DATA & AI BOOT-KON EVENT

FraudFix Use Case

Setup your environment: Notebooks & IAM

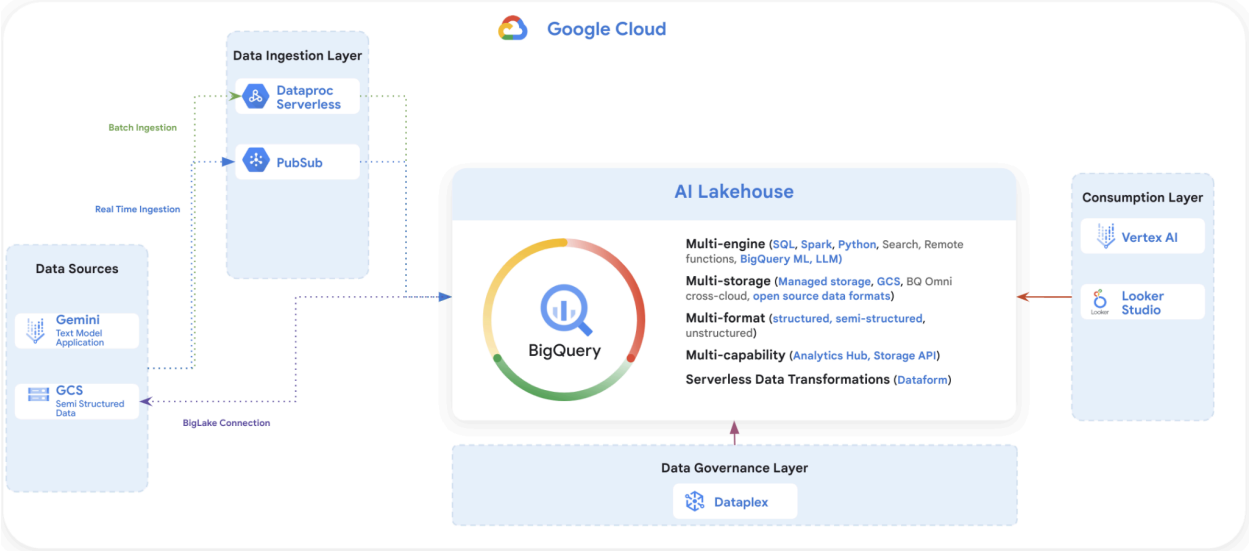
Lab Duration: 45 Minutes

CAUTION:
This lab is for educational purposes only and should be used with caution in production environments. Google Cloud Platform (GCP) products are changing frequently, and screenshots and instructions might become inaccurate over time. Always refer to the latest GCP documentation for the most up-to-date information.

Authors: Wissem Khlifi
Date : April 1, 2024

Github repository : *https://github.com/dace-de/bootkon-h2-2024*

Architecture Diagram



Goal of the Lab

- Enable Google cloud services APIs
- Ensure your GCP user and service account have access to the required resources.
- Create GCP default network
- Create Vertex AI notebook for the ML labs.

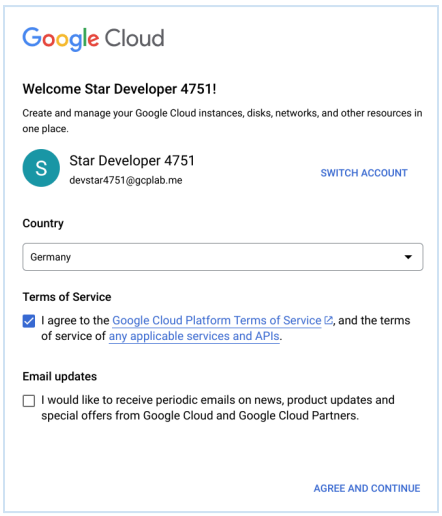
Choice of GCP Product and Service Location

You are free to choose any GCP region location for all labs. Ensure all your resources are created in the chosen location to avoid connectivity issues and minimize latency and cost. If you don't have a preferred GCP location, use ***us-central1*** for simplicity.

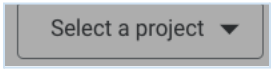
Setup your environment

Please note: Commands you need to execute are marked in **blue**. It's recommended to run them one at a time to prevent accidentally skipping steps.

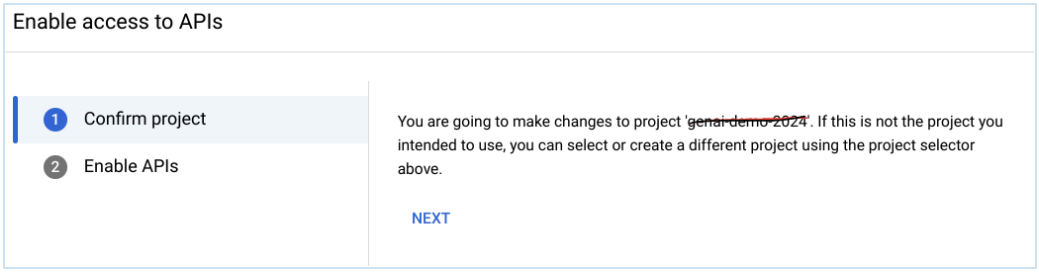
1. Open Web Browser in Incognito Mode.
2. Login to your GCP console. Use the provided credentials.
    a. Log in with your gcp_username and gcp_password.
    b. Open https://console.cloud.google.com/ in incognito mode
    c. Login with : gcp_username & gcp_password
    d. Accept the Terms of Service

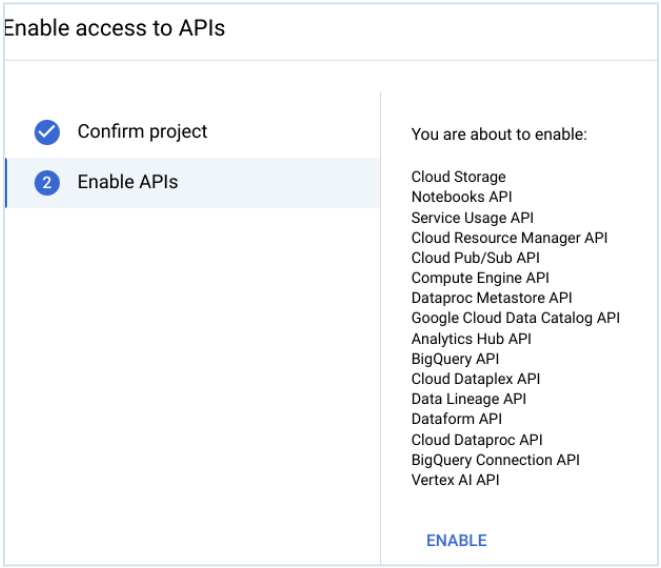


    e. Choose your project ID: it should be gcp_project_id you received by Email. Click on select a project and select the project ID

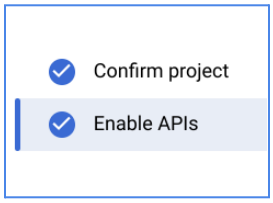


    f. initially you have been granted the project editor and IAM project admin roles.
3. Enable the necessary services APIs
    - Ensure all necessary APIs are *enabled* , Follow the link below in ***incognito mode*** and click "**Enable**".
      https://console.cloud.google.com/apis/enableflow?apiid=storage-component.googleapis.com,notebooks.googleapis.com,serviceusage.googleapis.com,cloudresourcemanager.googleapis.com,pubsub.googleapis.com,compute.googleapis.com,metastore.googleapis.com,datacatalog.googleapis.com,analyticshub.googleapis.com,bigquery.googleapis.com,dataplex.googleapis.com,datalineage.googleapis.com,dataform.googleapis.com,dataproc.googleapis.com,bigqueryconnection.googleapis.com,aiplatform.googleapis.com&_ga=2.132962701.243207769.1688884437-279425947.1688884437





- Wait until all APIs are enabled. (It should look like the following screenshot)



4. From the top corner of the GCP console, activate the Cloud Shell.

> Click "**Continue**" if prompted.

5. Ensure your project ID is set correctly:
   - Replace gcp_project_id by the GCP project ID.
   - Click on "**Authorize**" in the message box.

Linux command line : Set Project ID
gcloud config set project gcp_project_id

6. Install git & git lFS: Run the following commands to install Git and Git LFS:

Linux command line : Install git & git lFS

```
sudo apt-get install git
sudo apt-get install git-lfs
git lfs install
```

7. Clone the Repository Locally to your cloud shell. Run the following commands to clone the repository:

Linux command line :Clone the repository locally to your cloud shell

```
git clone https://github.com/dace-de/bootkon-h2-2024.git
cd bootkon-h2-2024/
git lfs pull
```

Note: If cloning fails, use these commands:

Linux command line : If for any reason the cloning from the repository does not work, run the following commands. Otherwise skip this step.

```
cd $HOME
mkdir  bootkon-h2-2024
cd bootkon-h2-2024/
gsutil cp -r gs://bootkon-labs/* .
```

[FOR YOUR INFORMATION ONLY] Description of the files / directories :

Parent Directory	***File / Directory Name***	***Description***
bootkon-h2-2024 (root)	*data-ingestion*	• *Contains datafiles to be ingested into Bigquery (CSV , Parquet)* • *Contains data Ingestion code into BigQuery (python)*
data-ingestion	*csv/ulb_fraud_detection*	• *Contains datafiles to be ingested into Bigquery (CSV format)*
data-ingestion	*parquet/ulb_fraud_detection*	• *Contains datafiles to be ingested into Bigquery (Parquet format)*
data-ingestion	*jar/spark-3.3-bigquery-0.37.0.jar*	*Spark BigQuery Jar file.*
data-ingestion	*src*	• *Contains data Ingestion code into BigQuery (python)*
src	*import_parquet_to_bigquery.py*	• *Batch ingestion code into BigQuery (python)*

src	*import_csv_to_bigquery_1.py*	• *Near Real time ingestion method1 : data Ingestion code into BigQuery (python)*
src	*import_csv_to_bigquery_2.py*	• *Near Real time ingestion method 2 : data Ingestion code into BigQuery (python)*
src	*my_avro_fraud_detection_schema.json*	• *Contains the Pubsub topic schema definition*
bootkon-h2-2024 (root)	*data-prediction*	• *Contains ML prediction datafiles that we will use as BigQuery data source for data quality checks in Dataplex and data sharing.*
bootkon-h2-2024 (root)	*dataform*	• *Contains dataform transformation code*
dataform	*definitions*	• *Contains SQLX definitions of tables, materialized views to run in BigQuery.*
definitions	*models*	• *Contains SQLX definitions of datasets, BigLake , Vertex AI connections to external sources of BigQuery.*
models	create_dataset.sqlx	• *SQLX that creates a BigQuery Dataset that we will use for transformations. (Curated data)*
models	*llm_model_connection.sqlx*	• *SQLX that creates a BigQuery external connection to Vertex AI , BigLake.*
definitions	*mview_ulb_fraud_detection.sqlx*	• *SQLX creates a materialized view that aggregates some data to be used for sentiment analysis.*
definitions	*sentiment_inference.sqlx*	• *SQLX creates a BigQuery table that contains the sentiment analysis results*
definitions	*ulb_fraud_detection.sqlx*	• *Config of BigQuery source table*
dataform	*dataform.json*	• *Config of BigQuery dataset, default location*
dataform	*package.json*	• *Default config of required dataform dependencies*
prepare-enviroment	*assign-roles.sh*	• *Grant the necessary IAM roles to your user and compute service accounts.*
metadata-mapping	*pca*	• *Metadata mapping of PCA metadata with actual real meaning. The file in parquet format.*

8. Run the following commands to verify checksums and remove checksum files:

(you can ignore the./checksums.sha256: FAILED error).

Linux command line : Check Datafiles check sum & delete the checksums files
*you can ignore the.**/checksums.sha256: FAILED** error*

```
cd data-prediction
sha256sum -c checksums.sha256
rm -f checksums.sha256
cd ..
cd data-ingestion/csv/ulb_fraud_detection/
sha256sum -c checksums.sha256
rm -f checksums.sha256
cd ../..
cd  parquet/ulb_fraud_detection/
sha256sum -c checksums.sha256
rm -f checksums.sha256
cd ../../..
cd metadata-mapping/
sha256sum -c checksums.sha256
rm -f checksums.sha256
```

Example of output should like this :

```
admin_@cloudshell:~/dace-de/bootkon-h2-2024/data-prediction (genai-demo-2024)$ sha256sum -c checksums.sha256
./part-00002-f05de483-6d3d-4041-9854-f233525dc231-c000.snappy.parquet: OK
./part-00001-f05de483-6d3d-4041-9854-f233525dc231-c000.snappy.parquet: OK
./checksums.sha256: FAILED
./part-00000-f05de483-6d3d-4041-9854-f233525dc231-c000.snappy.parquet: OK
sha256sum: WARNING: 1 computed checksum did NOT match
admin_@cloudshell:~/dace-de/bootkon-h2-2024/data-prediction (genai-demo-2024)$
```

9. Authenticate with GCP. Run the following command to login

Linux command line : gcloud auth login

Run: gcloud auth login
*Do you want to continue (Y/n)? => **Type:** Y*
*Follow the https link , **click on it***
Click on your user name
Click on Continue
*Click on **Allow***
*Click on **Copy***
*Return to cloud shell and **past the code***

```
Once finished, enter the verification code provided in your browser: 4/0AdLIrYcUzqHdcILqkPMt9cUcPvha9w9KUKD8Ziex7ZBvsJNy6GTEmTgGNzCaNfr
```

10. Setup IAM Permissions

Run the script to grant IAM roles. Replace PROJECT_ID and GCP_USERNAME *(Email format)* with your actual values.
This script grants your IAM user and the compute engine service account the necessary permissions.

Shell Script : Usage:

```
PROJECT_ID=your-project-id
GCP_USERNAME=gcp_username
cd $HOME/bootkon-h2-2024/prepare-environment
chmod 700 assign_roles.sh
./assign_roles.sh $PROJECT_ID $GCP_USERNAME
```

Replace :

- *"your-project-id" is your actual GCP project ID.*
- *gcp_username is your GCP user name (Email format)*

11. Create default VPC Network and enable private access, required for Dataproc.

To create a default VPC network in Google Cloud Platform (GCP) with automatic subnet creation and private Google access enabled using the gcloud command-line tool, you can use the following commands. *Make sure to replace "your-project-id" with your actual GCP project ID before running these commands.*

Linux command line : Create default VPC network and enable private access, required for Dataproc
Recommended: Execute one command at once.

```
# Set the GCP project ID
PROJECT_ID="your-project-id"
# Run authentication
gcloud auth login
# Create a default network with automatic subnet creation
export REGION="us-central1" # replace the region value with your selected region
SUBNET="default"

# Create a default network with automatic subnet creation
gcloud compute networks create $SUBNET --project=$PROJECT_ID --subnet-mode=auto
--bgp-routing-mode="regional"

# If you get ERROR: default already exists , you can ignore the error and run the following commands.
# Enable Private Google Access required by Dataproc Serverless
gcloud compute networks subnets update $SUBNET --region=$REGION --enable-private-ip-google-access

# Create a firewall rule for internal network communication
gcloud compute firewall-rules create "default-allow-all-internal" \
 --network="default" \
 --project=$PROJECT_ID \
 --direction=INGRESS \
 --priority=65534 \
 --source-ranges="10.128.0.0/9" \
 --allow=tcp:0-65535,udp:0-65535,icmp
```

Here's what each command does:

Network Creation:

- *gcloud compute networks create "default": This command creates a new VPC network named "default" in your project.*
  `--subnet-mode=auto`: Specifies that subnets are created automatically in each region.
  --bgp-routing-mode="regional": Sets the BGP routing mode for the network. You can also choose "global" if needed.
- *Enable Private Google Access:*

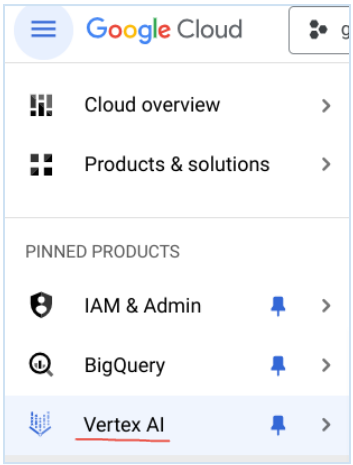
The second command block:

1. *first, lists all the subnets in the "default" network, filtering to include only those in the default network.*
2. *It then iterates over these subnets and enables Private Google Access on each one.*

Private Google Access allows VM instances in the subnet to communicate with Google services without an external IP address.

12. Create Vertex AI User Managed Notebook

a. Go to Vertex AI in the GCP console.



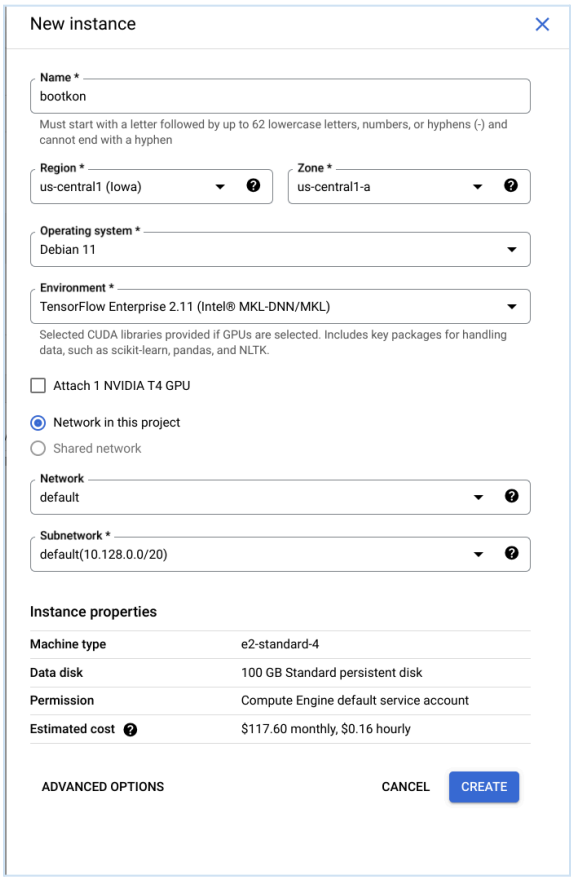
b. Click on the Workbench section.

c. Select "User managed notebooks"

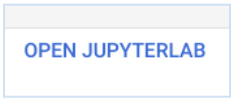
d. Create new instance

e. Name the notebook "***bootkon***" and leave the default network and environment. Leave the cheapest machine type; e2-standard-4 selected; 4 vCPUs and 16GB of RAM are more than enough to perform the

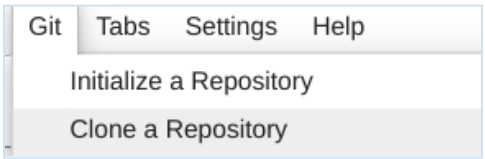
ML labs using jupyter notebooks. Do not attach a GPU. Normally it takes around 10 minutes to get the instance created.



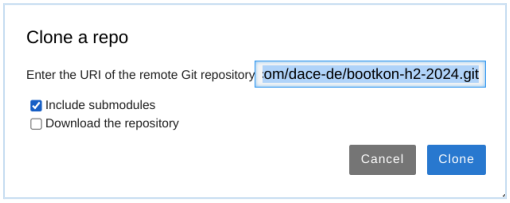
f. Open the Jupyter Lab;



g. From the Jupyter Lab top menu, click on Git -> Clone a Repository



h. Enter https://github.com/dace-de/bootkon-h2-2024.git and click on **clone**



11. Now let's create a Google Cloud storage bucket where we store the datafiles.
From your cloud shell, run the following gcloud command.
Make sure to replace "your-project-id" with your actual GCP project ID before running these commands.

Linux command line : Create a Google Cloud storage bucket

```
# Set the GCP project ID
PROJECT_ID=your-project-id

BUCKET_NAME="${PROJECT_ID}-bucket"
REGION=us-central1

# Create the bucket in the us-central1 region
gcloud storage buckets create "gs://${BUCKET_NAME}" \
  --project="${PROJECT_ID}" \
  --location="${REGION}" \
  --uniform-bucket-level-access
```

12. From your cloud shell, copy the csv and parquet files into a new data ingestion GCS location

Linux command line : Copy the csv and parquet files into a new data ingestion GCS location

```
gcloud storage ls
gsutil ls gs://${BUCKET_NAME}/

cd $HOME
cd bootkon-h2-2024/data-ingestion/
# List the files & directories
ls -rtl
gsutil cp -R  $HOME/bootkon-h2-2024/data-ingestion/csv/* gs://${BUCKET_NAME}/data-ingestion/csv/
gsutil cp -R  $HOME/bootkon-h2-2024/data-ingestion/jar/* gs://${BUCKET_NAME}/data-ingestion/jar/
gsutil cp -R  $HOME/bootkon-h2-2024/data-ingestion/src/* gs://${BUCKET_NAME}/data-ingestion/src/
gsutil cp -R  $HOME/bootkon-h2-2024/data-ingestion/parquet/* gs://${BUCKET_NAME}/data-ingestion/parquet/

gsutil ls -R gs://${BUCKET_NAME}/data-ingestion


cd ../data-prediction/
gsutil cp -R $HOME/bootkon-h2-2024/data-prediction/* gs://${BUCKET_NAME}/data-prediction/
gsutil ls -R gs://${BUCKET_NAME}/data-prediction

cd ../metadata-mapping/
gsutil cp  $HOME/bootkon-h2-2024/metadata-mapping/pca  gs://${BUCKET_NAME}/metadata-mapping/pca
gsutil ls  gs://${BUCKET_NAME}/metadata-mapping/pca
```

The result of the ls outputs should look similar to this ;

```
admin_@cloudshell:~ (bootkon-2024)$ gsutil ls -R gs://bootkon-2024-bucket/data-ingestion
gs://bootkon-2024-bucket/data-ingestion/:

gs://bootkon-2024-bucket/data-ingestion/csv/:

gs://bootkon-2024-bucket/data-ingestion/csv/ulb_fraud_detection/:
gs://bootkon-2024-bucket/data-ingestion/csv/ulb_fraud_detection/part-00000-8c401cf1-dd3e-492f-a7d2-a36c4b9b42a2-c000.csv
gs://bootkon-2024-bucket/data-ingestion/csv/ulb_fraud_detection/part-00001-8c401cf1-dd3e-492f-a7d2-a36c4b9b42a2-c000.csv

gs://bootkon-2024-bucket/data-ingestion/parquet/:

gs://bootkon-2024-bucket/data-ingestion/parquet/ulb_fraud_detection/:
gs://bootkon-2024-bucket/data-ingestion/parquet/ulb_fraud_detection/part-00000-5c0e25e0-c667-45ed-b796-36170778a09f-c000.snappy.parquet
gs://bootkon-2024-bucket/data-ingestion/parquet/ulb_fraud_detection/part-00001-5c0e25e0-c667-45ed-b796-36170778a09f-c000.snappy.parquet

gs://bootkon-2024-bucket/data-ingestion/src/:
gs://bootkon-2024-bucket/data-ingestion/src/import_csv_to_bigquery_1.py
gs://bootkon-2024-bucket/data-ingestion/src/import_csv_to_bigquery_2.py
gs://bootkon-2024-bucket/data-ingestion/src/my_avro_fraud_detection_schema.json
```

```
admin_@cloudshell:~ (bootkon-2024)$ gsutil ls -R gs://bootkon-2024-bucket/data-prediction
gs://bootkon-2024-bucket/data-prediction/:
gs://bootkon-2024-bucket/data-prediction/
gs://bootkon-2024-bucket/data-prediction/part-00000-6de0b1ea-ae55-4653-90fa-a5cf28df0a3b-c000.snappy.parquet
gs://bootkon-2024-bucket/data-prediction/part-00001-6de0b1ea-ae55-4653-90fa-a5cf28df0a3b-c000.snappy.parquet
gs://bootkon-2024-bucket/data-prediction/part-00002-6de0b1ea-ae55-4653-90fa-a5cf28df0a3b-c000.snappy.parquet
admin_@cloudshell:~ (bootkon-2024)$
```

```
admin_@cloudshell:~/dace-de/bootkon-h2-2024/metadata-mapping (bootkon-2024)$ gsutil ls gs://bootkon-2024-bucket/metadata-mapping/pca
gs://bootkon-2024-bucket/metadata-mapping/pca
```

13. Upload JAR File to GCS directory *gs://${BUCKET_NAME}*/jar/

Linux command line : Direct upload the JAR file to GCS
gsutil cp gs://spark-lib/bigquery/spark-3.3-bigquery-0.37.0.jar gs://${BUCKET_NAME}/jar/* *gsutil ls gs://${BUCKET_NAME}/jar/

Note: The Spark 3.3 Jar version can be found here:
https://github.com/GoogleCloudDataproc/spark-bigquery-connector

Benefits: Using the spark-3.3-bigquery-0.37.0.jar with Dataproc provides access to BigQuery's Storage API, which offers benefits such as improved performance for data reads, reduced latency, better parallelism, and more efficient handling of large datasets compared to traditional JDBC-based methods.

🥳🥳Congratulations on completing Lab 1!
You can now move on to Lab 2 for further practice.🥳🥳

[Further Reading, Home Work]

- *Firewall Best Practices for Dataproc;*
  https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/network