Spark in GCP

Use all available cores

Recursive (because it contains subfolders)

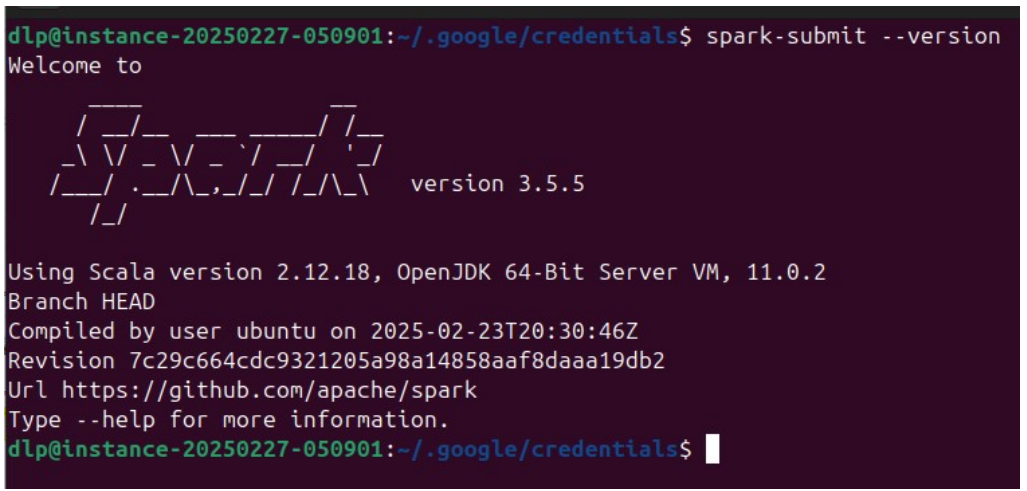
- First, the data must be uploaded to a bucket.
- Create a bucket and use this command:

  gsutil -m cp -r pq/ gs://dlp_2025_dateng_week_05/pq

- If the access is denied, try this:
  1) Stop VM
  2) goto --> VM instance details.
  3) in "Cloud API access scopes" select "Allow full access to all Cloud APIs" then Click "save".
  4) restart VM and Delete ~/.gsutil .

Spark in GCP

- We also need a hadoop connector to GCS, so that Spark can connect to the GCS bucket.
- The connector has to be compatible with the installed Spark version:




```
gsutil cp gs://hadoop-lib/gcs/gcs-connector-hadoop3-2.2.5.jar ~/lib/
```

Spark in GCP

localhost:8888/lab/tree/notebooks/09_spark_gcs_dlp.ipynb

gcp connector hadoop

Settings Help

Test_PySpark_Install.ipynb | 04_PySpark_dlp.ipynb | 05_PySpark_dlp.ipynb | 06_spark_sql_dlp.ipynb | 08_rdds_dlp.ipynb | 09_spark_gcs_dlp.ipynb

Code

[1]:
```
import pyspark
from pyspark.sql import SparkSession
from pyspark.conf import SparkConf
from pyspark.context import SparkContext
```

[2]:
```
credentials_location = '/home/dlp/.google/credentials/google_credentials.json'

conf = SparkConf() \
    .setMaster('local[*]') \
    .setAppName('test') \
    .set("spark.jars", "../lib/gcs-connector-hadoop3-2.2.5.jar") \
    .set("spark.hadoop.google.cloud.auth.service.account.enable", "true") \
    .set("spark.hadoop.google.cloud.auth.service.account.json.keyfile", credentials_location)
```

[3]:
```
sc = SparkContext(conf=conf)

hadoop_conf = sc._jsc.hadoopConfiguration()

hadoop_conf.set("fs.AbstractFileSystem.gs.impl",  "com.google.cloud.hadoop.fs.gcs.GoogleHadoopFS")
hadoop_conf.set("fs.gs.impl", "com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem")
hadoop_conf.set("fs.gs.auth.service.account.json.keyfile", credentials_location)
hadoop_conf.set("fs.gs.auth.service.account.enable", "true")
```

```
25/03/10 03:41:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

[4]:
```
spark = SparkSession.builder \
    .config(conf=sc.getConf()) \
    .getOrCreate()
```

[5]:
```
df_green = spark.read.parquet('gs://dlp_2025_dateng_week_05/pq/green/*/*')
```

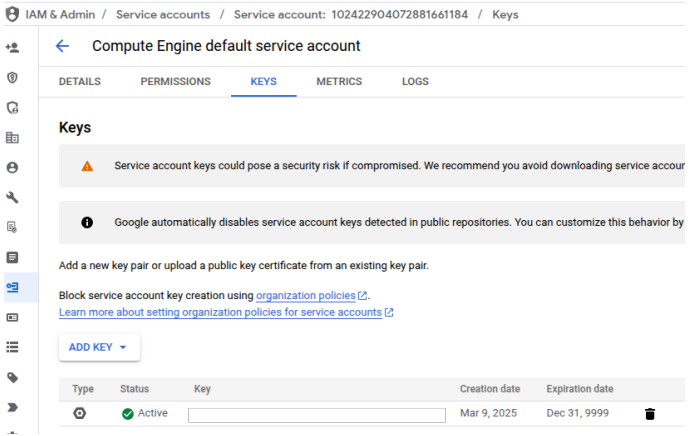
[6]:
```
df_green.count()
```

[6]:
```
2304517
```

Distribution of directories in VM

```
dlp@instance-20250227-050901: ~
dlp@instance-20250227-050901:~$ ls
Programs  data  de-zoomcamp  download_data.sh  lib  notebooks  snap  spark  tmp
dlp@instance-20250227-050901:~$
```

This credentials is the .json key for the VM instance service account



Creating a Local Spark Cluster

https://spark.apache.org/docs/latest/spark-standalone.html#installing-spark-standalone-to-a-cluster

`./sbin/start-master.sh`

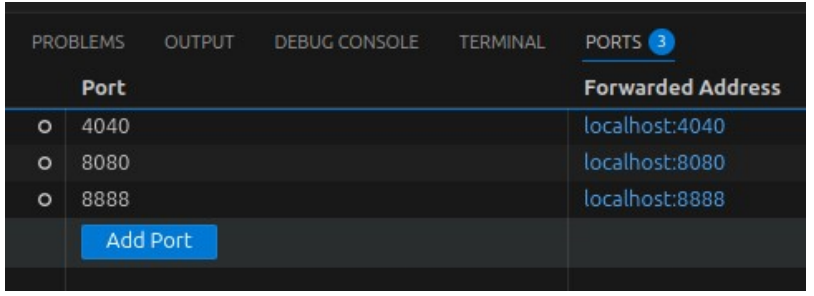
- Go to the directory saved as SPARK_HOME

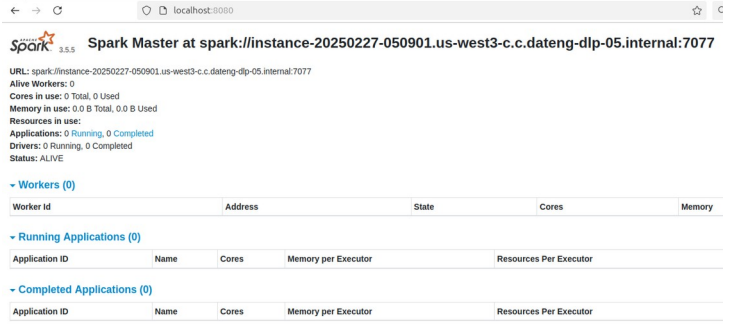
```
dlp@instance-20250227-050901:~/spark$ echo $SPARK_HOME
/home/dlp/spark/spark-3.5.5-bin-hadoop3
```

- `./sbin/start-master.sh`

```
dlp@instance-20250227-050901:~/spark/spark-3.5.5-bin-hadoop3$ ./sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /home/dlp/spark/spark-3.5.5-bin-hadoop3/logs
Master-1-instance-20250227-050901.out
dlp@instance-20250227-050901:~/spark/spark-3.5.5-bin-hadoop3$
```

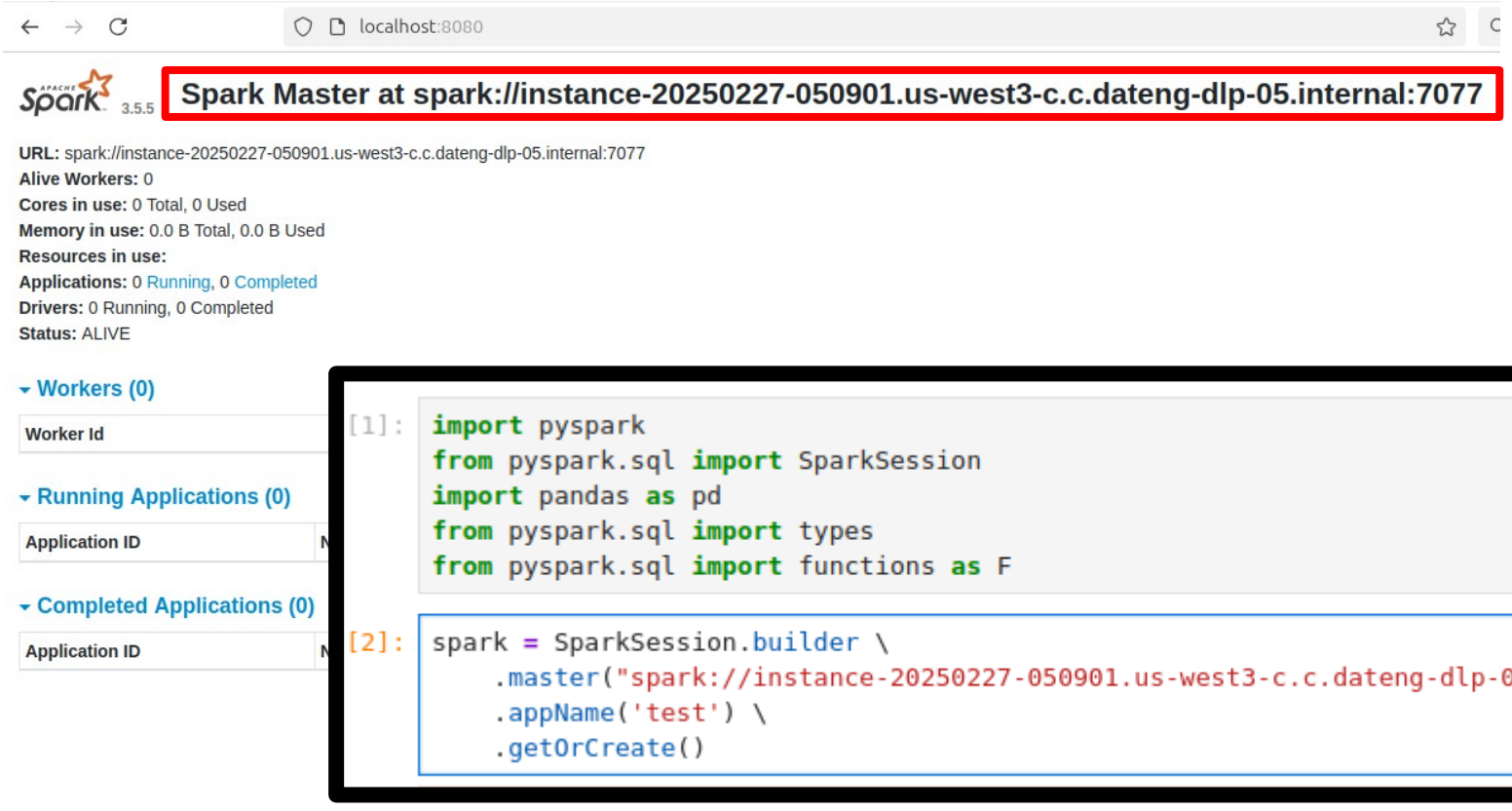
- This local cluster can be accessed on Port 8080, so map this port (VM Instance → local host) and check the UI in a browser





Creating a Local Spark Cluster

- Now change the master from local to the created cluster



Creating a Local Spark Cluster

- A Spark session can now be created but we need to manually create workers in order for the cluster to be able to do anything.

- Now the script to obtain the revenue can be run (with input parameters)

```
./sbin/start-worker.sh <master-spark-URL>
```

10_spark_sql_local_cluster_dlp.py

notebooks > 10_spark_sql_local_cluster_dlp.py

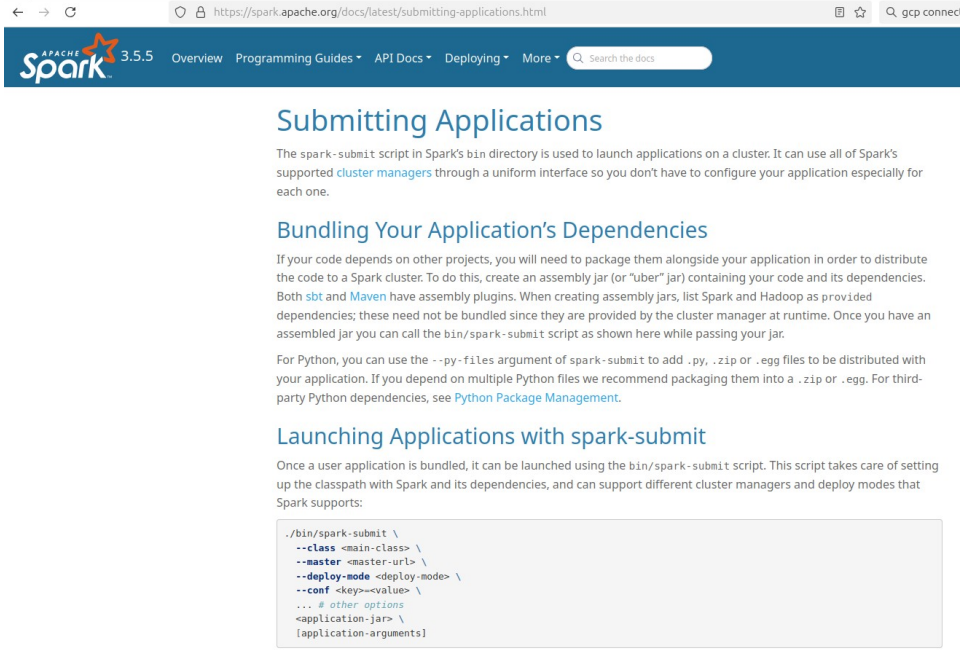
```python
import argparse
import pyspark
from pyspark.sql import SparkSession
import pandas as pd
from pyspark.sql import types
from pyspark.sql import functions as F

parser = argparse.ArgumentParser()

parser.add_argument('--input_green', required=True)
parser.add_argument('--input_yellow', required=True)
parser.add_argument('--output', required=True)

args = parser.parse_args()

input_green = args.input_green
input_yellow = args.input_yellow
output = args.output


spark = SparkSession.builder \
    .master("spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077") \
    .appName('test') \
    .getOrCreate()


# Read green taxi data
df_green = spark.read.parquet(input_green)
df_green.printSchema()
```

Creating a Local Spark Cluster

- However, we don't want to hard code the Spark master, we want to be able to define the number of executors and other settings.
- We use Spark Submit for this!

```
# spark = SparkSession.builder \
#     .master("spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077") \
#     .appName('test') \
#     .getOrCreate()

spark = SparkSession.builder \
    .appName('test') \
    .getOrCreate()
```

https://spark.apache.org/docs/latest/submitting-applications.html



Submitting jobs with spark-submit

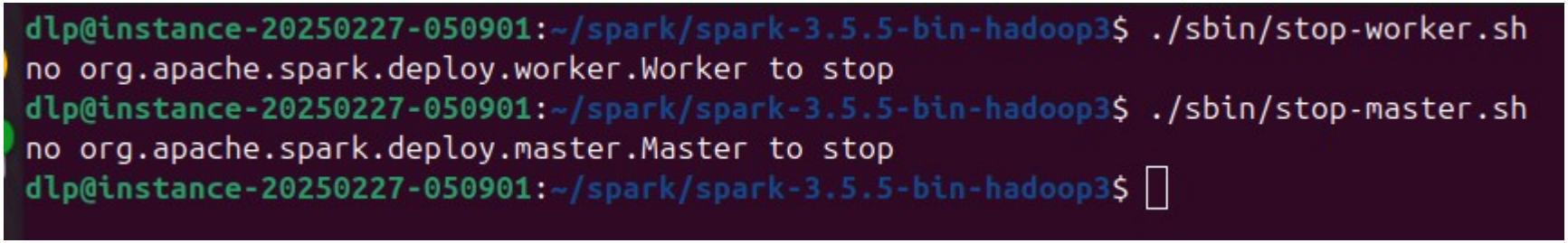
```
URL="spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077"

spark-submit \
    --master="${URL}" \
    10_spark_sql_local_cluster_dlp.py \
    --input_green='../data/pq/green/2021/*' \
    --input_yellow='../data/pq/yellow/2021/*' \
    --output='../data/report-2021'
```

```
(de-zoomcamp-py3.12) dlp@instance-20250227-050901:~/notebooks$ URL="spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077"
(de-zoomcamp-py3.12) dlp@instance-20250227-050901:~/notebooks$
(de-zoomcamp-py3.12) dlp@instance-20250227-050901:~/notebooks$ spark-submit \
> --master="${URL}" \
> 10_spark_sql_local_cluster_dlp.py \
> --input_green='../data/pq/green/2021/*' \
> --input_yellow='../data/pq/yellow/2021/*' \
> --output='../data/report-2021'
```

Shutting down the Spark Cluster

- Once the Spark job is done, both the workers/executors and the master/cluster must be closed.

- Navigate to the folder where Spark was installed

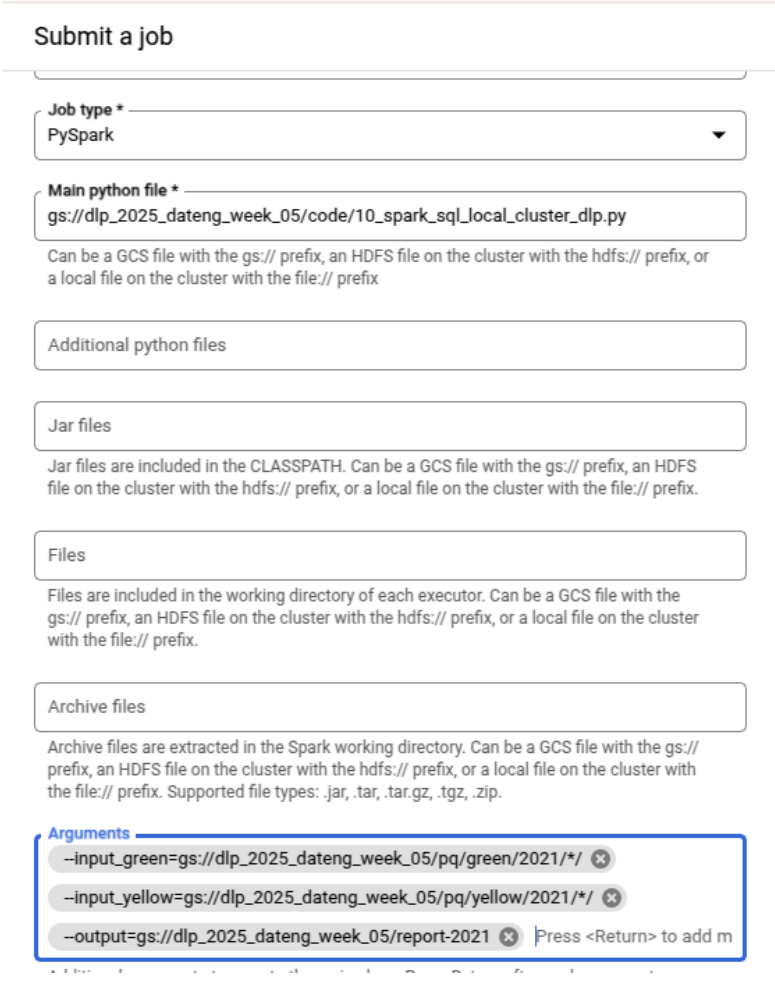


Create Dataproc cluster

- Enable API
- Add Dataproc Admin role
- Create Cluster
- Submit Job manually:
  - Select Job Type
  - Save a python script to a bucket and use this as the main python file:

    ```
    gsutil cp notebooks/10_spark_sql_local_cluster_dlp.py gs://dlp_2025_dateng_week_05/code/10_spark_sql_local_cluster_dlp.py
    ```

  - Specify arguments
- Copy commands to do it programatically:
  - Job → Configuration → Equivalent REST->Copy



Submitting job to Dataproc cluster

- Extract important info from REST API:


```
{
  "reference": {
    "jobId": "job-670f62fd",
    "projectId": "dateng-dlp-05"
  },
  "placement": {
    "clusterName": "dateng-dlp-cluster"
  },
  "status": {
    "state": "DONE",
    "stateStartTime": "2025-03-11T03:50:20.909443Z"
  },
  "yarnApplications": [
    {
      "name": "test",
      "state": "FINISHED",
      "progress": 1,
      "trackingUrl": "http://dateng-dlp-cluster-m.local.:8088/proxy/application_1741664101246_0001/"
    }
  ],
  "statusHistory": [
    {
      "state": "PENDING",
      "stateStartTime": "2025-03-11T03:48:50.311087Z"
    },
    {
      "state": "SETUP_DONE",
      "stateStartTime": "2025-03-11T03:48:50.342277Z"
    },
    {
      "state": "RUNNING",
      "details": "Agent reported job success",
      "stateStartTime": "2025-03-11T03:48:50.995136Z"
    }
  ],
  "driverControlFilesUri": "gs://dataproc-staging-us-west3-129694554536-2mbquz3u/google-cloud-dataproc-metainfo/bd096dcf-61a3-41b4-ac9f-bbd769f8445d/jobs/job-670f62fd/",
  "driverOutputResourceUri": "gs://dataproc-staging-us-west3-129694554536-2mbquz3u/google-cloud-dataproc-metainfo/bd096dcf-61a3-41b4-ac9f-bbd769f8445d/jobs/job-670f62fd/driveroutput",
  "jobUuid": "7a298efe-1c78-4975-a2a5-e362ccb2100c",
  "done": true,
  "pysparkJob": {
    "mainPythonFileUri": "gs://dlp_2025_dateng_week_05/code/10_spark_sql_local_cluster_dlp.py",
    "args": [
      "--input_green=gs://dlp_2025_dateng_week_05/pq/green/2021/*/",
      "--input_yellow=gs://dlp_2025_dateng_week_05/pq/yellow/2021/*/",
      "--output=gs://dlp_2025_dateng_week_05/report-2021"
    ]
  }
}
```

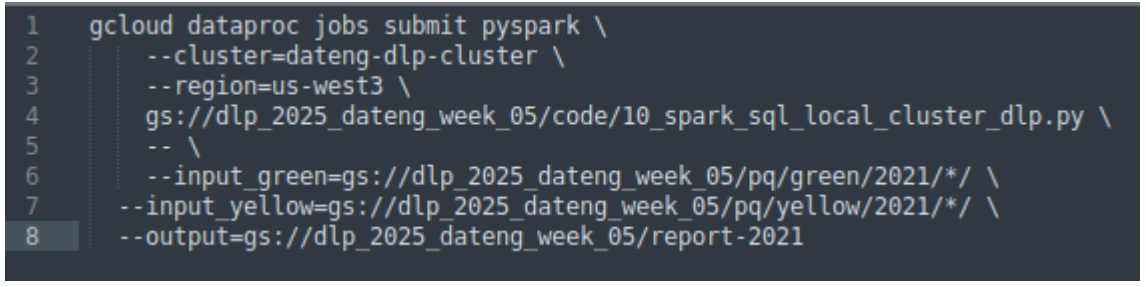
Submitting job to Dataproc cluster

- Use Google Cloud SDK to submit Dataproc job

https://cloud.google.com/dataproc/docs/guides/submit-job#dataproc-submit-job-gcloud

Anything after this -- is considered parameters for the script

```
gcloud dataproc jobs submit pyspark \
    --cluster=dateng-dlp-cluster \
    --region=us-west3 \
   gs://dlp_2025_dateng_week_05/code/10_spark_sql_local_cluster_dlp.py \
    -- \
        --input_green=gs://dlp_2025_dateng_week_05/pq/green/2021/*/ \
        --input_yellow=gs://dlp_2025_dateng_week_05/pq/yellow/2021/*/ \
        --output=gs://dlp_2025_dateng_week_05/report-2021
```



Connecting Spark to Big Query

- Create a dataset (same region as bucket and Dataproc cluster)
- Submit job:

```
gcloud dataproc jobs submit pyspark \
  --cluster=dateng-dlp-cluster \
  --region=us-west3 \
  --jars=file:///usr/lib/spark/examples/jars/spark-examples.jar \
  gs://dlp_2025_dateng_week_05/code/10_spark_sql_big_query_dlp.py \
  -- \
        --input_green=gs://dlp_2025_dateng_week_05/pq/green/2020/*/ \
        --input_yellow=gs://dlp_2025_dateng_week_05/pq/yellow/2020/*/ \
        --output=trips_data_all.reports-2020
```