Mukul Gaur edureka project

**code**

**Product Producer**

package snippet;

import org.apache.kafka.clients.producer.KafkaProducer;

import org.apache.kafka.clients.producer.ProducerConfig;

import org.apache.kafka.clients.producer.ProducerRecord;

import org.apache.kafka.common.serialization.StringSerializer;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.Properties;

public class ProductProducer {

private final static String TOPIC\_NAME = "products";

private final static String BOOTSTRAP\_SERVERS = "localhost:9092";

private final static String CSV\_FILE\_PATH = "/home/edureka/Desktop/products.csv";

public static void main(String args[]) throws JSONException {

// Set producer configuration properties in obj prop

Properties props = new Properties();

props.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, BOOTSTRAP\_SERVERS); //bootstrap

props.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

props.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

// Create Kafka producer instance

KafkaProducer<String, String> producer = new KafkaProducer<>(props);

try {

// Read products from CSV file

BufferedReader reader = new BufferedReader(new FileReader(CSV\_FILE\_PATH));

String line;

while ((line = reader.readLine()) != null) {

// Split CSV line by comma separator

String[] data = line.split(",");

// Create product JSON object

JSONObject json = new JSONObject();

json.put("PogId", data[0]);

json.put("Supc", data[1]);

json.put("Brand", data[2]);

json.put("Description", data[3]);

json.put("Size", data[4]);

json.put("Category", data[5]);

json.put("SubCategory", data[6]);

json.put("Price", data[7]);

json.put("Quantity", data[8]);

json.put("Country", data[9]);

json.put("SellerCode", data[10]);

json.put("CreationTime", data[11]);

json.put("Stock", data[12]);

// Send JSON data to Kafka

ProducerRecord<String, String> record = new ProducerRecord<>(TOPIC\_NAME, json.toString());

producer.send(record);

}

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

producer.close();

}

}

**KafkaConsumerClass**

package snippet;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Collections;

import java.util.Properties;

import org.apache.kafka.clients.consumer.ConsumerRecord;

import org.apache.kafka.clients.consumer.ConsumerRecords;

import org.apache.kafka.clients.consumer.KafkaConsumer;

import org.json.JSONException;

import org.json.JSONObject;

import com.datastax.driver.core.BoundStatement;

import com.datastax.driver.core.Cluster;

import com.datastax.driver.core.PreparedStatement;

import com.datastax.driver.core.Session;

public class KafkaConsumerClass {

private static final String CASSANDRA\_HOST = "localhost";

private static final String CASSANDRA\_KEYSPACE = "ecommerce";

private static final String MYSQL\_URL = "jdbc:mysql://localhost:9092/products";

private static final String MYSQL\_USER = "root";

private static final String MYSQL\_PASSWORD = "root";

public static void main(String[] args) throws JSONException, SQLException, InstantiationException, IllegalAccessException, ClassNotFoundException {

// set up Cassandra cluster

Cluster cassandraCluster = Cluster.builder().addContactPoint(CASSANDRA\_HOST).build();

Session cassandraSession = cassandraCluster.connect(CASSANDRA\_KEYSPACE);

// set up MySQL connection

Connection mysqlConnection = null;

try {

Class.forName("com.mysql.jdbc.Driver").newInstance();

mysqlConnection = DriverManager.getConnection(MYSQL\_URL, MYSQL\_USER, MYSQL\_PASSWORD);

} catch (SQLException e) {

e.printStackTrace();

}

// set up Kafka consumer

KafkaConsumer<String, String> consumer = new KafkaConsumer<String,String>(getProperties());

consumer.subscribe(Collections.singletonList("product-feed"));

while (true) {

ConsumerRecords<String, String> records = consumer.poll(1);

for (ConsumerRecord<String, String> record : records) {

// parse JSON dat

JSONObject json = new JSONObject(record.value());

// insert data into Cassandra

PreparedStatement cassandraStatement = cassandraSession.prepare(

"INSERT INTO products (pogid, supc, brand, description, size, category, sub\_category, country, seller\_code) " +

"VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");

BoundStatement cassandraBoundStatement = cassandraStatement.bind(

json.getString("PogId"), json.getString("Supc"), json.getString("Brand"), json.getString("Description"),

json.getString("Size"), json.getString("Category"), json.getString("Sub Category"), json.getString("Country"),

json.getString("Seller Code"));

cassandraSession.execute(cassandraBoundStatement);

// insert data into MySQL

java.sql.PreparedStatement mysqlStatement = mysqlConnection.prepareStatement(

"INSERT INTO products (pogid, supc, price, quantity) VALUES (?, ?, ?, ?)");

mysqlStatement.setString(1, json.getString("PogId"));

mysqlStatement.setString(2, json.getString("Supc"));

mysqlStatement.setDouble(3, json.getDouble("Price"));

mysqlStatement.setInt(4, json.getInt("Quantity"));

mysqlStatement.executeUpdate();

}

}

}

private static Properties getProperties() {

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("group.id", "test-group");

props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

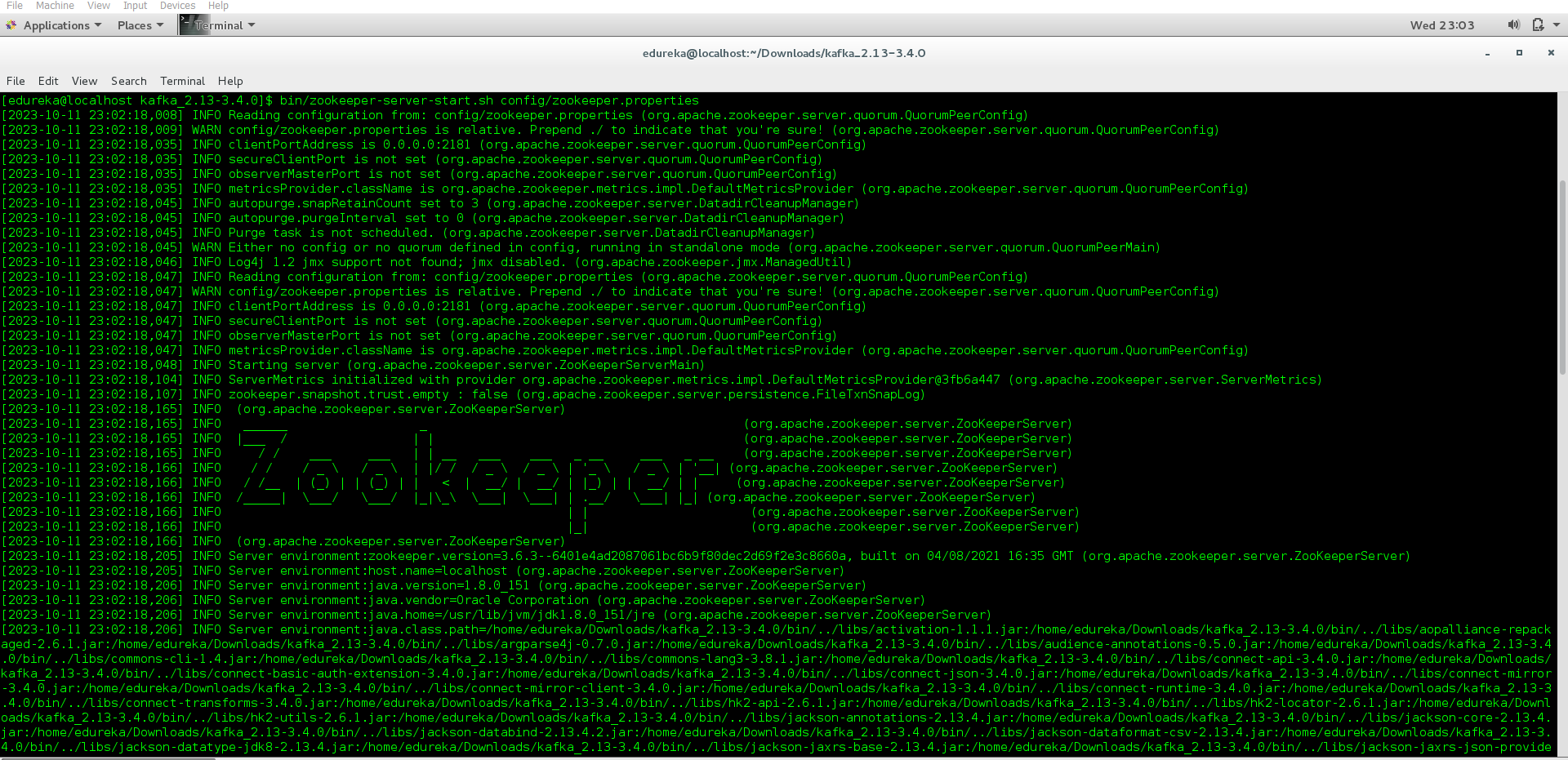
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

return props;

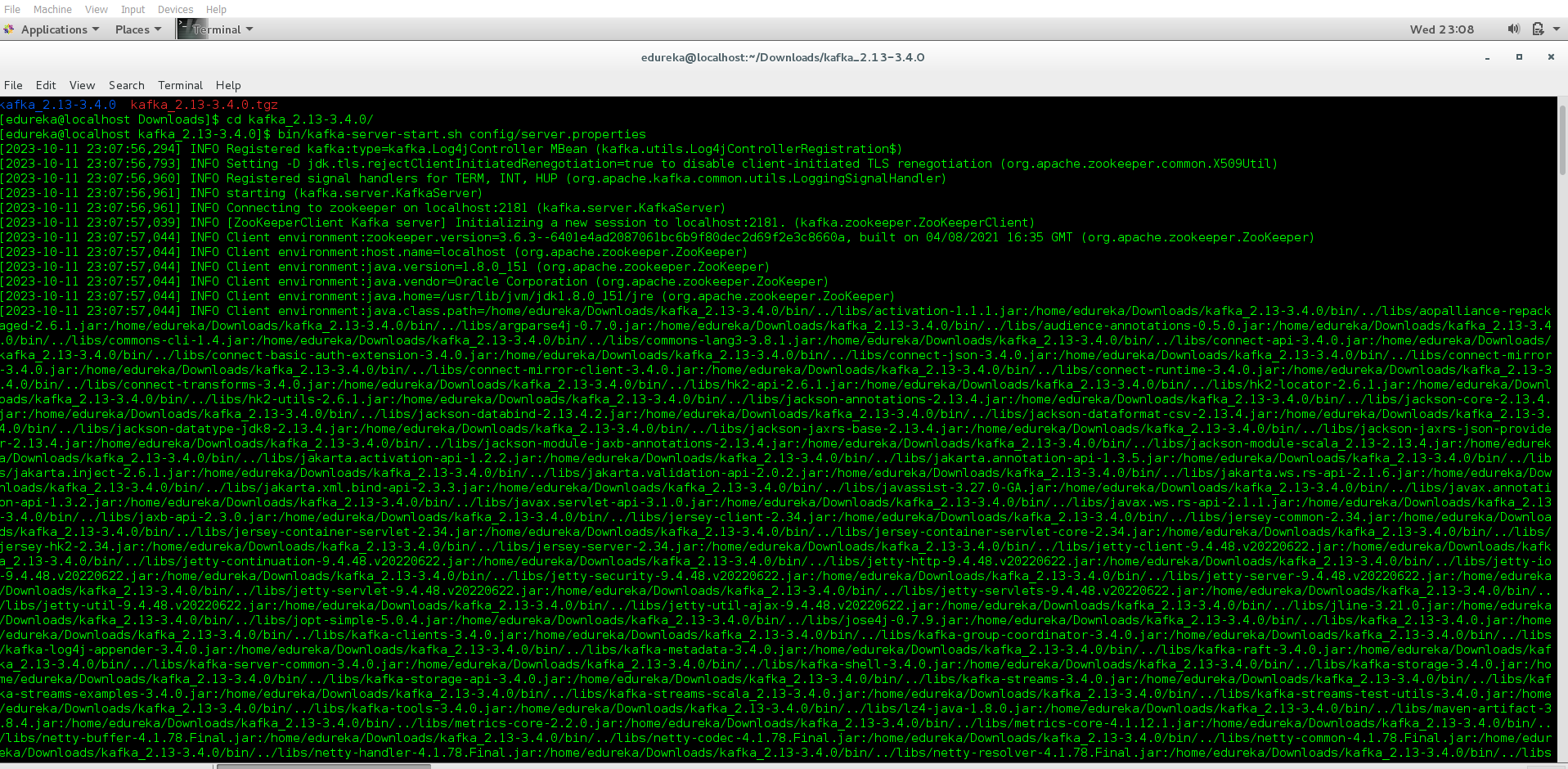
}

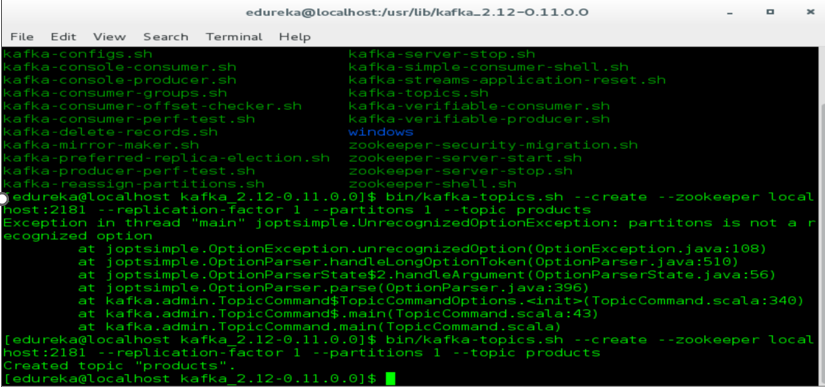
}

CLI   
zookeeper

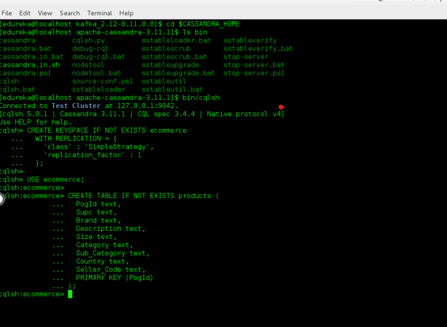


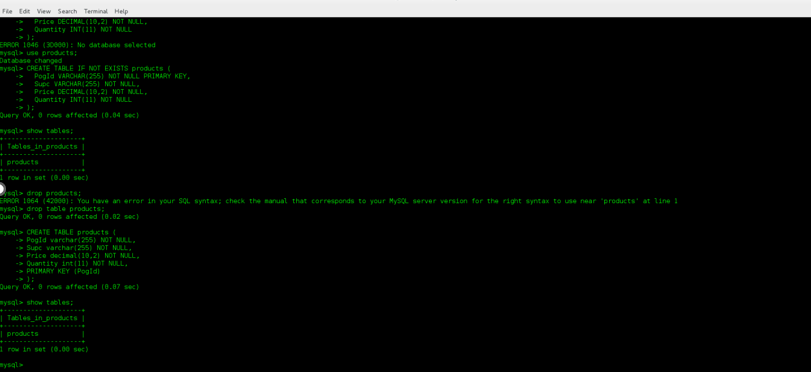
Kafka server



Topic name products

Cassandra





**Output**

**My sql**

mysql> use products;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> select count(\*) from products;

+-------------------+

| count(\*) |

+-------------------+

| 1040215 |

+-------------------+

1 row in set (0.00 sec)

**CASSANDRA**

[edureka@localhost apache-cassandra-3.11.1]$ bin/cqlsh

Connected to Test Cluster at 127.0.0.1:9042.

[cqlsh 5.0.1 | Cassandra 3.11.1 | CQL spec 3.4.4 | Native protocol v4]

Use HELP for help.

cqlsh> select count(\*) from ecommerce.products;

count

-------

1040215

(1 rows)