05a_retail_forecast_vertex_ai_notebook_execution.md

Retail Forecast using Serverless Spark through Vertex AI managed notebooks

Following are the lab modules:

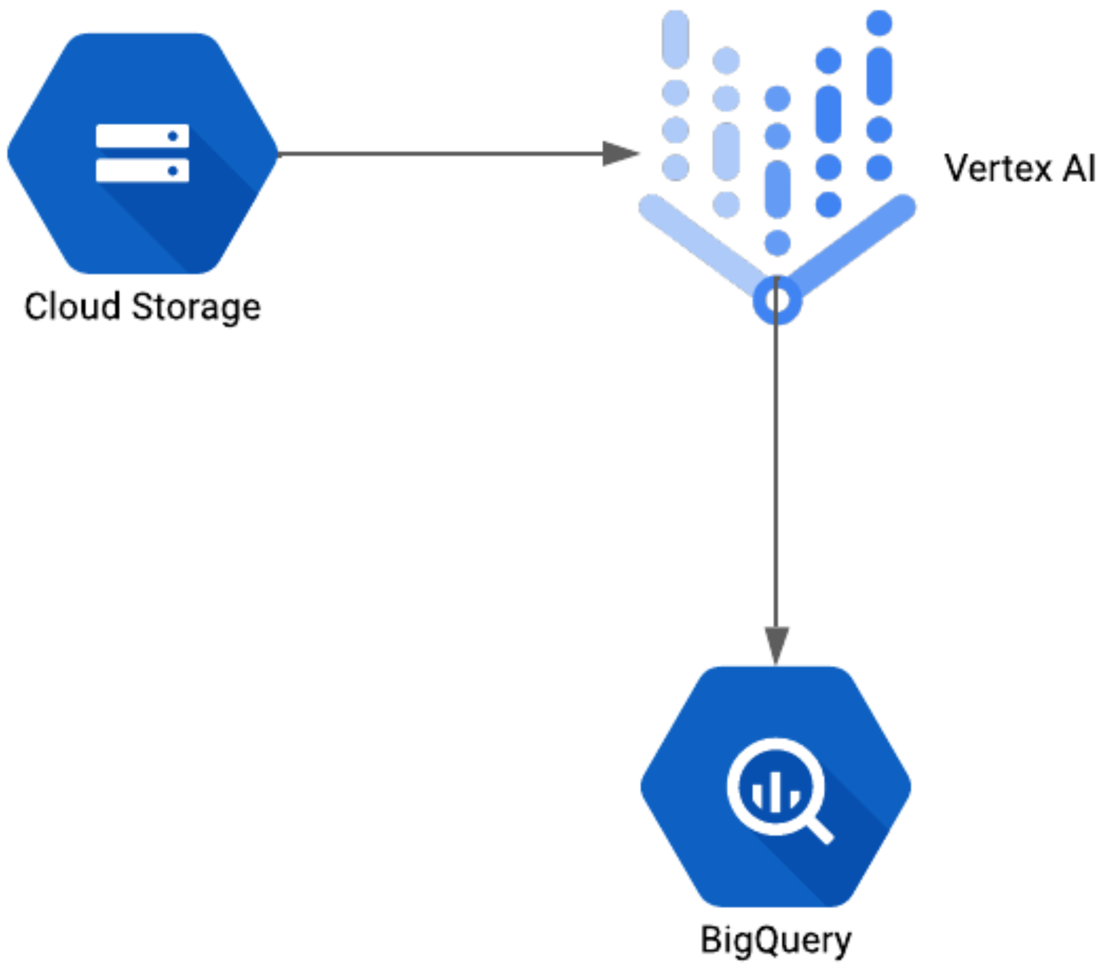
1. Understanding Data

The datasets used for this project are

1. Aisles data.
2. Departments data .
3. Orders data.
4. Products data.
5. Order_products__prior.
6. Order_products__train.

- Aisles: This table includes all aisles. It has a single primary key (aisle_id)
- Departments: This table includes all departments. It has a single primary key (department_id)
- Products: This table includes all products. It has a single primary key (product_id)
- Orders: This table includes all orders, namely prior, train, and test. It has single primary key (order_id).
- Order_products_train: This table includes training orders. It has a composite primary key (order_id and product_id) and indicates whether a product in an order is a reorder or not (through the reordered variable).
- Order_products_prior : This table includes prior orders. It has a composite primary key (order_id and product_id) and indicates whether a product in an order is a reorder or not (through the reordered variable).

2. Solution Architecture

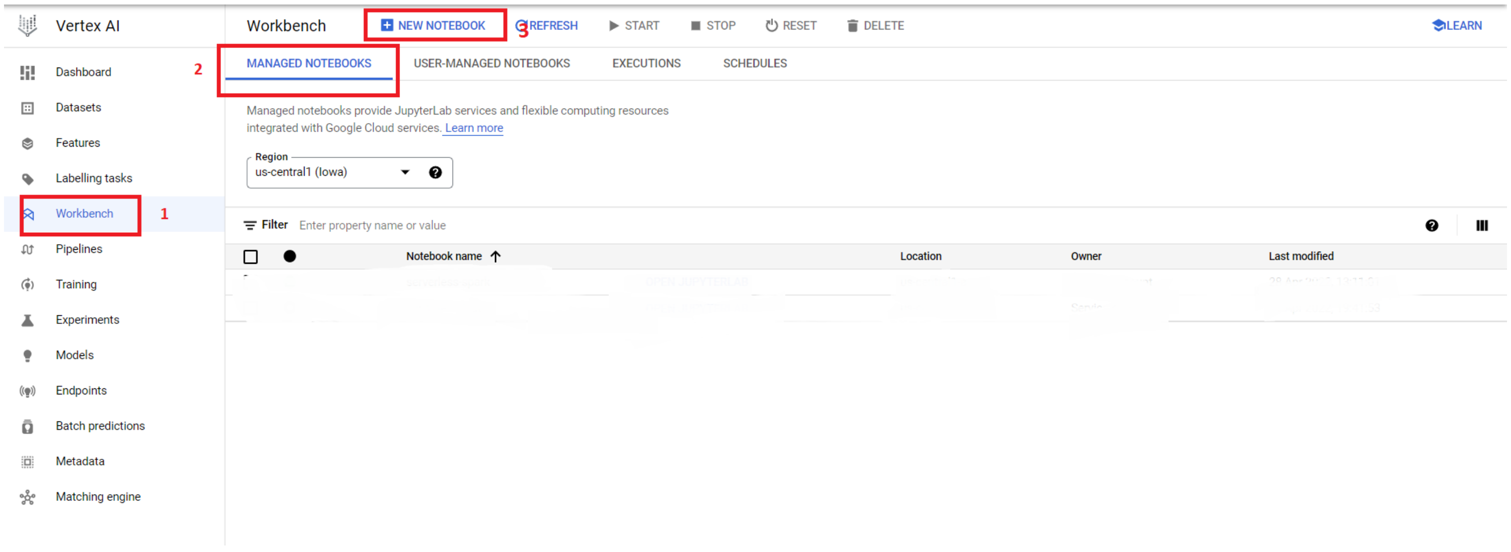


3. Execution

3.1. Run the Batch by creating session.

Creating Notebook in Vertex AI

Select Workbench from the left scroll bar of the Vertex AI main page. Select the Managed Notebooks tab. In the Managed Notebooks tab , click the New Notebook icon.

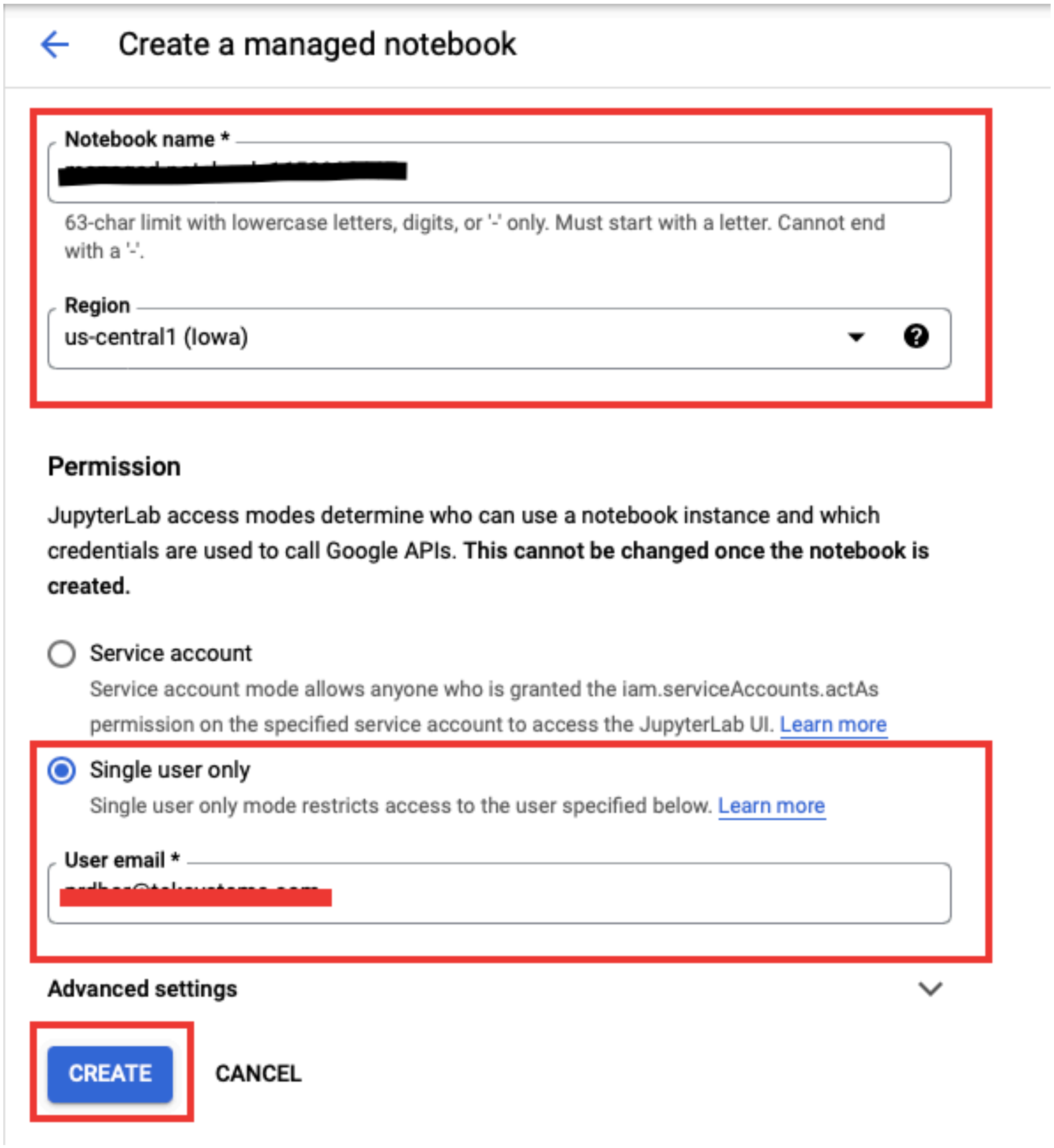


Managed notebooks provide JupyterLab services and flexible computing resources integrated with Google Cloud services. Learn more

Next, fill in the following values in the Notebook creation window as shown in the images below:

- **Notebook Name** - A unique identifier for your Notebook
- **Region** - The region name provided by the Admin team
- **Permission Type** - Single User Only (Single user only mode restricts access to the specified user)

- Provide a name and region to the notebook and select 'Single User Only' and click 'Create'. We will let the 'Advanced Settings' remain as the default values.


Create a managed notebook

Notebook name * [redacted]

63-char limit with lowercase letters, digits, or '-' only. Must start with a letter. Cannot end with a '-'.

Region: us-central1 (Iowa)

Permission

JupyterLab access modes determine who can use a notebook instance and which credentials are used to call Google APIs. **This cannot be changed once the notebook is created.**

Service account

Service account mode allows anyone who is granted the iam.serviceAccounts.actAs permission on the specified service account to access the JupyterLab UI. Learn more

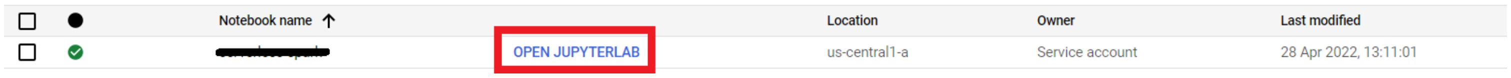
[✓ Single user only]

Single user only mode restricts access to the user specified below. Learn more

User email * [redacted]

Advanced settings

- Once the notebook is running, click the 'OPEN JUPYTERLAB' option next to the Notebook name as shown below

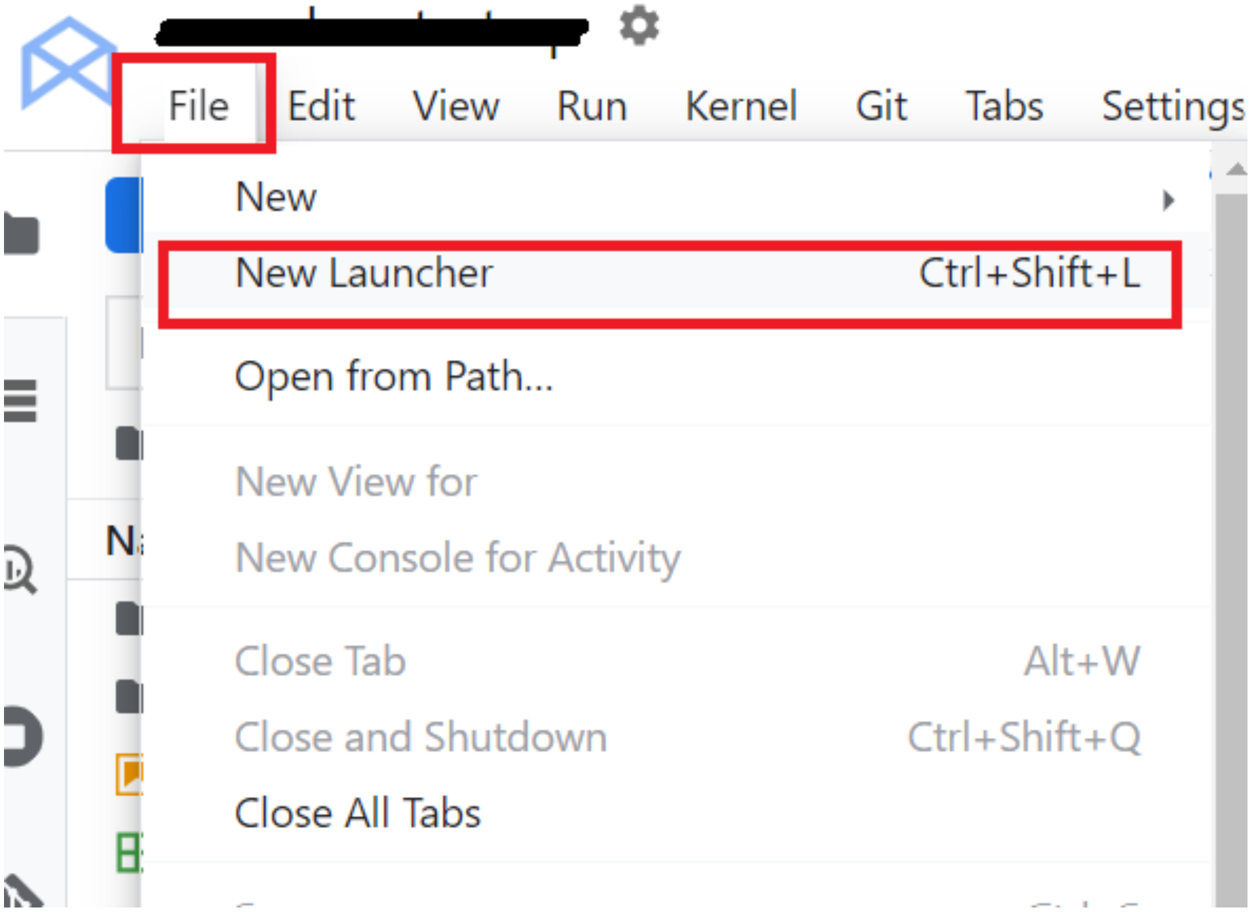


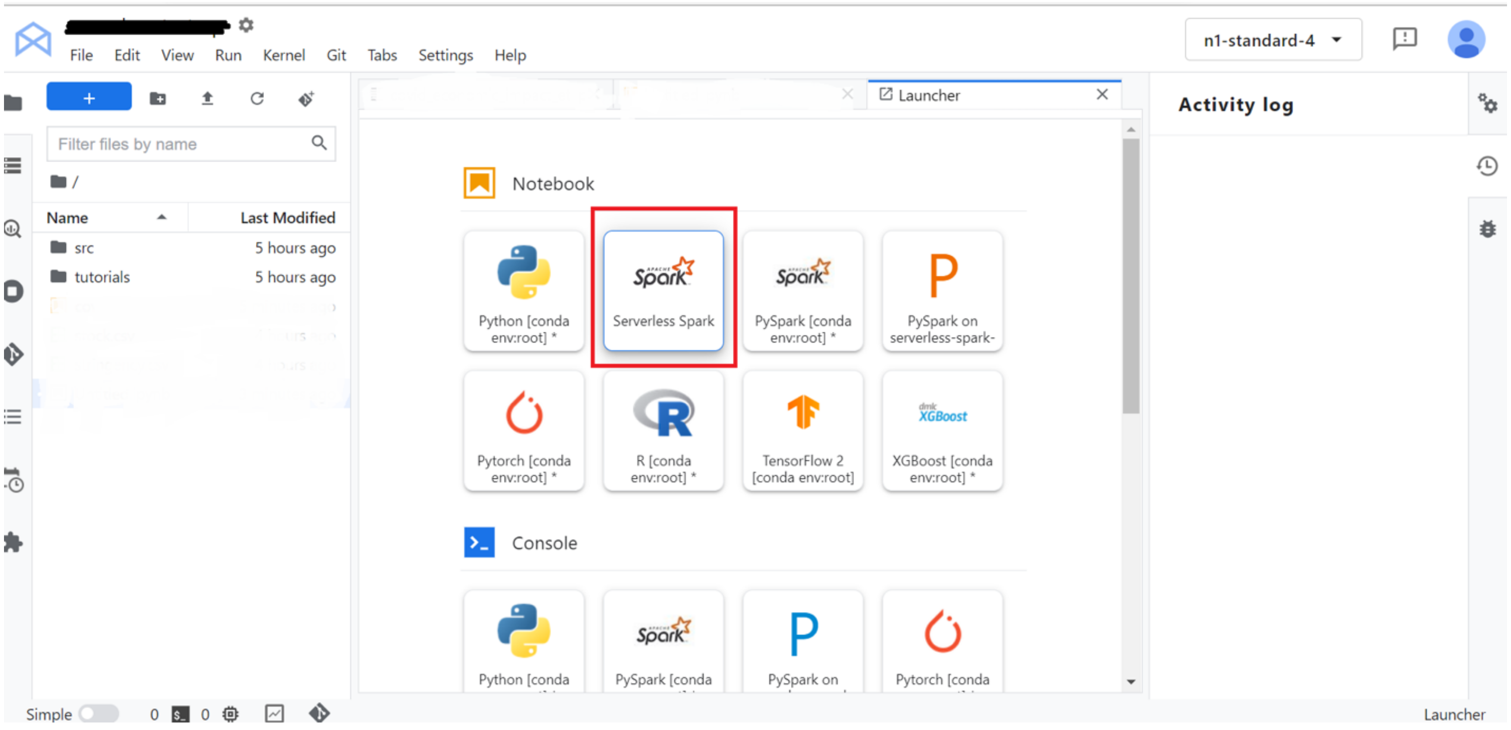
		Notebook name		Location	Owner	Last modified
☐	✓	[redacted]	OPEN JUPYTERLAB	us-central1-a	Service account	28 Apr 2022, 13:11:01

- Follow the on screen instructions to launch the JupyterLab session

Create Serverless Spark Session

- Click on the File and the New launcher and select Serverless Spark





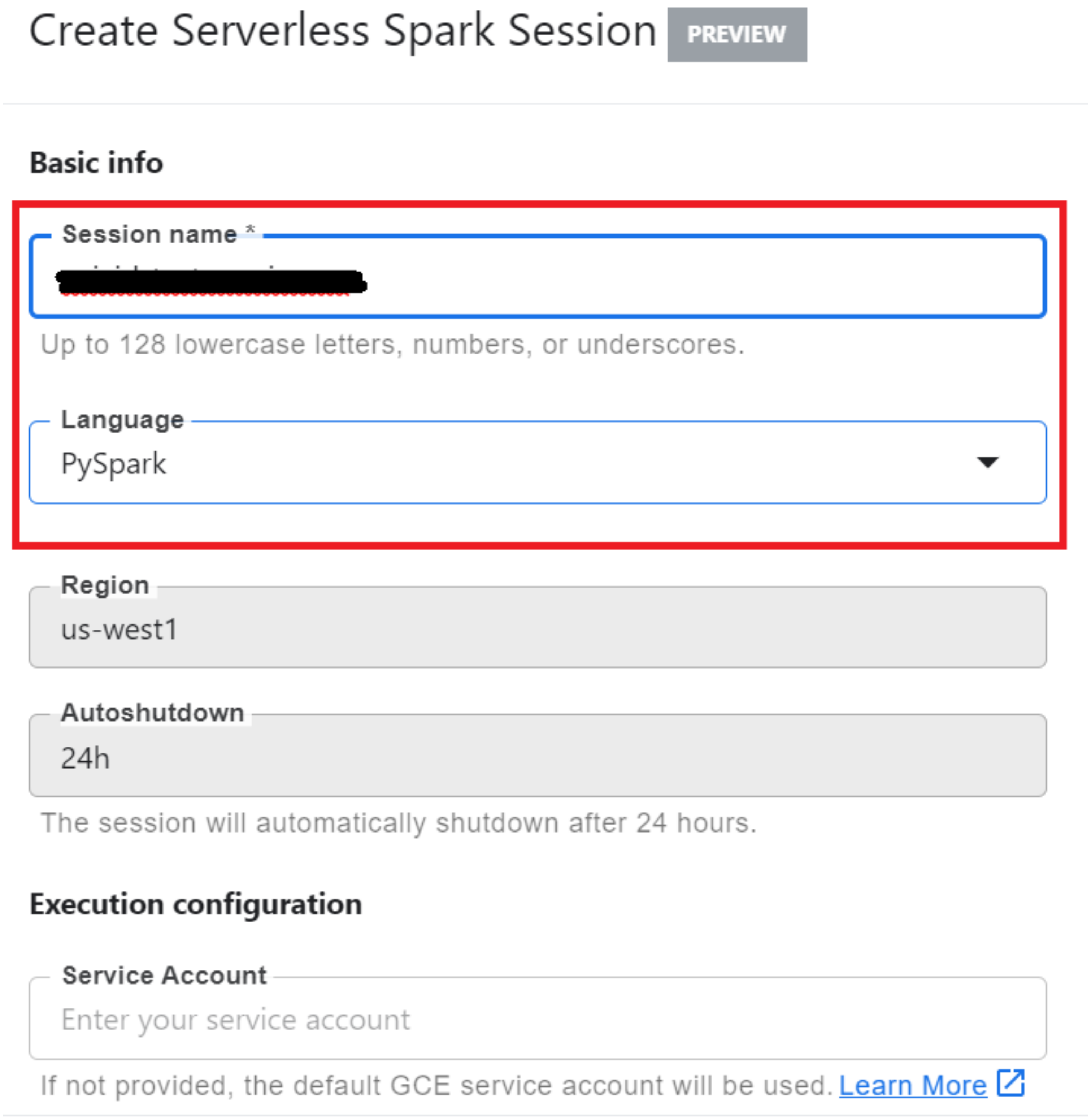
Follow the on screen instructions to create Session

3.2. Provide the details for the Session

Next, fill in the following values in the session creation window as shown in the images below:

- **Session Name** - A unique identifier for your session
- **Region** - The region name provided by the Admin team
- **Language** - Pyspark
- **Autoshutdown** - 24 hours
- **Service Account** - <UMSA_NAME>@<PROJECT_ID>.iam.gserviceaccount.com
- **Network Configuration** - Select the network and subnetwork provided by the Admin team
- **History Server Cluster** - projects/<PROJECT_ID>/regions/<REGION_NAME>/clusters/<HISTORY_SERVER_NAME>
- **Properties** - spark.jars=gs://spark-lib/bigquery/spark-bigquery-with-dependencies_2.12-0.22.2.jar

- Click the **SUBMIT** button to create the session.



Create Serverless Spark Session PREVIEW

Basic info

Session name * [redacted]

Up to 128 lowercase letters, numbers, or underscores.

Language: PySpark

Region: us-west1

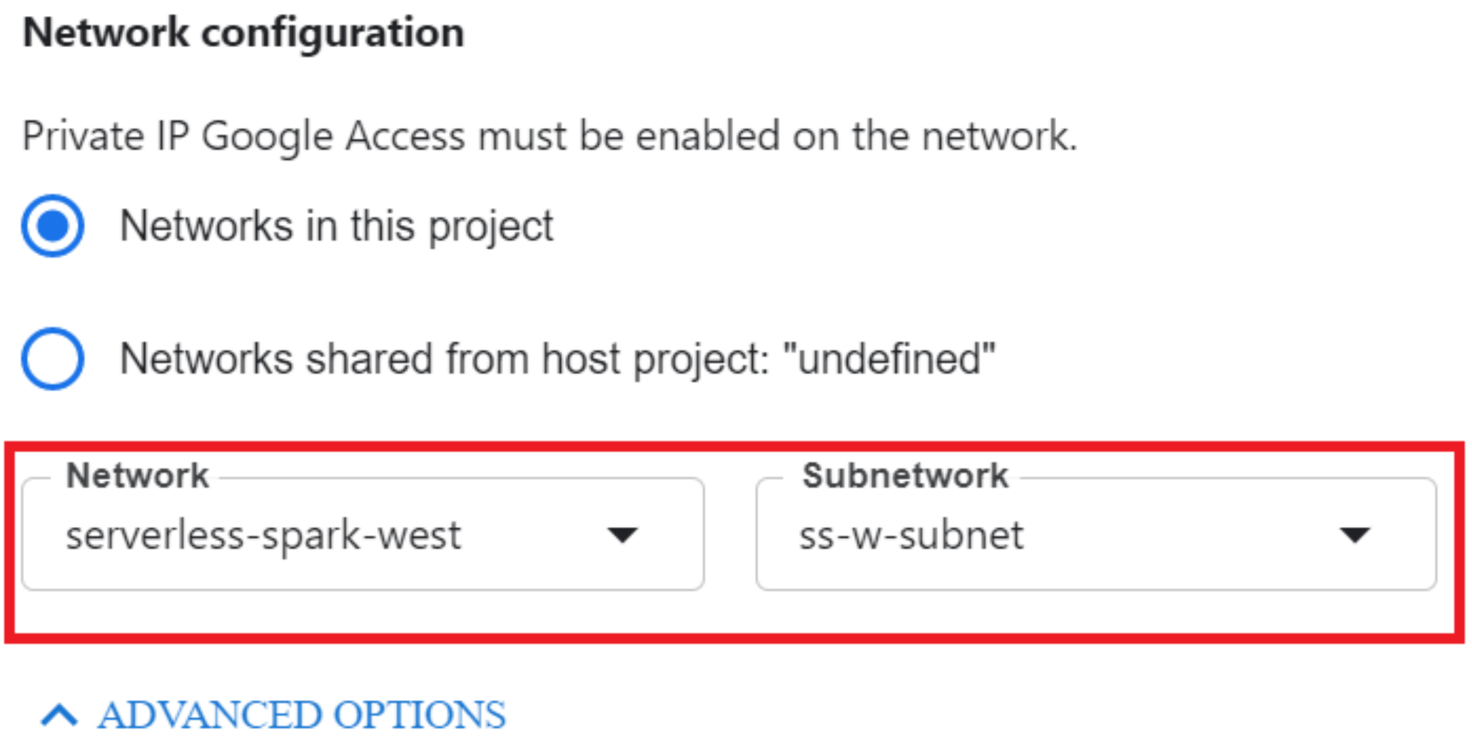
Autoshutdown: 24h

The session will automatically shutdown after 24 hours.

Execution configuration

Service Account: Enter your service account

If not provided, the default GCE service account will be used. Learn More



Network configuration

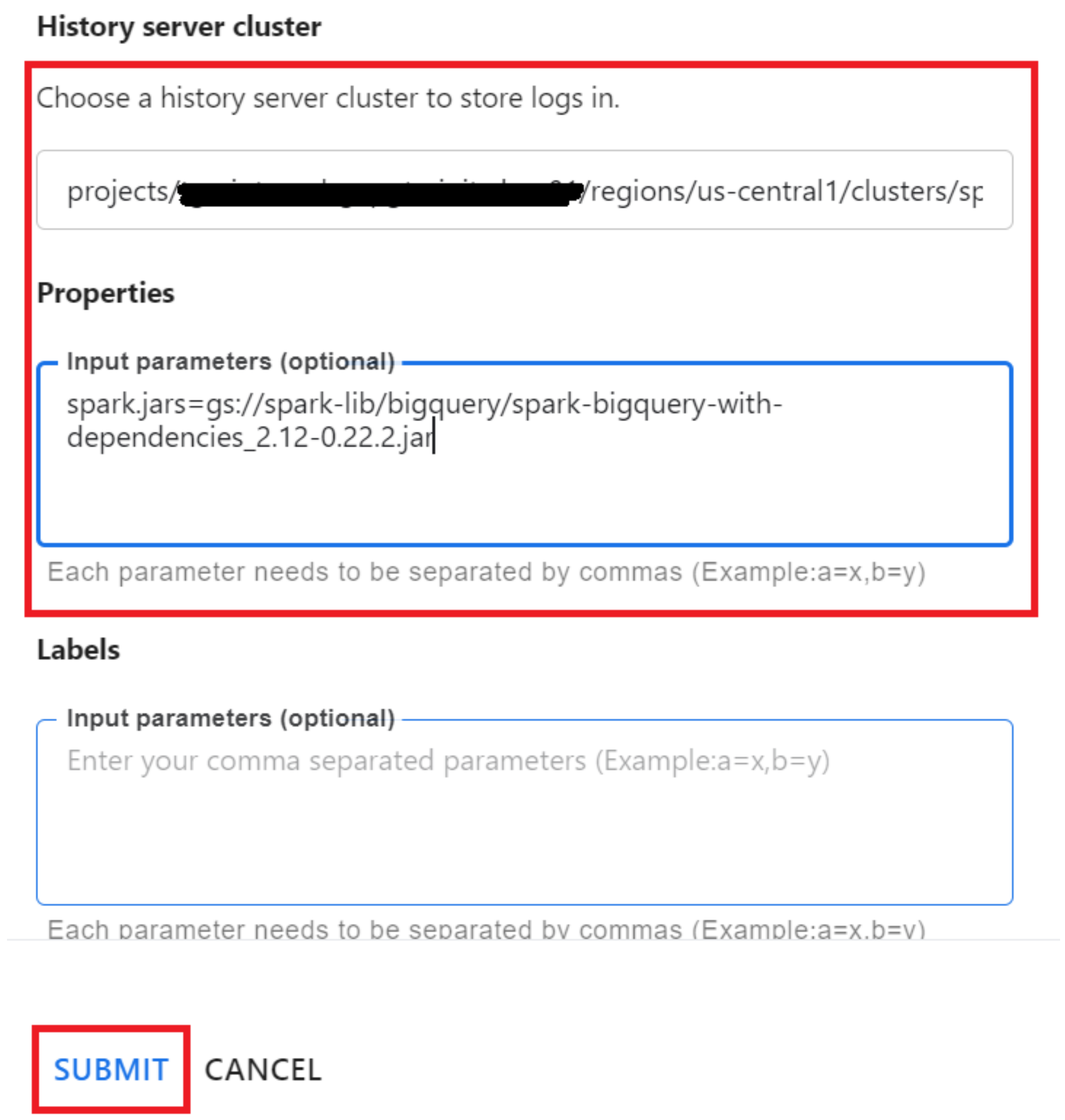
Private IP Google Access must be enabled on the network.

[✓ Networks in this project]

Networks shared from host project: "undefined"

Network: serverless-spark-west

Subnetwork: ss-w-subnet



History server cluster

Choose a history server cluster to store logs in.

projects/[redacted]/regions/us-central1/clusters/sp

Properties

Input parameters (optional)

```
spark.jars=gs://spark-lib/bigquery/spark-bigquery-with-dependencies_2.12-0.22.2.jar
```

Each parameter needs to be separated by commas (Example:a=x,b=y)

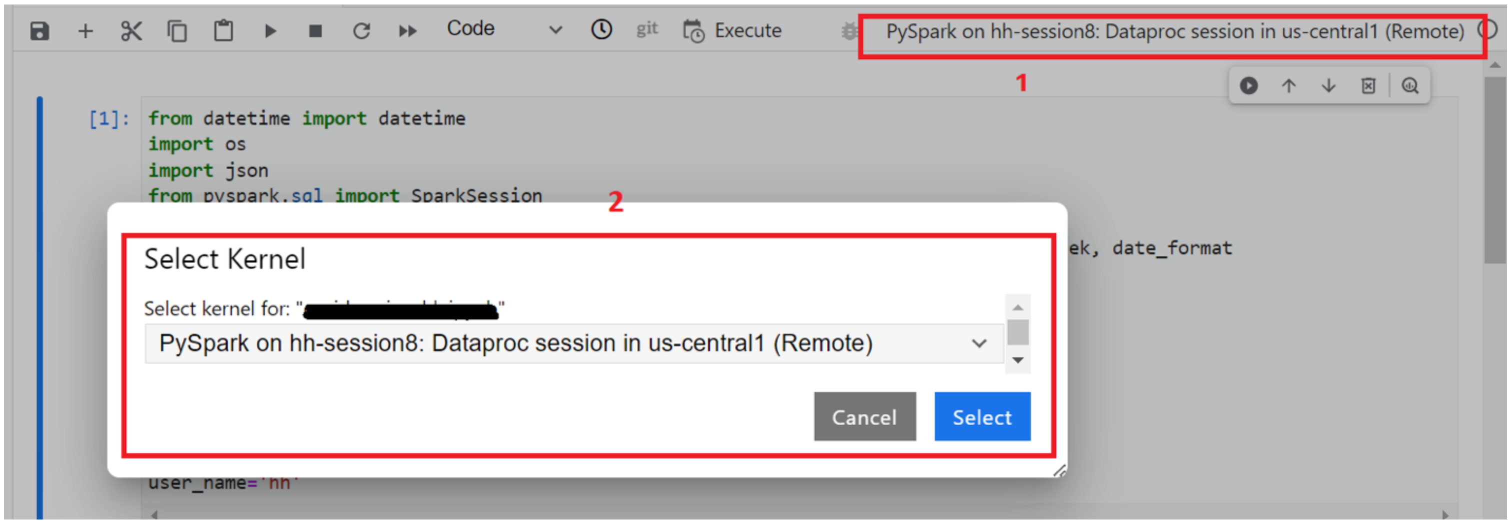
Labels

Input parameters (optional)

Enter your comma separated parameters (Example:a=x,b=y)

Each parameter needs to be separated by commas (Example:a=x,b=y)

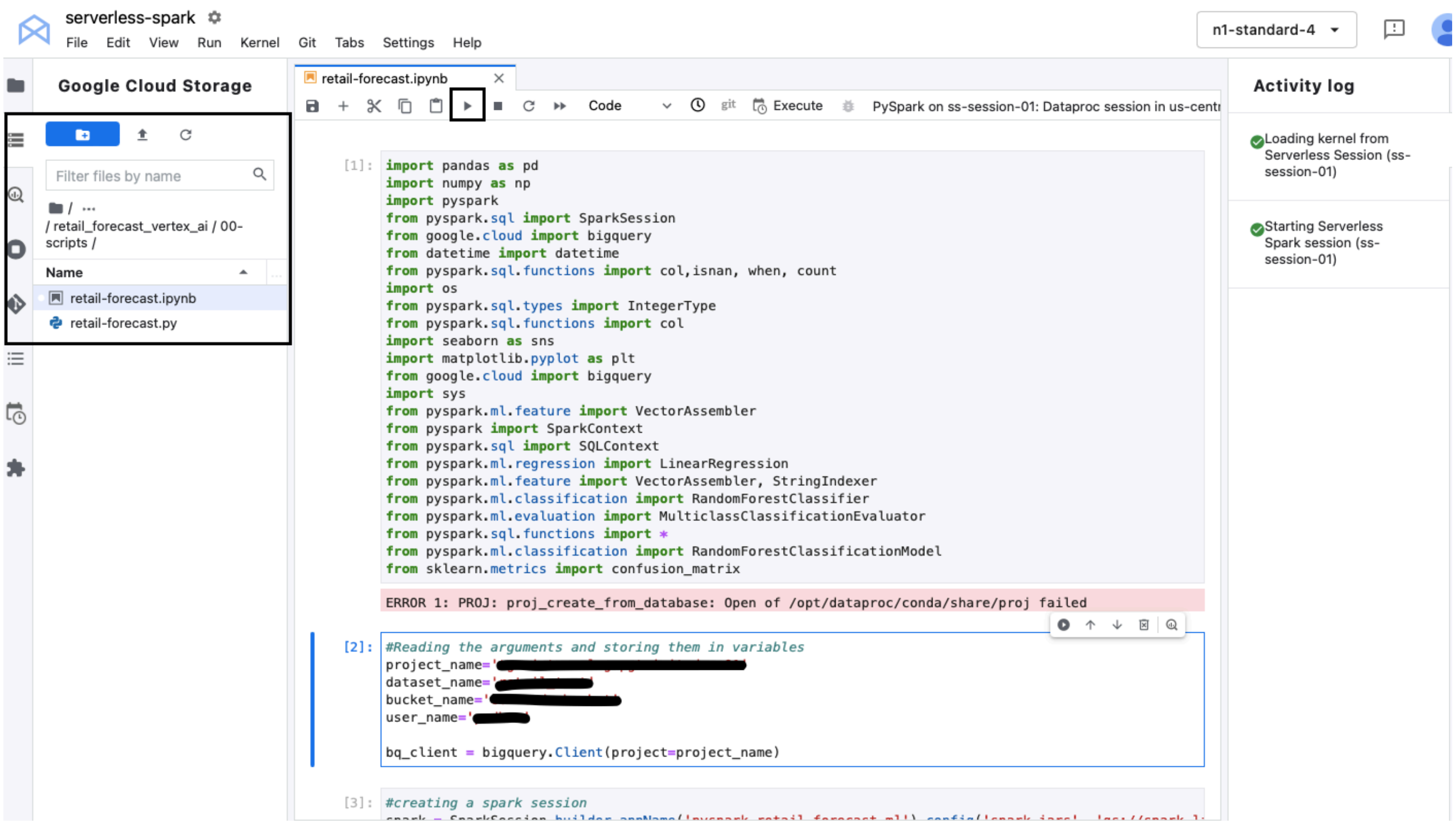
- Once the Session is created select 'No Kernel' from the kernel dropdown list and then delete the notebook


Code
git
Execute
PySpark on hh-session8: Dataproc session in us-central1 (Remote)
1
2
Select Kernel
Select kernel for: "[redacted]"
PySpark on hh-session8: Dataproc session in us-central1 (Remote)
Cancel
Select

```
[1]: from datetime import datetime
     import os
     import json
     from pyspark.sql import SparkSession
                                          ek, date_format

     user_name='hh'
```

- Next, using the browser option from JupyterLab, navigate to the Notebook file located at: <bucket_name> > 'retail_forecast_vertex_ai' > 00-scripts > retail-forecast.ipynb
- From the kernel dropdown list, select the kernel for the session created in section 3.2
- Pass the values to the variables project_name, dataset_name, bucket_name as provided by the Admin and replace user_name by your username
- Next, hit the **Execute** button as shown below to run the code in the notebook.



Loading kernel from Serverless Session (ss-session-01)

Starting Serverless Spark session (ss-session-01)

```
[1]: import pandas as pd
     import numpy as np
     import pyspark
     from pyspark.sql import SparkSession
     from google.cloud import bigquery
     from datetime import datetime
     from pyspark.sql.functions import col,isnan, when, count
     import os
     from pyspark.sql.types import IntegerType
     from pyspark.sql.functions import col
     import seaborn as sns
     import matplotlib.pyplot as plt
     from google.cloud import bigquery
     import sys
     from pyspark.ml.feature import VectorAssembler
     from pyspark import SparkContext
     from pyspark.sql import SQLContext
     from pyspark.ml.regression import LinearRegression
     from pyspark.ml.feature import VectorAssembler, StringIndexer
     from pyspark.ml.classification import RandomForestClassifier
     from pyspark.ml.evaluation import MulticlassClassificationEvaluator
     from pyspark.sql.functions import *
     from pyspark.ml.classification import RandomForestClassificationModel
     from sklearn.metrics import confusion_matrix

     ERROR 1: PROJ: proj_create_from_database: Open of /opt/dataproc/conda/share/proj failed

[2]: #Reading the arguments and storing them in variables
     project_name='[redacted]
     dataset_name='[redacted]
     bucket_name='[redacted]
     user_name='[redacted]

     bq_client = bigquery.Client(project=project_name)

[3]: #creating a spark session
```

3.3. Check the output table in BQ

Navigate to BigQuery Console, and check the **retail_forecast** dataset.
Once the data preparation batch is completed, four new tables '<your_name_here>_train_data', '<your_name_here>_test_data', '<your_name_here>_predictions_data' and '<your_name_here>_eval_output' will be created:

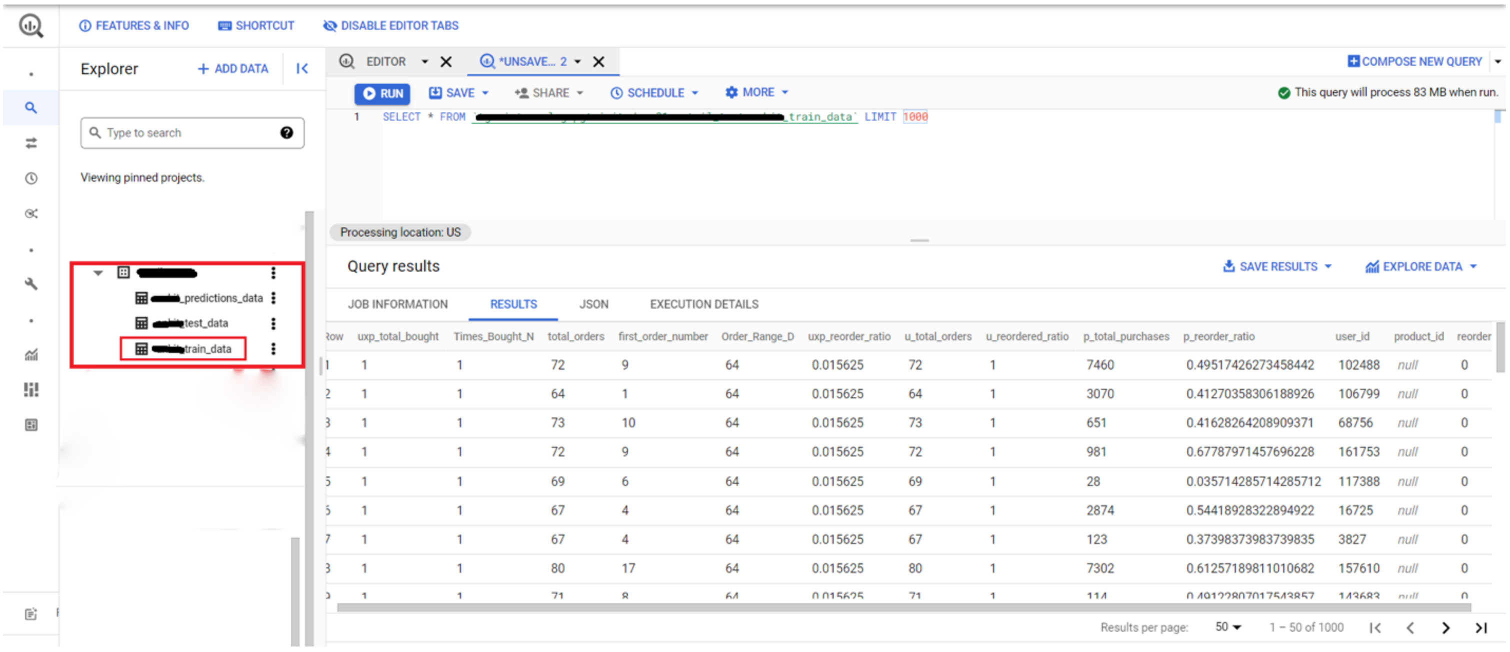
To view the data in these tables -

- Select the table from BigQuery Explorer by navigating 'project_id' **>** 'dataset' **>** 'table_name'
- Click on the **Preview** button to see the data in the table

Note: If the **Preview** button is not visible, run the below queries to view the data. However, these queries will be charged for the full table scan.

```
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_train_data` LIMIT 1000
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_test_data` LIMIT 1000
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_predictions_data` LIMIT 1000
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_eval_output` LIMIT 1000
```

Note: Edit all occurrences of <project_name> and <dataset_name> to match the values of the variables PROJECT_ID, and BQ_DATASET_NAME respectively



```
SELECT * FROM `[redacted]train_data` LIMIT 1000
```

Query results

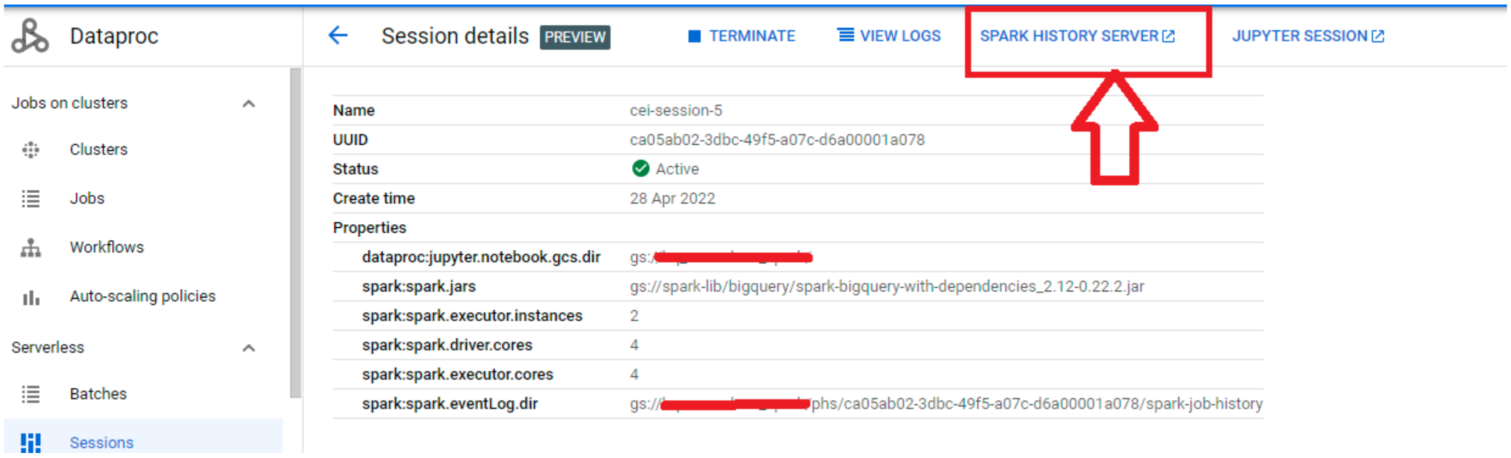
Row	uxp_total_bought	Times_Bought_N	total_orders	first_order_number	Order_Range_D	uxp_reorder_ratio	u_total_orders	u_reordered_ratio	p_total_purchases	p_reorder_ratio	user_id	product_id	reorder
1	1	1	72	9	64	0.015625	72	1	7460	0.49517426273458442	102488	null	0
2	1	1	64	1	64	0.015625	64	1	3070	0.41270358306188926	106799	null	0
3	1	1	73	10	64	0.015625	73	1	651	0.41628264208909371	68756	null	0
4	1	1	72	9	64	0.015625	72	1	981	0.67787971457696228	161753	null	0
5	1	1	69	6	64	0.015625	69	1	28	0.035714285714285712	117388	null	0
6	1	1	67	4	64	0.015625	67	1	2874	0.54418928322894922	16725	null	0
7	1	1	67	4	64	0.015625	67	1	123	0.37398373983739835	3827	null	0
8	1	1	80	17	64	0.015625	80	1	7302	0.61257189811010682	157610	null	0
9	1	1	71	8	64	0.015625	71	1	114	0.49122807017543857	143683	null	0

4. Logging

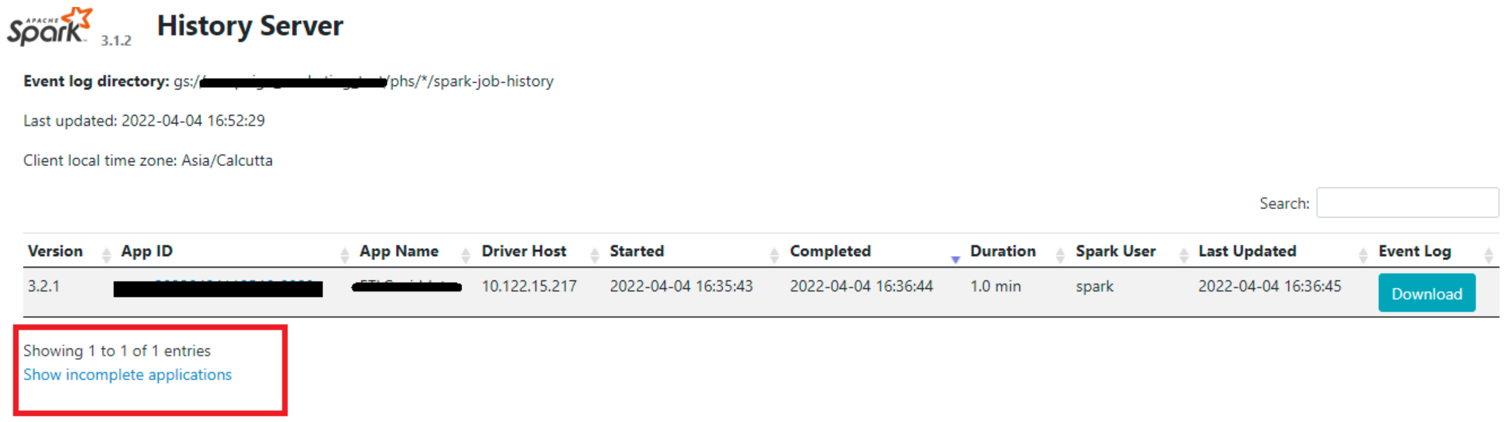
4.1 Persistent History Server logs

To view the Persistent History server logs, click the 'View History Server' button on the Sessions monitoring page and the logs will be shown as below:

As the session is still in active state , we will be able to find the logs in show incomplete applications.



Name	cei-session-5
UUID	ca05ab02-3dbc-49f5-a07c-d6a00001a078
Status	Active
Create time	28 Apr 2022
Properties	
dataproc:jupyter.notebook.gcs.dir	gs:/[redacted]
spark:spark.jars	gs://spark-lib/bigquery/spark-bigquery-with-dependencies_2.12-0.22.2.jar
spark:spark.executor.instances	2
spark:spark.driver.cores	4
spark:spark.executor.cores	4
spark:spark.eventLog.dir	gs://[redacted]phs/ca05ab02-3dbc-49f5-a07c-d6a00001a078/spark-job-history



History Server

Event log directory: gs:/[redacted]/phs/*/spark-job-history

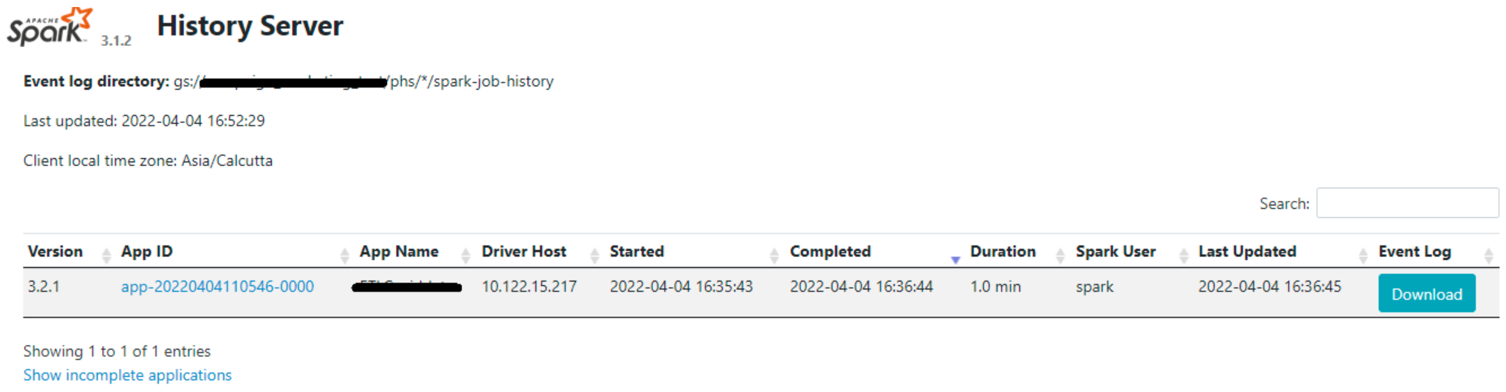
Last updated: 2022-04-04 16:52:29

Client local time zone: Asia/Calcutta

Search:

Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.2.1	[redacted]	[redacted]	10.122.15.217	2022-04-04 16:35:43	2022-04-04 16:36:44	1.0 min	spark	2022-04-04 16:36:45	Download

Showing 1 to 1 of 1 entries
Show incomplete applications



History Server

Event log directory: gs:/[redacted]/phs/*/spark-job-history

Last updated: 2022-04-04 16:52:29

Client local time zone: Asia/Calcutta

Search:

Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.2.1	app-20220404110546-0000	[redacted]	10.122.15.217	2022-04-04 16:35:43	2022-04-04 16:36:44	1.0 min	spark	2022-04-04 16:36:45	Download

Showing 1 to 1 of 1 entries
Show incomplete applications