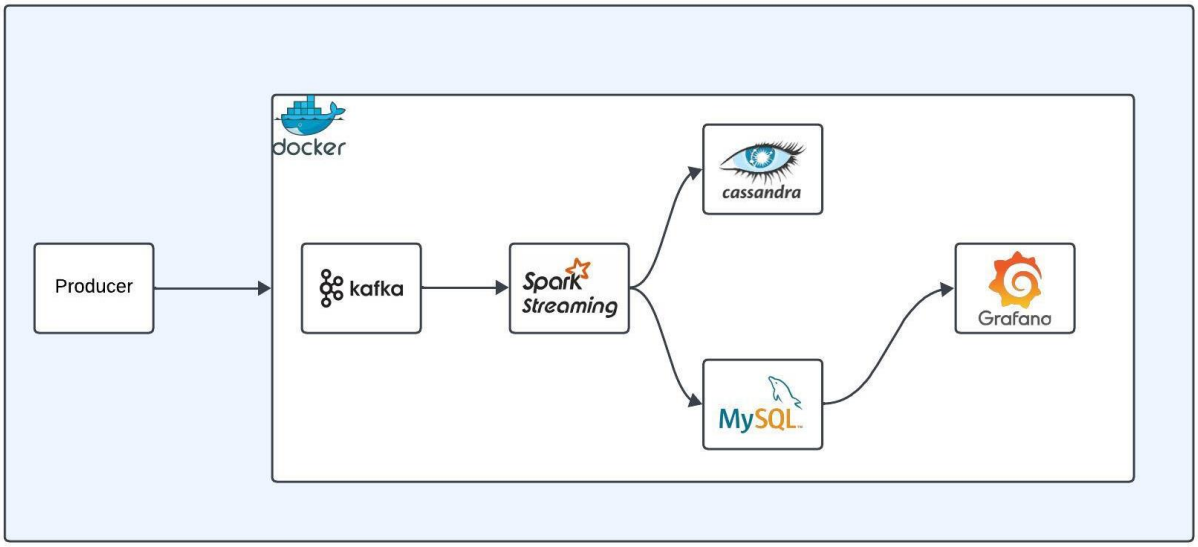
Case Study:

The chosen domain for the data pipeline is user interactions, with data fields such as event types, user IDs, timestamps, and page IDs. A Python producer script generates the data, randomly sending events to the Kafka topic. Subsequently, the streaming job processes the data at 10-second intervals, streaming it to both Cassandra and MySQL databases. Real-time analysis of views is facilitated through Grafana Visualization, which is also hosted within the Docker container environment.

Architecture:



The solution is containerized using Docker, encompassing various components within separate containers.

The docker-compose.yml file comprises eight services: Zookeeper, Kafka, Spark, Spark-worker, Cassandra, MySQL and Grafana. Each service is configured with specific attributes such as image, container_name, ports, environment, volumes, depends_on, restart, and command.

To initiate the Docker containers, execute the command:

```bash

docker-compose up -d
```

```

### 1. Data Source Simulation:

Begin by accessing the Kafka container and creating the required topic. Inside the Kafka container, execute:

```bash

docker exec -it kafka bash

kafka-topics.sh --create --topic user_interactions --bootstrap-server kafka:9092 --partitions 1 --replication-factor 1

```

Then exit the container.

2. Data Transformation:

Data processing involves utilizing PySpark transformations and constructing a schema for aggregated queries. The PySpark script resides in the `spark_script` directory within the project.

```bash

docker exec -it spark bash

spark-submit --jars /opt/bitnami/spark/jars/spark-cassandra-connector_2.12-3.3.0.jar,/opt/bitnami/spark/jars/mysql-connector-java-8.0.27.jar --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.2.0,org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.0 /opt/bitnami/spark/spark_script/data_streaming.py

```

3. Message Broker (Kafka):

Execute the Python script from the project's directory to simulate data streaming into Kafka:

```bash

python producer.py user_interactions

```

4. Data Storage:

Data is stored in two distinct data sources:

- Cassandra
- MySQL

5. **Data Visualization**:

Grafana serves as the designated data visualization tool. A dashboard has been crafted to present useful insights.

Grafana is hosted on http://localhost:3000/

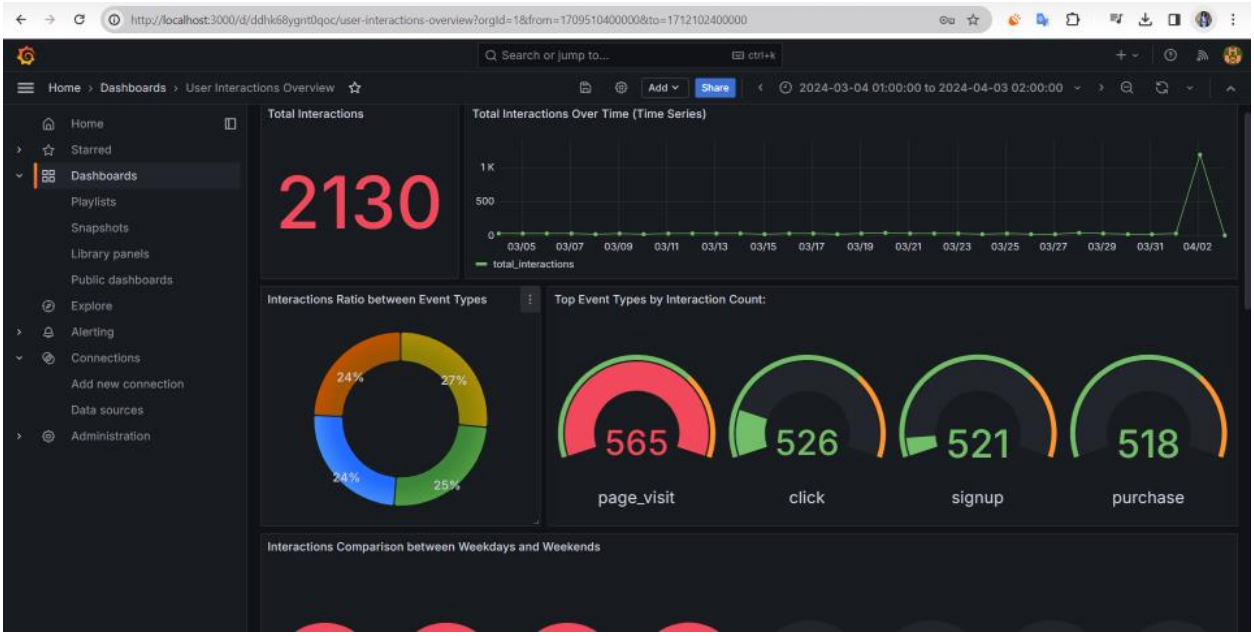
Adding the Mysql Datasource:

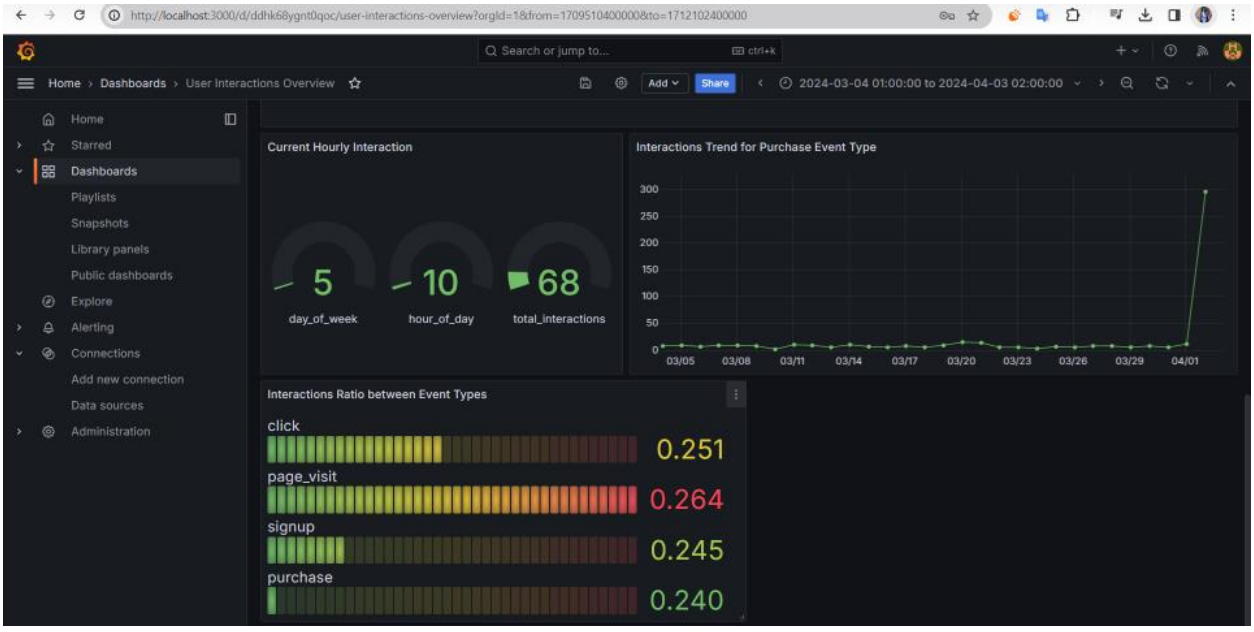
1. HostURL: host.docker.internal:3307
2. Database Name: user_interactions
3. Username & Password : root

The User Interaction Dashboard will look like this for representational purpose, I have included some of the key interactions overview. With more factual data, the analysis can be made in depth.

The view explains,

1. The total interactions so far.
2. Time series graph of the total interactions over time.
3. Total percentage of event based interactions.
4. Top Events Interaction Types with Count.
5. Interactions count Mapping on a weekday vs weekend.
6. Current hourly Interactions Count
7. Interactions time series of the Purchase Event Type
8. Interaction Ratio between Event types.
```





This setup facilitates the ingestion, transformation, storage, and visualization of data within the containerized environment, streamlining the entire data pipeline process.