://dataproc-temp-us-west1-337569798961-xw3othoe/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/spark-job-history/application 1712126793334 0004)
Job [d7c5bc092e62461792dd2f2d286871e4] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-west1-337569798961-n0unt61y/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/d7c5bc092e62461792dd2f2d286871e4/
driverOutputResourceUri: gs://dataproc-staging-us-west1-337569798961-n0unt61y/google-cloud-dataproc-metainfo/6ddfecf2-f663-42d9-8e8c-cd03ce67b960/jobs/d7c5bc092e62461792dd2f2d286871e4/driveroutput
jobUuid: 92bca57f-47a6-3b68-8757-d65694c75c19
```

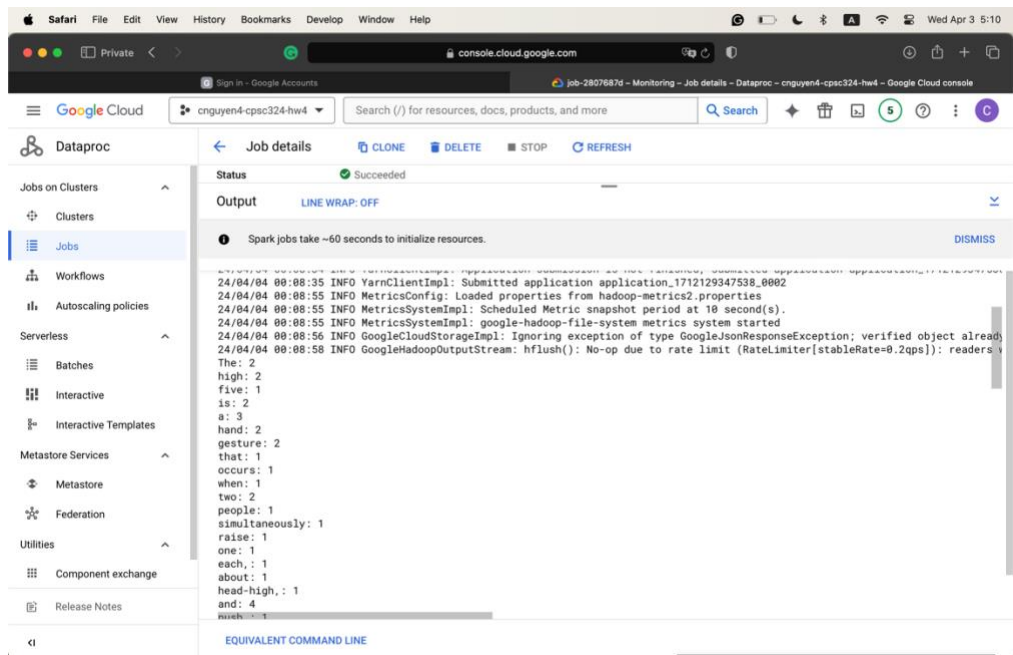
3.

- Steps
 - Enable APIs
 - Upload the script file to a bucket
 - Create a cluster
 - Submit job
- Screenshot



4.

- I don't encounter any issues while running this question
- For this script, I noticed that it runs much faster
- Screenshot



5.

- Compared to other steps
 - When creating a cluster
 - Change the Versioning section to 2.1
 - Under “Component,” enable “Jupyter Notebook”
- Commands
 - Clone the git repo
 - `git -C ~ clone https://github.com/GoogleCloudPlatform/training-data-analyst`
 - Export the default Cloud Storage Bucket
 - `export DP_STORAGE="gs://$(gcloud dataproc clusters describe sparktodb --region=europe-west1 --format=json | jq -r '.config.configBucket')"`
 - Copy the sample notebooks into the Jupyter working folder
 - `gcloud storage cp ~/training-data-analyst/quests/sparktodb/*.ipynb $DP_STORAGE/notebooks/jupyter`
 - Copy the Python script to run as a Cloud Dataproc Job
 - `gcloud storage cp gs://$PROJECT_ID/sparktodb/spark_analysis.py spark_analysis.py`
 - Create a launch script
 - `nano submit_onejob.sh`
 - `#!/bin/bash`
 - `gcloud dataproc jobs submit pyspark \`

```
--cluster sparktodb \  
--region europe-west1 \  
spark_analysis.py \  
-- --bucket=$1
```

- Press CTRL+X then Y and Enter key to exit and save.
- Make the script executable
 - `chmod +x submit_onejob.sh`
- Launch the PySpark Analysis job
 - `./submit_onejob.sh $PROJECT_ID`
- Steps to open the Jupyter Notebook
 - Get into the cluster main menu, click on “Web Interfaces”
 - Under “Component Gateway,” choose “Jupyter”

6.

- Bucket name after creating the cluster
 - `qwiklabs-gcp-00-01249133f629-dsongcp`
- Screenshot showing the result of the first spark command

The screenshot shows a Databricks notebook interface. The left sidebar displays the file explorer with a search bar and a list of files. The main area contains a code editor with the following code:

```

24/04/04 04:12:32 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/04 04:12:32 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator

Create a Spark DataFrame

[4]: from pyspark.mllib.classification import LogisticRegressionWithLBFGS
    from pyspark.mllib.regression import LabeledPoint

Read the dataset

[5]: traindays = spark.read \
    .option("header", "true") \
    .csv('gs://[...]/flights/trainday.csv'.format(BUCKET))
    traindays.createOrReplaceTempView('traindays')

Create a SparkSQL view

[6]: traindays.createOrReplaceTempView('traindays')

[7]: spark.sql("SELECT * from traindays LIMIT 5").show()

+-----+-----+
| FL_DATE | is_train_day |
+-----+-----+
| 2015-01-01 | True |
| 2015-01-02 | False |
| 2015-01-03 | False |
| 2015-01-04 | True |
| 2015-01-05 | True |
+-----+-----+

```

The bottom status bar indicates "Saving completed" and "Mode: Edit".

- Hi

The screenshot shows a Databricks notebook interface. The left sidebar displays the file explorer with a search bar and a list of files. The main area contains a code editor with the following code:

```

+-----+-----+
| 2015-01-01 | True |
| 2015-01-02 | False |
| 2015-01-03 | False |
| 2015-01-04 | True |
| 2015-01-05 | True |
+-----+-----+

[8]: inputs = 'gs://[...]/flights/tzcorr/all_flights-00000-*.format(BUCKET)

Read the data into Spark SQL from the input file you created:

[9]: flights = spark.read.json(inputs)
    flights.createOrReplaceTempView('flights')

[10]: trainquery = """
    SELECT
    DEP_DELAY, TAXI_OUT, ARR_DELAY, DISTANCE
    FROM flights f
    JOIN traindays t
    ON f.FL_DATE == t.FL_DATE
    WHERE
    t.is_train_day == 'True'
    """
    traindata = spark.sql(trainquery)

[11]: print(traindata.head(2))

[Row(DEP_DELAY=-3.0, TAXI_OUT=14.0, ARR_DELAY=-16.0, DISTANCE='370.00'), Row(DEP_DELAY=24.0, TAXI_OUT=12.0, ARR_DELAY=12.0, DISTANCE='370.00')]

```

The bottom status bar indicates "Saving completed" and "Mode: Edit".

- I think these attributes represent

- Departure delay
- Taxi out (time)

- Arrival delay
- Distance
- The result of running the statement *traindata.describe().show()*

summary	DEP_DELAY	TAXI_OUT	ARR_DELAY	DISTANCE
count	46439	46422	46355	46936
mean	8.561769202609876	15.427685149282668	3.2853413871211306	916.0707133117437
stddev	30.752752455053308	8.427384168645757	32.98848343691196	591.9164453757172
min	-22.0	2.0	-77.0	1009.00
max	711.0	178.0	719.0	980.00

- This table shows the summary statistics over the values of those four attributes.
- Data cleaning using SQL
 - The first query
 - This query joins the master flight list with the traindays set to find the flights that are used to train. It is used to calculate the summary statistics.
 - The query filters out dep_delay and arr_delay that are empty and is_train_day that is false.

summary	DEP_DELAY	TAXI_OUT	ARR_DELAY	DISTANCE
count	46355	46355	46355	46355
mean	8.539531873584295	15.421507927947363	3.2853413871211306	917.660230827311
stddev	30.700034730525516	8.41130660980497	32.98848343691196	592.0960248192869
min	-22.0	2.0	-77.0	1009.00
max	711.0	178.0	719.0	980.00

- The second query
 - This query joins the master flight list with the traindays set to find the flights that are used to train. It is used to calculate the summary statistics.

- This query filters out those that are canceled and diverted as well as those who belong to is_train_day equal to true.

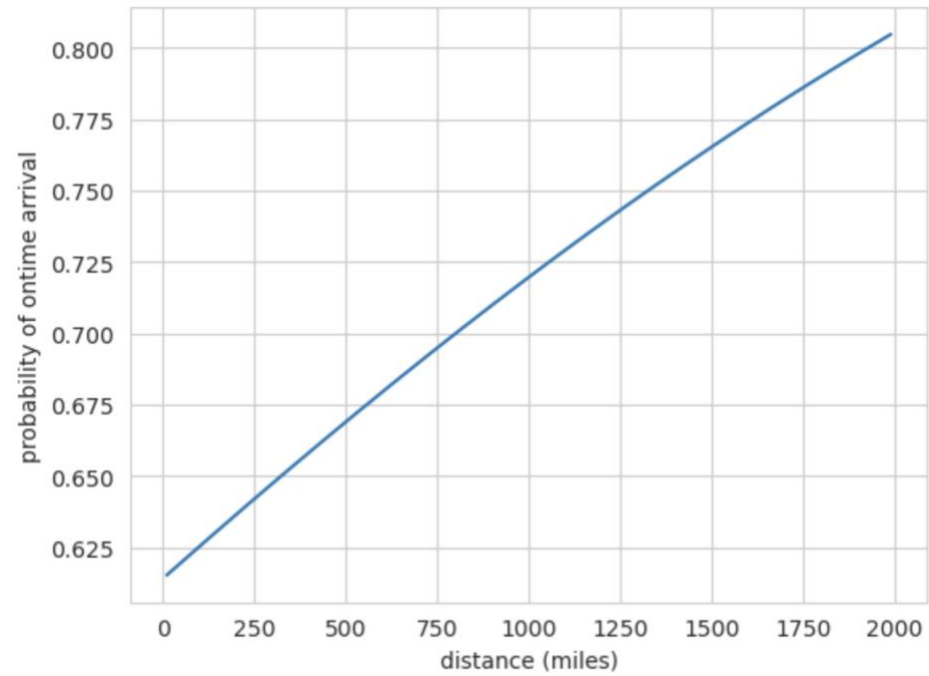
summary	DEP_DELAY	TAXI_OUT	ARR_DELAY	DISTANCE
count	46355	46355	46355	46355
mean	8.539531873584295	15.421507927947363	3.2853413871211306	917.660230827311
stddev	30.700034730525516	8.41130660980497	32.98848343691196	592.0960248192869
min	-22.0	2.0	-77.0	1009.00
max	711.0	178.0	719.0	980.00

- I don't personally like cleaning the dataset using SQL. I think one of the biggest problems is that the cleaning operations are not saved. Hence, every time we need to run the report again, we have to perform the cleaning operation another time.
- The map function acts as a “proxy” to apply the to_example function on the traindata set. In this case, the to_example acts as a filter that picks out the flights that match the condition within the function. The result is saved to the example variable.
- The result of two predictions after clearing the threshold

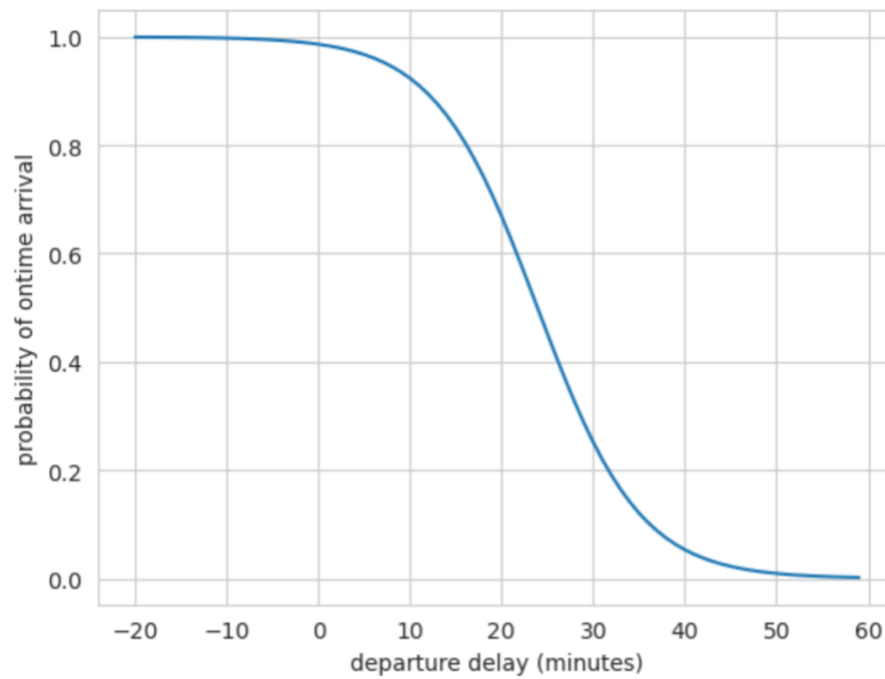
```
lrmodel.clearThreshold()
print(lrmodel.predict([6.0,12.0,594.0]))
print(lrmodel.predict([36.0,12.0,594.0]))
```

0.9520080900763146
0.08390675828170738

-
- Two arrival probability graphs



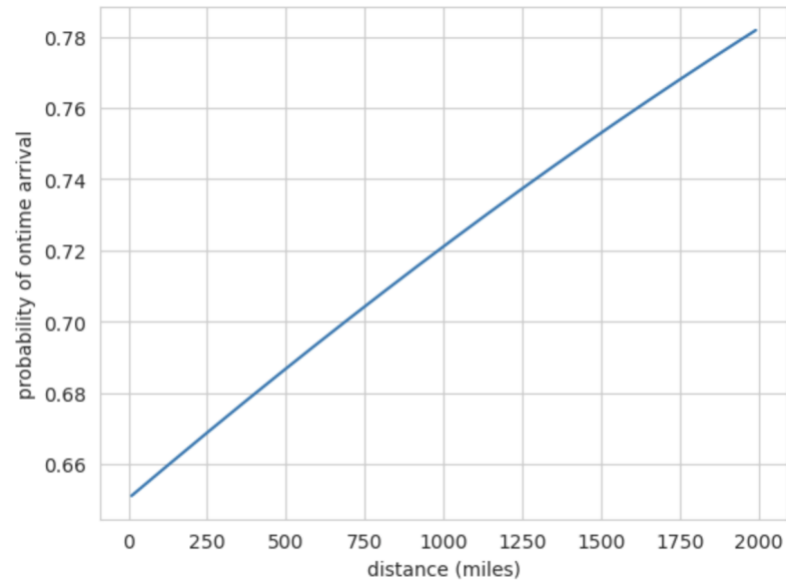
○



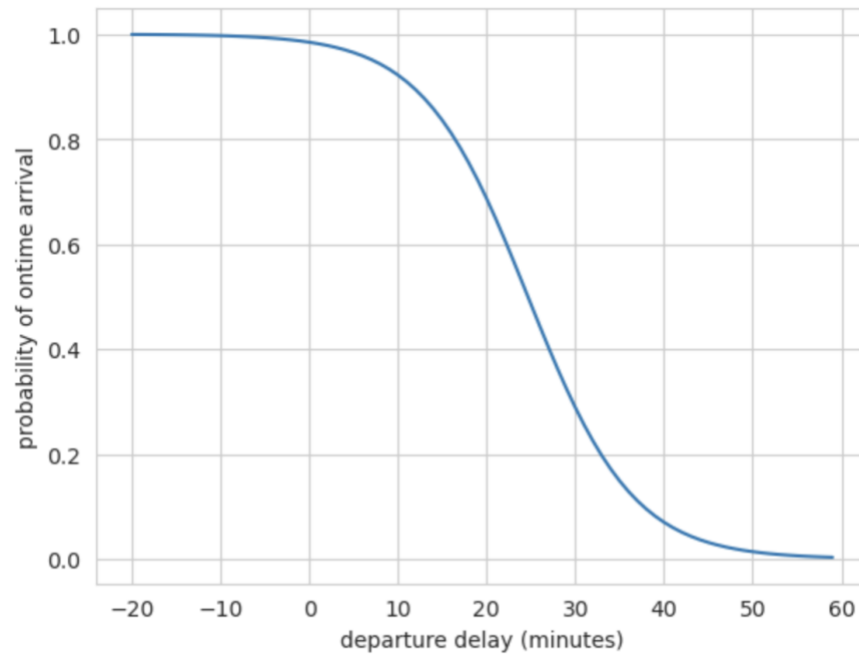
○

- The map and filter functions
 - The map function maps the ground truth and the predicted label of the testing set.

- The filter function filters out results that are within the 65-75% range.
- Running on the full-flight dataset



○



○

7.

- RDD operations
 - Transformations: create a new dataset from an existing one

- flatMap: process each element into multiple output elements
 - reduceByKey: aggregate values of a key using a specified function
- Actions: return a value to the driver program after running a computation on the dataset
 - collect: returns all elements of the RDD to the driver
 - countByKey: counts the occurrences of each distinct key
- SQL functions in Spark
 - Offer a seamless and efficient way to perform complex data transformations and analyses. Somewhat similar to SQL
 - Interesting examples
 - Use SQL directly on DataFrame to filter and aggregate result
 - Use the window function in SparkSQL to perform computations over a range of rows related to the current rows.

8.

- I find the ML support in Spark impressively extensive. It offers a wide range of algorithms and utilities for machine-learning tasks involving scalability and integration with Spark's ecosystem. The system also simplifies the development process of machine learning systems.
- Basic statistics
 - This article gives a summary of basic statistic computation, including correlation, hypothesis testing with ChiSquareTest, and data summarization through Summarizer.
- Extracting, transforming, and selecting features

- Spark supports many feature-selecting tools, such as TF-IDF and Word2Vec, for text analysis, which is good for developing generative AI models.
- Classification and regression
 - Spark ML provides frameworks for complex algorithms better suited to handle real-world data. It promises to be more efficient than the current Python library as it can solve performance issues in many datasets.
- Clustering
 - This is mainly supported by the k-means algorithm to group instances that are in common. This can be helpful for handling large data structures and relationships.
- ML Tuning
 - This one emphasizes the importance of feature selection used in machine learning algorithms and pipelines. This can be used to find the attributes that are impactful in order to improve performance and accuracy.

9.

- Steps
 - Check Project ID
 - `echo ${GOOGLE_CLOUD_PROJECT}`
 - If it is not set
 - `export \`
`GOOGLE_CLOUD_PROJECT=PROJECT-ID`
 - Configure it to the current Project ID
 - `gcloud config set project \`

cnguyen4-cpsc324-hw4

- Set region
 - export REGION=us-west1
- Enable private access
 - gcloud compute networks subnets \

update default \

--region=\${REGION} \

--enable-private-ip-google-access
 - gcloud compute networks subnets \

describe default \

--region=\${REGION} \

--format="get(privateIpGoogleAccess)"
- Choose a bucket
 - export BUCKET=gs://cnguyen4-cpsc324-hw4-bucket1
 - export BUCKET=BUCKET

gsutil mb -l \${REGION} \

gs://\${BUCKET}
 - To create a new bucket
- Choose a dataset
 - export DATASET=DATASET

bq --location=\${REGION} mk -d \

\${DATASET}
- Set a name for the persistent history

- PHS_CLUSTER_NAME=cnguyen4-cpsc324-hw4-cluster1
- Create a new cluster
 - gcloud dataproc clusters create \

 \${PHS_CLUSTER_NAME} \

 --region=\${REGION} \

 --single-node \

 --enable-component-gateway \

 --

 properties=spark:spark.history.fs.logDirectory=gs://\${BUCKET}/phs/

 */spark-job-history
- Set a batch workload
 - BATCH_NAME=batch1
- Submit job
 - gcloud dataproc batches submit \

 pyspark citibike.py \

 --batch=\${BATCH_NAME} \

 --region=\${REGION} \

 --deps-bucket=gs://\${BUCKET} \

 --version=1.1 \

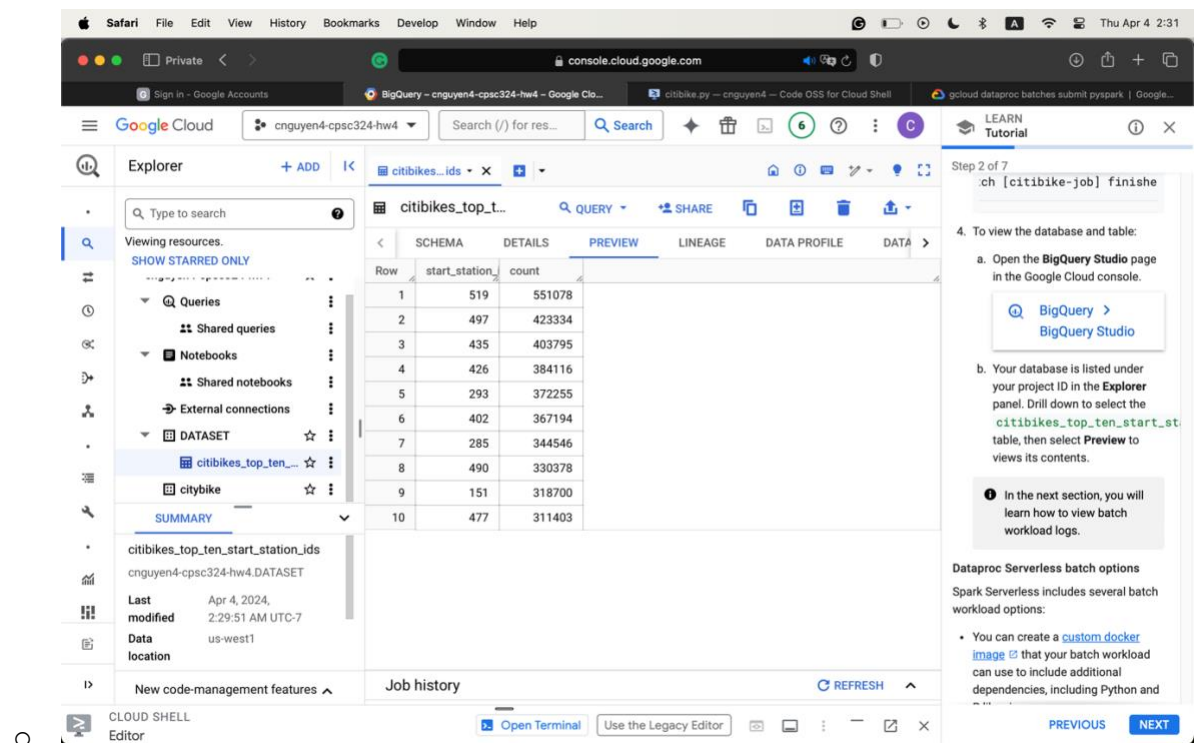
 --history-server-

 cluster=projects/\${GOOGLE_CLOUD_PROJECT}/regions/\${REGIO

 N}/clusters/\${PHS_CLUSTER_NAME} \

 -- \${DATASET}

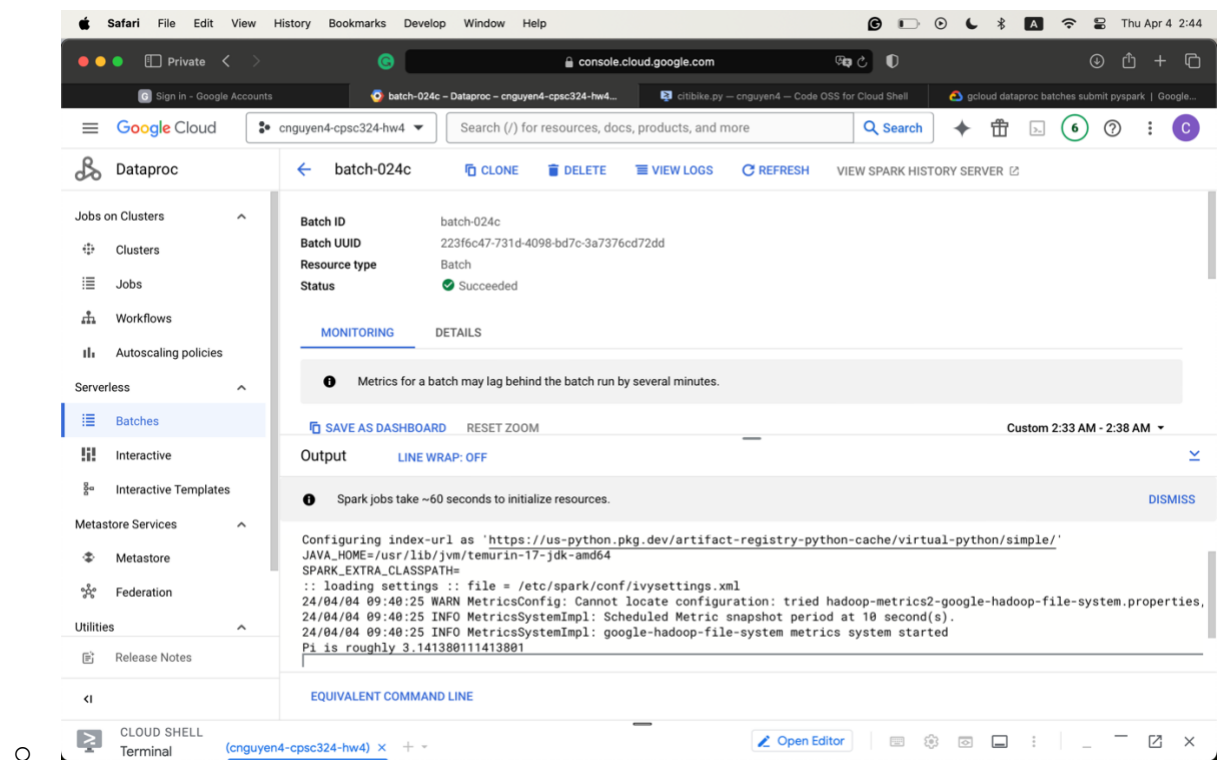
- Statement to create the data frame
 - `df = spark.read.format("bigquery").load(table)`
- Top ten data frame SQL equivalent:
 - `SELECT start_station_id, COUNT(*) AS count`
`FROM df`
`GROUP BY start_station_id`
`WHERE start_station_id IS NOT NULL`
`ORDER BY count ASC LIMIT 10;`
- Screenshot of BigQuery Studio



- I noticed that the command to submit the job was weird. It returned an HTTP 400 error without telling me exactly what happened. It turned out that the BATCH_NAME only allows (a-z) and numbers, and I did not know that. I also hard-coded the command.

10.

- Equivalent command line
 - `gcloud dataproc batches submit --project cnguyen4-cpsc324-hw4 --region us-west1 spark --batch batch-024c --class org.apache.spark.examples.SparkPi --version 2.2 --jars file:///usr/lib/spark/examples/jars/spark-examples.jar --subnet default -- 1000`
- Screenshot of resulting pi computation



- Running again but using Cloud Shell

The screenshot shows the Google Cloud console interface. The top navigation bar includes the Google Cloud logo, a search bar, and a user profile icon. The left sidebar contains navigation links for Jobs on Clusters, Clusters, Jobs, Workflows, and Release Notes. The main content area is titled 'Dataproc Batches' and includes buttons for 'CREATE', 'CREATE FROM TEMPLATE', and 'DELETE'. A warning message states: 'Starting March 31, 2024, special requirements apply to using CMEK with a batch.' Below this, a table lists batches with columns for Batch ID, Location, Status, Creation time, Elapsed time, and Type. Two batches are shown: 'batch-024d' and 'batch-024c', both in 'Succeeded' status. At the bottom, a 'CLOUD SHELL' terminal window is open, displaying the output of a Spark job. The output includes log messages, a warning about configuration, and a final message indicating the batch is finished. The terminal also shows the batch's metadata, including its ID, UID, location, creation time, and labels.

Batch ID	Location	Status	Creation time	Elapsed time	Type
batch-024d	us-west1	Succeeded	Apr 4, 2024, 2:46:07 AM	2 min 21 sec	Spark
batch-024c	us-west1	Succeeded	Apr 4, 2024, 2:38:21 AM	2 min 30 sec	Spark

```

JAVA_HOME=/usr/lib/jvm/temurin-17-jdk-amd64
SPARK_EXTRA_CLASSPATH=
:: loading settings :: file = /etc/spark/conf/ivysettings.xml
24/04/04 09:48:01 WARN MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-google-hadoop-file-system.properties,hadoop-metrics2.properties
24/04/04 09:48:01 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/04/04 09:48:01 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
Pi is roughly 3.14170799141708
Batch [batch-024d] finished.
metadata:
  @type: type.googleapis.com/google.cloud.dataproc.v1.BatchOperationMetadata
  batch: projects/cnguyen4-cpsc324-hw4/locations/us-west1/batches/batch-024d
  batchUuid: dc2cacc6-2e36-4fc2-8835-49e7ea91476b
  createTime: '2024-04-04T09:46:07.316976Z'
  description: Batch
  labels:
    goog-dataproc-batch-id: batch-024d
    goog-dataproc-batch-uid: dc2cacc6-2e36-4fc2-8835-49e7ea91476b
    goog-dataproc-location: us-west1
  operationType: BATCH
  name: projects/cnguyen4-cpsc324-hw4/regions/us-west1/operations/f3a695dc-15e2-3941-aal9-46e0a5f034b6
cnguyen4@cloudshell:~/devrel-demos/data-analytics/next-2022-workshop/dataproc-secnguyen4@cloudshell:~/devrel-demos/data-analytics/ncnguyen4@cloudshell:~/devrel-demos/data
cnguyen4@cloudshell:~/devrel-demos/data-analytics/next-2022-workshop/dataproc-serverless (cnguyen4-cpsc324-hw4) $

```

11.

- How is it different?
 - The wordcount.py of the other approaches uses the Spark functions liberally to match and count the word.
 - The wordcount.py, on the other hand, focuses on using SQL to find what the word count process is like.
- Paste a screenshot

The screenshot shows a Google Cloud Shell terminal window with the following content:

```

(cpsc322) tony@Tony's-MacBook-Pro: CPSC224 % bq ls
datasetid
-----
DATASET
citybike
wordcount_dataset
(cpsc322) tony@Tony's-MacBook-Pro: CPSC224 % gcloud dataproc batches submit --project cnguyen4-cpsc324-hw4 --region us-west1 spark --batch batch147c --class org.apache.spark.examples.SparkPi --version 2.2 --jars file:///usr/lib/spark/examples/jars/spark-examples.jar --subnet default -- 1000 --jars gs://spark-lib/bigquery/spark-bigquery-with-dependencies_2.13.jar
zsh: command not found: g
(cpsc322) tony@Tony's-MacBook-Pro: CPSC224 % gcloud dataproc batches submit --project cnguyen4-cpsc324-hw4 --region us-west1 spark --batch batch147c --class org.apache.spark.examples.SparkPi --version 2.2 --jars file:///usr/lib/spark/examples/jars/spark-examples.jar --subnet default -- 1000 --jars gs://spark-lib/bigquery/spark-bigquery-with-dependencies_2.13.jar
Batch [batch147c] submitted.
Using the default container image
Waiting for container log creation
PYSPARK_PYTHON=/opt/dataproc/conda/bin/python
Generating /home/spark/.pip/pip.conf
Configuring index-url as 'https://us-python.pkg.dev/artifact-registry-python-cache/virtual-python/simple/'
JAVA_HOME=/usr/lib/jvm/temurin-17-jdk-amd64
SPARK_EXTRA_CLASSPATH=
:: loading settings :: file = /etc/spark/conf/ivysettings.xml
24/04/04 16:11:42 WARN MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-google-hadoop-file-system.
properties.hadoop.metrics2.properties
24/04/04 16:11:42 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/04/04 16:11:42 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
P1 is roughly 3.141592653589793
Batch [batch147c] finished.
metadata:
  @type: type.googleapis.com/google.cloud.dataproc.v1.BatchOperationMetadata
  batch: projects/cnguyen4-cpsc324-hw4/locations/us-west1/batches/batch147c
  batchUuid: 4c23b900-d800-4522-bd86-c1940e4a295f
  createTime: '2024-04-04T16:09:56.867257Z'
  description: Batch
  labels:
    goog-dataproc-batch-id: batch147c
    goog-dataproc-batch-uuid: 4c23b900-d800-4522-bd86-c1940e4a295f
    goog-dataproc-location: us-west1
  operationType: BATCH
  name: projects/cnguyen4-cpsc324-hw4/regions/us-west1/operations/b9e7fa5f-7864-3f1f-bcfb-781b2e6de949
(cpsc322) tony@Tony's-MacBook-Pro: CPSC224 %
gcloud dataproc batches describe --project cnguyen4-cpsc324-hw4 --region us-west1 --batch batch147c
Running auto diagnostics on the batch. It may take few minutes before diagnostics output is available. Please check dia
gnostics output by running 'gcloud dataproc batches describe --project cnguyen4-cpsc324-hw4 --region us-west1 --batch batch147c'
cnguyen4@cloudshell:~ (cnguyen4-cpsc324-hw4) $
  
```

The file explorer on the left shows the project structure for 'CNGUYEN4' with files like README.md, CONTRIBUTING.md, LICENSE, and various scripts.