GROUP PROJECT
MAHARISHI INTERNATIONAL UNIVERSITY

BIG DATA TECHNOLOGY FINAL PROJECT

MIU UNIVERSITY

MAHARISHI INTERNATIONAL UNIVERSITY

MEET OUR TEAM

BINIAM GHEBREMEDHIN
SOFTWARE ENGINEER

NATSNET GHIRMAY
SOFTWARE ENGINEER

HIDRI GHEBRETNSAE
SOFTWARE ENGINEER

MAHARISHI INTERNATIONAL UNIVERSITY

INTRODUCTION

This project is all about the design and implementation of a process to process real-time data from an eCommerce store and visualization of the processed data.

It involves the processing and analyzing of the data and storing it in a scalable distributed database model like HBase for further usage of the data.

The data analysis is done by sparkSQL by executing queries and then the processed data is stored in SQL and NoSQL DBs.

TECHNOLOGIES USED

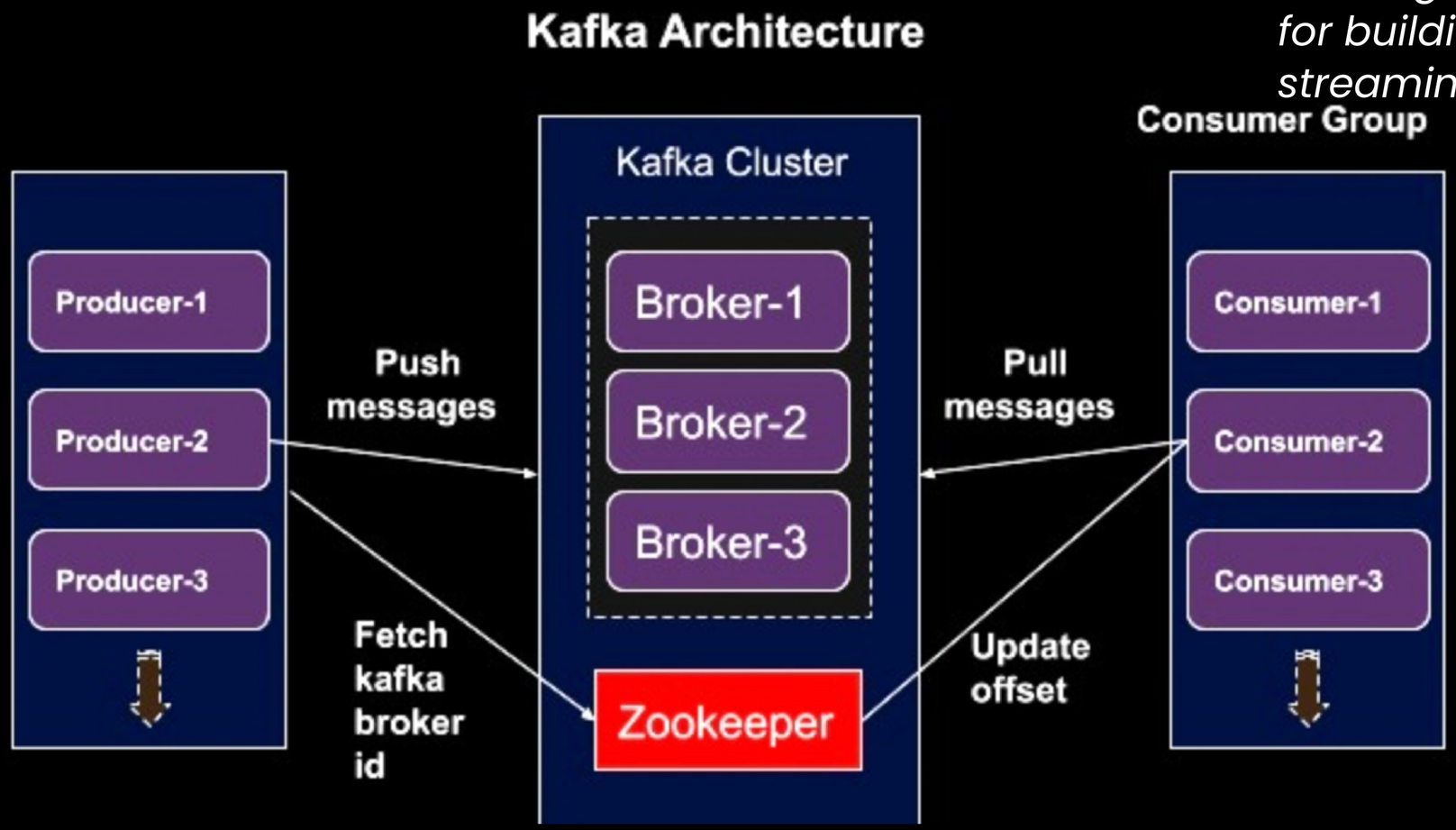
KAFKA
HBASE
SPARKSQL
SPARAK STREAMING
HIVE
HADOOP
DOCKER
JAVA
PYTHON
INTELLIJ
VSCODE
ECLIPSE

LEARN MORE

MAHARISHI INTERNATIONAL UNIVERSITY

KAFKA

Apache Kafka is an open-source stream processing platform developed by LinkedIn and donated to the Apache Software Foundation. It is designed for high-throughput, fault-tolerant, and scalable messaging systems. Kafka is commonly used for building real-time data pipelines and streaming applications.



Kafka's architecture consists of producers, topics, brokers, partitions, and consumers. Producers publish messages to topics, which are partitions of data. Brokers manage the storage and communication of these topics, and consumers subscribe to topics to receive the messages.

MAHARISHI INTERNATIONAL UNIVERSITY

DOCKER IS A PLATFORM FOR DEVELOPING, SHIPPING, AND RUNNING APPLICATIONS IN CONTAINERS. CONTAINERS ARE LIGHTWEIGHT, PORTABLE UNITS THAT PACKAGE APPLICATIONS AND THEIR DEPENDENCIES TOGETHER, ENSURING CONSISTENCY ACROSS DIFFERENT ENVIRONMENTS. HERE'S A BRIEF EXPLANATION OF DOCKER AND ITS KEY USES:

01

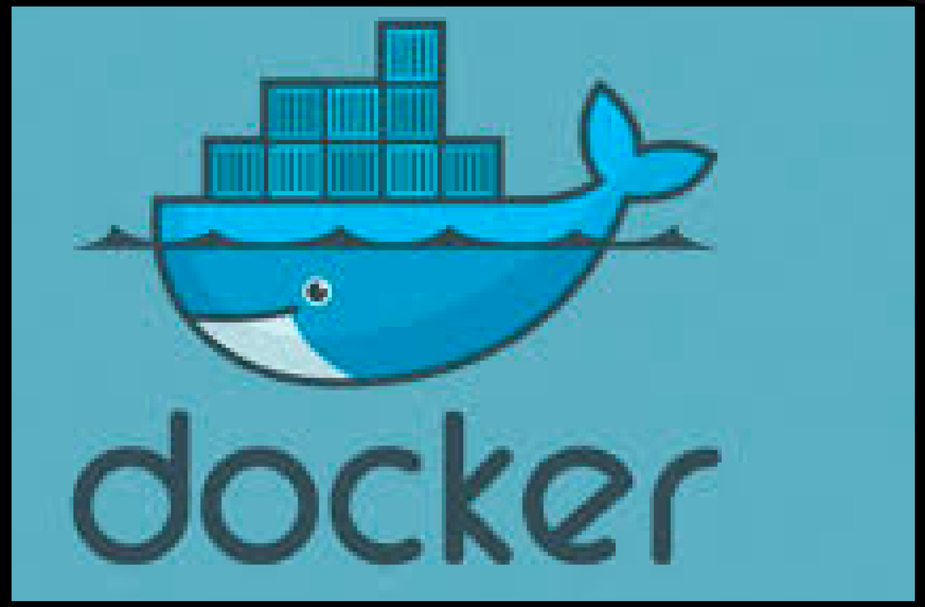
Isolation: Docker provides application isolation by encapsulating them in containers, preventing conflicts between applications and their dependencies.

02

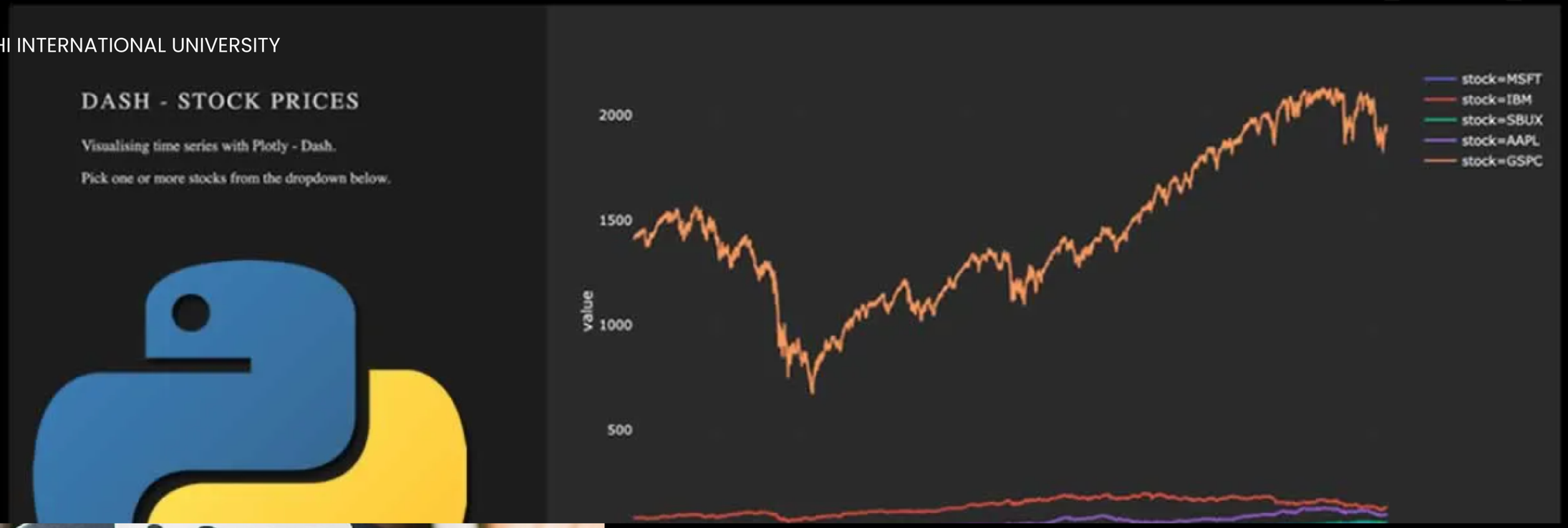
Portability: Containers can run on any system that supports Docker, ensuring applications work consistently across various environments, from development to production.

03

Rapid Deployment: Docker allows quick deployment of applications, reducing the time from development to deployment significantly.



Collaboration: Docker simplifies collaboration between development and operations teams by providing a consistent environment for both. Developers can create containers locally, and operations teams can deploy them without worrying about compatibility issues.

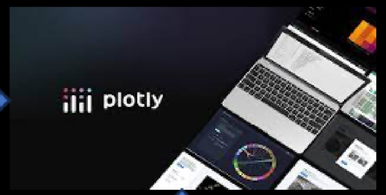


- Plotly is a versatile Python graphing library.
- It creates interactive, high-quality graphs.
- Supports various chart types: lines, bars, 3D plots, and heat maps.
- Allows zooming, panning, and hovering over data points for insights.
- Operates seamlessly online and offline.
- In offline mode, visualize plots in local environments like Jupyter Notebooks.
- In online mode, it offers collaborative features for storing, sharing, and teamwork.

PROJECT FLOW - STREAMING PIPELINE

E-commerce store

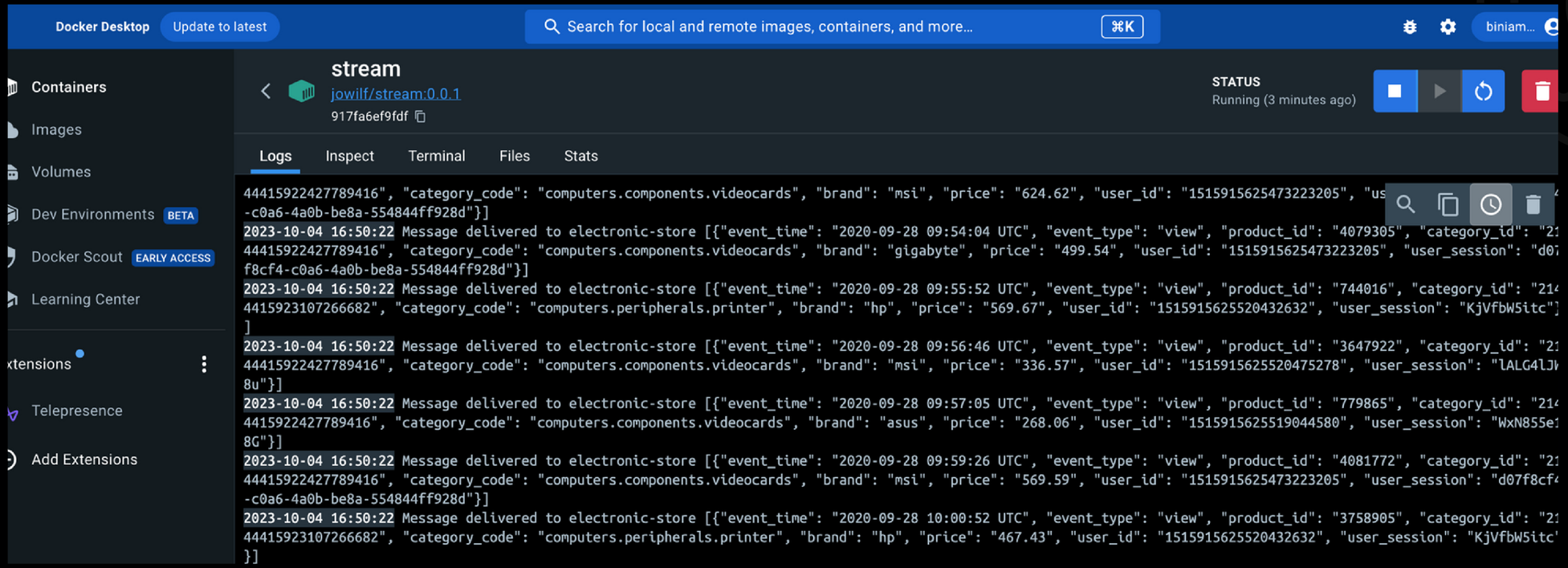
E-commerce Analysis



E-commerce store

E-commerce Analysis

MAHARISHI INTERNATIONAL UNIVERSITY

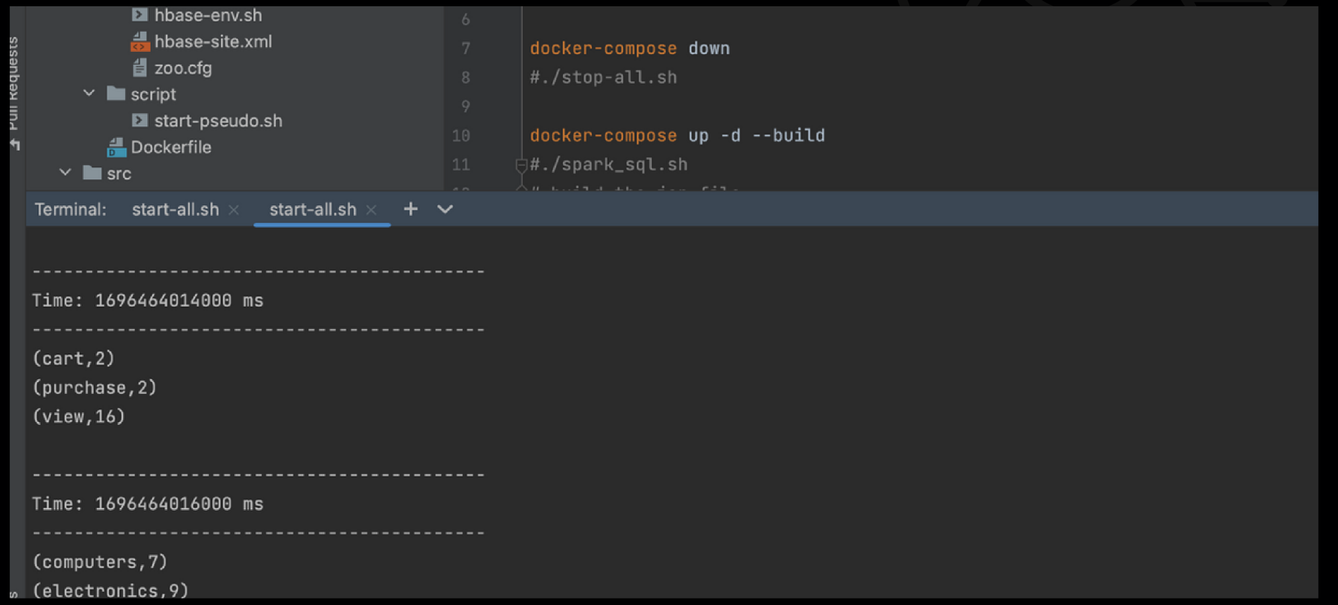


SAMPLE STREAMING DATA TO APACHE KAFKA

The Data is streamed to Apache Kafka from the producers where the Kafka cluster puts it into different brokers and notifies the different consumers that have subscribed to the specific topic.

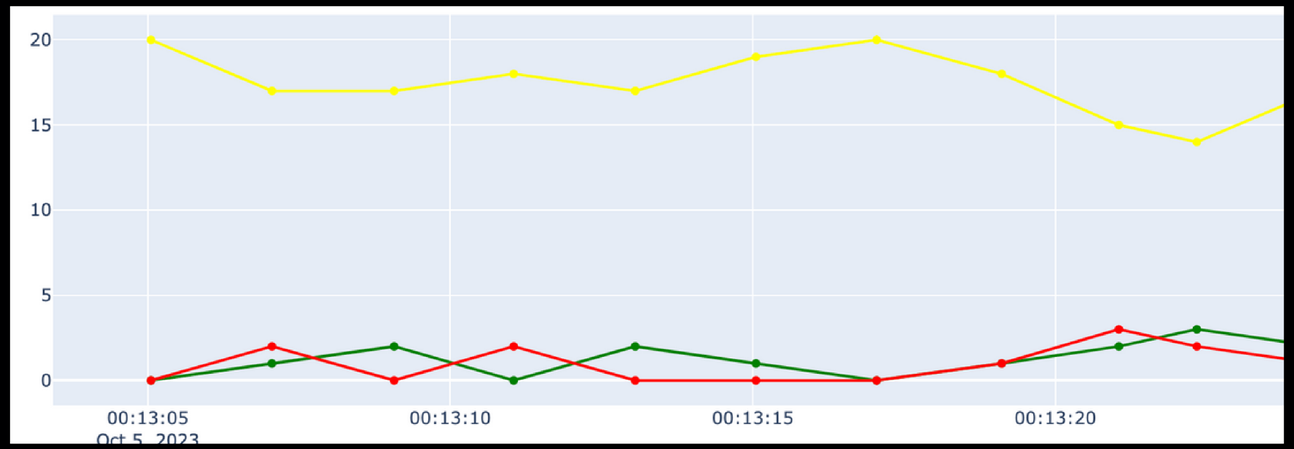
SALFORD & CO.

```java
public static void main(String[] args) throws InterruptedException, IOException {
    TableUtils tableUtils = new TableUtils();
    tableUtils.createTable();

    SparkConf sparkConf = new SparkConf().setAppName("JavaNetworkWordCount");

    JavaStreamingContext ssc = new JavaStreamingContext(sparkConf,
            Durations.seconds(2));

    ssc.sparkContext().setLogLevel("ERROR");

    Map<String, Object> kafkaParams = new HashMap<>();
    kafkaParams.put("bootstrap.servers", "kafka:29092");
    kafkaParams.put("key.deserializer", StringDeserializer.class);
    kafkaParams.put("value.deserializer", StringDeserializer.class);
    kafkaParams.put("group.id", "spark");
    kafkaParams.put("auto.offset.reset", "earliest");
    kafkaParams.put("enable.auto.commit", false);
```



STREAMING WITH SPARK

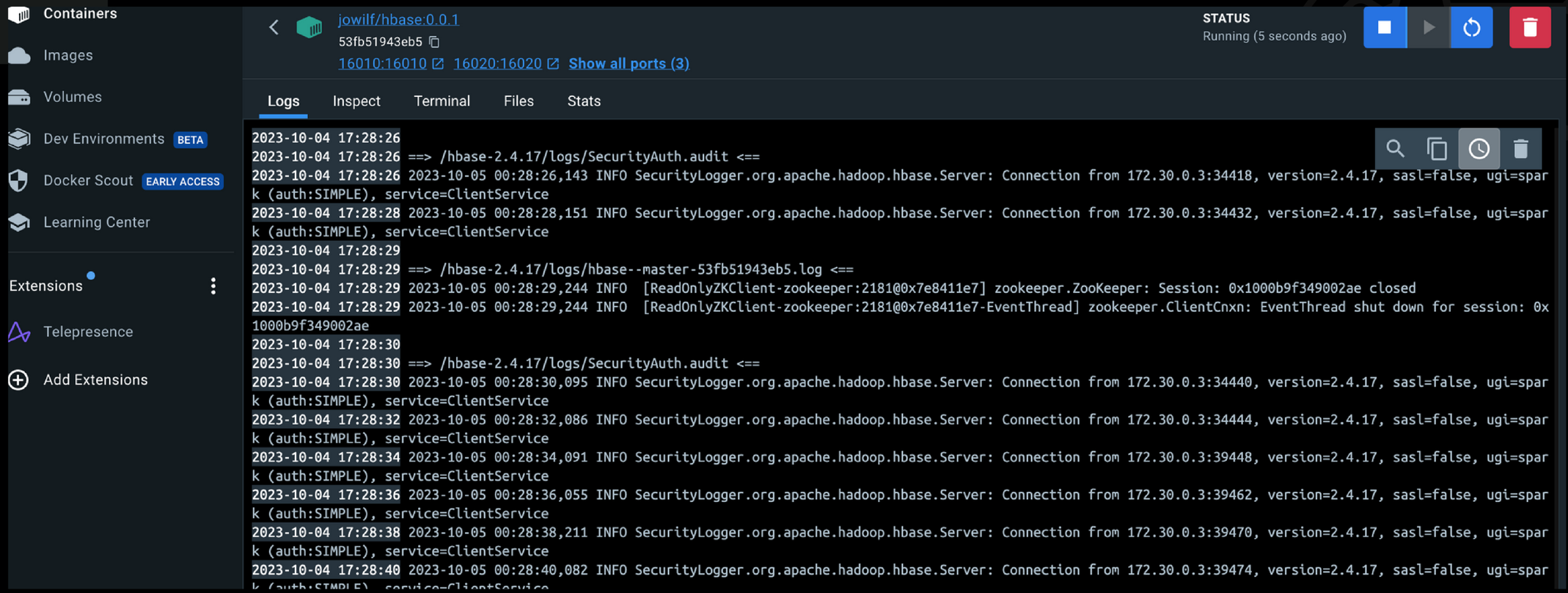
SALFORD & CO.




```
def fig(self):
    return go.Figure(
        data=[
            go.Scatter(
                x=self.dates,
                y=self.views,
                line=dict(color='yellow'),  # Set line color to green
                legendgroup="view",
                name="View",
                legendgrouptitle=Legendgrouptitle(text="View"),
            ),
            go.Scatter(
                x=self.dates,
                y=self.carts,
                line=dict(color='green'),  # Set line color to green
                name="Cart",
                legendgroup="cart",
                legendgrouptitle=Legendgrouptitle(text="Cart"),
            ),
            go.Scatter(
                x=self.dates,
                y=self.purchases,
```

REAL-TIME DATA VISUALIZATION OR PRESENTATION

MAHARISHI INTERNATIONAL UNIVERSITY



SPARKSTREAM - HBASE

Integrating Apache Spark Streaming with Apache HBase enables real-time data processing and storage capabilities. Spark Streaming allows the processing of live data streams, while HBase serves as a scalable, distributed NoSQL database. By combining these technologies, organizations can analyze and process real-time data streams using Spark Streaming and then store the results in HBase for efficient retrieval and analysis. This integration facilitates seamless handling of large volumes of real-time data, making it ideal for applications requiring instant data processing, analysis, and storage.

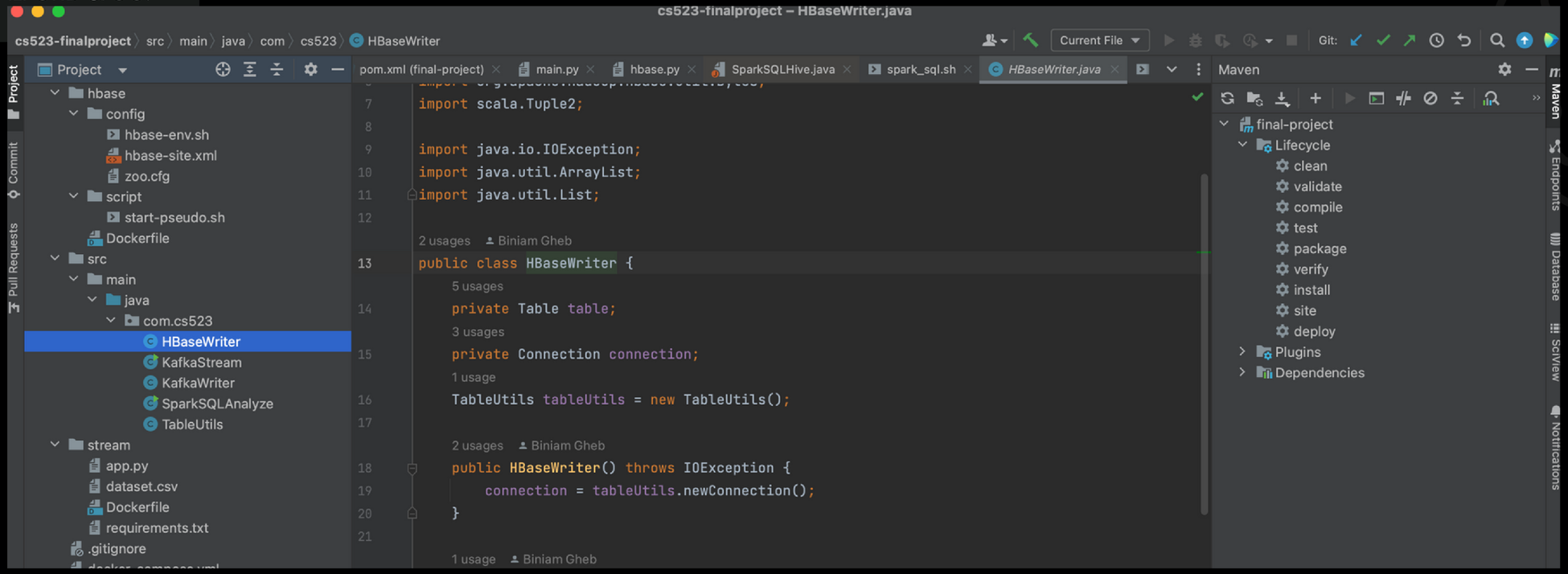
SALFORD & CO.

```
#./stop-all.sh
#docker-compose up -d --build
#./spark_sql.sh
#docker-compose restart dashboard
#./spark.sh

docker-compose down
#./stop-all.sh

docker-compose up -d --build
#./spark_sql.sh
# build the jar file
mvn package
# copy our dataset inside the spark docker container
docker cp stream/dataset.csv spark:/dataset.csv
# copy the generated jar file inside the spark docker container
docker cp final-project-1.0-SNAPSHOT.jar spark:/app_sql.jar
# Run spark submit inside the docker container
docker exec -it spark spark-submit --master=local --conf spark.sql.shuffle.partitions=1 --class com.cs523.Spa
docker-compose restart dashboard
#./spark.sh
```

HOW TO RUN THE APPLICATION

To run the application you have to start all shell scripts and it will stop all the running containers. Then it will start again all the containers in the same network by creating new images(if not available) otherwise it will run the existing ones as docker containers.or you can explicitly run by docker-compose command.

SALFORD & CO.



SAMPLE JAVA CODE
HBASE WRITER

MAHARISHI INTERNATIONAL UNIVERSITY

```
sudo cp /etc/hive/conf/hive-site.xml /usr/lib/spark/conf/

spark-submit --class "cs523.finalProject.SparkSQLHive"
    --master yarn /home/cloudera/workspace/finalProject/target/finalProject-0.0.1-SNAPSHOT.jar


SELECT brand, COUNT(event_type) AS viewed FROM electronics WHERE event_type="view"
GROUP BY brand ORDER BY viewed DESC


SELECT brand, MAX(price) AS highestPrice FROM electronics GROUP BY brand ORDER BY brand LIMIT 7;


SELECT brand, SUM(price) AS highestPrice FROM electronics GROUP BY brand ORDER BY brand LIMIT 7;
```

SPARKSQL COMMANDS

```
import ...


public class SparkSQLHive {

    public static void main(String[] args) {
        JavaSparkContext sc = new JavaSparkContext(new SparkConf().setAppName("electronics").setMaster("local").set("spark.sql.warehouse
        HiveContext hiveContext =  new HiveContext(sc);

        //total sale per brand
        DataFrame mostPurchased = hiveContext
                .sql("SELECT brand, SUM(cast(price as double)) AS totalSale FROM electronics WHERE event_type='purchase' GROUP BY brand
        mostPurchased.show();
    }

}
```

WHOLENESS OF PROJECT

In Spark Streaming, real-time data is processed in micro-batches, enabling rapid analysis and response to live data streams. In this project we have been actively involved with different technologies, we have observed the power of individual technologies but more than that we have emphasized how all those technologies can solve realtime processing of huge amounts of data, monitor them using different visualization tools, and how this amount of data can be stored into a scalable Databases for future use.

Similarly, the Science of Consciousness delves into the continuous, ever-changing nature of human awareness and perception. Both concepts emphasize the importance of understanding and adapting to ongoing, evolving information. In a metaphorical sense, Spark Streaming can be seen as analyzing the constant flow of data in the digital world, mirroring the intricate processes underlying the ever-changing landscape of human consciousness explored in the Science of Consciousness. Both highlight the significance of real-time adaptation and analysis in their respective domains, emphasizing the dynamic nature of knowledge and understanding and this can be achieved by diving deep into pure consciousness through Transcendental meditation.

```
CREATE EXTERNAL TABLE electronics (event_time STRING, event_type STRING, product_id int, category_id int, category_code STRING, brand STRING, price FLOAT, user_id int, user_session STRING ) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' LOCATION '/home/cloudera/cs523/inputProject' TBLPROPERTIES ('skip.header.line.count'='1');

//Load command
LOAD DATA INPATH '/user/cloudera/cs523/inputProject/popular_brand_only.csv' OVERWRITE INTO TABLE employee;
```

HIVE TABLE

MAHARISHI INTERNATIONAL UNIVERSITY

THANK YOU

BIG DATA TECHNOLOGY WITH
PROF.MRUDULA MUKADAM