

2019-fall-group-project-ricardo-jacob-khalil

2019-fall-group-project-Ricardo-Jacob-Khalil created by GitHub Classroom

Project Members

- Jacob Jablonski
- Khalil Hanna
- Ricardo Rocha

Communication Platform

- [Slack \(https://app.slack.com/client/THRKMGT55/GPHGO9SH0\)](https://app.slack.com/client/THRKMGT55/GPHGO9SH0)

Contents

- [Collisions Routieres Dataset](#)
- [Downloading the data](#)
- [Creating Volume](#)
- [Services on Spark Cluster](#)
- [Preparing Data and Work Environment](#)
- [Store the data in HDFS and MongoDB](#)
- [Automated Storage](#)
- [Using Jupyter Notebook](#)
- [Google Cloud Platform](#)

Collisions Routieres Dataset



The screenshot shows the Montréal Open Data Portal interface. At the top, the Montréal logo is on the left, and navigation links 'Nous joindre' and 'Suivez-nous' with social media icons are on the right. Below this is a red banner with 'PORTAIL DONNÉES OUVERTES' and icons for a circular refresh and a user profile. A navigation bar contains links for 'Données', 'Démarche', 'Applications', 'Aide et entraide', 'Actualités', and 'English', along with a search icon. The main content area has a sidebar on the left with 'Thématiques' (Loi, justice et sécurité publique; Transport) and 'Partager' (Facebook, Google+, Twitter). The main panel shows the 'Collisions routières' dataset, with tabs for 'Ensemble de données', 'Thématiques', and 'Flux d'Activité'. The text describes the dataset as a list of collisions in Montréal since 2012, including descriptive, contextual, and location data, as well as severity levels like fatalities, serious injuries, minor injuries, and property damage.

Description

List of collisions that have occurred in Montreal since 2012.

This set includes collisions involving at least one motor vehicle circulating on the network and which have been the subject of a police report. It includes descriptive, contextual and event location information, including seriousness in terms of death, serious injury, minor injury and property damage only.

Dataset Source

[Collisions Routieres \(http://donnees.ville.montreal.qc.ca/dataset/collisions-routieres\)](http://donnees.ville.montreal.qc.ca/dataset/collisions-routieres)

Dataset Characteristics

Number of Instances: 171,271

Number of Attributes: 68

Publishers: Service de l'urbanisme et de la mobilité - Direction de la mobilité

Frequency of update: Annual

Language: French

Geographic coverage: Territory of the city of Montreal

Temporal coverage: 2012-01-01 / 2018-12-31

Last edit: 2019-09-17 09:40

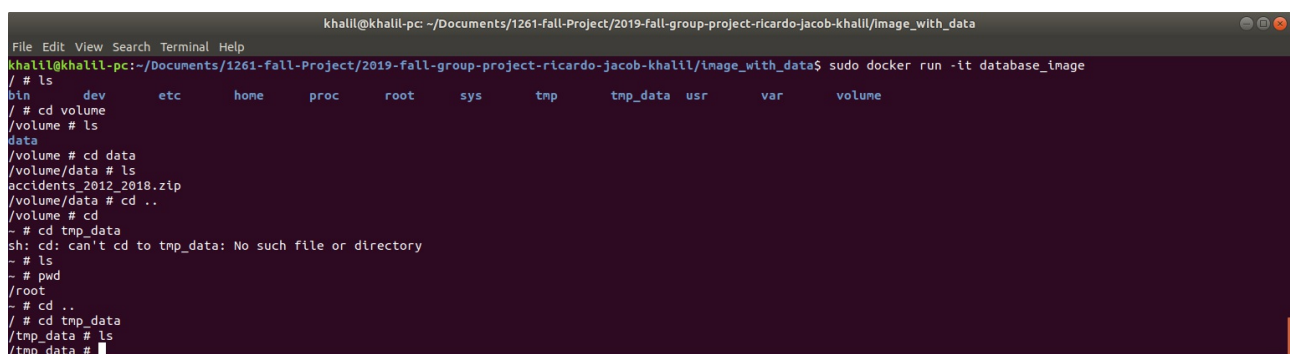
Created on: 2018-11-11 21:39

License: Creative Commons Attribution 4.0 International

Fetching the data and download it into volume using an image

using Dockerfile [data_dockerfile \(image with data/data_dockerfile\)](#) to build an image downloads the data from the source to tmp_data directory then move this data to data directory inside a volume when the container will be created by running the following commands:

```
docker build -t database_image -f data_dockerfile .
docker run -it database_image
```



```
khalil@khalil-pc: ~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil/image_with_data
File Edit View Search Terminal Help
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil/image_with_data$ sudo docker run -it database_image
/# ls
bin      dev      etc      hone     proc     root     sys      tmp      tmp_data  usr      var      volume
/# cd volume
/volume # ls
data
/volume # cd data
/volume/data # ls
accidents_2012_2018.zip
/volume/data # cd ..
/volume # cd
- # cd tmp_data
sh: cd: can't cd to tmp_data: No such file or directory
- # ls
- # pwd
/root
- # cd ..
/# cd tmp_data
/tmp_data # ls
/tmp_data #
```

Creating Volume

Create Data volume

```
docker volume create project-scripts-volume
```

Copy Database to Volume

```
docker run --rm -v project-scripts-volume:/volume database_image
```

Copy Data Folder to Volume

```
docker run --rm -v "$(pwd)"/data:/data \
-v project-scripts-volume:/volume busybox \
cp -r /data/ /volume
```

Volume Contents

```
docker run -it --rm -v project-scripts-volume:/volume busybox ls -l /volume
```

```
docker run -it --rm -v project-scripts-volume:/volume busybox ls -l
/volume/data
```

```
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil$ docker run -it --rm -v Project-scripts-volume:/volume busybox ls -l /volume
WARNING: Error loading config file: /home/khalil/.docker/config.json: open /home/khalil/.docker/config.json: permission denied
total 8
drwxr-xr-x  2 root   root       4096 Nov 16 05:05 data
drwxr-xr-x  2 root   root       4096 Nov 16 07:10 script
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil$ docker run -it --rm -v Project-scripts-volume:/volume busybox ls -l /volume/data
WARNING: Error loading config file: /home/khalil/.docker/config.json: open /home/khalil/.docker/config.json: permission denied
total 7348
-rw-r--r--  1 root   root      7520751 Nov  8 04:28 accidents_2012_2018.zip
```

Services on Spark Cluster

Create Spark Network

Using the following command a spark network will be created as "spark-network"

```
docker network create spark-network
```

Spark Cluster with HDFS and MongoDB

using docker compose to create spark cluster by running [spark-compose.yml](#) (./spark-compose.yml) file using the below command:

```
env user_mongo=root pass_mongo=password docker-compose --file spark-compose.yml
up --scale spark-worker=2
```

Preparing Data and Work Environment

Checking if the volume accessible by the cluster

- check the containers of the cluster
- execute the worker container to check the volume using the command:

```
docker exec -it containerID sh
```

```

khalil@khalil-pc: ~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil
File Edit View Search Terminal Help
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil$ docker ps
WARNING: Error loading config file: /home/khalil/.docker/config.json: open /home/khalil/.docker/config.json: permission denied
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
09b5e24f572c       mjhea0/spark:2.4.1  "bin/spark-class org..."  10 minutes ago     Up 10 minutes      0.0.0.0:8081->8081/tcp
2019-fall-group-project-ricardo-jacob-khalil_worker_1
085b8779c66d       harisekhon/hadoop:2.8  "/bin/sh -c \"entryp..."  10 minutes ago     Up 10 minutes      0.0.0.0:8042->8042/tcp, 8020/tcp, 9
000/tcp, 0.0.0.0:8088->8088/tcp, 0.0.0.0:19888->19888/tcp, 0.0.0.0:50010->50010/tcp, 0.0.0.0:50020->50020/tcp, 0.0.0.0:50070->50070/tcp, 0.0.0.0:
50075->50075/tcp, 10020/tcp, 0.0.0.0:50090->50090/tcp 2019-fall-group-project-ricardo-jacob-khalil_hadoop_1
362bc220e884       mjhea0/spark:2.4.1  "bin/spark-class org..."  10 minutes ago     Up 10 minutes      0.0.0.0:4040->4040/tcp, 0.0.0.0:606
6->6066/tcp, 0.0.0.0:7077->7077/tcp, 0.0.0.0:8080->8080/tcp
2019-fall-group-project-ricardo-jacob-khalil_master_1
8ae1c5d69ec7       mongo-express         "tini -- /docker-ent..."  10 minutes ago     Up 10 minutes      0.0.0.0:8181->8081/tcp
2019-fall-group-project-ricardo-jacob-khalil_mongo-express_1
4aef60b50653       mongo                 "docker-entrypoint.s..."  10 minutes ago     Up 10 minutes      27017/tcp
2019-fall-group-project-ricardo-jacob-khalil_mongo_1
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil$ docker exec -it 09b5e24f572c sh
WARNING: Error loading config file: /home/khalil/.docker/config.json: open /home/khalil/.docker/config.json: permission denied
# ls /volume
data script
# ls /volume/data
accidents_2012_2018.zip
#

```

Starting Spark Shell

Spark shell will be used to unzip the data inside the volume, upload the data on HDFS and MongoDB.

to start spark shell the below command will be run:

```

docker run -it --rm \
-v project-scripts-volume:/volume \
--network=spark-network \
mjhea0/spark:2.4.1 \
./bin/pyspark \
--master spark://master:7077 \
--packages org.mongodb.spark:mongo-spark-connector_2.11:2.4.0

```

```

khalil@khalil-pc: ~
File Edit View Search Terminal Help
-----
|               | modules | artifacts |
| conf | number | search | dwnlded | evicted | number | dwnlded |
-----
| default | 2 | 2 | 2 | 0 | 2 | 2 |
-----
:: retrieving :: org.apache.spark#spark-submit-parent-de76b109-c170-4f42-94b3-af7322acea7a
  confs: [default]
  2 artifacts copied, 0 already retrieved (2833kB/15ms)
19/11/18 19:52:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
ng builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____
 /  __ \
/   /  \
/_____/

version 2.4.1

Using Python version 3.4.2 (default, Feb  7 2019 06:08:06)
SparkSession available as 'spark'.
>>>

```

Unzip the Data File:

using the following command in spark shell:

```
from zipfile import *
with ZipFile("/volume/data/accidents_2012_2018.zip", 'r') as zipObj:
...     zipObj.extractall('/volume/data')
```

```
>>> from zipfile import *
>>> with ZipFile("/volume/data/accidents_2012_2018.zip", 'r') as zipObj:
...     zipObj.extractall('/volume/data')
...
>>>
```

Store the data in HDFS and MongoDB

Store the data in HDFS as parquet

first read the unzipped data from the volume then push it as parquet into HDFS. to achieve that the following command to be run in spark shell

```
acc_data = spark.read.csv("/volume/data")
acc_data.write.parquet("hdfs://hadoop/acc_data_parquet")
```

```
>>> acc_data = spark.read.csv("/volume/data/")
>>> acc_data.write.parquet("hdfs://hadoop/acc_data_parquet")
19/11/17 00:54:54 WARN util.Utils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields' in SparkEnv.conf.
>>>
```

to check the data file on HDFS open <http://localhost:50070> (<http://localhost:50070>) then navigate to "Utilities" in main bar, select "Browse the file system" then the below page will open.

Browse Directory

/ Go!

Show: 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Nov 16 19:55	0	0 B	acc_data_parquet

Showing 1 to 1 of 1 entries Previous 1 Next

Hadoop, 2017.

Store the data in MongoDB

first in spark shell run the following command to be able pushing the data to MongoDB

```

spark = SparkSession \
    .builder \
    .appName("mongodb") \
    .master("spark://master:7077") \
    .config("spark.mongodb.input.uri",
"mongodb://root:password@mongo/test.coll?authSource=admin") \
    .config("spark.mongodb.output.uri",
"mongodb://root:password@mongo/test.coll?authSource=admin") \
    .config('spark.jars.packages', 'org.mongodb.spark:mongo-spark-
connector_2.11:2.4.0')\
    .getOrCreate()

```

Read the data from the volume and store it in MongoDB

```

acc_mongo = spark.read.csv("/volume/data")
acc_mongo.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").sa
ve()

```

Read the data from the HDFS and store it in MongoDB

```

acc_mongo = spark.read.csv("hdfs://hadoop/acc_data_parquet")
acc_mongo.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").sa
ve()

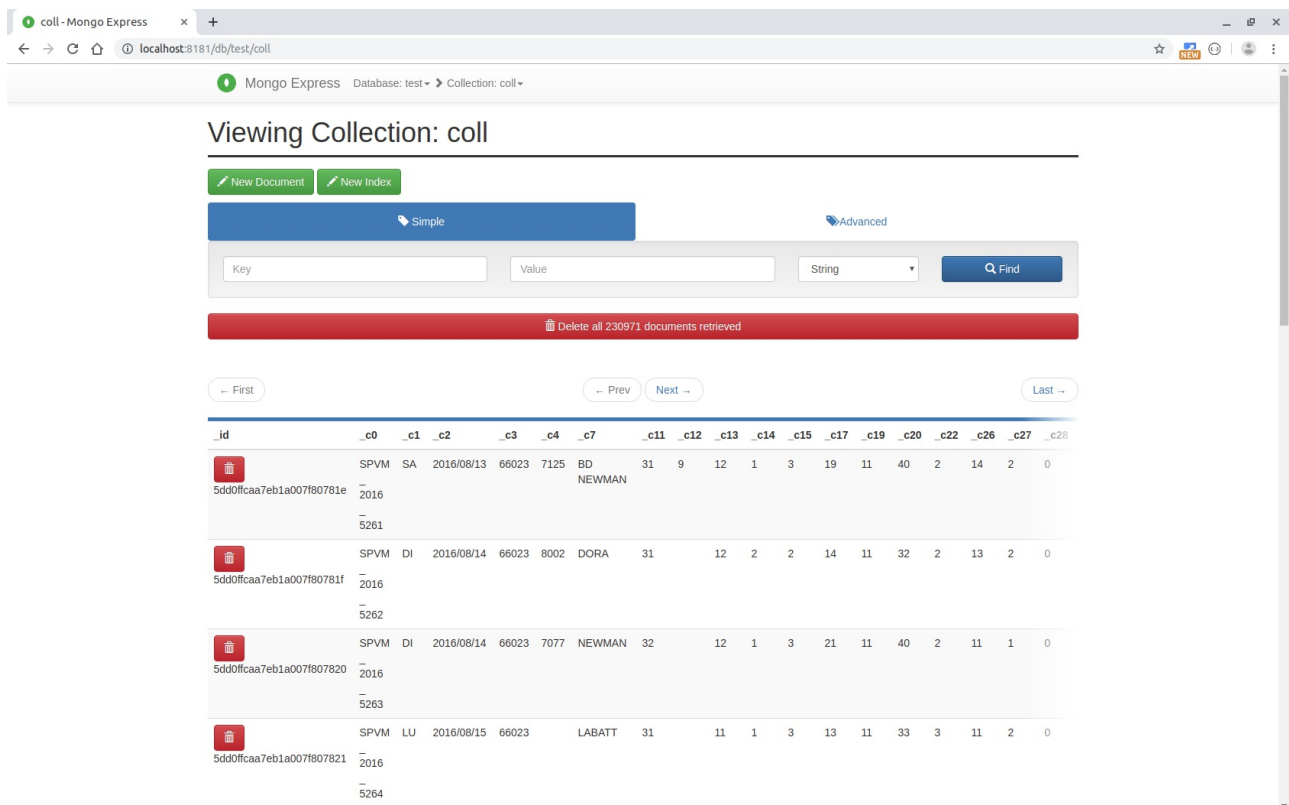
```

Open <http://localhost:8181> (<http://localhost:8181>) for Mongo Express.

The screenshot shows the Mongo Express web interface in a browser window. The address bar shows 'localhost:8181'. The page title is 'Mongo Express'. Below the title, there's a 'Databases' section with a search bar and a '+ Create Database' button. It lists four databases: 'admin', 'config', 'local', and 'test'. Each database has a 'View' button and a 'Del' button. Below the databases, there's a 'Server Status' section with a table of server metrics.

Server Status			
Hostname	803d4be5f216	MongoDB Version	4.2.1
Uptime	848 seconds	Server Time	Sun, 17 Nov 2019 08:08:28 GMT
Current Connections	4	Available Connections	838856
Active Clients	0	Queued Operations	0
Clients Reading	0	Clients Writing	0
Read Lock Queue	0	Write Lock Queue	0
Disk Flushes		Last Flush	
Time Spent Flushing	ms	Average Flush Time	ms
Total Inserts	230971	Total Queries	4

Then click on "test" for database



Automated Storage

Copy the scripts into the volume

In the volume a directory named "script" will be created and copied all the required scripts into that directory by executing the following command:

```
docker run --rm -v "$(pwd)"/scripts:/script \
-v project-scripts-volume:/volume busybox \
cp -r /script/ /volume
```

Checking the scripts in the volume:

```
docker run -it --rm -v project-scripts-volume:/volume busybox ls -l
/volume/script
```

```
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil$ sudo docker run -it --rm -v project-scripts-volume:/volume busybox ls -l
/volume/script
total 8
-rw-r--r-- 1 root root 924 Nov 19 05:30 hdfs_store.py
-rw-r--r-- 1 root root 1189 Nov 19 05:30 mongodb_store.py
khalil@khalil-pc:~/Documents/1261-fall-Project/2019-fall-group-project-ricardo-jacob-khalil$
```

Store the data in HDFS as parquet

By execute [hdfs_store.py](#) (./scripts/hdfs_store.py) script as following:

```
docker run -t --rm \
  -v project-scripts-volume:/volume \
  --network=spark-network \
  mjhea0/spark:2.4.1 \
  bin/spark-submit \
    --master spark://master:7077 \
    --class endpoint \
    /volume/script/hdfs_store.py
```

Store the data in MongoDB

By execute [mongodb_store.py \(./scripts/mongodb_store.py\)](#) script as following:

```
docker run -t --rm \
  -v project-scripts-volume:/volume \
  --network=spark-network \
  mjhea0/spark:2.4.1 \
  bin/spark-submit \
    --master spark://master:7077 \
    --class endpoint \
    --packages org.mongodb.spark:mongo-spark-connector_2.11:2.4.0 \
    /volume/script/mongodb_store.py
```

Using Jupyter Notebook

Create Volume

Creating a volume to store the notebooks Which will be created

```
docker volume create notebooks
```

Jupyter in the Cluster

Keeping the cluster up now to deploy jupyter in cluster using docker compose by running [jupyter-compose.yml \(jupyter-compose.yml\)](#) file using the below command::

```
env TOKEN=project1261 docker-compose --file jupyter-compose.yml up
```

Open <http://localhost:8889/?token=project1261>

jupyter have access to the volume where the data and scripts are stored.

Jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them. Upload New ↻

	Name ↓	Last Modified	File size
0	/ work		
	..	seconds ago	
	volume	2 hours ago	
	hdfs_mongo_notebook.ipynb	Running a minute ago	29.8 kB
	write_from_jupyter.png	7 minutes ago	62.3 kB

Opening new notebook and prepare the environment.

Reading the Data from volume

Read the data from the volume as csv

Read Data from Volume

```
In [3]: acc_data = spark.read.csv("./volume/data")
```

```
In [11]: acc_data.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
```

Store the Data on HDFS

Push the data to HDFS

Write Data on HDFS

```
In [4]: acc_data.write.parquet("hdfs://hadoop/acc_data_parquet_2")
```

The screenshot shows the Hadoop web interface at localhost:50070/explorer.html. The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main content area is titled "Browse Directory" and shows a file browser for the root directory (/). The file list displays two files: "acc_data_parquet" and "acc_data_parquet_2", both with permissions "drwxr-xr-x", owned by "root", and belonging to the "supergroup". The file "acc_data_parquet_2" is highlighted in blue, indicating it is the current selection. The interface also shows a search bar, a "Go!" button, and pagination controls for the file list.

Store the Data in MongoDB

Push the data from hdfs into MongoDB

Read The Data from HDFS

```
In [7]: acc_mongo_jupyter = spark.read.parquet("hdfs://hadoop/acc_data_parquet_2")
```

Push The Data to MongoDB

```
In [8]: acc_mongo_jupyter.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").save()
```

[The Full Notebook \(./hdfs_mongo_notebook.ipynb\)](#)

Google Cloud Platform

Environment Setup

- First lets create a project.



New Project



You have 19 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *

project1261



Project ID: project1261-259823. It cannot be changed later. [EDIT](#)

Location *



No organization

[BROWSE](#)

Parent organization or folder

[CREATE](#)

[CANCEL](#)

- Enabling the Cloud Dataproc and Google Compute Engine APIs.
- Creating bucket where notebook and the data will be stored.

Google Cloud PlatformProject1261

Storage

Create a bucket

Name your bucket

Pick a globally unique, permanent name. [Naming guidelines](#)

project1261_bucket

Tip: Don't include any sensitive information

CONTINUE

Choose where to store your data

Choose a default storage class for your data

Choose how to control access to objects

Advanced settings (optional)

CREATECANCEL

Google Cloud PlatformProject1261

Storage

Bucket details

EDIT BUCKETREFRESH BUCKET

project1261_bucket

ObjectsOverviewPermissionsBucket Lock

Upload filesUpload folderCreate folderManage holdsDelete

Filter by prefix...

Buckets / project1261_bucket / notebooks / data

<input type="checkbox"/>	Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
<input type="checkbox"/>	accidents_2012_2018.csv	49.94 MB	text/csv	Standard	11/22/19, 6:30:05 PM UTC-5	Not public	Google-managed key	-	None

- Creating cluster and selecting the the bucket to which was created to have access to the data and where the notebooks will be stored.

 Clusters

Jobs

 Workflows

Name ?

project1261-cluster

Region ?

global

Zone ?

us-central1-c

Cluster mode

Standard (1 master, N workers)

Master node

Contains the YARN Resource Manager, HDFS NameNode, and all job drivers

Machine type ?

4 vCPUs

15 GB memory

Customize

Primary disk size (minimum 15 GB) ?

500

GB

Primary disk type ?

Standard persistent disk

Worker nodes

Each contains a YARN NodeManager and a HDFS DataNode.
The HDFS replication factor is 2.

Machine type ?

4 vCPUs

15 GB memory

[Customize](#)

Primary disk size (minimum 15 GB) ?

500

GB

Primary disk type ?

Standard persistent disk

Google Cloud Platform

Project1261

Dataproc

Clusters

Jobs

Workflows

Create a cluster

Enhanced flexibility mode (Optional)

☐ Enable enhanced flexibility mode.
Only supported by image version 1.4.

Network

default

Subnetwork

default (10.128.0.0/20)

Network tags (Optional)

Internal IP only


☐ Configure all instances to have only internal IP addresses. [Learn more](#)

Cloud Storage staging bucket (Optional)

☒ project1261_bucket

Browse

Image



Cloud Dataproc image version: 1.3 (Debian 9, Hadoop 2.9, Spark 2.3)
First released on 8/16/2018.

Change

Optional components (Optional)

Install optional open source components on the cluster. [Learn more](#)

Select component

In advanced options section Anaconda and jupyter components should be selected to run jupyter in the cluster.

Google Cloud Platform

Dataproc

Clusters

Jobs

Workflows

Optional components

Select one or multiple components. [Learn more](#)

- ☒ **Anaconda**
Anaconda is a Python distribution and Package Manager with over 1000 popular data science packages. Anaconda becomes the default Python interpreter.
- ☐ **Hive WebHCat**
The Hive WebHCat server provides a REST API for HCatalog. The REST service is available on port 50111 on the cluster's first master node..
- ☒ **Jupyter Notebook**
Jupyter, a Web-based notebook for interactive data analytics. The Jupyter Web UI is available on port 8123 on the cluster's first master node. Python and PySpark kernels are available.
- ☐ **Zeppelin Notebook**
Zeppelin Notebook is a Web-based notebook for interactive data analytics. The Zeppelin Web UI is available on port 8080 on the cluster's first master node.
- ☐ **Druid**
The Apache Druid component is an open source distributed OLAP data store. The Druid component installs Druid services on the Cloud Dataproc cluster master(Coordinator, Broker, and Overlord) and worker (Historical, Realtime and MiddleManager)nodes.
- ☐ **Presto**
The Presto component is an open source distributed SQL query engine. The Presto server and Web UI are available on port 8060 (or port 7778 if Kerberos is enabled) on the cluster's first master node.
- ☐ **ZooKeeper**
The Apache ZooKeeper component is a centralized service for providing distributed synchronization of data.

Select **Cancel**

- From the web interfaces in created cluster we can open jupyter notebook which is running on the cluster and have access to the bucket where our data is.

Analytics on GCP

Opening Jupyter on GCP cluster

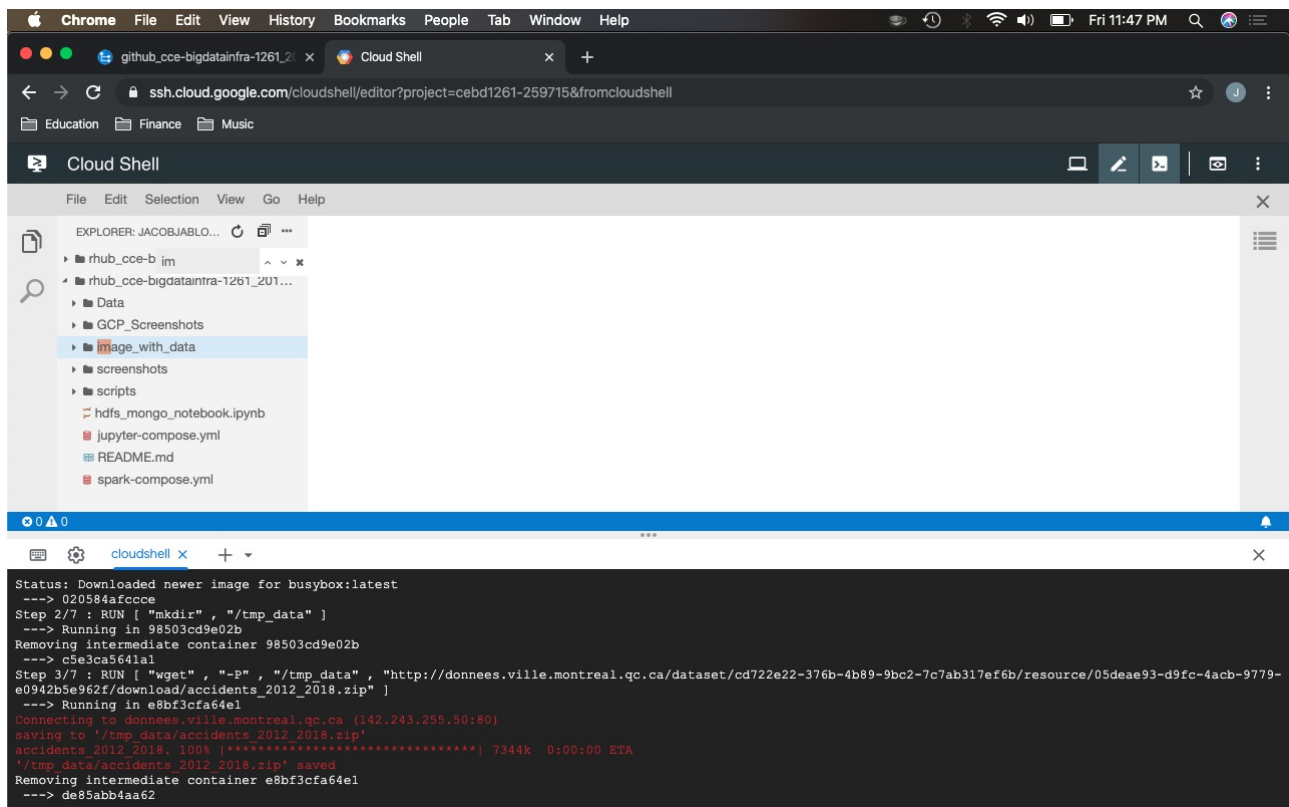
Running the Project App

[Exploratory Data Analysis on the Collisions Routieres dataset \(./project_app.ipynb\)](#)

Github Mirror and Deploy via GCP

Using the exact same process as our local docker stack, we will deploy our application on the GCP. The only difference is that we skip the manual hdf5/mongo-express builds and go right to automated builds.

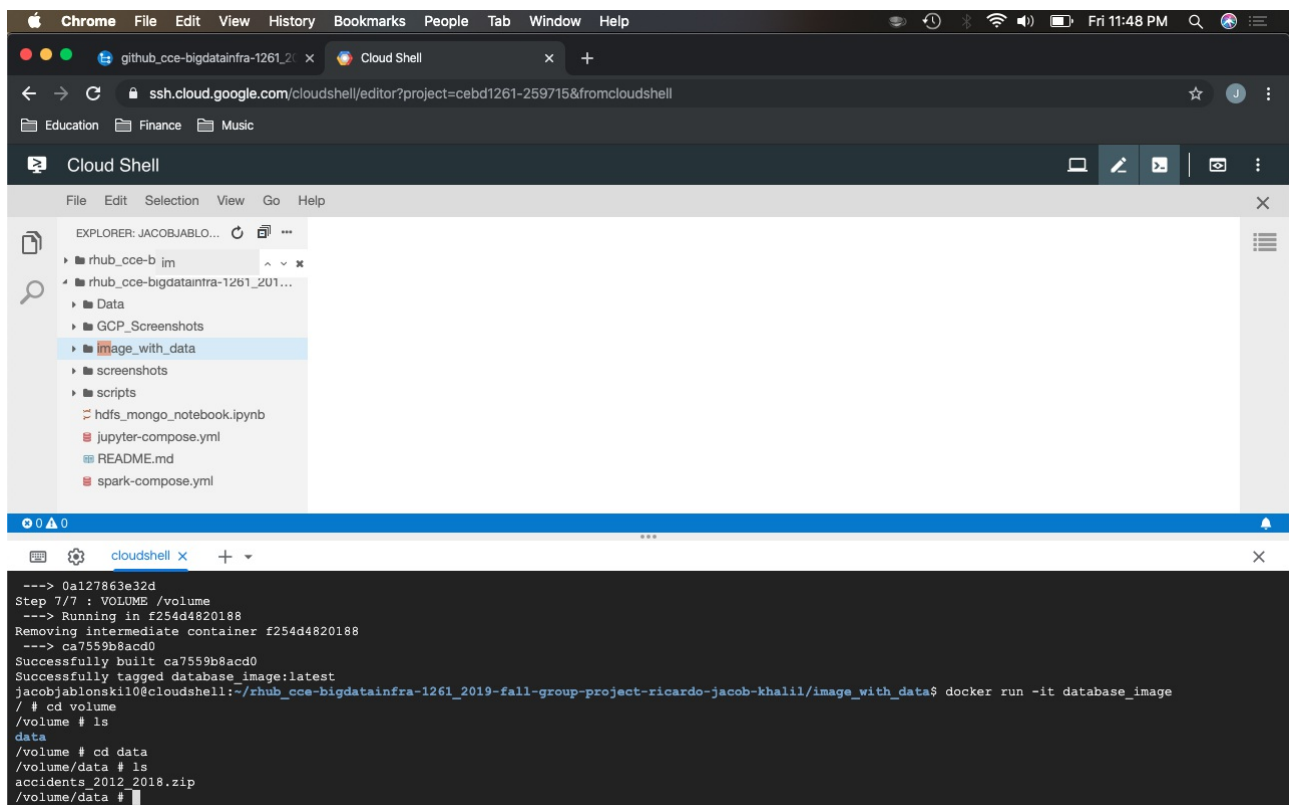
- Pulling data from github hosted docker image through google shell



The screenshot shows the Google Cloud Shell interface. The file explorer on the left shows a directory structure with 'image_with_data' selected. The terminal window displays the following output:

```
Status: Downloaded newer image for busybox:latest
---> 020584afccce
Step 2/7 : RUN [ "mkdir", "/tmp_data" ]
---> Running in 98503cd9e02b
Removing intermediate container 98503cd9e02b
---> c5e3ca5641a1
Step 3/7 : RUN [ "wget", "-P", "/tmp_data", "http://donnees.ville.montreal.qc.ca/dataset/cd722e22-376b-4b89-9bc2-7c7ab317ef6b/resource/05deae93-d9fc-4acb-9779-e0942b5e962f/download/accidents_2012_2018.zip" ]
---> Running in e8bf3cfa64e1
Connecting to donnees.ville.montreal.qc.ca (142.243.255.50:80)
saving to '/tmp_data/accidents_2012_2018.zip'
accidents_2012_2018.zip 100% [*****] 7344k 0:00:00 ETA
'/tmp_data/accidents_2012_2018.zip' saved
Removing intermediate container e8bf3cfa64e1
---> de85abb4aa62
```

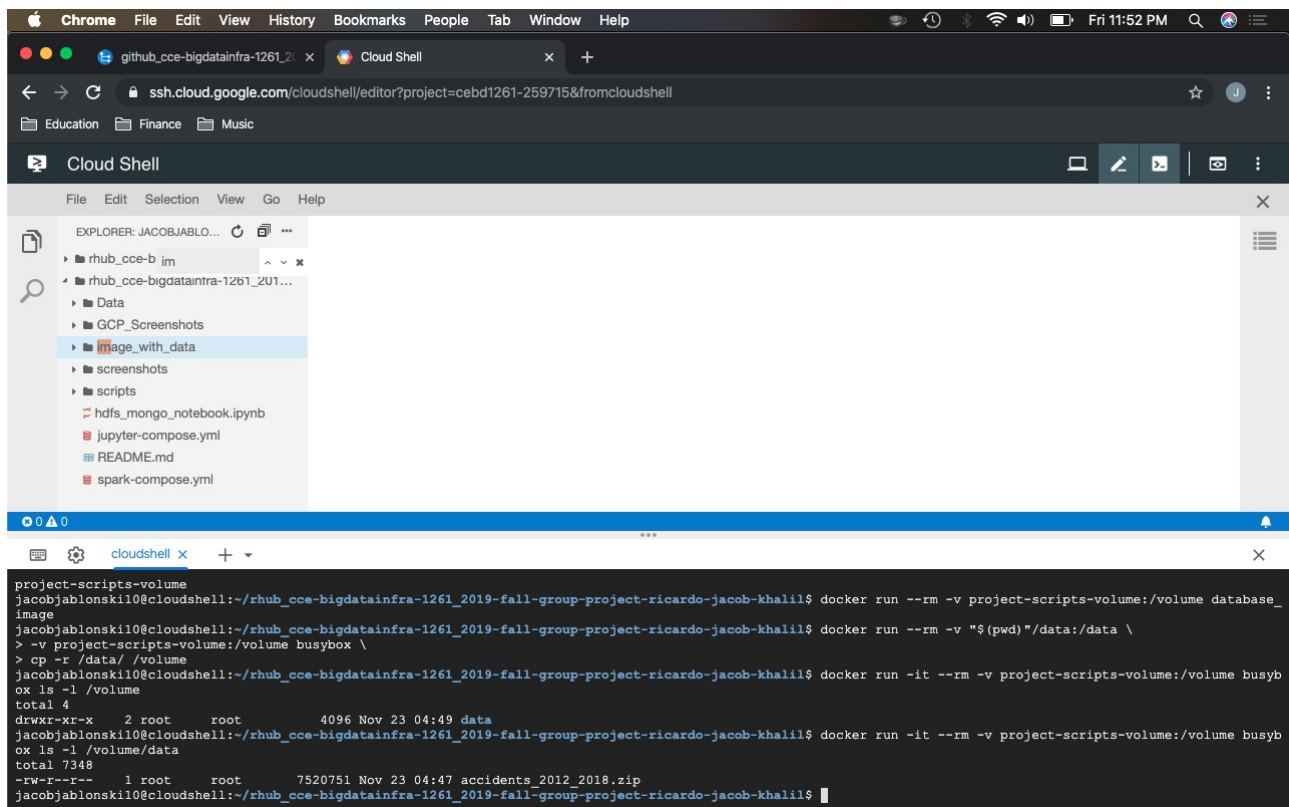
- Running the docker image and making sure the data is in the correct directory



The screenshot shows the Google Cloud Shell interface. The file explorer on the left shows a directory structure with 'image_with_data' selected. The terminal window displays the following output:

```
---> 0a127863e32d
Step 7/7 : VOLUME /volume
---> Running in f254d4820188
Removing intermediate container f254d4820188
---> ca7559b8acd0
Successfully built ca7559b8acd0
Successfully tagged database_image:latest
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil/image_with_data$ docker run -it database_image
/ # cd volume
/volume # ls
data
/volume # cd data
/volume/data # ls
accidents_2012_2018.zip
/volume/data #
```

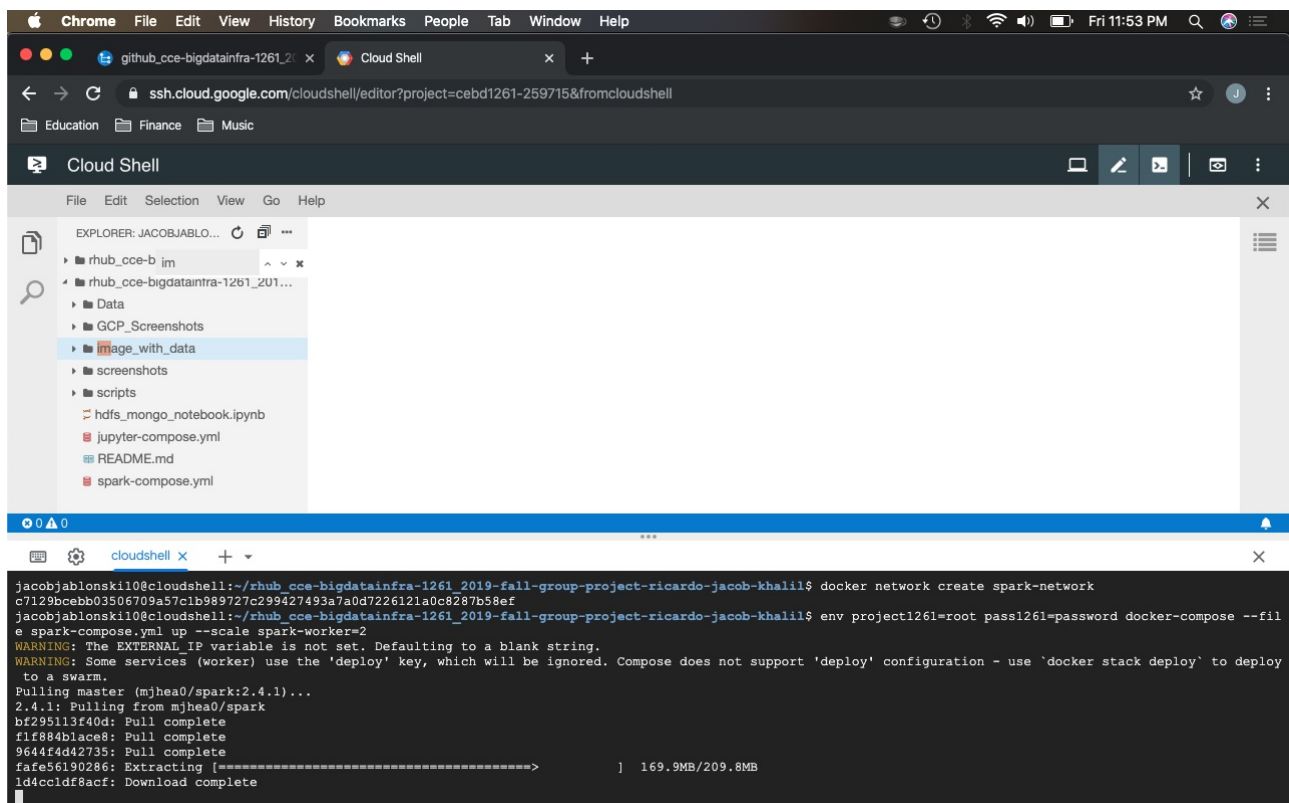
- Copying the scripts



The screenshot shows the Google Cloud Shell interface. The file explorer on the left lists a directory structure including 'rhub_cce-b jm', 'rhub_cce-bigdatainfra-1261_201...', 'Data', 'GCP_Screenshots', 'image_with_data', 'screenshots', 'scripts', 'hdfs_mongo_notebook.ipynb', 'jupyter-compose.yml', 'README.md', and 'spark-compose.yml'. The terminal window shows the following commands and output:

```
project-scripts-volume
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$ docker run --rm -v project-scripts-volume:/volume database_
image
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$ docker run --rm -v "$(pwd)"/data:/data \
> -v project-scripts-volume:/volume busybox \
> cp -r /data/ /volume
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$ docker run -it --rm -v project-scripts-volume:/volume busyb
ox ls -l /volume
total 4
drwxr-xr-x  2 root   root           4096 Nov 23 04:49 data
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$ docker run -it --rm -v project-scripts-volume:/volume busyb
ox ls -l /volume/data
total 7348
-rw-r--r--  1 root   root       7520751 Nov 23 04:47 accidents_2012_2018.zip
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$
```

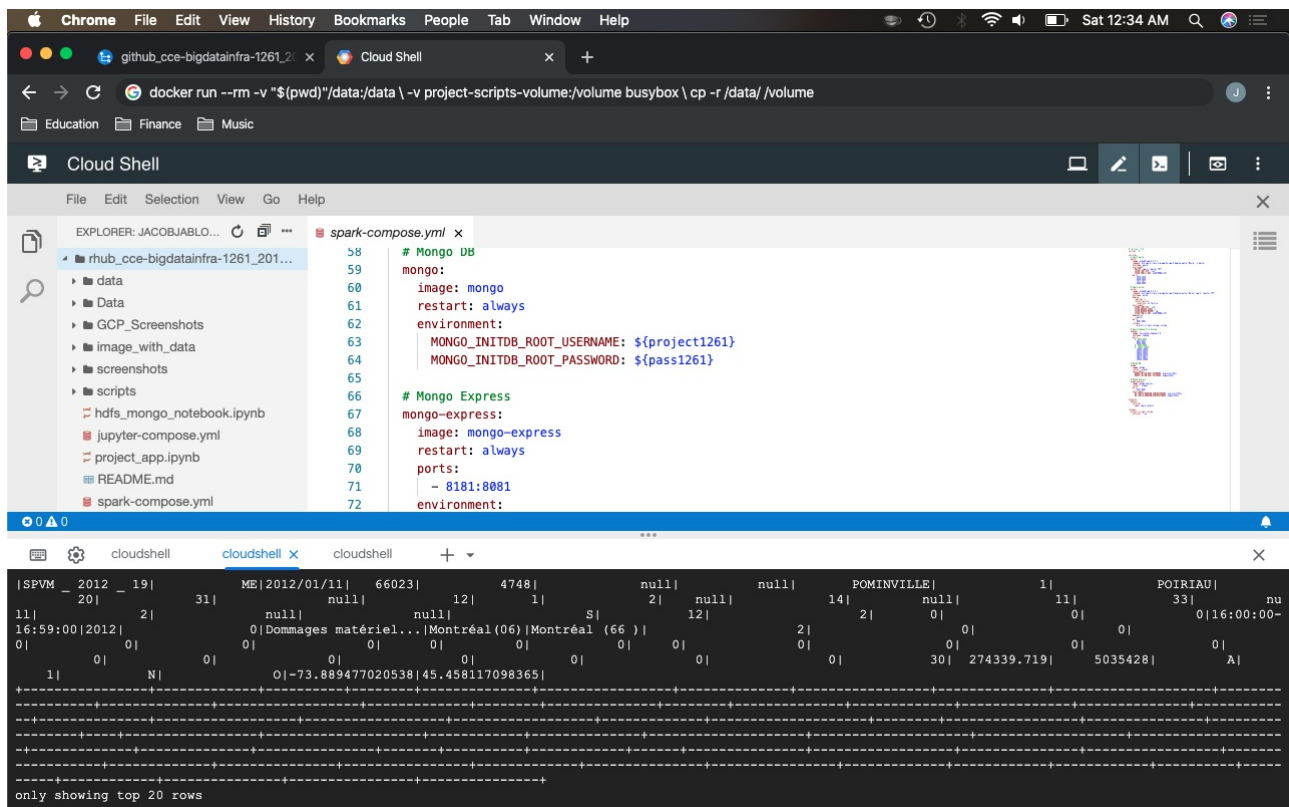
- Deploying the spark compose file with created network



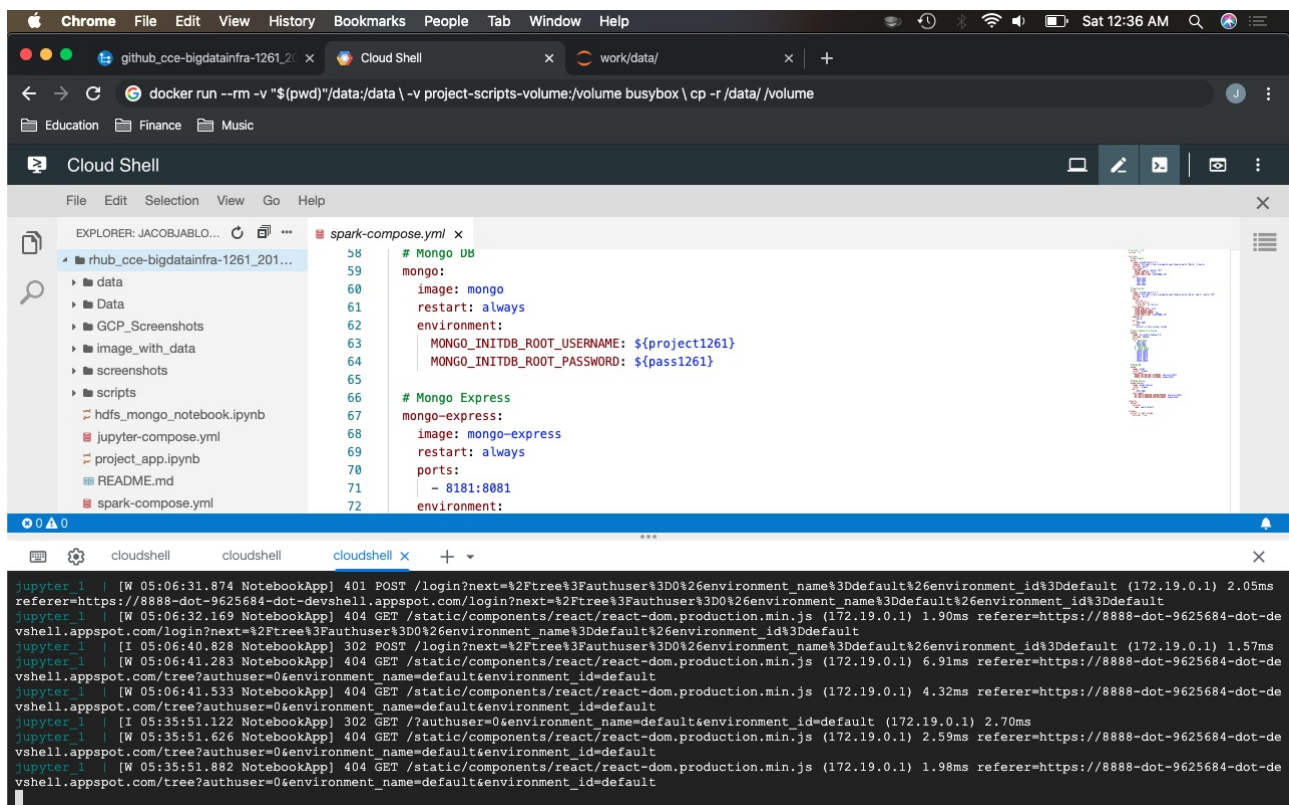
The screenshot shows the Google Cloud Shell interface. The file explorer on the left is the same as in the previous screenshot. The terminal window shows the following commands and output:

```
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$ docker network create spark-network
c7129bcebb03506709a57c1b989727c299427493a7a0d7226121a0c8287b58ef
jacobjablonski10@cloudshell:~/rhub_cce-bigdatainfra-1261_2019-fall-group-project-ricardo-jacob-khalil$ env project1261=root pass1261=password docker-compose --fil
e spark-compose.yml up --scale spark-worker=2
WARNING: The EXTERNAL_IP variable is not set. Defaulting to a blank string.
WARNING: Some services (worker) use the 'deploy' key, which will be ignored. Compose does not support 'deploy' configuration - use 'docker stack deploy' to deploy
to a swarm.
Pulling master (mjhea0/spark:2.4.1)...
2.4.1: Pulling from mjhea0/spark
bf295113f40d: Pull complete
1f8884b1ace8: Pull complete
9644f4d42735: Pull complete
f4fe56190286: Extracting [=====>] 169.9MB/209.8MB
1d4ccdf8ac6f: Download complete
```

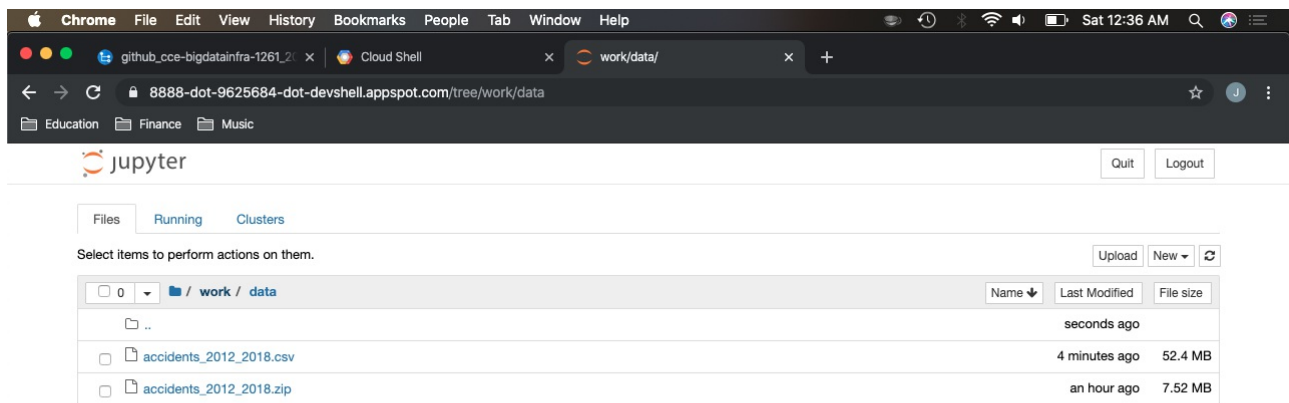
- Creating jupyter volume



- Jupyter compose up to be able to reach our notebook



- Proof of the notebook in action on our GCP project cluster



<https://8888-dot-9625684-dot-devshell.appspot.com/tree/work>

The Results Obtained

In our project, we chose to work with the dataset about traffic accidents that happened from 2012 to 2018 in the city of Montreal. We built our infrastructure using a docker container to create an image gathering that dataset from the Montreal Open Data Website. We decompressed, read and wrote the dataset in HDFS (Parquet files) and MongoDB using a spark console running on a Spark Cluster and making use of Docker Volumes. Also, we wrote scripts to make those operations automatically. Plus, we ran a Jupyter Notebook using the same volume where the data was saved and we implemented some exploratory data analysis. The sensitive data as passwords and tokens were handled safely. After doing everything locally as a Docker Stack, we connected our GitHub repository on the Google Cloud Platform and deployed our solution on the Cloud!

Conclusion

After doing the proposed tutorials during the course classes we could imagine the complexity of Big Data infrastructure. However, after doing the group project, we started to understand such complexity due to the great challenge that was to deploy a simple application both locally and on the Cloud. The use of a lot of structures that characterize Big Data infrastructure solutions can easily become hard work even when we have good tools to help us out.