Data Engineer Project - Dibimbing

CREDIT DATA PIPELINE

Muhammad Ariq Arfina

Content

01 Problems

02 Data Pipeline

03 Requirements & Steps

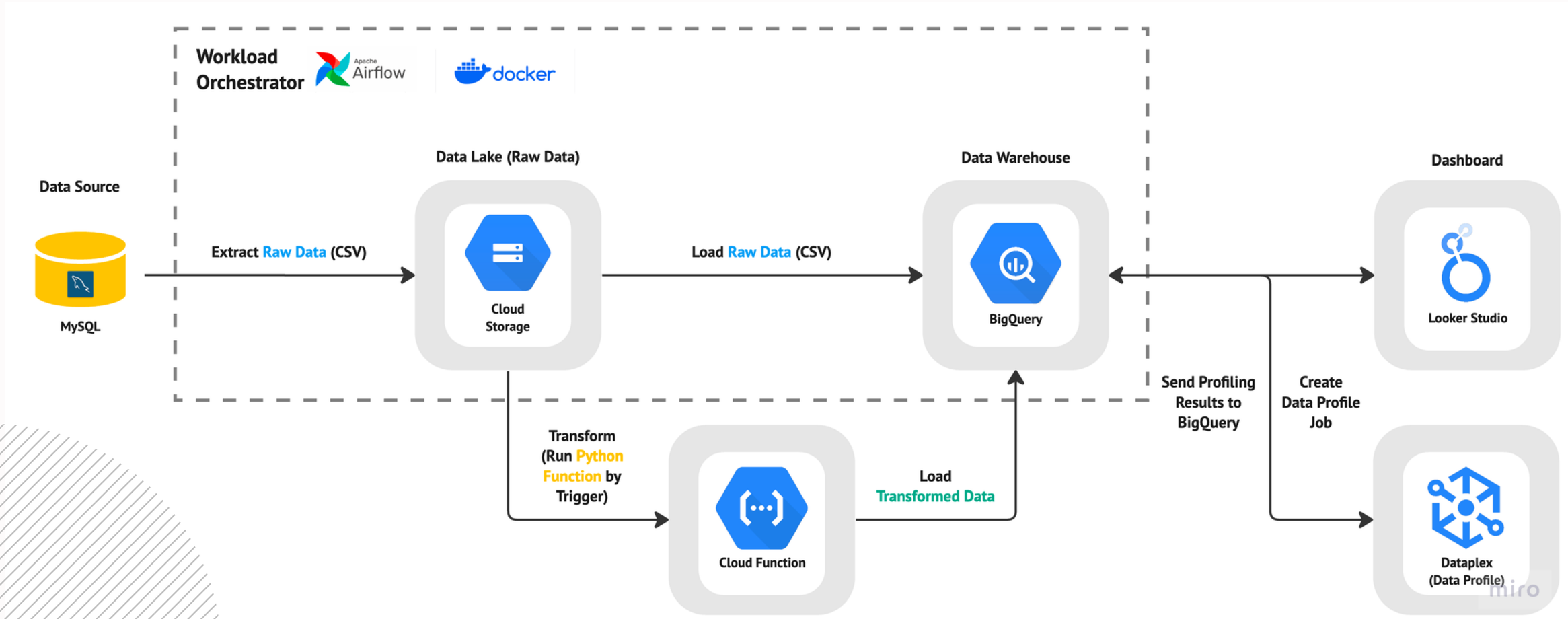
04 Output

Problems

Credivo, like many modern companies, deals with a lot of data from different sources. This data can help make smart decisions and find important insights, but it's hard to use effectively.

To fix this, Credivo knows it needs a better data pipeline for managing data. This pipeline will organize the data better, making it easier to use and trust. With this pipeline, Credivo wants to give its team the right information at the right time, so they can make better decisions and make the company better overall.

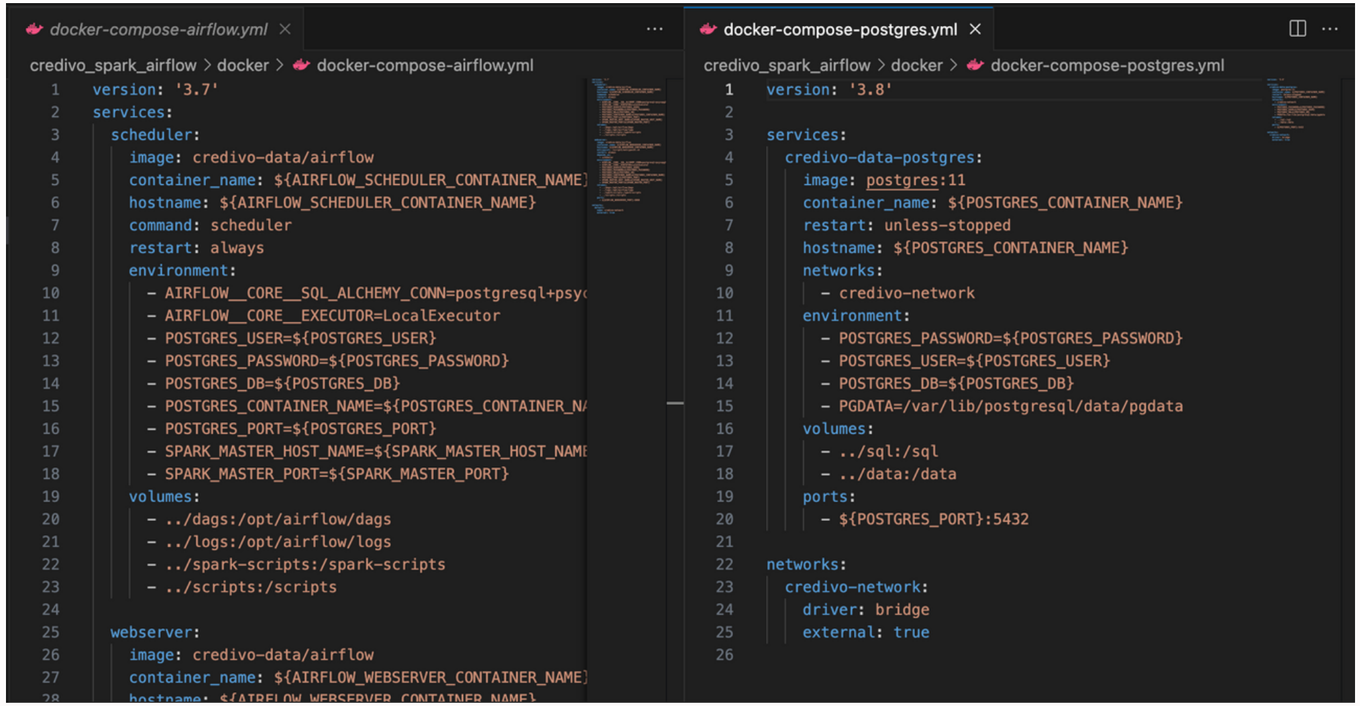
Data Pipeline



Requirement

1. Docker Compose Configuration:

- File:
  - **docker-compose-airflow.yml**
  - **docker-compose-postgres.yml**
  - **docker-compose-jupyter.yml**
- Services:
  - Airflow Scheduler
  - Airflow Webserver
  - Postgres
  - Jupyter



docker-compose-airflow.yml

credivo_spark_airflow > docker > docker-compose-airflow.yml

```yaml
version: '3.7'
services:
  scheduler:
    image: credivo-data/airflow
    container_name: ${AIRFLOW_SCHEDULER_CONTAINER_NAME}
    hostname: ${AIRFLOW_SCHEDULER_CONTAINER_NAME}
    command: scheduler
    restart: always
    environment:
      - AIRFLOW__CORE__SQL_ALCHEMY_CONN=postgresql+psyc
      - AIRFLOW__CORE__EXECUTOR=LocalExecutor
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
      - POSTGRES_CONTAINER_NAME=${POSTGRES_CONTAINER_NA
      - POSTGRES_PORT=${POSTGRES_PORT}
      - SPARK_MASTER_HOST_NAME=${SPARK_MASTER_HOST_NAME
      - SPARK_MASTER_PORT=${SPARK_MASTER_PORT}
    volumes:
      - ../dags:/opt/airflow/dags
      - ../logs:/opt/airflow/logs
      - ../spark-scripts:/spark-scripts
      - ../scripts:/scripts

  webserver:
    image: credivo-data/airflow
    container_name: ${AIRFLOW_WEBSERVER_CONTAINER_NAME}
    hostname: ${AIRFLOW_WEBSERVER_CONTAINER_NAME}
```

docker-compose-postgres.yml

credivo_spark_airflow > docker > docker-compose-postgres.yml

```yaml
version: '3.8'

services:
  credivo-data-postgres:
    image: postgres:11
    container_name: ${POSTGRES_CONTAINER_NAME}
    restart: unless-stopped
    hostname: ${POSTGRES_CONTAINER_NAME}
    networks:
      - credivo-network
    environment:
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_DB=${POSTGRES_DB}
      - PGDATA=/var/lib/postgresql/data/pgdata
    volumes:
      - ../sql:/sql
      - ../data:/data
    ports:
      - ${POSTGRES_PORT}:5432

networks:
  credivo-network:
    driver: bridge
    external: true
```

Requirement

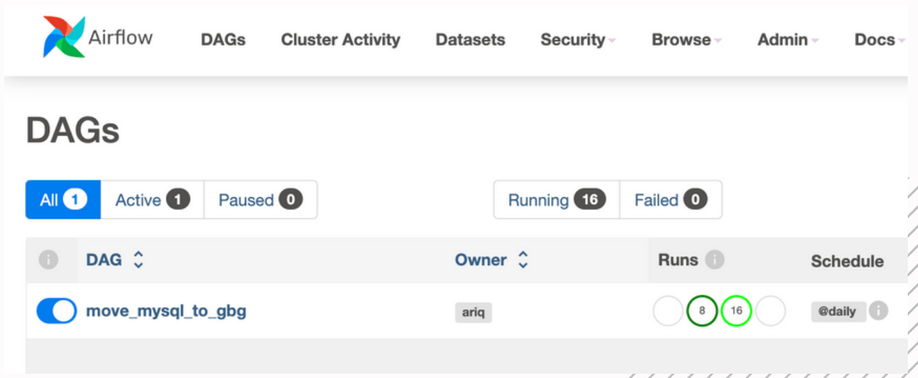
2. Airflow DAG (Directed Acyclic Graph):

- File:
  - **[run]credivo-dag-gcs.py**
- Steps:
  - Define an Airflow DAG named **move_mysql_to_gbg** for loading raw data from **MySQL to Google Cloud Storage (Raw Data)** and **BigQuery (Raw Data)**
  - Import Variable and define variable in DAG file such as **BUCKET_NAME**, **SQL_QUERY**, **FILENAME**, **GCP_CONN_ID**, **MYSQL_CONN_ID**, and **DESTINATION_DATASET_TABLE**
  - Use **MySQLToGCSOperator** to load raw data from **MySQL to Google Cloud Storage**
  - Use **GCSToBigQueryOperator** to load raw data from **Google Cloud Storage to BigQuery**
  - Schedules the DAG to run daily and set dependecies

[run]credivo-dag-gcs.py 4, M

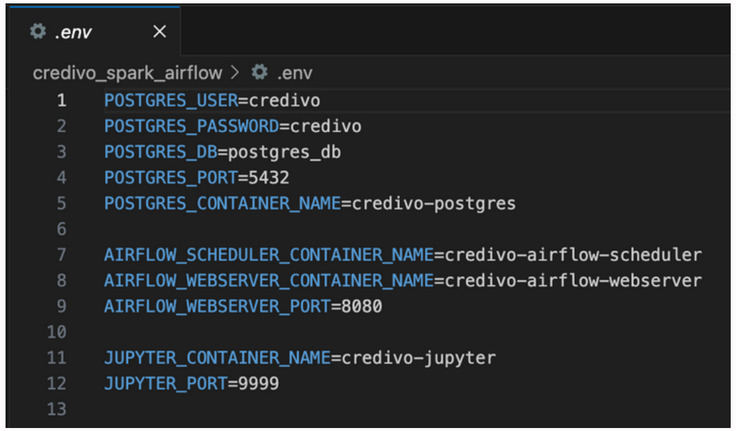
credivo_spark_airflow > dags > [run]credivo-dag-gcs.py > ...

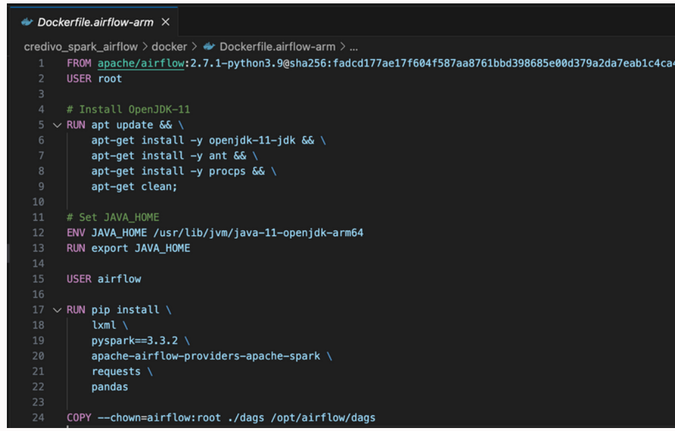
```python
import os
from airflow.providers.google.cloud.transfers.mysql_to_gcs import MySQLToGCSOperator
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSToBigQueryOperator
from airflow.models import Variable

#dag arguments
default_args = {
    'owner': 'ariq',
    'start_date': datetime(2024, 1, 1),
    'retries': 1,
    'retry_delay': timedelta(minutes=5)
}

#define dag
dag = DAG('move_mysql_to_gbg',
          schedule_interval= "@daily",
          default_args= default_args,
          description="Moving Data from MySQL to BigQuery",
          dagrun_timeout=timedelta(minutes=60)
          )

BUCKET_NAME = Variable.get("bucket_name")
SQL_QUERY = Variable.get("simple_query_all")
FILENAME = Variable.get("gcs_file")
GCP_CONN_ID = Variable.get("gcp_conn_id")
MYSQL_CONN_ID = Variable.get("mysql_conn_id")
DESTINATION_DATASET_TABLE = Variable.get("destination_dataset_table")
```

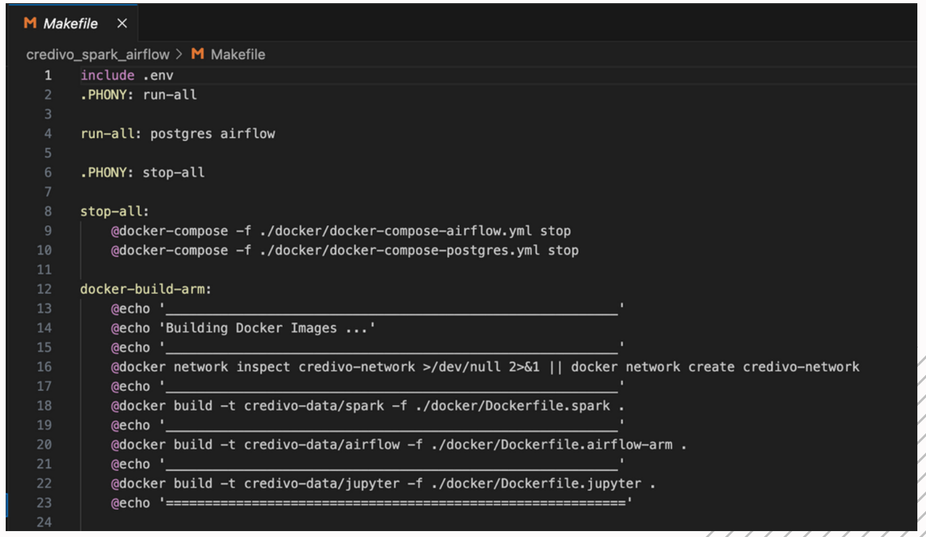


Requirement

3. Environment:
   - File:
     - **.env**
4. Dockerfile:
   - File:
     - **Dockerfile.airflow-arm**
     - **Dockerfile.jupyter**
   - **Descriptions:**
     - **Utilizes the Apache Airflow version 2.7.1**
     - **Installs dependencies**
5. Makefile:
   - File:
     - **Makefile**
   - Commands:
     - run-all: Starts all the containers
     - stop-all: Stops all the containers

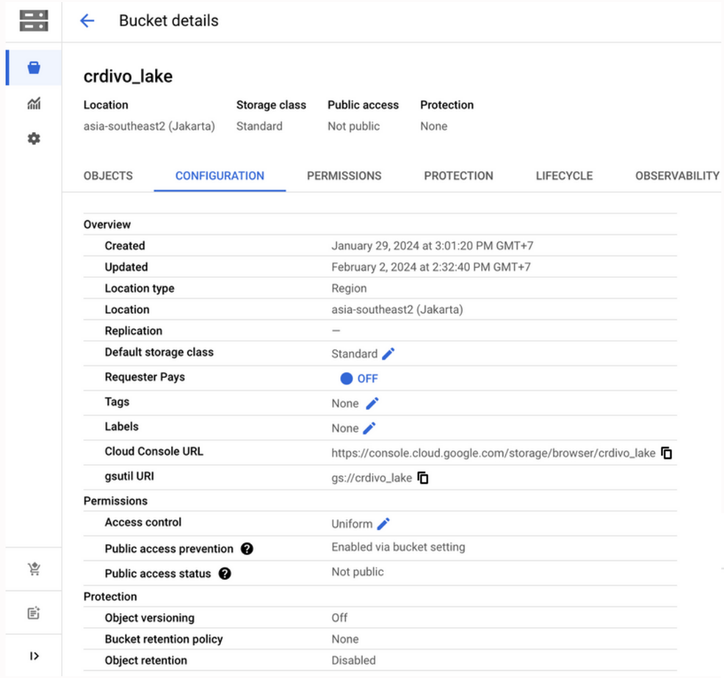


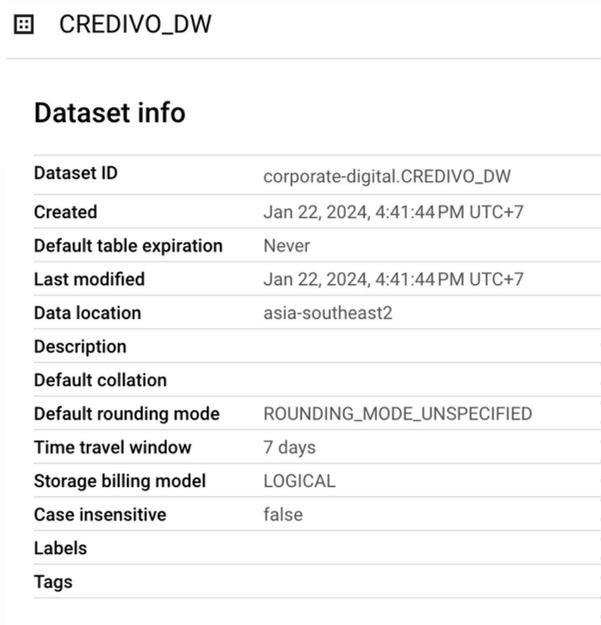




Requirement

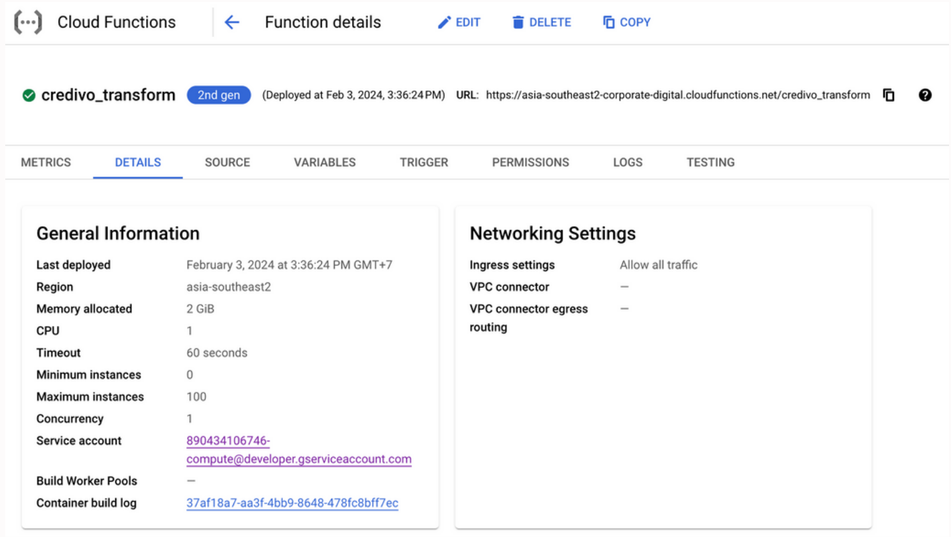
6. Google Cloud Bucket
   - Description:
     - Create and define bucket name as crdivo_lake
     - Location type : Region
     - Location : asia-southeast2 (Jakarta)
     - Storage Class : Standard (We consider to use standard because of the usage activity is actively used)
7. BigQuery Dataset
   - Description:
     - Create New dataset and define name as CREDIVO_DW to store Raw Data and Transformed Data
     - Data location : asia-southeast2 (Jakarta)

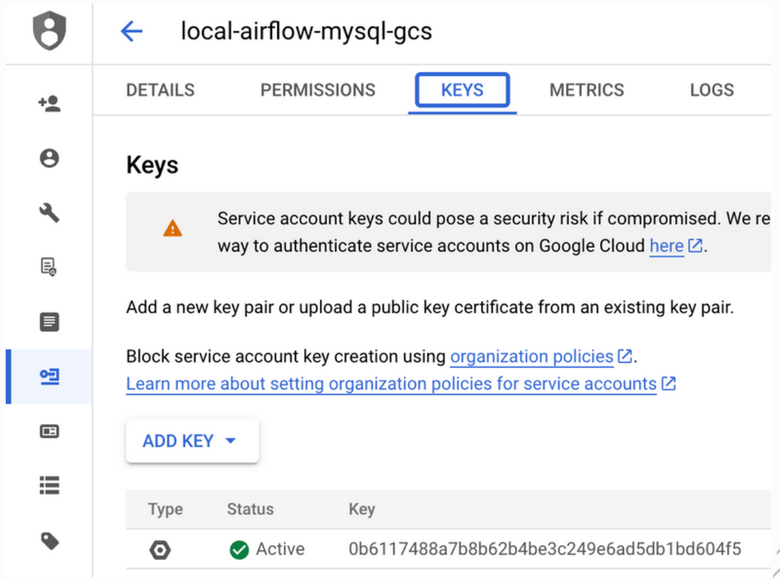




Requirement

8. Cloud Function
   - Description:
     - Create new function named credivo_transform to transform data that stored in Google Cloud Storage to BigQuery
     - Region : asia-southeast2
     - Memory allocated : 2 GiB
     - CPU : 1
     - Timeout : 60 seconds
     - Minimum instances : 0
     - Maximum instances: 100
9. Service Account for Credentials
   - Description:
     - Create and define service account name as local-airflow-mysql-gcs to generate google cloud credential key

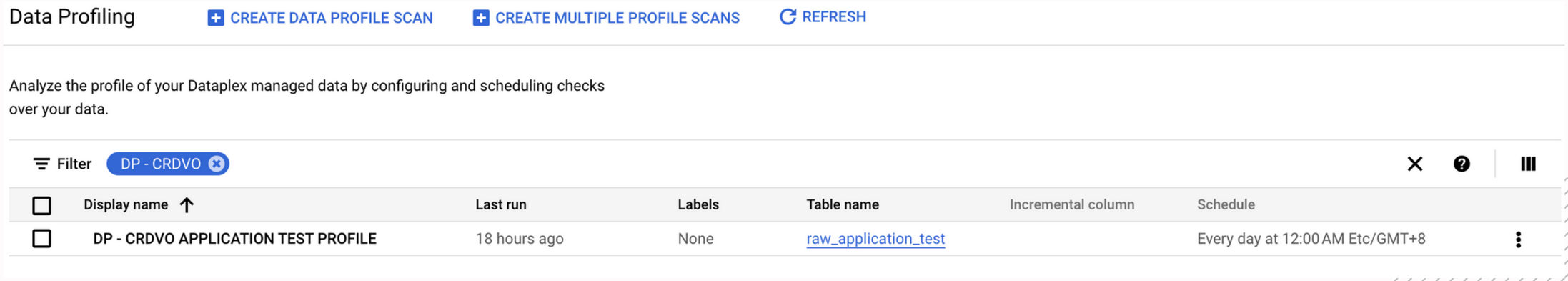




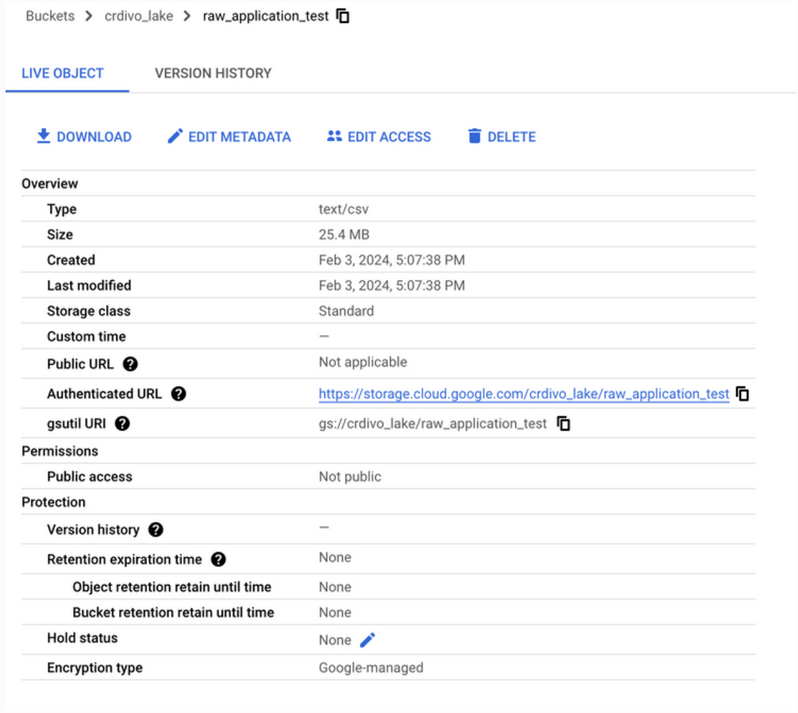
Requirement

10. Data Profiling Scan on Google Cloud Dataplex:

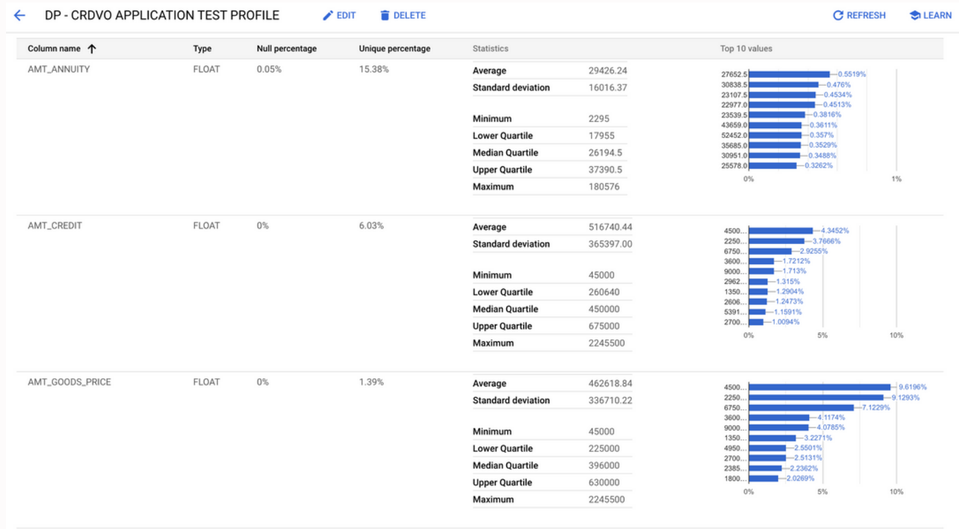
- Description:
  - Create and define data profile name as **DP - CRDV APPLICATION TEST**
  - Set profiling scope to **Entire Data** and sampling size to **All Data**
  - Send the profile test to BigQuery Dataset named CREDIVO_DW and name it as DP_{Table_Name} (In this case, we use application_test data to run the whole process
  - Set repeat check on Daily



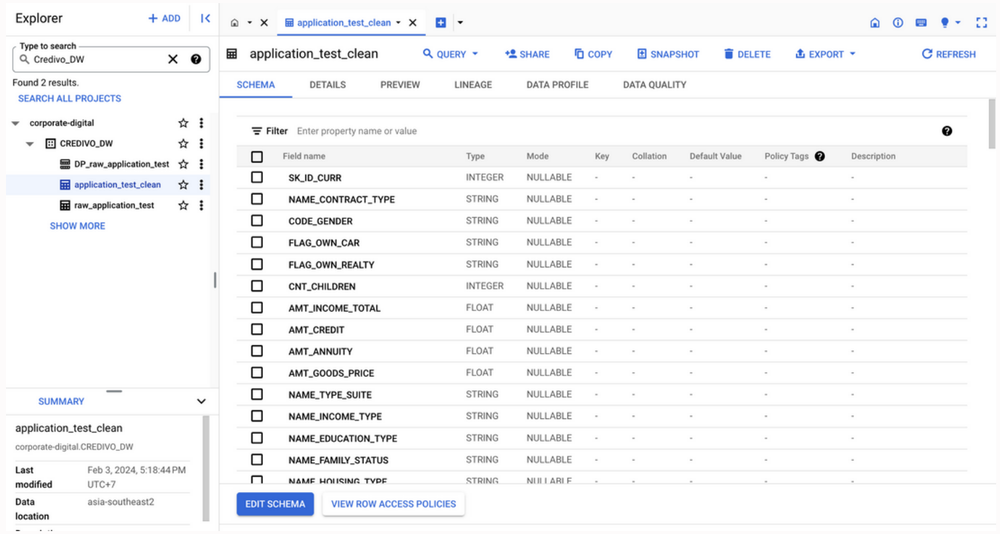
Output



Data Lake in Google Cloud Storage



Data Profiling in Dataplex

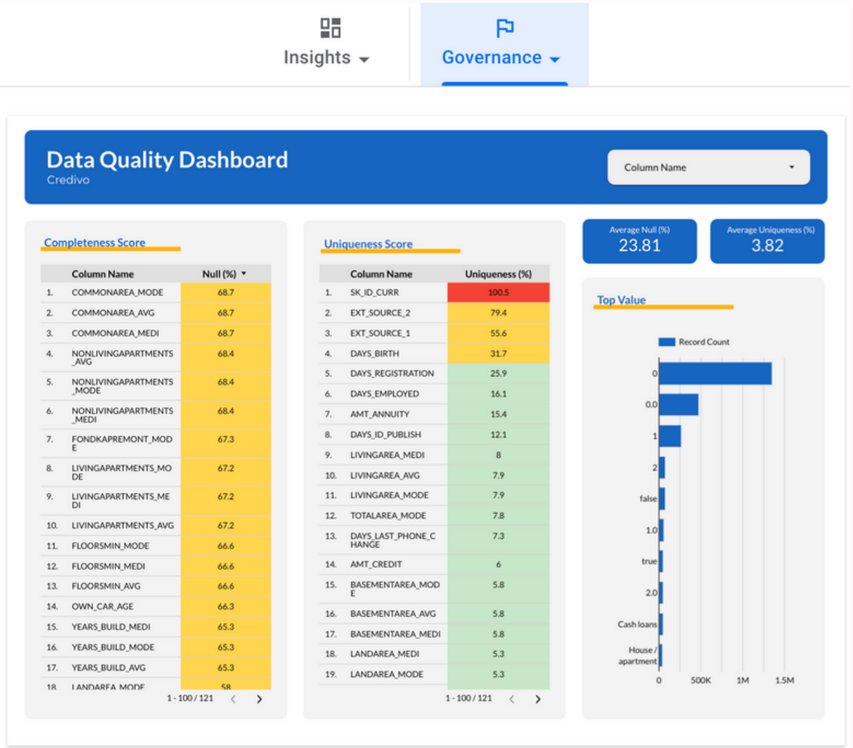


Raw and Clean Data in BigQUery

Output

Insight Dashboard

Data Quality Dashboard




THANK YOU