👩‍💻 How to Run the Project

1. Clone the Repository

bash

```
git clone
https://github.com/silva-thiagomoura/Kantox-Data-Engineer-Challenge
```

2. Set Up the Environment

Make sure you have **Docker** and **Docker Compose** installed on your machine.

3. Start the Environment

Use Docker Compose to bring up the required services (Kafka, Zookeeper, Minio, and Spark).

bash

```
docker-compose up -d --build
```

This will start all the necessary services in the background.

4. Run the FastAPI Service

Access the API container:

bash

```
docker exec -it api-container bash
```

Then, start the FastAPI service:

bash

```
uvicorn api.collect:app --host 0.0.0.0 --port 8000
```

The API will be available at: http://localhost:8000/collect.

5. Run Spark Streaming

Access the Spark container:

bash

```
docker exec -it spark-master bash
```

Then, run the Spark Streaming job:

bash

```
/opt/bitnami/spark/bin/spark-submit \
  --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0 \
  /opt/spark-apps/spark_streaming.py
```

This will start consuming events from Kafka, transforming and enriching them, and sending the enriched events to the `acme.clickstream.latest.events` topic.

6. Monitor Kafka

Access the Kafka UI at: http://localhost:8080 to monitor the Kafka topics and their messages.

📂 Configuration Files

The configuration files for Kafka and logging can be found in the `config/` directory.

- `config/kafka_config.py`: Contains Kafka-related configurations such as topic names and broker URLs.
- `config/logging_config.py`: Contains logging configuration for monitoring and debugging.

🧪 Tests

Automated tests are located in the `tests/` directory. To run the tests, use:

bash

```
pytest tests/
```

Test API:

Test the FastAPI endpoint by simulating events being sent via the `/collect` endpoint.

Test Streaming:

Test Spark Streaming's functionality, ensuring it processes and enriches the data correctly.

📈 Monitoring

The project includes monitoring configurations for **Prometheus** and **Grafana** to visualize system metrics:

- **Prometheus**: http://localhost:9090
- **Grafana**: http://localhost:3000

These tools provide visibility into Kafka and Spark performance, such as consumer lag, data processing rate, and more.

📋 License

This project is licensed under the MIT License.

🔧 Additional Notes

Kafka Topics

- `acme.clickstream.raw.events`: This topic receives raw clickstream events from the API.
- `acme.clickstream.latest.events`: This topic receives enriched events after processing in Spark.

Running in Production

In a production environment, you can scale this setup by:

- Adding more **Kafka brokers** for better throughput.
- **Scaling Spark** to handle larger volumes of data.
- Using a more robust **storage solution** (such as AWS S3) instead of Minio.

💡 Improvements and Future Enhancements

- **Batch Processing**: You could add batch processing alongside real-time processing for cases where streaming isn't necessary.
- **Data Validation Enhancements**: Expand the validation in Schema Registry to ensure that incoming events are more rigorously checked.
- **Real-time Analytics**: Implement real-time analytics and aggregation on the processed data for insights (e.g., unique users per day).
- **Fault Tolerance**: Improve the fault tolerance by implementing retry logic and ensuring the system can handle partial failures.