

Yog Chaudhary

11727095

ADTA 5240 Week 13'th (harvesting, Storing, And Retrieving Data)

Professor: Dr. Zeynep Orhan

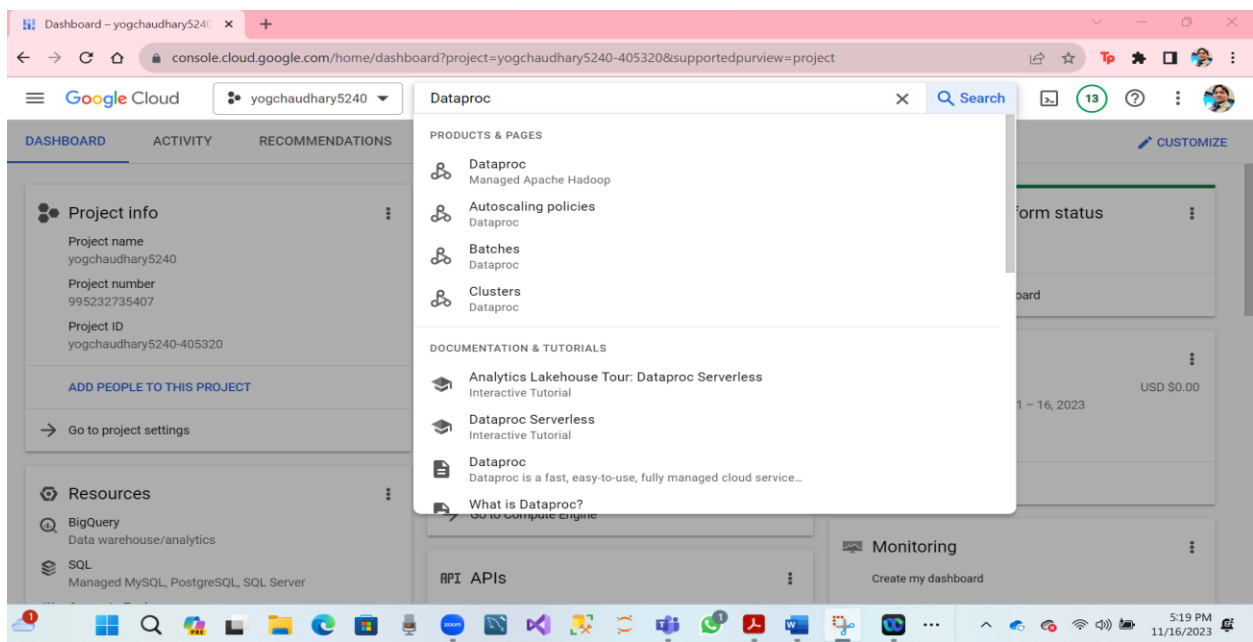
University Of North Texas

Nov 16, 2023

Homework Assignment: Hive Queries

Step 1. For a Google Console.

- I click on three horizontal lines.
- In the search bar I typed Cluster and clicked on Clusters Dataproc.
- Here screenshot.



- After clicking on cluster data proc. It will be mentioned that I have previously created, and it will show that was it running.
- Here screenshot.

The screenshot shows the Google Cloud Marketplace page for the Dataproc Metastore API. The page title is "Dataproc Metastore API" with a subtitle "Google Enterprise API". Below the title, it says "Fully-managed, OSS-native technical metadata management." There are two buttons: "ENABLE" and "TRY THIS API". The page has tabs for "OVERVIEW", "PRICING", "DOCUMENTATION", and "RELATED PRODUCTS". The "OVERVIEW" tab is selected, showing a description of the service and its details. The "Additional details" section lists the type as "SaaS & APIs", the last product update as "4/30/22", and the category as "Big data, Google Enterprise APIs".

Dataproc Metastore API
Google Enterprise API

Fully-managed, OSS-native technical metadata management.

[ENABLE](#) [TRY THIS API](#)

OVERVIEW PRICING DOCUMENTATION RELATED PRODUCTS

Overview

Dataproc Metastore is a fully managed, highly available, auto-scaled, auto-healing, OSS-native metastore service that greatly simplifies technical metadata management. Dataproc Metastore service is based on Apache Hive metastore and serves as a critical component towards enterprise data lakes.

Additional details

Type: [SaaS & APIs](#)
Last product update: 4/30/22
Category: [Big data](#), [Google Enterprise APIs](#)

The screenshot shows the Google Cloud Dataproc Clusters page. The left sidebar contains a navigation menu with options like "Jobs on Clusters", "Clusters", "Jobs", "Workflows", "Autoscaling policies", "Serverless", "Batches", "Interactive", "Metastore Services", "Metastore", and "Release Notes". The "Clusters" option is selected. The main content area shows a table of clusters. The table has columns for Name, Status, Region, Zone, Total worker nodes, Flexible VMs?, Scheduled deletion, and Cloud Storage staging location. There is one cluster listed: "hadoop-spark-4-cluster" with a status of "Running", region of "us-central1", zone of "us-central1-a", and 2 total worker nodes. The table also includes a "Filter" bar and a "Search clusters, press Enter" prompt.

Dataproc

Jobs on Clusters

- Clusters
- Jobs
- Workflows
- Autoscaling policies

Serverless

- Batches
- Interactive

Metastore Services

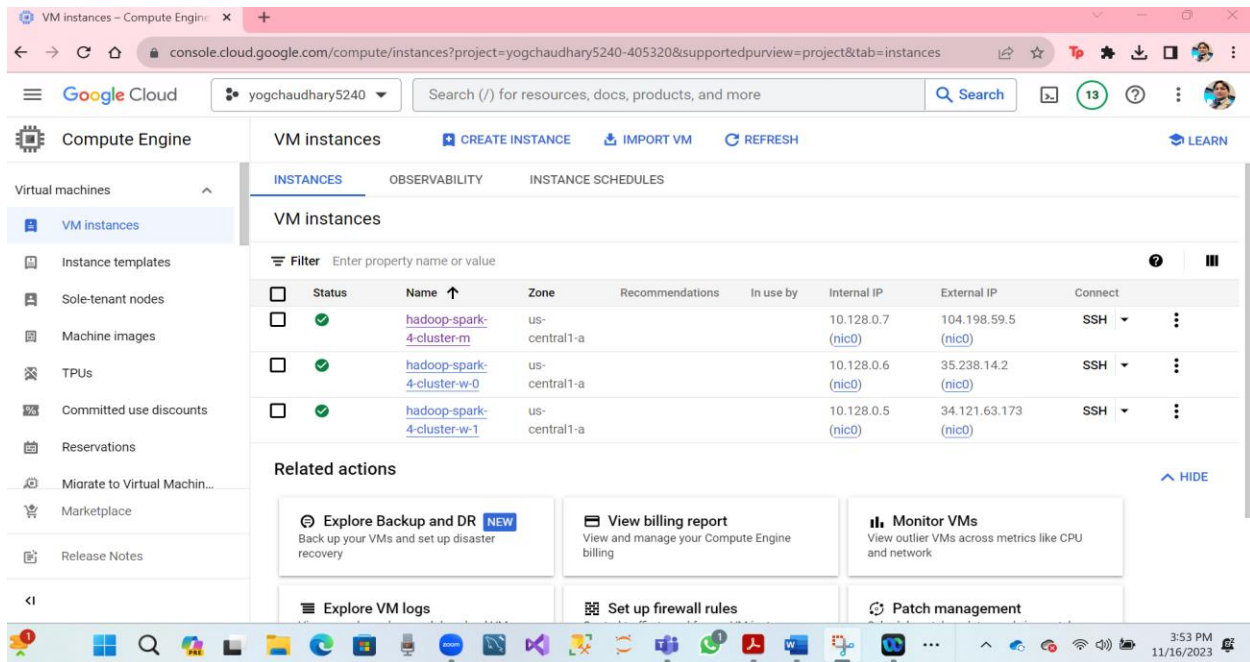
- Metastore
- Release Notes

Clusters [CREATE CLUSTER](#) [REFRESH](#) [START](#) [STOP](#) [DELETE](#) [REGIONS](#) [+ 5 RECOMMENDED ALERTS](#)

Filter Search clusters, press Enter

<input type="checkbox"/>	Name ↑	Status	Region	Zone	Total worker nodes	Flexible VMs?	Scheduled deletion	Cloud Storage staging location
<input type="checkbox"/>	hadoop-spark-4-cluster	Running	us-central1	us-central1-a	2	No	Off	yogchaudhary5240

- When I scroll down it will show how the monitoring is changing according to the usage.
- By scrolling up and clicking on the virtual machine the screen will show the cluster details.
- After that I accessed the master node through SSH.
- Click on SSH
- Click on open in a browser.
- Click on the drop-down button next to SSH.
- Click on open in a new browser window.



- Click authorized.

Step 2. We open the SSH terminal. This connected our local to the remote server. We can use an SSH terminal to work with the cluster.

- First basic Linux Commands
- Whoami
- Pwd
- Hdfs dfs -ls /
- Hdfs dfs -ls /user
- Hdfs dfs -ls /user/yogchaudhary2459
- Ls

```
ssh.cloud.google.com/v2/ssh/projects/yogchaudhary5240-405320/zones/us-central1-a/instances/hadoop-spark-4-cluster-m?authuse...
SSH-in-browser
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ whoami
yogchaudhary2459
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ pwd
/home/yogchaudhary2459
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ hdfs dfs -ls /
Found 3 items
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /tmp
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /var
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ hdfs dfs -ls /user
Found 11 items
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/dataproc
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/hbase
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/hdfs
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/hive
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/mapred
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/pig
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/solr
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/spark
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/yarn
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/zeppelin
drwxrwxrwt - hdfs hadoop 0 2023-11-16 21:35 /user/zookeeper
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ hdfs dfs -ls /user/yogchaudhary2459
ls: '/user/yogchaudhary2459': No such file or directory
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ ls
userdata.csv weblog.csv
yogchaudhary2459@hadoop-spark-4-cluster-m:~$
```

Step 3. We created the interface commands. It uses the following Hive command.

- beeline -u jdbc:hive2://localhost:10000

```
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ ls
userdata.csv weblog.csv
yogchaudhary2459@hadoop-spark-4-cluster-m:~$ beeline -u jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://localhost:10000>
```

Step 4 We Created the following command table names.

- Show tables.
- Hive command ends with Pay attention a semicolon. It will get an error.
- We have not created band any tables (tab_name) in our database in our cluster, .
- Hare screenshot.

```

yogchaudhary2459@hadoop-spark-4-cluster-m:~$ beeline -u jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://localhost:10000> show tables;
INFO : Compiling command(queryId=hive_20231116215951_dde0108d-e0d0-419b-bc38-9fcf20dfc39e): show tables
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from deserializer)], properties:null)
INFO : Completed compiling command(queryId=hive_20231116215951_dde0108d-e0d0-419b-bc38-9fcf20dfc39e); Time taken: 1.095 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hive_20231116215951_dde0108d-e0d0-419b-bc38-9fcf20dfc39e): show tables
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20231116215951_dde0108d-e0d0-419b-bc38-9fcf20dfc39e); Time taken: 0.046 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+
| tab_name |
+-----+
+-----+
No rows selected (1.479 seconds)
0: jdbc:hive2://localhost:10000>

```

Step 5 Create a table (schema) with SQL Commands. Hive can access the data; we need to create a table or schema that maps the database saved in HDFS.

- Using the following script, we will create a very simple table or schema to map the dataset weblog.csv in HDFS. It is best to type this script in a text editor so you can correct any mistakes before copying and pasting into Hive.
- **CREATE EXTERNAL TABLE IF NOT EXISTS weblogs_3 (weblog string) ROW FORMAT DELIMITED STORED AS TEXTFILE LOCATION '/user/yogchaudhary2459/data/weblog/';**

Insert the script:

show tables;

- Hare screenshot

```

0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE IF NOT EXISTS weblogs_3
. . . . .> (weblog string)
. . . . .> ROW FORMAT DELIMITED STORED AS TEXTFILE
. . . . .> LOCATION '/user/yogchaudhary2459/data/weblog/';
INFO : Compiling command(queryId=hive_20231116230302_c0fe20d1-09cb-43de-840b-fc4c8ba2792e): CREATE EXTERNAL TABLE IF NOT EXISTS weblogs_3
(weblog string)
ROW FORMAT DELIMITED STORED AS TEXTFILE
LOCATION '/user/yogchaudhary2459/data/weblog/'
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO : Completed compiling command(queryId=hive_20231116230302_c0fe20d1-09cb-43de-840b-fc4c8ba2792e); Time taken: 0.036 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hive_20231116230302_c0fe20d1-09cb-43de-840b-fc4c8ba2792e): CREATE EXTERNAL TABLE IF NOT EXISTS weblogs_3
(weblog string)
ROW FORMAT DELIMITED STORED AS TEXTFILE
LOCATION '/user/yogchaudhary2459/data/weblog/'
INFO : Completed executing command(queryId=hive_20231116230302_c0fe20d1-09cb-43de-840b-fc4c8ba2792e); Time taken: 0.001 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
No rows affected (0.051 seconds)
0: jdbc:hive2://localhost:10000>

```

- We have created tables. And we can see below that the table was created. This allows us to access the data through Hive and make queries.
- Hare screenshot.

```

yogchaudhary2459@hadoop-spark-4-cluster-m:~$ beeline -u jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://localhost:10000> show tables;
INFO : Compiling command(queryId=hive_20231116224700_77ae9c02-5272-4c09-add5-cble933298bc): show tables
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from deserializer)], properties:null)
INFO : Completed compiling command(queryId=hive_20231116224700_77ae9c02-5272-4c09-add5-cble933298bc); Time taken: 0.026 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hive_20231116224700_77ae9c02-5272-4c09-add5-cble933298bc): show tables
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20231116224700_77ae9c02-5272-4c09-add5-cble933298bc); Time taken: 0.014 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+
| tab_name |
+-----+
| weblogs_3 |
+-----+
1 row selected (0.256 seconds)
0: jdbc:hive2://localhost:10000> 

```

Step 6. We used to insert a command to query the tables.

- SELECT * FROM weblogs_3 LIMIT 1;

```

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1          container  SUCCEEDED    0         0         0         0         0         0
-----
VERTICES: 00/01 [>-----] 0%    ELAPSED TIME: 0.83 s
-----
+-----+
| weblogs_3.weblog |
+-----+
+-----+
No rows selected (8.939 seconds)
0: jdbc:hive2://localhost:10000> 

```

Step 7. We created a complex table (schema) following the command:

- SELECT userid, COUNT(userid) AS log_count FROM weblog_8 GROUP BY userid ORDER BY log_count DESC LIMIT 5;
- Hare screenshot

```

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1          container  SUCCEEDED    0         0         0         0         0         0
Reducer 2 ..... container  SUCCEEDED    1         1         0         0         0         0
Reducer 3 ..... container  SUCCEEDED    1         1         0         0         0         0
-----
VERTICES: 02/03 [=====] 100%    ELAPSED TIME: 7.91 s
-----
+-----+-----+
| userid | log_count |
+-----+-----+
+-----+-----+
No rows selected (17.457 seconds)
0: jdbc:hive2://localhost:10000> 
0: jdbc:hive2://localhost:10000> 

```

- Finally, Successfully Hive cluster and now exit.