

Spark in GCP

Use all available cores

- 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
```

Recursive (because it contains subfolders)

- 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:

```
dlp@instance-20250227-050901:~/.google/credentials$ spark-submit --version
Welcome to

  ____      _
 / ___|    / \
| |  | |  / _ \
| |  | | / ___ \
| |  | || |___| |
| |  | || |___| |
| |  | || |___| |
|_|  |_| \_____/

version 3.5.5

Using Scala version 2.12.18, OpenJDK 64-Bit Server VM, 11.0.2
Branch HEAD
Compiled by user ubuntu on 2025-02-23T20:30:46Z
Revision 7c29c664cdc9321205a98a14858aaf8daaa19db2
Url https://github.com/apache/spark
Type --help for more information.
dlp@instance-20250227-050901:~/.google/credentials$
```

```
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
[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:~$ 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

IAM & Admin / Service accounts / Service account: 102422904072881661184 / Keys

Compute Engine default service account

DETAILS PERMISSIONS KEYS METRICS LOGS

Keys

Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys.

Google automatically disables service account keys detected in public repositories. You can customize this behavior by [configuring Google Cloud IAM](#).

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using [organization policies](#) (2). [Learn more about setting organization policies for service accounts](#) (2)

ADD KEY

Type	Status	Key	Creation date	Expiration date	
Key	Active		Mar 9, 2025	Dec 31, 9999	

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

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS 3

| Port     | Forwarded Address |
|----------|-------------------|
| ○ 4040   | localhost:4040    |
| ○ 8080   | localhost:8080    |
| ○ 8888   | localhost:8888    |
| Add Port |                   |

Spark 3.5.5 Spark Master at spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077

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

Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used

Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

Workers (0)

| Worker Id | Address | State | Cores | Memory |
|-----------|---------|-------|-------|--------|
|-----------|---------|-------|-------|--------|

Running Applications (0)

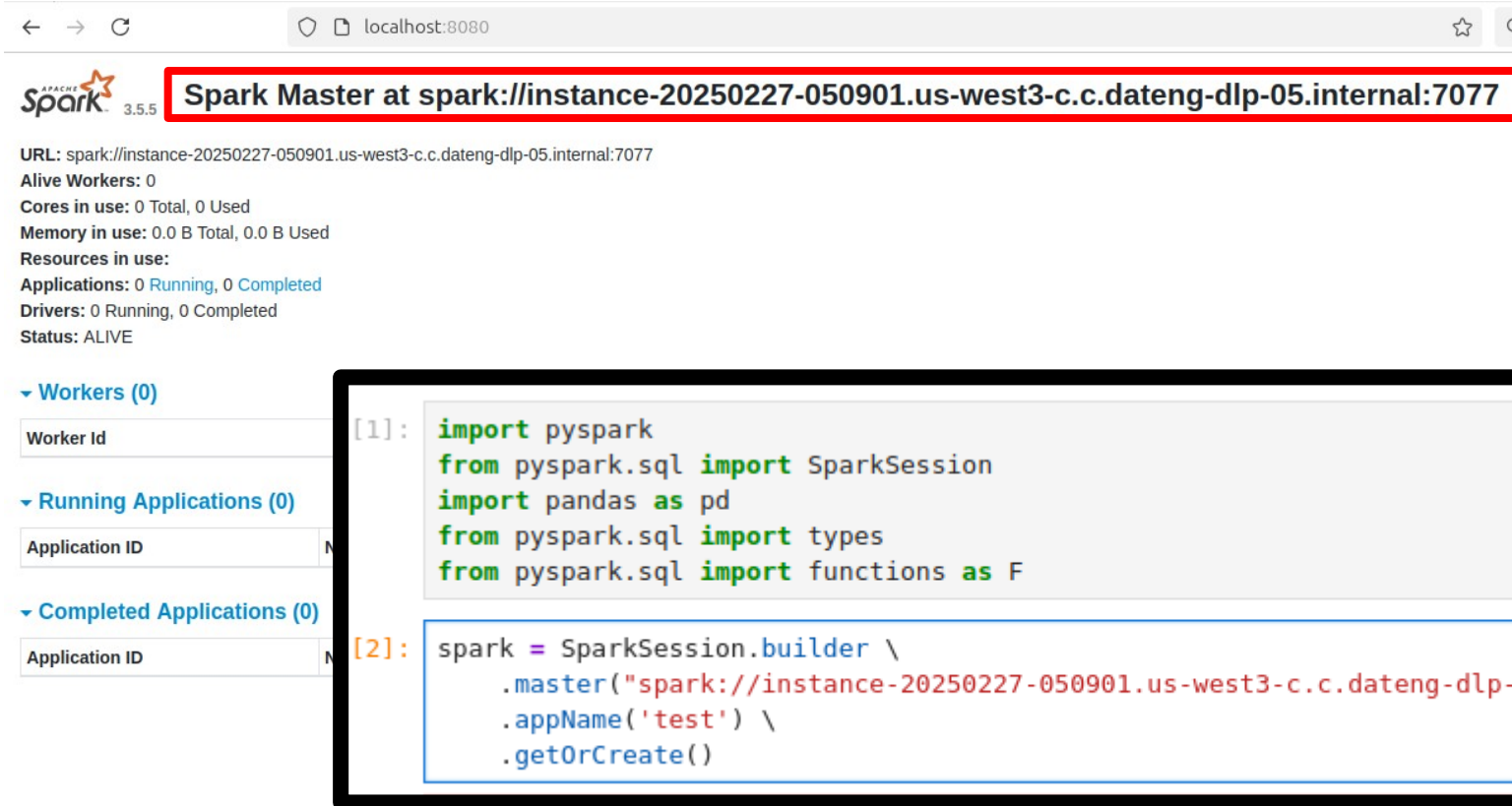
| Application ID | Name | Cores | Memory per Executor | Resources Per Executor |
|----------------|------|-------|---------------------|------------------------|
|----------------|------|-------|---------------------|------------------------|

Completed Applications (0)

| Application ID | Name | Cores | Memory per Executor | Resources Per Executor |
|----------------|------|-------|---------------------|------------------------|
|----------------|------|-------|---------------------|------------------------|

# Creating a Local Spark Cluster

- Now change the master from local to the created cluster



The screenshot shows the Databricks Spark UI interface. The browser address bar displays `localhost:8080`. The main header shows the Apache Spark logo and version 3.5.5, followed by the Spark Master URL: `spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077`, which is highlighted with a red box. Below the header, the URL is repeated, and various status metrics are listed: `Alive Workers: 0`, `Cores in use: 0 Total, 0 Used`, `Memory in use: 0.0 B Total, 0.0 B Used`, `Resources in use:`, `Applications: 0 Running, 0 Completed`, `Drivers: 0 Running, 0 Completed`, and `Status: ALIVE`. On the left sidebar, there are sections for `Workers (0)`, `Running Applications (0)`, and `Completed Applications (0)`. The main content area displays two code blocks. The first block, labeled `[1]:`, contains the following Python code: 

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

 The second block, labeled `[2]:`, contains the following Python code: 

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

# 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>
```

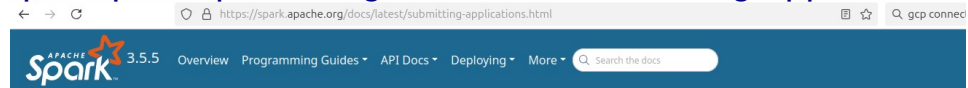
```
10_spark_sql_local_cluster_dlp.py X
notebooks > 10_spark_sql_local_cluster_dlp.py
1 import argparse
2 import pyspark
3 from pyspark.sql import SparkSession
4 import pandas as pd
5 from pyspark.sql import types
6 from pyspark.sql import functions as F
7
8 parser = argparse.ArgumentParser()
9
10 parser.add_argument('--input_green', required=True)
11 parser.add_argument('--input_yellow', required=True)
12 parser.add_argument('--output', required=True)
13
14 args = parser.parse_args()
15
16 input_green = args.input_green
17 input_yellow = args.input_yellow
18 output = args.output
19
20
21 spark = SparkSession.builder \
22 .master("spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077") \
23 .appName('test') \
24 .getOrCreate()
25
26
27 # Read green taxi data
28 df_green = spark.read.parquet(input_green)
29 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!

```
21 # spark = SparkSession.builder \
22 # .master("spark://instance-20250227-050901.us-west3-c.c.dateng-dlp-05.internal:7077") \
23 # .appName('test') \
24 # .getOrCreate() \
25 \
26 spark = SparkSession.builder \
27 .appName('test') \
28 .getOrCreate()
```

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



## Submitting Applications

The `spark-submit` script in Spark's bin directory is used to launch applications on a cluster. It can use all of Spark's supported [cluster managers](#) through a uniform interface so you don't have to configure your application especially for each one.

## Bundling Your Application's Dependencies

If your code depends on other projects, you will need to package them alongside your application in order to distribute the code to a Spark cluster. To do this, create an assembly jar (or "uber" jar) containing your code and its dependencies. Both [sbt](#) and [Maven](#) have assembly plugins. When creating assembly jars, list Spark and Hadoop as provided dependencies; these need not be bundled since they are provided by the cluster manager at runtime. Once you have an assembled jar you can call the `bin/spark-submit` script as shown here while passing your jar.

For Python, you can use the `--py-files` argument of `spark-submit` to add `.py`, `.zip` or `.egg` files to be distributed with your application. If you depend on multiple Python files we recommend packaging them into a `.zip` or `.egg`. For third-party Python dependencies, see [Python Package Management](#).

## Launching Applications with spark-submit

Once a user application is bundled, it can be launched using the `bin/spark-submit` script. This script takes care of setting up the classpath with Spark and its dependencies, and can support different cluster managers and deploy modes that Spark supports:

```
./bin/spark-submit \
--class <main-class> \
--master <master-url> \
--deploy-mode <deploy-mode> \
--conf <key>=<value> \
... # other options \
<application-jar> \
[application-arguments]
```

# 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

```
dlp@instance-20250227-050901:~/spark/spark-3.5.5-bin-hadoop3$./sbin/stop-worker.sh
no org.apache.spark.deploy.worker.Worker to stop
dlp@instance-20250227-050901:~/spark/spark-3.5.5-bin-hadoop3$./sbin/stop-master.sh
no org.apache.spark.deploy.master.Master to stop
dlp@instance-20250227-050901:~/spark/spark-3.5.5-bin-hadoop3$
```

# 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 programmatically:
  - Job → Configuration → Equivalent REST->Copy

## Submit a job

Job type \*

PySpark

Main python file \*

gs://dlp\_2025\_dateng\_week\_05/code/10\_spark\_sql\_local\_cluster\_dlp.py

Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix

Additional python files

Jar files

Jar files are included in the CLASSPATH. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Files

Files are included in the working directory of each executor. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Archive files

Archive files are extracted in the Spark working directory. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix. Supported file types: .jar, .tar, .tar.gz, .tgz, .zip.

Arguments

--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 ✕ Press <Return> to add m

# Submitting job to Dataproc cluster

- Extract important info from REST API:

```
1 {
2 "reference": {
3 "jobId": "job-670f62fd",
4 "projectId": "dateng-dlp-05"
5 },
6 "placement": {
7 "clusterName": "dateng-dlp-cluster"
8 },
9 "status": {
10 "state": "DONE",
11 "stateStartTime": "2025-03-11T03:50:20.909443Z"
12 },
13 "yarnApplications": [
14 {
15 "name": "test",
16 "state": "FINISHED",
17 "progress": 1,
18 "trackingUrl": "http://dateng-dlp-cluster-m.local.:8088/proxy/application_1741664101246_0001/"
19 }
20],
21 "statusHistory": [
22 {
23 "state": "PENDING",
24 "stateStartTime": "2025-03-11T03:48:50.311087Z"
25 },
26 {
27 "state": "SETUP_DONE",
28 "stateStartTime": "2025-03-11T03:48:50.342277Z"
29 },
30 {
31 "state": "RUNNING",
32 "details": "Agent reported job success",
33 "stateStartTime": "2025-03-11T03:48:50.995136Z"
34 }
35],
36 "driverControlFilesUri": "gs://dataproc-staging-us-west3-129694554536-2mbquz3u/google-cloud-dataproc-metainfo/bd096dcf-61a3-41b4-ac9f-bbd769f8445d/jobs/job-670f62fd/",
37 "driverOutputResourceUri": "gs://dataproc-staging-us-west3-129694554536-2mbquz3u/google-cloud-dataproc-metainfo/bd096dcf-61a3-41b4-ac9f-bbd769f8445d/jobs/job-670f62fd/driveroutput",
38 "jobUuid": "7a298efe-1c78-4975-a2a5-e362ccb2100c",
39 "done": true
40 "pysparkJob": {
41 "mainPythonFileUri": "gs://dlp_2025_dateng_week_05/code/10_spark_sql_local_cluster_dlp.py",
42 "args": [
43 "--input_green=gs://dlp_2025_dateng_week_05/pq/green/2021/*/",
44 "--input_yellow=gs://dlp_2025_dateng_week_05/pq/yellow/2021/*/",
45 "--output=gs://dlp_2025_dateng_week_05/report-2021"
46]
47 }
48 }
```

# 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>

```
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
```

Anything after this -- is  
considered parameters for  
the script

```
1 gcloud dataproc jobs submit pyspark \
2 --cluster=dateng-dlp-cluster \
3 --region=us-west3 \
4 gs://dlp_2025_dateng_week_05/code/10_spark_sql_local_cluster_dlp.py \
5 -- \
6 --input_green=gs://dlp_2025_dateng_week_05/pq/green/2021/*/ \
7 --input_yellow=gs://dlp_2025_dateng_week_05/pq/yellow/2021/*/ \
8 --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
```