Minor End-to-End ETL Data Engineering Project

Table of Contents

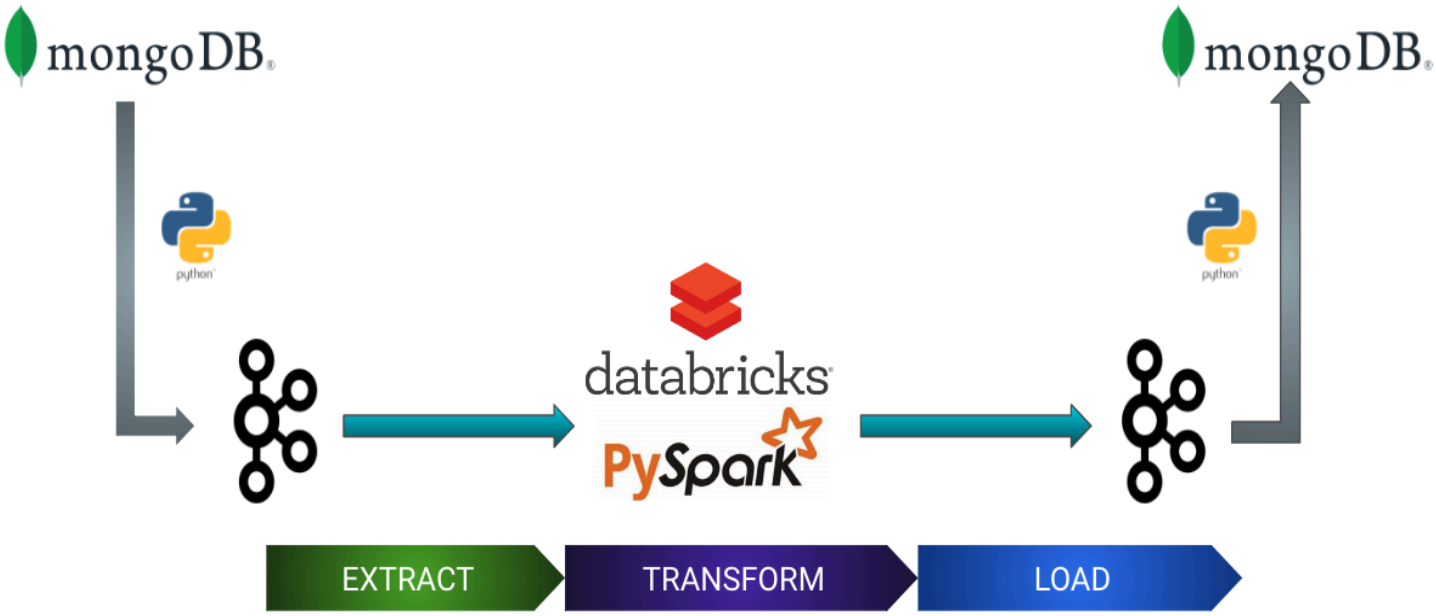
---

1. Summary

This project is a data engineering endeavor focused on integrating Kafka, MongoDB, PySpark, and Databricks to build an effective data pipeline. It commences with data ingestion from a Kafka topic into MongoDB, followed by storage in a Delta table on Databricks. PySpark is then employed for data transformation tasks like duplicate removal, null handling, and sorting. The transformed data is stored in Delta format for optimized storage. While the process doesn't involve real-time streaming, it demonstrates a robust data processing workflow that harnesses the strengths of these technologies for efficient data handling and storage.

---

2. System Architecture



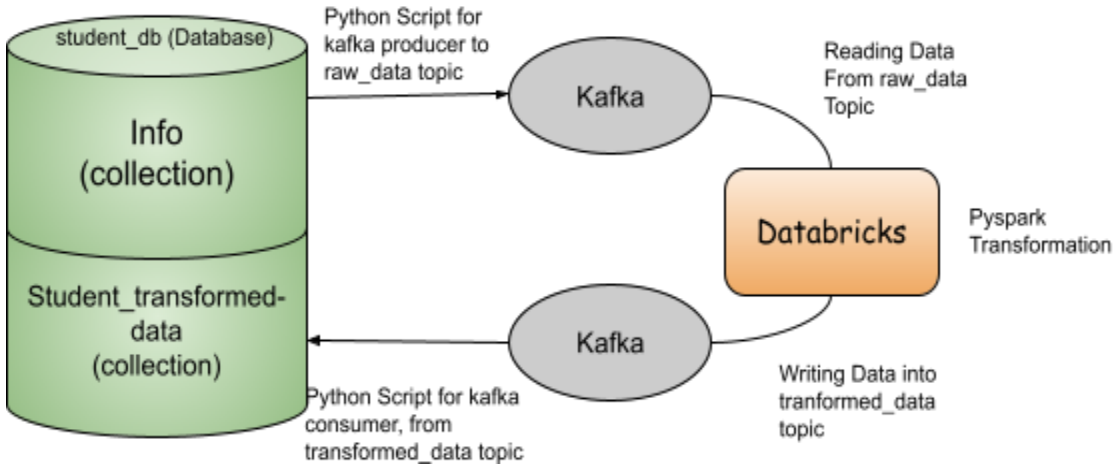
3. Technologies Used

- Kafka
- MongoDB
- Databricks(PySpark)
- Databricks
- Python

---

4. DataFlow and Integrations

DataFlow Diagram



1. **MongoDB to Kafka Integration(Producing)**
    - Produces data from a MongoDB collection ("info") to Kafka topic ("raw_data") using Python scripts.

- Establishes an initial data flow from MongoDB to Kafka for reading and further processing.

2. **Kafka to Databricks Delta Table Integration(Reading)**

    - Reads data from Kafka ("raw_data") to a Databricks notebook using PySpark and Kafka cluster API credentials.
    - Converts read data into a Delta table ("student_delta") stored in DBFS for optimized storage and management.

3. **PySpark Data Transformation**

    - Utilizes PySpark for data transformations, including duplicate removal, null value handling, column name capitalization, and data sorting.
    - Ensures data quality and consistency for downstream processing.

4. **Delta Table to Kafka Integration(Writing)**

    - Produces transformed data from the Delta table into a new Kafka topic ("transformed_data") using PySpark.

5. **Kafka to MongoDB Collection Integration (Consumption)**

    - Consumes transformed data from the Kafka topic ("transformed_data") using Python scripts.
    - Deserializes and processes messages, storing data in a new MongoDB collection ("Student_transformed_data") within "student_db."
    - Completes the integration loop by storing processed data in MongoDB for analysis or reporting purposes.

Conclusion

In conclusion, the integration of Kafka, MongoDB, PySpark, and Databricks in this data engineering project has facilitated a seamless data pipeline. Starting from data ingestion from Kafka into MongoDB, through transformation and storage in Databricks Delta tables, to real-time data streaming with Kafka, and finally consumption into MongoDB for analysis, the project demonstrates a robust and efficient approach to handling data across different technologies. This integration not only ensures data integrity and quality but also enables scalable and real-time data processing capabilities essential for modern data-driven applications.

****