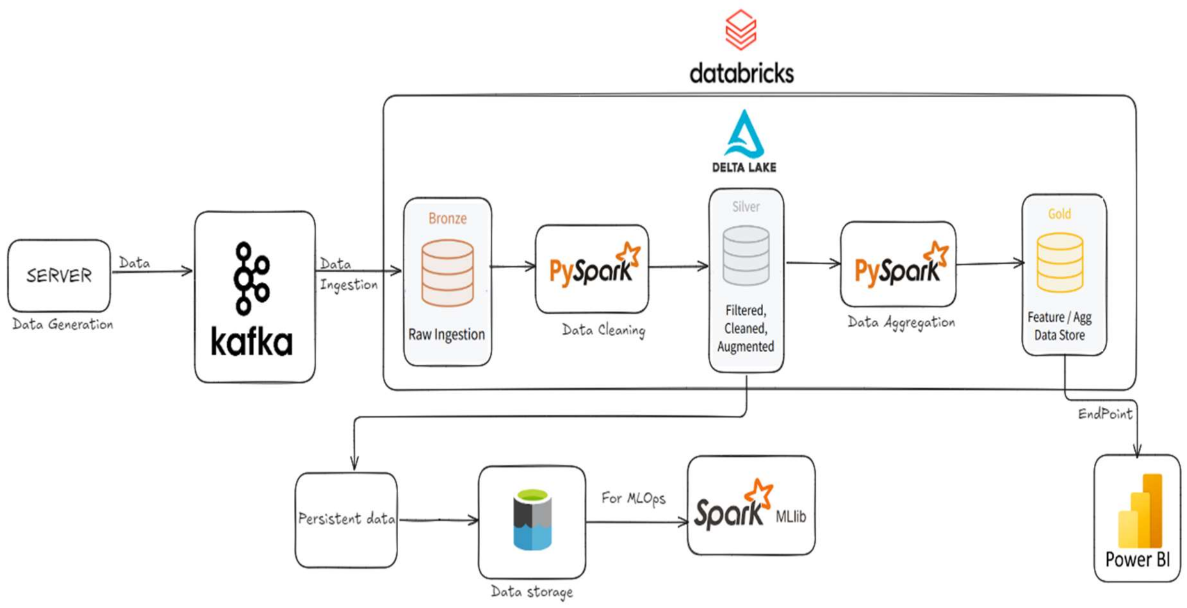
Problem Statement - To build a network performance dashboard using a real time streaming data using power Bi.

Tech Stack:

1. Kafka (Alternative - Event Hubs, Auto-loader)
2. Databricks (Alternative -HDInsight/Synapse Analytics)
3. Delta Tables (Alternative - Apache Hive/Parquet file)
4. PySpark (Alternative - Dask/Ray)
5. Delta Live Tables (Alternative -Apache Airflow)
6. Azure Storage (Alternative -Amazon S3/Google Cloud Storage)
7. PowerBi (Alternative - Tableau/Qlik)
8. ML lib (Alternative -Scikit-learn/TensorFlow)

Architecture -



Steps:

1. Start zookeeper and Kafka.

2. Create Topic in Kafka.

3. Create producer.py file.

4. Run Producer.py

5. Connection of Kafka with databricks using Pyspark script.

6. Storing streaming data into bronze delta table.

7. Filtration of the data.

8. Store the filtered data into the silver layer.

9. Aggregate the data for the silver layer according to the business requirement.

10. Store it in gold layer.

11. Use that data in Power BI dashboard.

12. Also from silver layer, the data is stored in Azure Data-lake GEN2 to keep a record of historical data.

13. This historical data can be used in ML Model implementation and prediction.

Implementation:

1. Started with Setting up Kafka and zookeeper on local system for testing code locally.

2. Then created a topic and ran producer.py and consumer.py to check the connection and data generation.

3. Ran a cron job to send real time data to topic.

4. Once it was tested locally, did the same configuration on VM and ran Kafka zookeeper, created topic, ran producer.py there.

5. To send real time data from Kafka Producer to databricks, we needed to build a connection using public IP.

6. Using this IP, we sent data to databricks through streaming through port 9092.

7. Once the data is received in the databricks, it is stored into bronze layer.

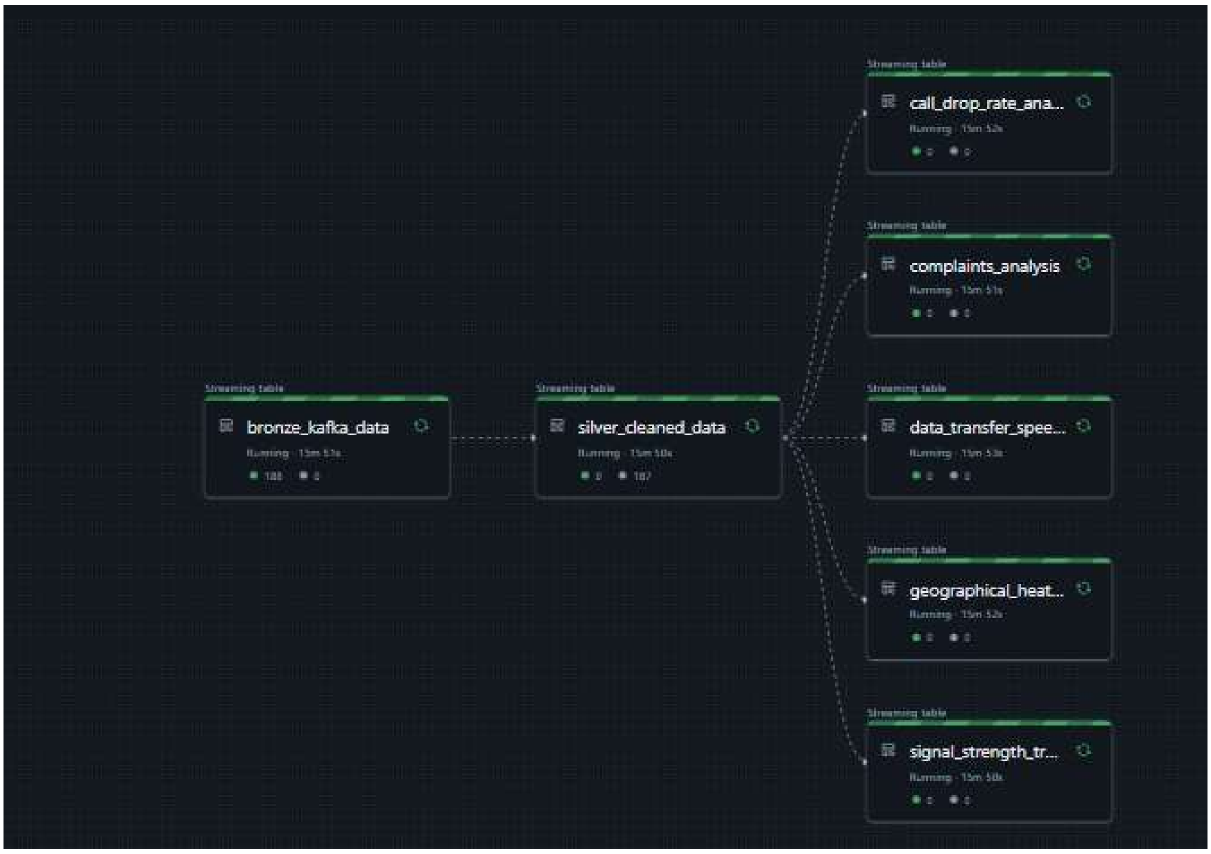
8. Then we wrote script for the filtration process on bronze layer data and aggregation script on silver layer data.

9. The streaming was real time, but the filtration and aggregation was working as a cron job which is batch processing.

10. So, after testing this code manually, we implemented this complete process into a pipeline using Delta Live Tables (DLT).

11. We first ran a sample pipeline given by Databricks themselves and read through the notebook it produced then we correlated that with our existing code and updated our code accordingly with the usage of python's Delta Live Tables (DLT) module, utilizing its wrappers to live stream the data through the tables, updating them dynamically.

12. Then we provided this notebook to the pipeline and ran it, this automates the complete process with the option to schedule it.



13. Further we are planning to connect this database to power Bi and get some insights about the network using dashboard.

Challenges

1. The exposure of public IP (Unext Team)

2. Automation of the process (DLT)

3. PowerBi and databricks connection