**Recommendation Engine on GCP**

**Goals**

1. Proof-of-concept of Serverless Recommendation Engine for Market Data Lake & Catalog
2. Evaluate Big Data/ML capabilities, performance, cost, and security of GCP
3. Provide reusable codebase and template to speed up future ML development

**Technologies Evaluated**



**Cloud SQL**

**Cloud SDK**

**Cloud Shell**

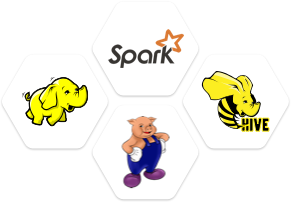
**Jupyter Notebooks**

**Apache MLlib**

**Cloud Datalab**

**Cloud Dataproc**

**Cloud Storage**



Resources:

[Cloud Storage Options](https://cloud.google.com/storage-options/)

[Cloud SQL MySQL Second Generation](https://cloud.google.com/sql/docs/mysql/1st-2nd-gen-differences)

[Cloud Dataproc Version 1.1](https://cloud.google.com/dataproc/docs/concepts/dataproc-versions)

[Cloud Datalab Preinstalled Packages](https://cloud.google.com/datalab/docs/concepts/key-concepts)

[Apache MLlib Collaborative Filtering](http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html)

**Assumptions**

* Focus is on Collaborative Filtering for now, stay tuned for additional improvements using Content-based, Knowledge-based, and Deep learning approaches.
* Fake data has been created for purpose of this POC that resembles our problem space but with dummy names for Vendors, Products, Services. Data format and column names can be used for final technical implementation.
* Assume that all necessary ETL, Feature Selection, and Feature Engineering has already taken place and input data is in a Model training-ready state.
* Assume Ratings can either be actual user star ratings, or some derived category based on some metric we are capturing, such as # of search hit matches or # of downloads. For e.g. 5 Star 🡪 Searched >= 50 times, 4 Star 🡪 40 =< Searched < 50, 3 Star 🡪 30 =< Searched < 40, etc.

**Takeaways**

* Serverless workloads can be customized and automated via cloud with great performance & cost savings
* Huge time savings in preinstalled Spark/Hadoop infra on Dataproc and Data Science/ML/NLP packages on Datalab
* Security layer for explicit access to connect to Cloud SQL Data
* Compress data (gz) to reduce costs to Cloud Storage
* Spark 1.4 allows direct writes to Google Cloud SQL DB from PySpark
* Spark SQL context connects to Google Cloud SQL via JDBC connector and returns back a DataFrame
* We then convert DataFrame to RDD to be used by Spark for parallelizing the ML workload
* Finally we use MLlib to train and tune the model

Metrics:

* Cluster Spec:
  + 1 Master, 2 Worker nodes
  + All n1-standard-2 nodes: 2CPU x 7.5GB Memory x 1 GB Disk
  + Rate per node: $0.0950/hour, $48.55/month
* Data Size: 5000 service ratings, 600 services, 150 users
* Spinning up Dataproc Cluster: 1-2 min
* Running ML model: 2 min

**Recommender System Approaches**

1. **Content-based Filtering**: Recommend products that have similar attributes to what the user searched for or used in the past.

* Generic Example: Recommended items based on your recently viewed items and browsing history
* Market Data Example: User who has searched “Index constituents” 3 times a week and used Service A exclusively could get recommendation to use Service B which also has similar indices data.
* Models: TF-IDF + Cosine Similarities, Bayesian Classifiers, Clustering, Neural Networks
* Pros: Addresses cold start problem, when you don’t have enough user rating data but have item attribute data
* Cons:  The system is limited to recommending content of the same type as the user is already using

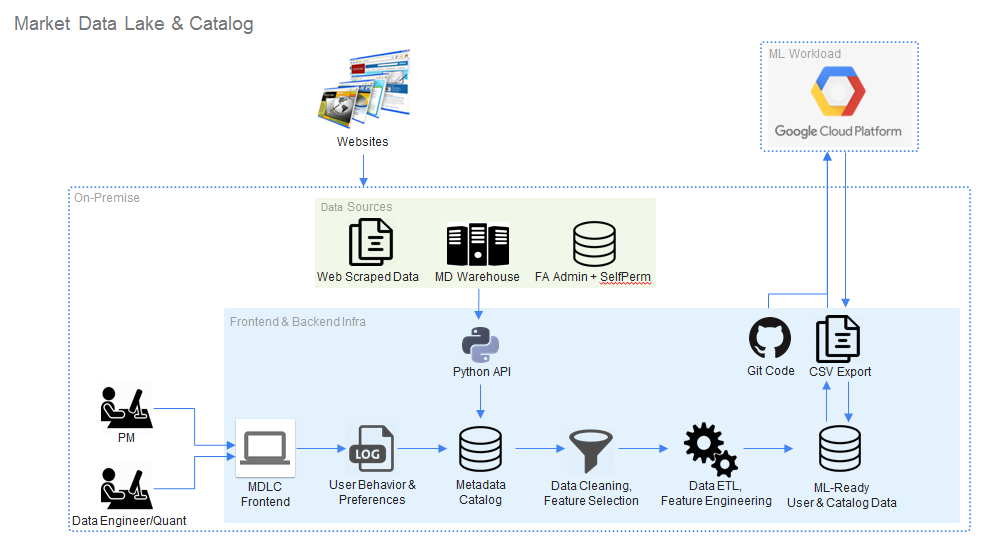
1. **Collaborative Filtering**: Two or more different users who liked the same products in the past will probably like the same ones now and in the future.

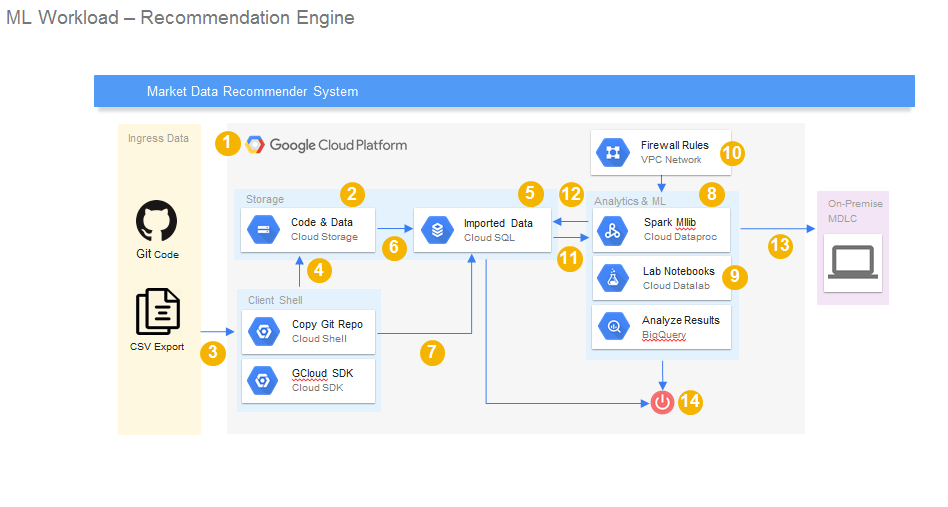
* Generic Example: Customers who bought Item X also bought Item Y (Item-to-item)
* Market Data Example: Users who bought and used Service X also use Service Y.
* Models: ALS
* Pros: Very popular, can provide recommendations without understanding content of service and only user behavior
* Cons: Cold start problem, not as accurate when we don’t have enough user rating data and only item attribute data

1. **Clustering**: Recommend products that go well together, no matter what other users have done.

* Generic Example: Items commonly bought together
* Market Data Example: Two services that are supposed to be used together, one containing reference data for a set of securities and the other containing the prices and fundamentals for the same set of Securities. Regardless of what users have used or searched in the past, these two services are meant to be used together.
* Models: K-means
* Pros:
* Cons: Not good by itself, needs to be combined with other filtering methods for cross-category selling

**System Flow Diagram**

****

****

Numbers in above diagram correspond to the Steps below

**Steps**

**Create Project**

1. Go to [**Cloud Platform Console**](https://console.cloud.google.com/)(Henceforth called GCP Console) **🡪** Create a new Project by selecting a Project from top left button 🡪 Create a new **Project** 🡪 Choose a globally unique project name 🡪 Click **Create 🡪** We will refer to this as <PROJECT-NAME> in future steps.

**Create Cloud Storage**

1. Switch to your new Project in **GCP Console** 🡪 Select **Storage** from top left menu 🡪 **Create Bucket** 🡪 Choose a globally unique bucket name (your project name is unique, so you could use that) 🡪 Click **Create 🡪** We will refer to this as <BUCKET-NAME> in future steps.

CLI alternative:

gsutil mb -c regional -l us-central1 gs://$DEVSHELL\_PROJECT\_ID

**Activate Cloud Shell and Copy Git Repo Code**

1. From **GCP Console** 🡪 Click **Activate** **Cloud Shell** on the top right menu to start a local unix shell 🡪 On the command-line, you can copy over the code assets from internal MLP Github branch to your local Cloud Sandbox environment. Request for permission from Market Data Engineering Team if your access is denied.

git clone http://github.mlp.com/md/catalog/tree/master/gcp/recommendation

Alternatively, the code will also be uploaded to our team shared drive at M:\MLP\_Data\_Catalog\GCP\POC1 which you can make a copy and download

**Copy Cloud Shell code to Cloud Storage**

1. Check that the directory contents was successfully copied over using ls first, then type the below to copy code assets to Cloud Storage. Cloud Shell comes with many connectors and prebuilt functions and “gsutil” is one of the APIs that lets you copy over data from local Cloud sandbox to Cloud Storage buckets, which are prefixed with “gs://” to indicate Google Storage. For this example we’ll also create a subdirectory under our Bucket called recommendation.

gsutil cp gcp/recommendation/\* gs://<BUCKET-NAME>/recommendation/

From **GCP Console** navigate to Storage 🡪 Click your <BUCKET-NAME> and make sure you can now see the code stored in Google Cloud Bucket.

**Create Cloud SQL**

1. From **GCP Console** 🡪 Select **SQL** under the Storage section 🡪 Create Instance 🡪 Choose **MySQL** 🡪 Choose **Cloud SQL Second Generation** (Recommended) 🡪 Name the instance ID <CLOUDSQL-NAME> 🡪 Scroll down and specify a **Root Password** of your choice to protect access to your MySQL DB, just be sure to write it down in the table below 🡪 Select region as **us-central1,** zone as **Any**

|  |  |
| --- | --- |
| **Password/IP Address** | **Fill in actual values** |
| Root Password | <CLOUDSQL-PW> |
| Cloud Shell IP Address | <CLOUDSHELL-IP> |
| Cloud SQL IP Address | <CLOUDSQL-IP> |

CLI alternative:

gcloud sql tiers list #This shows you what tier machine instance you want to create a MySQL db in

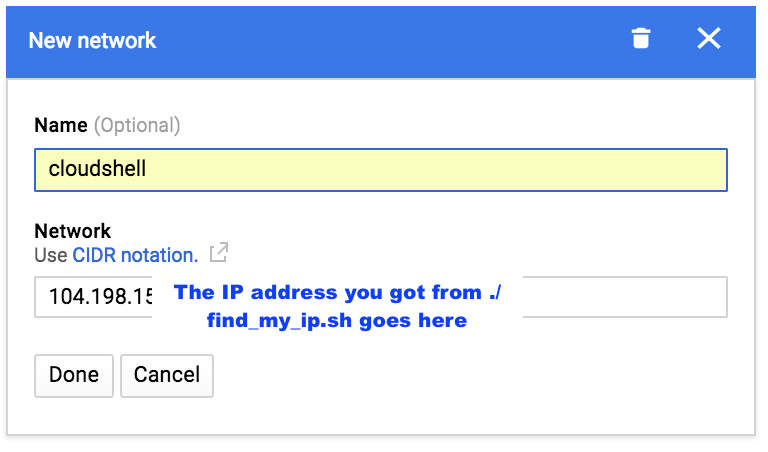
gcloud sql instances create <CLOUDSQL-NAME> --tier=db-n1-standard-1 --region=us-central1

Within Cloud Shell under ~/gcp/recommendation/ directory that you copied over from Git, find your client IP address by calling the below script.:

./find\_my\_ip.sh

Scroll further down under **Advanced Settings** 🡪 **Authorize Networks** 🡪 Click **Add network** 🡪 Paste your IP address (name is optional, put as cloudshell) 🡪 Click **Done 🡪** Click **Create** at the very bottom. This will give your CloudShell access to connect to and perform queries on the CloudSQL instance.

Once created, note down the IP address of your newly provisioned Cloud SQL instance from GCP Console 🡪 SQL 🡪 Column called “IP address” to the table above. Also take note of the column “Location”, ours is **us-central1-c**. Later we want to make sure our Dataproc cluster is provisioned in the same location to minimize latency.



**Optional**: If you lose your Cloud Shell VM due to inactivity, you will have to reauthorize your new Cloud Shell VM with Cloud SQL. You can do this programmatically using the script below in ~/gcp/recommendation/

./authorize\_cloudshell\_to\_cloudsql.sh <CLOUDSQL-NAME>

**Import Files and Validate Loaded Data**

1. Click on the hyperlink of your <CLOUDSQL-NAME> from the SQL page 🡪 Click **Import** on the top menu bar 🡪 Click **Browse** and browse to your previously created <BUCKET-NAME> subdirectory and **Select** the script **create\_tables.sql**  🡪 Make sure Format of import is **SQL** 🡪 Click **Import**

Once that’s done 🡪 Click **Import** again 🡪 Select **services.csv** 🡪 Select Format of import as **CSV** 🡪 Fill out field **Database** as recommendation\_spark, and field **Table** as Services 🡪 Click **Import.**

Repeat the **Import** for **rating\_services.csv**, changing the table name to **Rating\_Services**.

More on Data import/export can be found at <https://cloud.google.com/sql/docs/mysql/import-export/>

To explore Cloud SQL, you can use the tool mysql from Cloud Shell by typing:

mysql --host=<CLOUDSQL-IP> --user=root --password

MySQL will prompt you for the root password which you created earlier so enter that when prompted 🡪 Once you are in the mysql prompt, type the below commands to set the database, show the tables, and do some selects to make sure the data and tables are properly loaded.

use recommendation\_spark;

mysql> show tables;

+--------------------------------+

| Tables\_in\_recommendation\_spark |

+--------------------------------+

| Rating\_Services |

| Recommendation |

| Services |

+--------------------------------+

3 rows in set (0.00 sec)

mysql> select count(\*) from Rating\_Services;

+----------+

| count(\*) |

+----------+

| 5000 |

+----------+

1 row in set (0.00 sec)

mysql> select count(\*) from Services;

+----------+

| count(\*) |

+----------+

| 600 |

+----------+

1 row in set (0.00 sec)

mysql> select \* from Rating\_Services limit 3;

+---------+------------+--------+

| user\_id | service\_id | rating |

+---------+------------+--------+

| 125 | 1 | 2 |

| 13 | 1 | 4 |

| 132 | 1 | 1 |

+---------+------------+--------+

3 rows in set (0.00 sec)

mysql> select \* from Services limit 3;

+-----+--------------+-------------+----------------------+-----------------------+-----------------------+--------------------+----------------------+------------+

| id | service\_name | vendor\_name | dominant\_asset\_class | dominant\_content\_type | dominant\_super\_region | num\_times\_searched | num\_users\_subscribed | avg\_rating |

+-----+--------------+-------------+----------------------+-----------------------+-----------------------+--------------------+----------------------+------------+

| 1 | Service\_1 | Vendor\_1 | OPTIONS | FINANCIAL MODEL | GLOBAL | 17 | 5 | 1.06 |

| 10 | Service\_10 | Vendor\_1 | FIXED INCOME | PRICING | GLOBAL | 13 | 5 | 2.81 |

| 100 | Service\_100 | Vendor\_2 | FIXED INCOME | FINANCIAL MODEL | GLOBAL | 6 | 91 | 4.63 |

+-----+--------------+-------------+----------------------+-----------------------+-----------------------+--------------------+----------------------

3 rows in set (0.00 sec)

List of other mysql commands to try- <https://www.computerhope.com/unix/mysql.htm>

**Update PySpark Script and Copy to Cloud Storage**

1. In the same directory, open up the recommendation\_engine.py file and modify the lines with #CHANGE marked next to it, which is the variables CLOUDSQL\_INSTANCE\_IP = <CLOUDSQL-IP>, CLOUDSQL\_PWD = <CLOUDSQL-PW>. Save the script changes and copy the updated script back to Cloud Storage Bucket.

gsutil cp gcp/recommendation/recommendation\_engine.py gs://<BUCKET-NAME>/recommendation/

**Create Dataproc Cluster and check pre-installed Software**

1. From GCP Console 🡪 Select **Dataproc** 🡪 Click **Create cluster** (Name will be referred as <DATAPROC-NAME> going forward) 🡪 Change the **Zone** to be in the same region as your Cloud SQL instance from Step 6 🡪 Set others attributes like below and click **Create**

Name: cluster-1

Zone: us-central1-c

Master node Machine type: n1-standard2 (2vCPU, 3.75GB memory)

Master node Primary disk size: 10 GB

Worker nodes Machine type: n1-standard2 (2vCPU, 3.75GB memory)

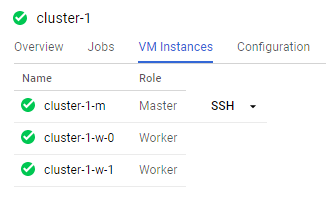
Worker nodes: 2

Work nodes primary disk size: 10 GB

CLI alternative:

gcloud dataproc clusters create cluster-1 --zone us-central1-c --master-machine-type n1-standard-2 --master-boot-disk-size 10 --num-workers 2 --worker-machine-type n1-standard-2 --worker-boot-disk-size 10

Once you see green check mark next to created cluster, click on the name to open **Cluster Details** 🡪 Click **VM Instances** tab🡪Click on the master node (cluster-1-m) 🡪 Click **SSH** and a new pop-up will appear and you are now logged into the cluster.



Type the below commands to check the pre-installed programs that you can immediately use 🡪 Type Exit to close the SSH window once you are done checking out the program versions that you require.

python --version

java -version

scala -version

pyspark --version

pig --version

hive --version

**Optional: Create Dataproc Cluster with pre-installed Cloud Datalab and Google Python APIs**

1. You could create barebones Dataproc clusters as in step 10, but the real power of Dataproc is that it allows you to leverage ‘Initialization Actions’ to pre-install useful APIs and Cloud Datalab to any Cluster you create. To do this, we follow same step as Step 10, except we pass in extra arguments in our CLI create cluster command to install Cloud Datalab and Google Python Client APIs.

gcloud dataproc clusters create cluster-1 --zone us-central1-c \

--master-machine-type n1-standard-2 --master-boot-disk-size 10 \

--num-workers 2 --worker-machine-type n1-standard-2 \

--worker-boot-disk-size 10 --network=default \

--scopes 'https://www.googleapis.com/auth/cloud-platform' \

--project $DEVSHELL\_PROJECT\_ID \

--initialization-actions 'gs://dataproc-initialization-actions/datalab/datalab.sh', \

'gs://mdlc\_ml/recommendation/install\_python\_api\_to\_dataproc.sh'

The script from ‘gs://dataproc-initialization-actions/datalab/datalab.sh’ above installs Google Cloud Datalab on the Cluster's Master node, while ‘gs://mdlc\_ml/unstructured/init-script.sh’ installs Google Python Client API on all machines in the Cluster.

Initialization actions are just executables that run when a cluster is being created. They are used to install additional software or customize your cluster. You can browse <https://github.com/GoogleCloudPlatform/dataproc-initialization-actions> where Google has pre-written many other initialization actions to be leveraged.

**Access Dataproc Cluster with Browser**

1. You are going to allow access to your Dataproc cluster, but only to your machine. Go to this URL to find your local IP address: <http://ip4.me/>

From GCP Console 🡪 Click **VPC** **Networks (**Previously called Networking) from left hand menu 🡪 **Firewall rules** tab 🡪 Click **Create Firewall Rule** 🡪 Enter the below 🡪 Click **Create**

Name: **default-allow-dataproc-access**

Targets: **All instances in the network**

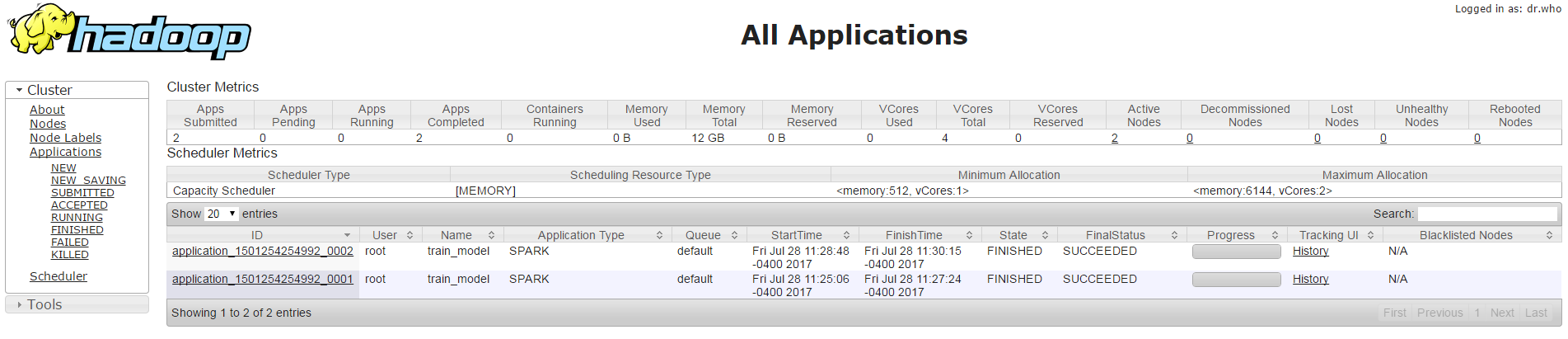
**Source filter:** **IP ranges**

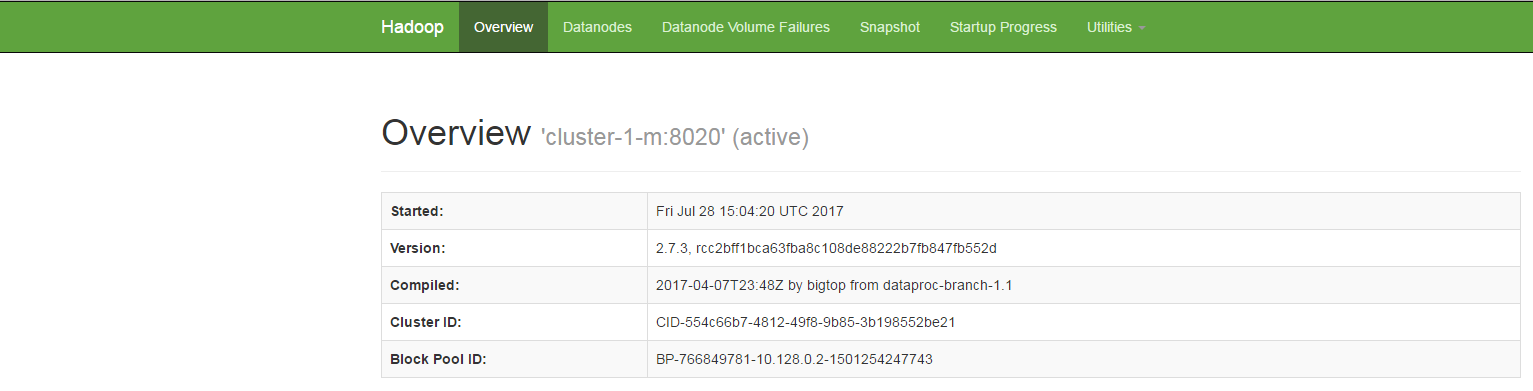
Source IP ranges: **Enter your IP address followed by /32. E.g. 1.2.3.4/32**

P**rotocols and ports**: **Specified Protocols and ports** 🡪 **tcp:8088;tcp:50070;tcp:8080**

Go back to the **Dataproc** 🡪 Click Cluster name to open **Cluster details 🡪** Click **VM Instances** tab 🡪 Click **master node 🡪** Scroll down to see master node's **External IP** address, and copy it to your clipboard.

**Note**: All the nodes in the Dataproc cluster are really Compute Engine virtual machines. You could also find the master node's IP address from **Products and Services** 🡪 Compute Engine 🡪 Find your master node named my-first-cluster-m 🡪 Copy the external IP address.

****Open new browser tab and paste in the master node's IP address, followed by the port **8088** to access Hadoop console. E.g. 1.2.3.4:8088 🡪 Should open page like below 🡪 Click the various links on left to explore

****Open another new browser tab and paste in the master node's IP address, followed by the port **50070** to access your HDFS cluster similar to below. Explore this as well.

If you did Optional Step #11 above to install Cloud Datalab, open another new browser tab and paste in the master node’s IP address, followed by the port **8080** to access your installed Google Cloud Datalab environment.

**Authorize Dataproc access to Cloud SQL**

1. In Cloud Shell 🡪 Exit mysql prompt by typing “exit” 🡪 cd to ~/gcp/recommendation/ 🡪 Run the below script with your actual Cluster and CloudSQL names to authorize the Cluster access to query the MySQL database.

cd ~/gcp/recommendation/\*

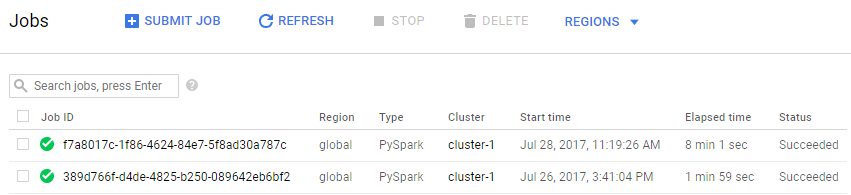
./authorize\_dataproc\_to\_cloudsql.sh <DATAPROC-NAME> <ZONE> <NUM-WORKER-NODES> <CLOUDSQL-NAME>

# Replace the args like ./authorize\_dataproc\_to\_cloudsql.sh cluster-1 us-central1-c 2 <CLOUDSQL-NAME>

**Submit PySpark Job**

1. On the left-hand menu of the Dataproc section, click on **Jobs** 🡪 Click **Submit job** on the top menu 🡪 Specify Job Type as **PySpark** 🡪 Main python file as **gs://<BUCKET-NAME>/recommendation/recommendation\_engine.py** 🡪 Click **Submit**

Once the job completes, you can click the Job ID to view the program log and see it go to completion.If the job failed, please troubleshoot using the logs and fix the errors. You may need to re-copy the updated Python file to Cloud Storage and clone the failed job and resubmit.



CLI alternative:

gcloud dataproc jobs submit pyspark --cluster cluster-1 \

gs://<BUCKET-NAME>/recommendation/recommendation\_engine.py

**Explore and Export Resulting Data**

1. In Cloud Shell, connect back to the mysql instance prompt and recommendation\_spark DB from Step 9 🡪 Perform some select statements to make sure you see new records inserted into Recommendation table.

select r.userid, r.accoid, r.prediction, a.title, a.location, a.price, a.rooms, a.rating, a.type from Recommendation as r, Accommodation as a where r.accoid = a.id and r.userid = 10;

You can then export the results from the MySQL table to a CSV or on-premise DB

**Clean Up**

1. Delete both the Cloud SQL and Dataproc instances from their respective console tabs to not get charged for them.

CLI alternative:

gcloud sql instances delete <CLOUDSQL-NAME>

gcloud dataproc clusters delete cluster-1