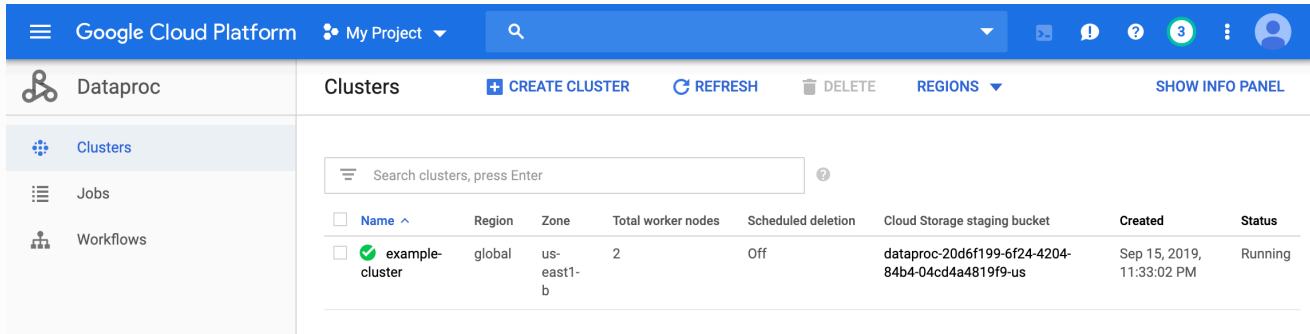


HOMEWORK 0 (E6893)

Jing Qian (jq2282)

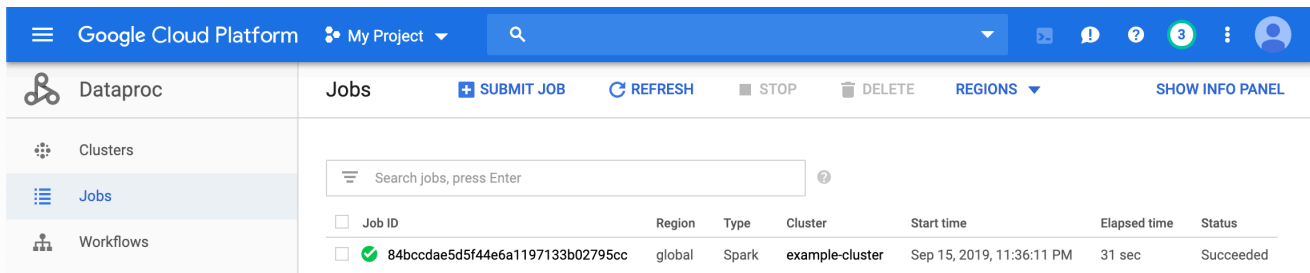
1. Warm-up exercises

1) Screenshots of exercise 3:



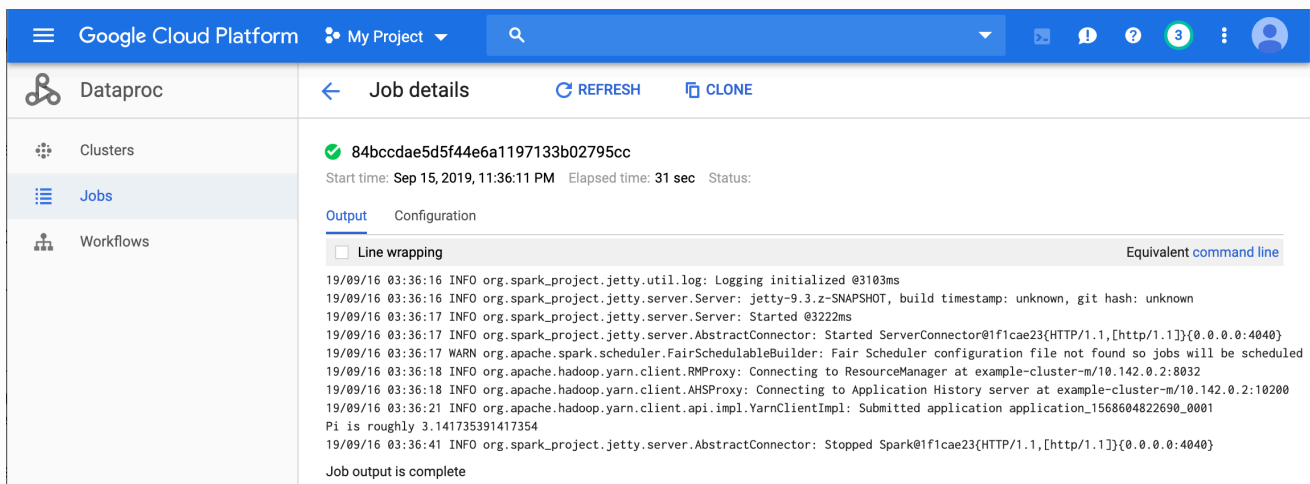
This screenshot shows the Google Cloud Platform interface for the 'Clusters' page. The left sidebar contains 'Dataproc', 'Clusters', 'Jobs', and 'Workflows'. The main content area has a search bar and a table of clusters. The table has columns: Name, Region, Zone, Total worker nodes, Scheduled deletion, Cloud Storage staging bucket, Created, and Status. One cluster is listed: 'example-cluster' in the 'global' region, 'us-east1-b' zone, with 2 worker nodes, scheduled for deletion 'Off', and a staging bucket 'dataproc-20d6f199-6f24-4204-84b4-04cd4a4819f9-us'. It was created on 'Sep 15, 2019, 11:33:02 PM' and is in 'Running' status.

Name	Region	Zone	Total worker nodes	Scheduled deletion	Cloud Storage staging bucket	Created	Status
example-cluster	global	us-east1-b	2	Off	dataproc-20d6f199-6f24-4204-84b4-04cd4a4819f9-us	Sep 15, 2019, 11:33:02 PM	Running



This screenshot shows the Google Cloud Platform interface for the 'Jobs' page. The left sidebar contains 'Dataproc', 'Clusters', 'Jobs', and 'Workflows'. The main content area has a search bar and a table of jobs. The table has columns: Job ID, Region, Type, Cluster, Start time, Elapsed time, and Status. One job is listed: '84bccdae5d5f44e6a1197133b02795cc' in the 'global' region, 'Spark' type, on 'example-cluster', started on 'Sep 15, 2019, 11:36:11 PM', with an elapsed time of '31 sec', and status 'Succeeded'.

Job ID	Region	Type	Cluster	Start time	Elapsed time	Status
84bccdae5d5f44e6a1197133b02795cc	global	Spark	example-cluster	Sep 15, 2019, 11:36:11 PM	31 sec	Succeeded



This screenshot shows the Google Cloud Platform interface for the 'Job details' page. The left sidebar contains 'Dataproc', 'Clusters', 'Jobs', and 'Workflows'. The main content area shows details for job '84bccdae5d5f44e6a1197133b02795cc'. It includes the start time 'Sep 15, 2019, 11:36:11 PM', elapsed time '31 sec', and status 'Succeeded'. There are tabs for 'Output' and 'Configuration'. The 'Output' tab is selected, showing a log of Spark job execution. The log includes information about logging initialization, server startup, and the execution of a Spark job. The job output is complete.

Job output is complete

Screenshots of exercise 4:

```
trackingUrl: http://example-cluster-m:8088/proxy/application_1568604822690_0003/
jq2282@cloudshell:~ (winged-plate-252922) $ gsutil cat gs://example-bucket-jq/output/*
(u'a', 2)
(u'we', 1)
(u'would', 1)
(u'What's', 1)
(u'sweet.', 1)
(u'as', 1)
(u'call', 1)
(u'which', 1)
(u'smell', 1)
(u'name', 1)
(u'That', 1)
(u'rose', 1)
(u'any', 1)
```

2)

Transformations in Exercise3: filter().

Actions in Exercise3: count(). The RDD operation that triggers the program to execute is the actions and hence "count()".

Transformations in Exercise4: flatMap(), map(), reduceByKey().

Actions in Exercise4: saveAsTextFile(). The RDD operation that triggers the program to execute is the actions and hence "saveAsTextFile()".

*Exercise 3 is an inside example and corresponding code is found at: <https://spark.apache.org/examples.html>. The code for Exercise 4 is provided in the given link: <https://cloud.google.com/dataprocs/docs/tutorials/gcs-connector-spark-tutorial>.

2. NYC Bike expert

1) There are 843 unique station_ids in this dataset.

Google Cloud Platform My Project

BigQuery FEATURES & INFO SHORTCUTS + COMPOSE NEW QUERY

Query history

Query editor

```
1 SELECT Count(DISTINCT station_id) FROM Bikes.Bikes
```

Valid.

Run Save query Save view Schedule query More

This query will process 6.6 KB when run.

Query results SAVE RESULTS EXPLORE WITH DATA STUDIO

Query complete (1.7 sec elapsed, 6.6 KB processed)

Job information Results JSON Execution details

Row	fo_
1	843

2) The largest capacity for a station is 79. The *station_id* of stations that have the largest capacity are: 445, 422, 501.

Google Cloud Platform My Project

BigQuery FEATURES & INFO SHORTCUTS + COMPOSE NEW QUERY

Query history

Query editor

```
1 SELECT station_id, capacity
2 FROM Bikes.Bikes
3 ORDER BY capacity DESC
4 LIMIT 5
```

Valid.

Run Save query Save view Schedule query More

This query will process 13.2 KB when run.

Query results SAVE RESULTS EXPLORE WITH DATA STUDIO

Query complete (0.5 sec elapsed, 13.2 KB processed)

Job information Results JSON Execution details

Row	station_id	capacity
1	422	79
2	501	79
3	445	79
4	442	77

3) The total number of bikes available in region_id 70 is 394.

Google Cloud Platform My Project

BigQuery FEATURES & INFO SHORTCUTS + COMPOSE NEW QUERY

Query history

Query editor

```

1 SELECT Sum(num_bikes_available)
2 FROM bikes.Bikes AS B
3 WHERE B.region_id = 70

```

Valid.

Run Save query Save view Schedule query More

This query will process 13.2 KB when run.

Query results SAVE RESULTS EXPLORE WITH DATA STUDIO

Query complete (0.3 sec elapsed, 13.2 KB processed)

Job information Results JSON Execution details

Row	f0_
1	394

3. Understanding William Shakespeare

1) The top 5 frequent words without any text preprocessing are: (('the', 620), ('and', 427), ('of', 396), ('to', 367), ('I', 326)).

Jupyter Shakespeare (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted PySpark

Run Code

```

In [22]: # Import text from bucket shakes0
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
url = "gs://shakes0/shakes.txt"
txt = sc.textFile(url)

In [25]: # Method 1: learn from wordCounts, without text preprocessing
words = txt.flatMap(lambda line: line.split())
# words.collect()
wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, c
wordCounts.takeOrdered(5, key=lambda x: -x[1])

Out[25]: [(u'the', 620), (u'and', 427), (u'of', 396), (u'to', 367), (u'I', 326)]

```

2) Top 5 frequent words by filtering out stop words provided by NLTK package are: (('macb.', 137), ('haue', 119), ('thou', 85), ('enter', 74), ('shall', 67)).

```
In [2]: from pyspark import SparkContext
sc = SparkContext.getOrCreate()
url = 'gs://jings/shakes.txt'
txt = sc.textFile(url)
words = txt.flatMap(lambda line: line.split())
words = words.map(lambda word: word.lower())
```

```
In [3]: import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
words = words.filter(lambda word : word not in stop_words)
wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda count1, c
wordCounts.takeOrdered(5, key=lambda x: -x[1])
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
Out[3]: [(u'macb.', 137),
         (u'haue', 119),
         (u'thou', 85),
         (u'enter', 74),
         (u'shall', 67)]
```