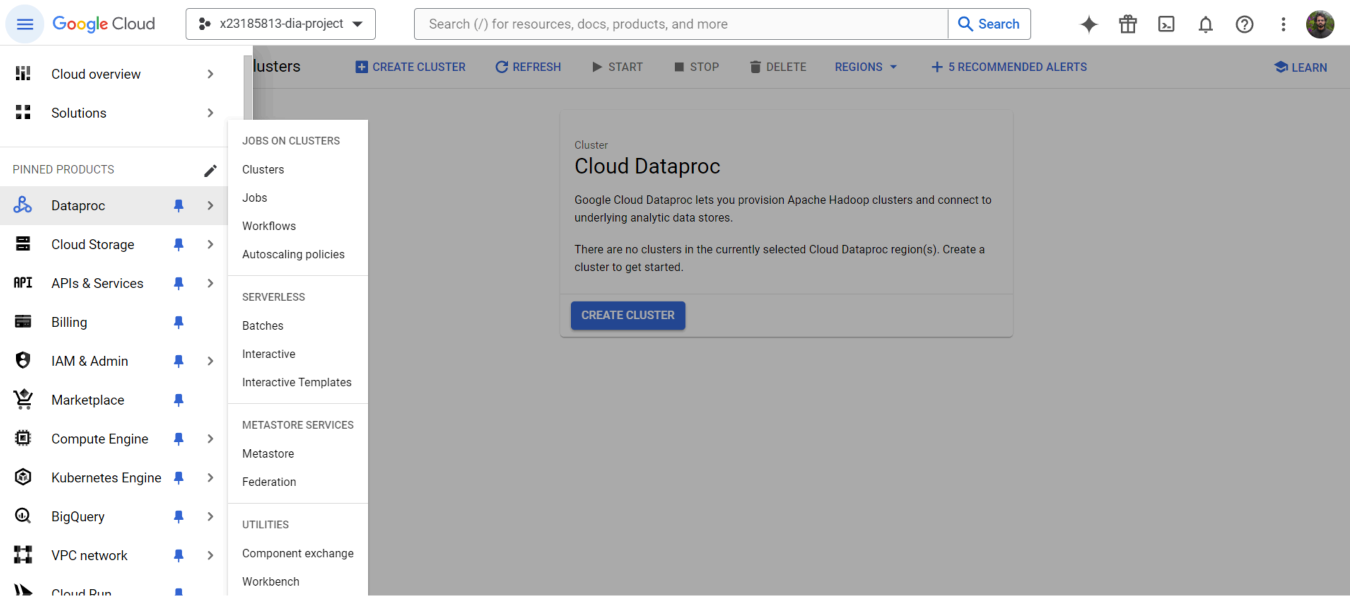
Analysis of Motor Vehicle Collisions in New York City with PySpark and MapReduce

This guide will help setup the dataproc cluster and apache spark used for this project. We use Google Cloud for this, so as prerequisite you need a google account and a billing account associated with it to access the google cloud console

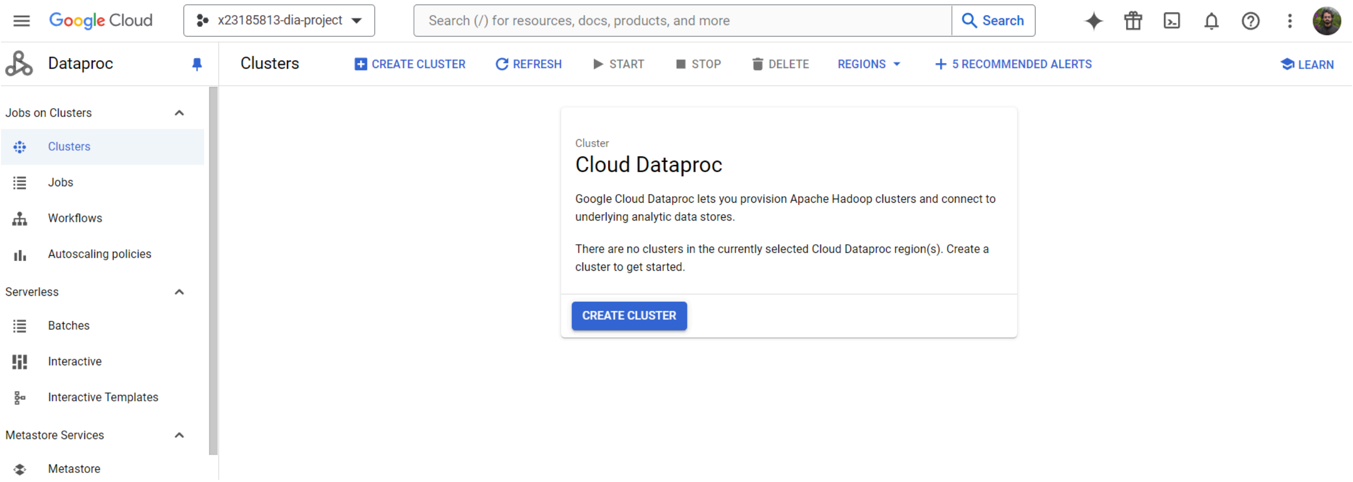
1. Log in to the google cloud console and select dataproc from the side panel or use the search bar



Google Cloud Dataproc lets you provision Apache Hadoop clusters and connect to underlying analytic data stores.

There are no clusters in the currently selected Cloud Dataproc region(s). Create a cluster to get started.

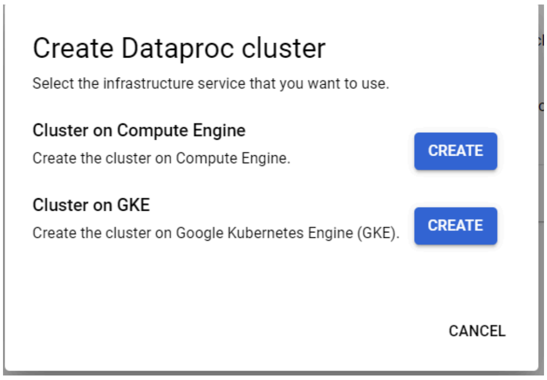
1. The dataproc interface will look like this proceed with **CREATE CLUSTER**



Google Cloud Dataproc lets you provision Apache Hadoop clusters and connect to underlying analytic data stores.

There are no clusters in the currently selected Cloud Dataproc region(s). Create a cluster to get started.

3. Select Cluster on Compute engine this will use the google compute engine to create the virtual machines used for master and worker nodes



Create Dataproc cluster

Select the infrastructure service that you want to use.

Cluster on Compute Engine

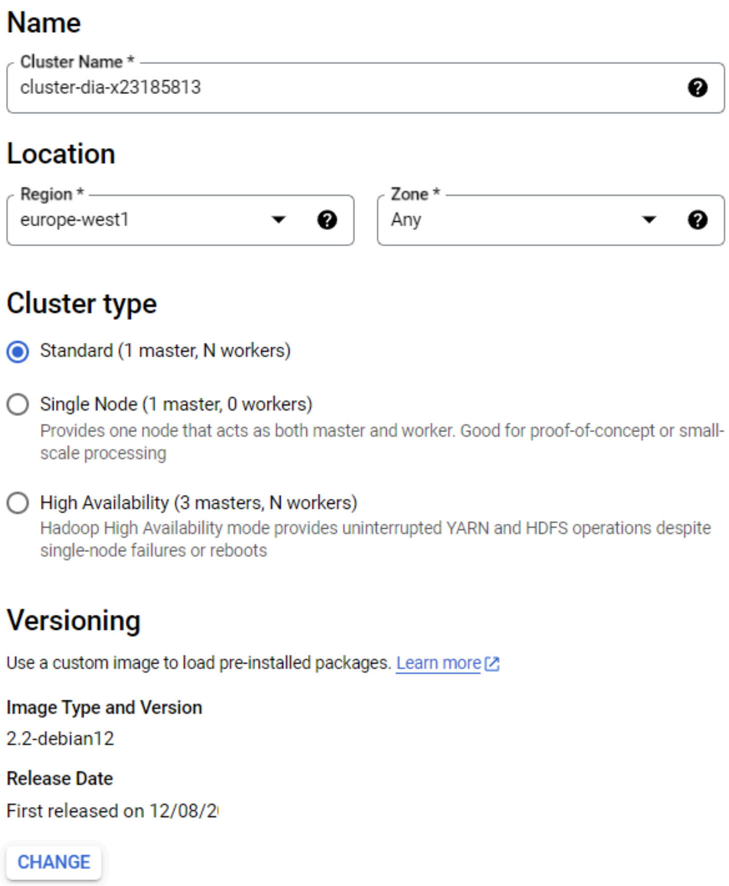
Create the cluster on Compute Engine.

Cluster on GKE

Create the cluster on Google Kubernetes Engine (GKE).

4. In the following steps select the settings for the cluster.

select a name for the cluster and select the cluster type we use 'Standard'



Name

Cluster Name *: cluster-dia-x23185813

Location

Region *: europe-west1

Zone *: Any

Cluster type

● Standard (1 master, N workers)

○ Single Node (1 master, 0 workers)
Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing

○ High Availability (3 masters, N workers)
Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Versioning

Use a custom image to load pre-installed packages. Learn more

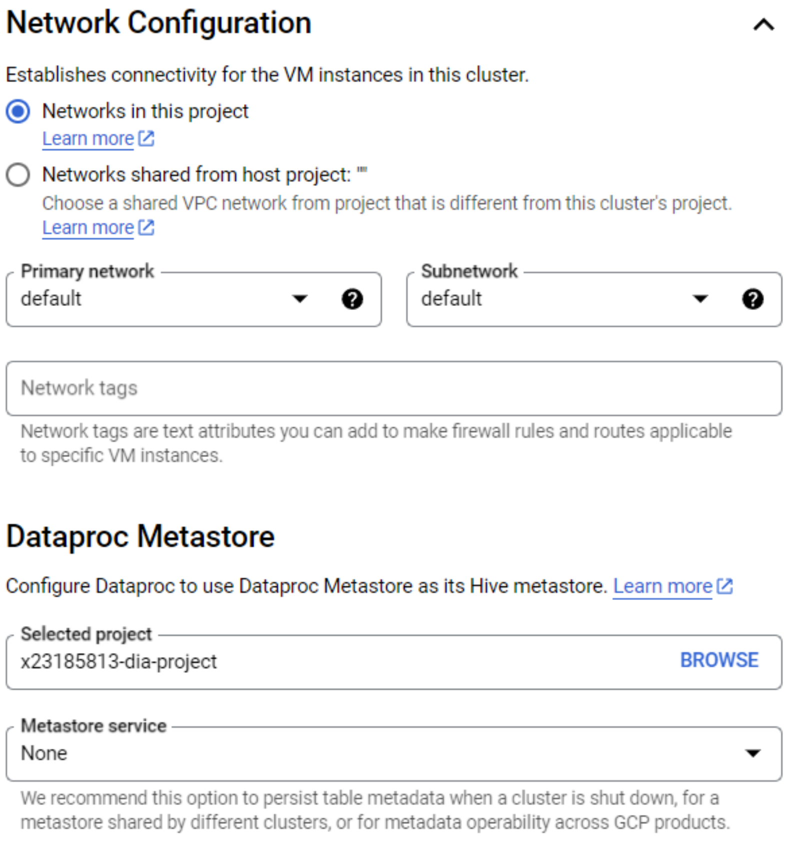
Image Type and Version

2.2-debian12

Release Date

First released on 12/08/2

Choose the subnetwork as default



Network Configuration

Establishes connectivity for the VM instances in this cluster.

● Networks in this project
Learn more

○ Networks shared from host project: ""
Choose a shared VPC network from project that is different from this cluster's project.
Learn more

Primary network: default

Subnetwork: default

Network tags

Network tags are text attributes you can add to make firewall rules and routes applicable to specific VM instances.

Dataproc Metastore

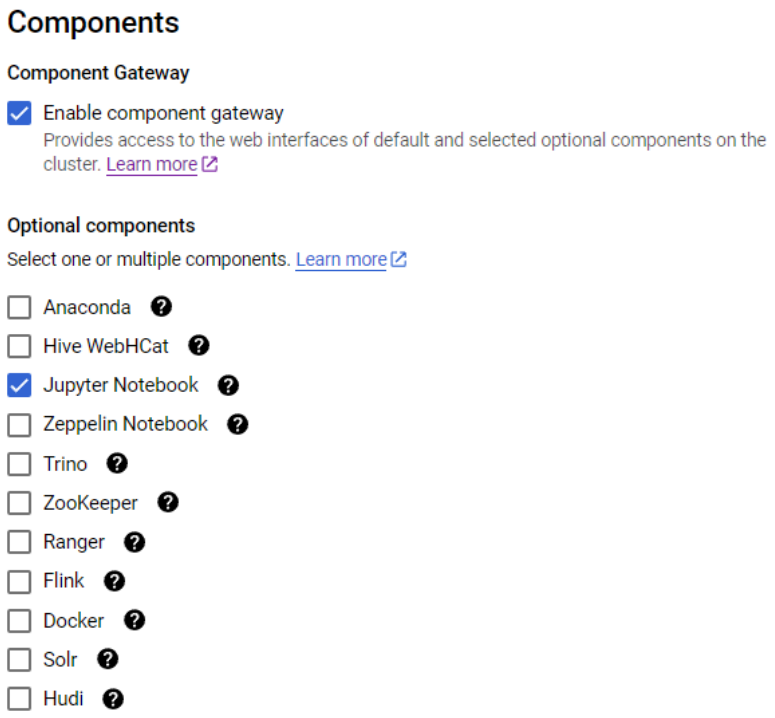
Configure Dataproc to use Dataproc Metastore as its Hive metastore. Learn more

Selected project: x23185813-dia-project

Metastore service: None

We recommend this option to persist table metadata when a cluster is shut down, for a metastore shared by different clusters, or for metadata operability across GCP products.

Check the Component gateway and select Jupyter Notebook



Components

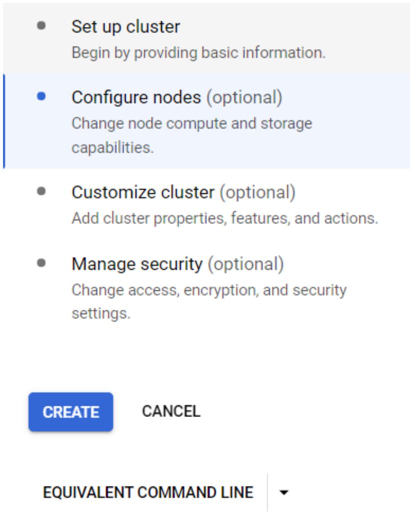
Component Gateway

☑ Enable component gateway
Provides access to the web interfaces of default and selected optional components on the cluster. Learn more

Optional components

Select one or multiple components. Learn more

- ☐ Anaconda
- ☐ Hive WebHCat
- ☑ Jupyter Notebook
- ☐ Zeppelin Notebook
- ☐ Trino
- ☐ ZooKeeper
- ☐ Ranger
- ☐ Flink
- ☐ Docker
- ☐ Solr
- ☐ Hudi

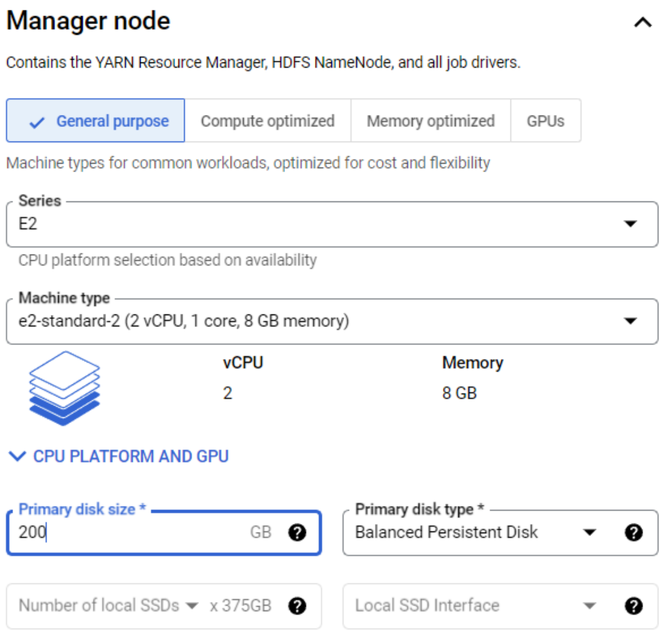


- Set up cluster
  Begin by providing basic information.
- Configure nodes (optional)
  Change node compute and storage capabilities.
- Customize cluster (optional)
  Add cluster properties, features, and actions.
- Manage security (optional)
  Change access, encryption, and security settings.

In the Configure node select the node configurations

we will use E2 instances

1 Master node and 2 Worker nodes



Manager node

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

Machine types for common workloads, optimized for cost and flexibility

Series: E2

CPU platform selection based on availability

Machine type: e2-standard-2 (2 vCPU, 1 core, 8 GB memory)

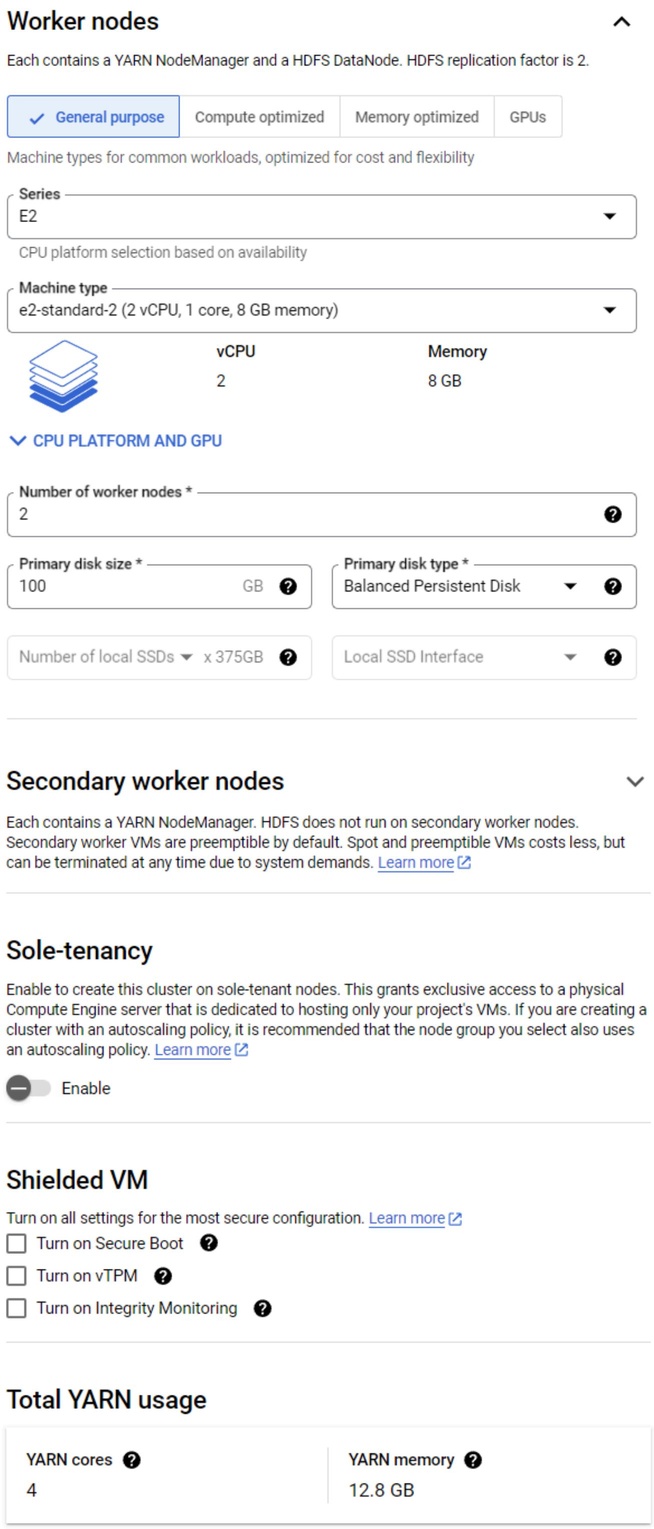
vCPU	Memory
2	8 GB

Primary disk size *: 200 GB

Primary disk type *: Balanced Persistent Disk

Number of local SSDs x 375GB

Local SSD Interface



Worker nodes

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

✓ General purpose | Compute optimized | Memory optimized | GPUs

Machine types for common workloads, optimized for cost and flexibility

Series: E2

CPU platform selection based on availability

Machine type: e2-standard-2 (2 vCPU, 1 core, 8 GB memory)

vCPU	Memory
2	8 GB

Number of worker nodes *: 2

Primary disk size *: 100 GB

Primary disk type *: Balanced Persistent Disk

Number of local SSDs x 375GB

Local SSD Interface

Secondary worker nodes

Each contains a YARN NodeManager. HDFS does not run on secondary worker nodes. Secondary worker VMs are preemptible by default. Spot and preemptible VMs costs less, but can be terminated at any time due to system demands. Learn more

Sole-tenancy

Enable to create this cluster on sole-tenant nodes. This grants exclusive access to a physical Compute Engine server that is dedicated to hosting only your project's VMs. If you are creating a cluster with an autoscaling policy, it is recommended that the node group you select also uses an autoscaling policy. Learn more

Enable

Shielded VM

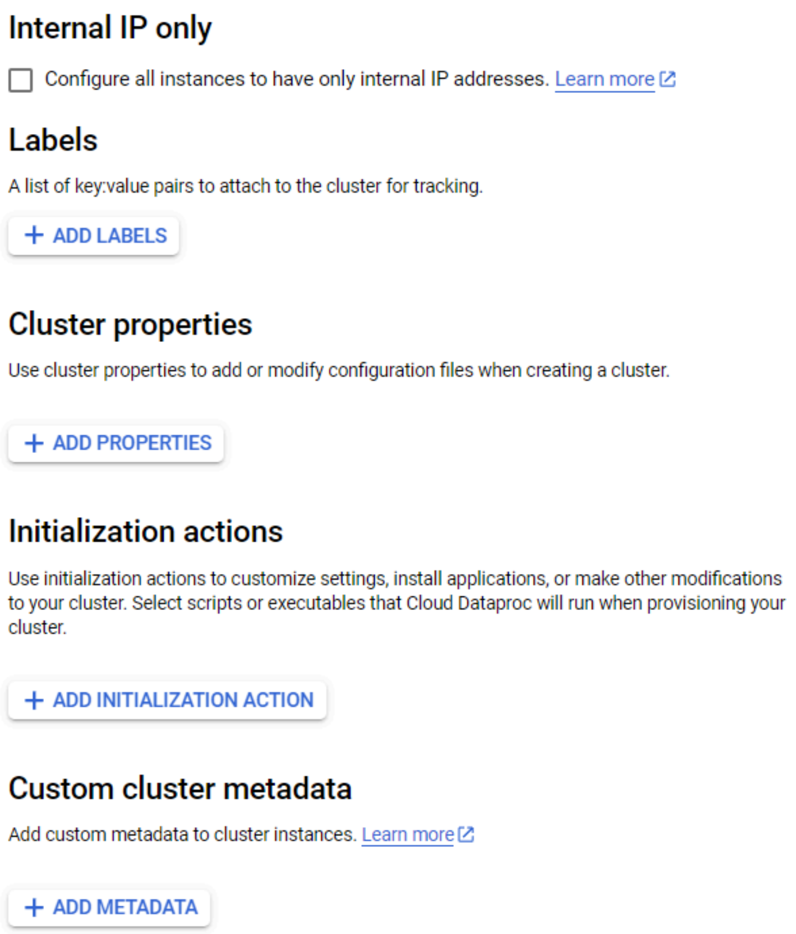
Turn on all settings for the most secure configuration. Learn more

- ☐ Turn on Secure Boot
- ☐ Turn on vTPM
- ☐ Turn on Integrity Monitoring

Total YARN usage

YARN cores	YARN memory
4	12.8 GB

Uncheck the internal IP only option



Internal IP only

□ Configure all instances to have only internal IP addresses. Learn more

Labels

A list of key:value pairs to attach to the cluster for tracking.

+ ADD LABELS

Cluster properties

Use cluster properties to add or modify configuration files when creating a cluster.

+ ADD PROPERTIES

Initialization actions

Use initialization actions to customize settings, install applications, or make other modifications to your cluster. Select scripts or executables that Cloud Dataproc will run when provisioning your cluster.

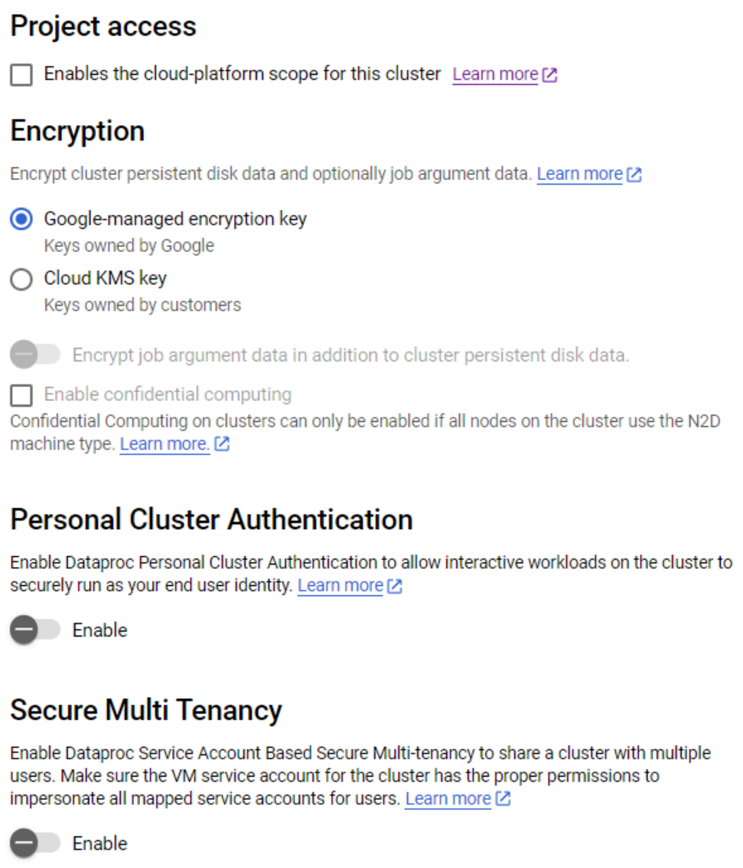
+ ADD INITIALIZATION ACTION

Custom cluster metadata

Add custom metadata to cluster instances. Learn more

+ ADD METADATA

Rest settings keep it as default



Project access

□ Enables the cloud-platform scope for this cluster Learn more

Encryption

Encrypt cluster persistent disk data and optionally job argument data. Learn more

● Google-managed encryption key
Keys owned by Google

○ Cloud KMS key
Keys owned by customers

Encrypt job argument data in addition to cluster persistent disk data.

□ Enable confidential computing
Confidential Computing on clusters can only be enabled if all nodes on the cluster use the N2D machine type. Learn more.

Personal Cluster Authentication

Enable Dataproc Personal Cluster Authentication to allow interactive workloads on the cluster to securely run as your end user identity. Learn more

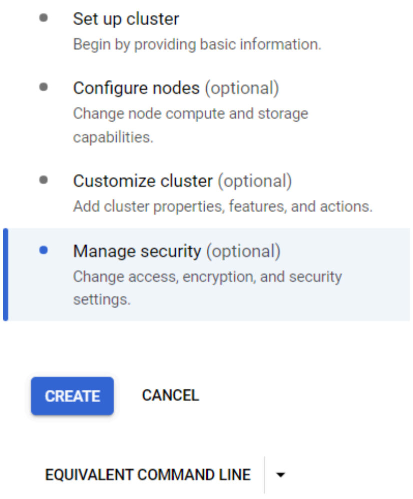
Enable

Secure Multi Tenancy

Enable Dataproc Service Account Based Secure Multi-tenancy to share a cluster with multiple users. Make sure the VM service account for the cluster has the proper permissions to impersonate all mapped service accounts for users. Learn more

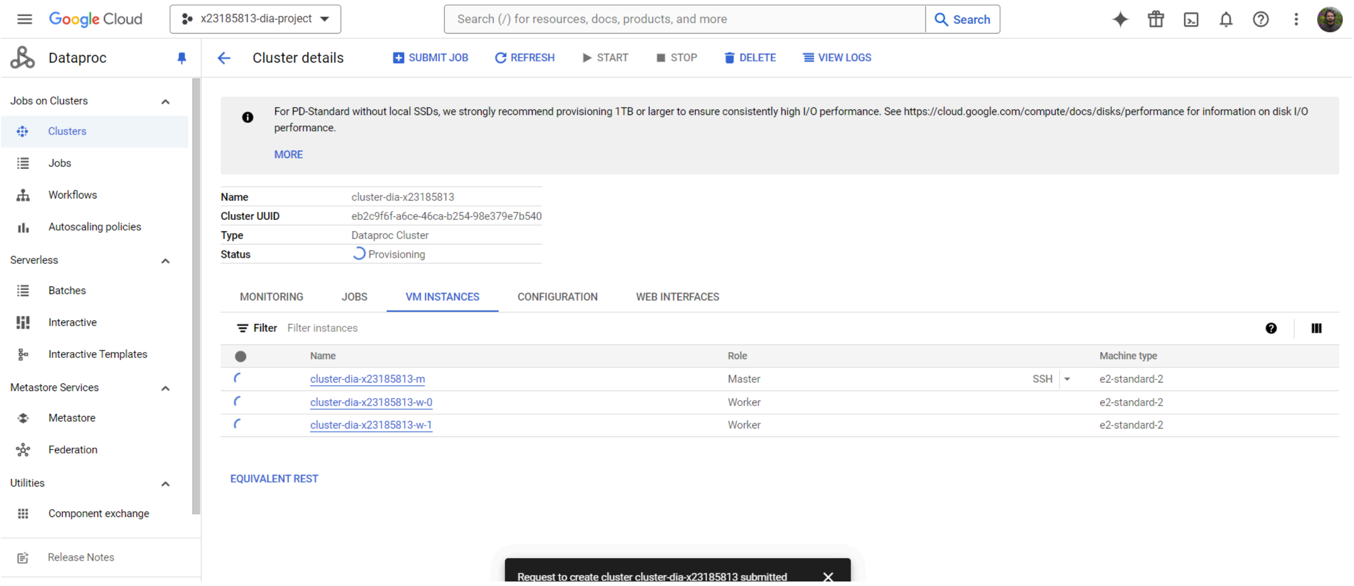
Enable

5. Create the cluster using **CREATE**



- Set up cluster
  Begin by providing basic information.
- Configure nodes (optional)
  Change node compute and storage capabilities.
- Customize cluster (optional)
  Add cluster properties, features, and actions.
- Manage security (optional)
  Change access, encryption, and security settings.

The dataproc will start provisioning the resources and will install all the required softwares.



For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/performance for information on disk I/O performance.

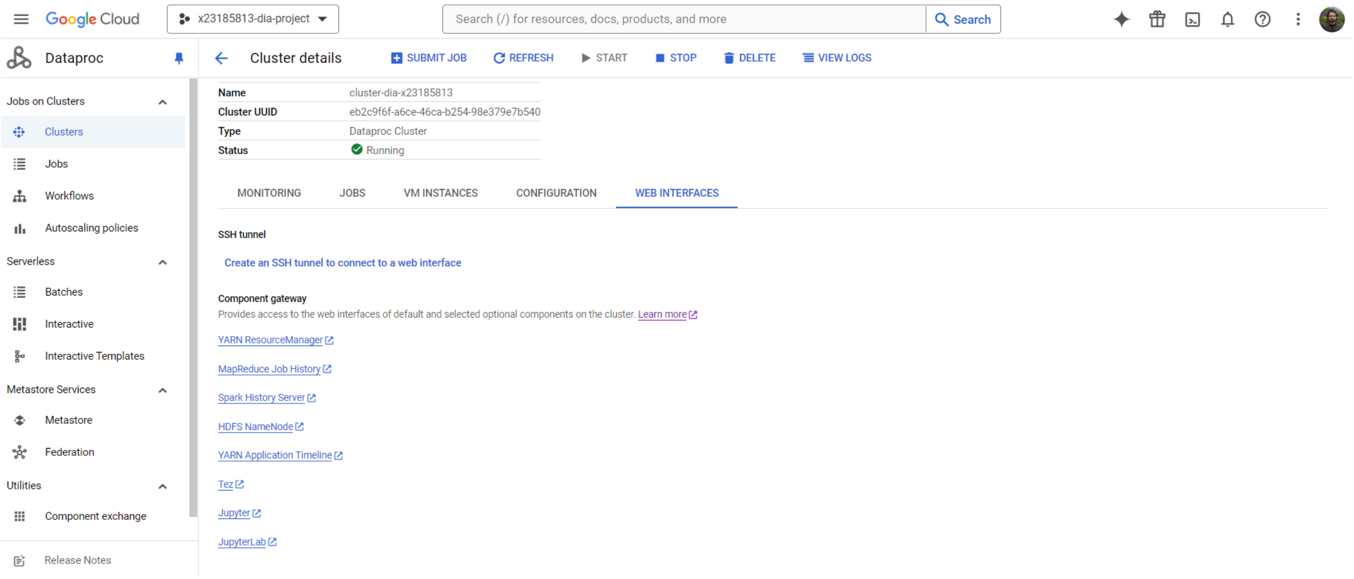
MORE

Name	cluster-dia-x23185813
Cluster UUID	eb2c9f6f-a6ce-46ca-b254-98e379e7b540
Type	Dataproc Cluster
Status	Provisioning

Name	Role		Machine type
cluster-dia-x23185813-m	Master	SSH	e2-standard-2
cluster-dia-x23185813-w-0	Worker		e2-standard-2
cluster-dia-x23185813-w-1	Worker		e2-standard-2

Request to create cluster cluster-dia-x23185813 submitted

6. Once the Cluster is up and running go to **WEB INTERFACES** select jupyterlab to open the jupyter interface



Name	cluster-dia-x23185813
Cluster UUID	eb2c9f6f-a6ce-46ca-b254-98e379e7b540
Type	Dataproc Cluster
Status	Running

SSH tunnel

Create an SSH tunnel to connect to a web interface

Component gateway

Provides access to the web interfaces of default and selected optional components on the cluster. Learn more

YARN ResourceManager

MapReduce Job History

Spark History Server

HDFS NameNode

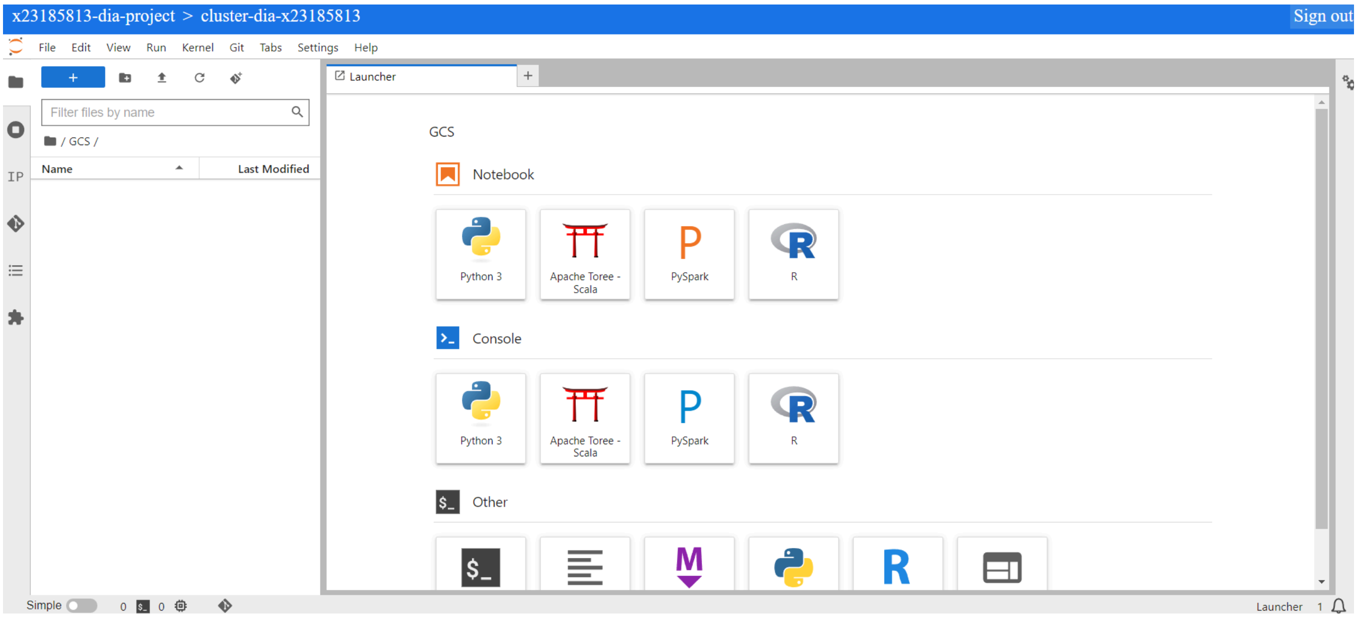
YARN Application Timeline

Tez

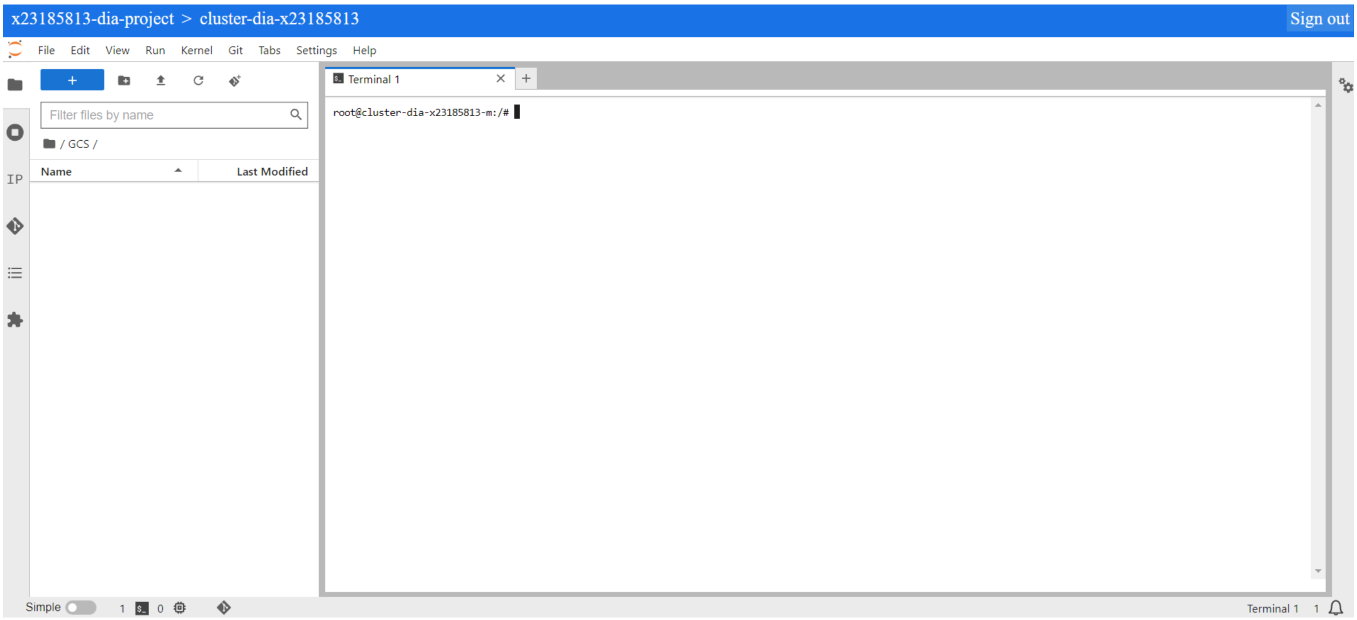
Jupyter

JupyterLab

7. The jupyterLab interface will look like this, In launcher select the console option



This will open a terminal window with root user



```
root@cluster-dia-x23185813-m:/#
```

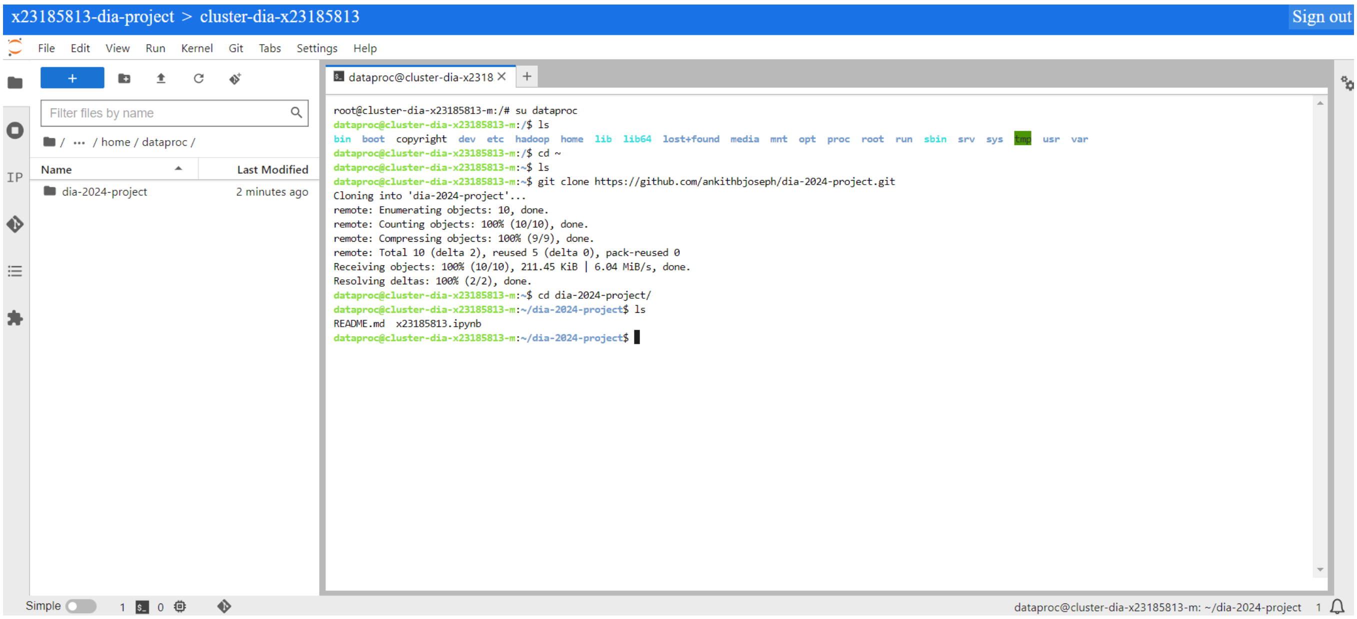
8. We will execute the following commands to setup the files from the git repository

```
su dataproc
```

```
cd ~
```

```
git clone https://github.com/ankithbjoseph/dia-2024-project.git
```

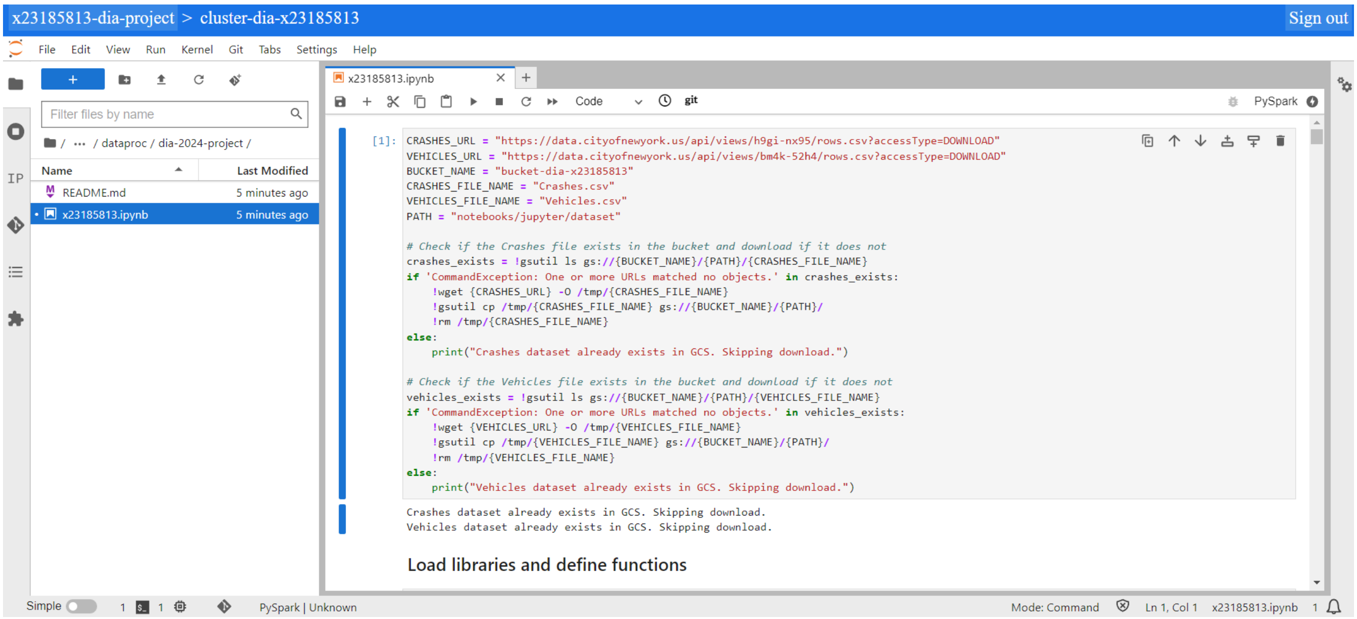
```
cd dia-2024-project/
```


x23185813-dia-project > cluster-dia-x23185813
Sign out
File Edit View Run Kernel Git Tabs Settings Help
Filter files by name
/ … / home / dataproc /
Name
Last Modified
dia-2024-project
2 minutes ago
dataproc@cluster-dia-x2318
Simple
dataproc@cluster-dia-x23185813-m: ~/dia-2024-project

```
root@cluster-dia-x23185813-m:/# su dataproc
dataproc@cluster-dia-x23185813-m:/$ ls
bin  boot  copyright  dev  etc  hadoop  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
dataproc@cluster-dia-x23185813-m:/$ cd ~
dataproc@cluster-dia-x23185813-m:~$ ls
dataproc@cluster-dia-x23185813-m:~$ git clone https://github.com/ankithbjoseph/dia-2024-project.git
Cloning into 'dia-2024-project'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 2), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), 211.45 KiB | 6.04 MiB/s, done.
Resolving deltas: 100% (2/2), done.
dataproc@cluster-dia-x23185813-m:~$ cd dia-2024-project/
dataproc@cluster-dia-x23185813-m:~/dia-2024-project$ ls
README.md  x23185813.ipynb
dataproc@cluster-dia-x23185813-m:~/dia-2024-project$ 
```

9. This will download the required files , In the file explorer navigate to the folder `home/dataproc/dia-2024-project/`

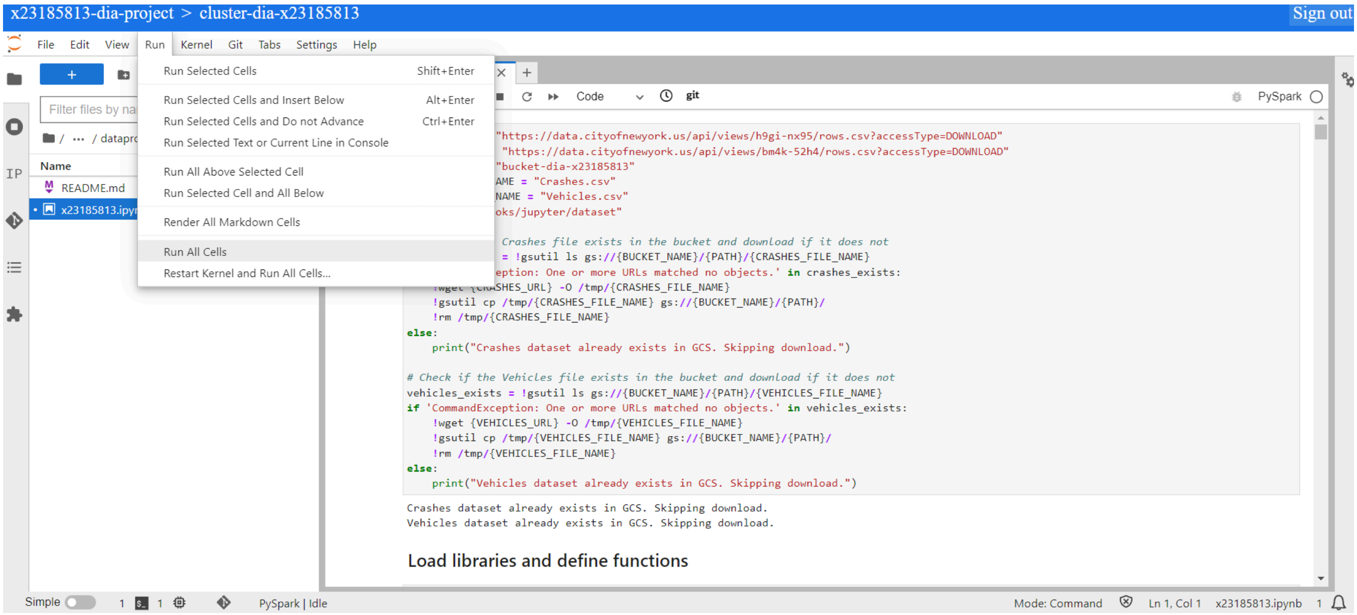
10. Open the notebook `x23185813.ipynb`



```
[1]: CRASHES_URL = "https://data.cityofnewyork.us/api/views/h9gi-nx95/rows.csv?accessType=DOWNLOAD"
     VEHICLES_URL = "https://data.cityofnewyork.us/api/views/bm4k-52h4/rows.csv?accessType=DOWNLOAD"
     BUCKET_NAME = "bucket-dia-x23185813"
     CRASHES_FILE_NAME = "Crashes.csv"
     VEHICLES_FILE_NAME = "Vehicles.csv"
     PATH = "notebooks/jupyter/dataset"

     # Check if the Crashes file exists in the bucket and download if it does not
     crashes_exists = !gsutil ls gs://{BUCKET_NAME}/{PATH}/{CRASHES_FILE_NAME}
     if 'CommandException: One or more URLs matched no objects.' in crashes_exists:
         !wget {CRASHES_URL} -O /tmp/{CRASHES_FILE_NAME}
         !gsutil cp /tmp/{CRASHES_FILE_NAME} gs://{BUCKET_NAME}/{PATH}/
         !rm /tmp/{CRASHES_FILE_NAME}
     else:
         print("Crashes dataset already exists in GCS. Skipping download.")

     # Check if the Vehicles file exists in the bucket and download if it does not
     vehicles_exists = !gsutil ls gs://{BUCKET_NAME}/{PATH}/{VEHICLES_FILE_NAME}
     if 'CommandException: One or more URLs matched no objects.' in vehicles_exists:
         !wget {VEHICLES_URL} -O /tmp/{VEHICLES_FILE_NAME}
         !gsutil cp /tmp/{VEHICLES_FILE_NAME} gs://{BUCKET_NAME}/{PATH}/
         !rm /tmp/{VEHICLES_FILE_NAME}
     else:
         print("Vehicles dataset already exists in GCS. Skipping download.")

     Crashes dataset already exists in GCS. Skipping download.
     Vehicles dataset already exists in GCS. Skipping download.
```

Load libraries and define functions

11. Run All cells


x23185813-dia-project > cluster-dia-x23185813
Sign out
File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
+
Filter files by na
/ ··· / datapro
Name
README.md
x23185813.ipyn
Run Selected Cells Shift+Enter
Run Selected Cells and Insert Below Alt+Enter
Run Selected Cells and Do not Advance Ctrl+Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
Code
git
PySpark
Simple
1
1
PySpark | Idle
Mode: Command
Ln 1, Col 1
x23185813.ipynb
1

```
"https://data.cityofnewyork.us/api/views/h9gi-nx95/rows.csv?accessType=DOWNLOAD"
 "https://data.cityofnewyork.us/api/views/bm4k-52h4/rows.csv?accessType=DOWNLOAD"
"bucket-dia-x23185813"
AME = "Crashes.csv"
NAME = "Vehicles.csv"
oks/jupyter/dataset"

 Crashes file exists in the bucket and download if it does not
 = !gsutil ls gs://{BUCKET_NAME}/{PATH}/{CRASHES_FILE_NAME}
eption: One or more URLs matched no objects.' in crashes_exists:
    !wget {CRASHES_URL} -O /tmp/{CRASHES_FILE_NAME}
    !gsutil cp /tmp/{CRASHES_FILE_NAME} gs://{BUCKET_NAME}/{PATH}/
    !rm /tmp/{CRASHES_FILE_NAME}
else:
    print("Crashes dataset already exists in GCS. Skipping download.")

# Check if the Vehicles file exists in the bucket and download if it does not
vehicles_exists = !gsutil ls gs://{BUCKET_NAME}/{PATH}/{VEHICLES_FILE_NAME}
if 'CommandException: One or more URLs matched no objects.' in vehicles_exists:
    !wget {VEHICLES_URL} -O /tmp/{VEHICLES_FILE_NAME}
    !gsutil cp /tmp/{VEHICLES_FILE_NAME} gs://{BUCKET_NAME}/{PATH}/
    !rm /tmp/{VEHICLES_FILE_NAME}
else:
    print("Vehicles dataset already exists in GCS. Skipping download.")
```

```
Crashes dataset already exists in GCS. Skipping download.
Vehicles dataset already exists in GCS. Skipping download.
```

Load libraries and define functions