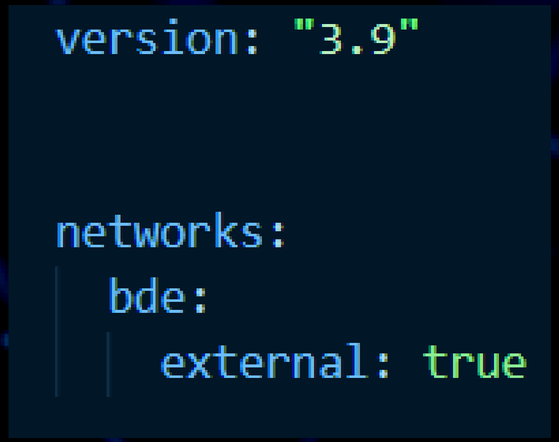
Kafka-Spark-Flink -Projekat 2- Bike Washington

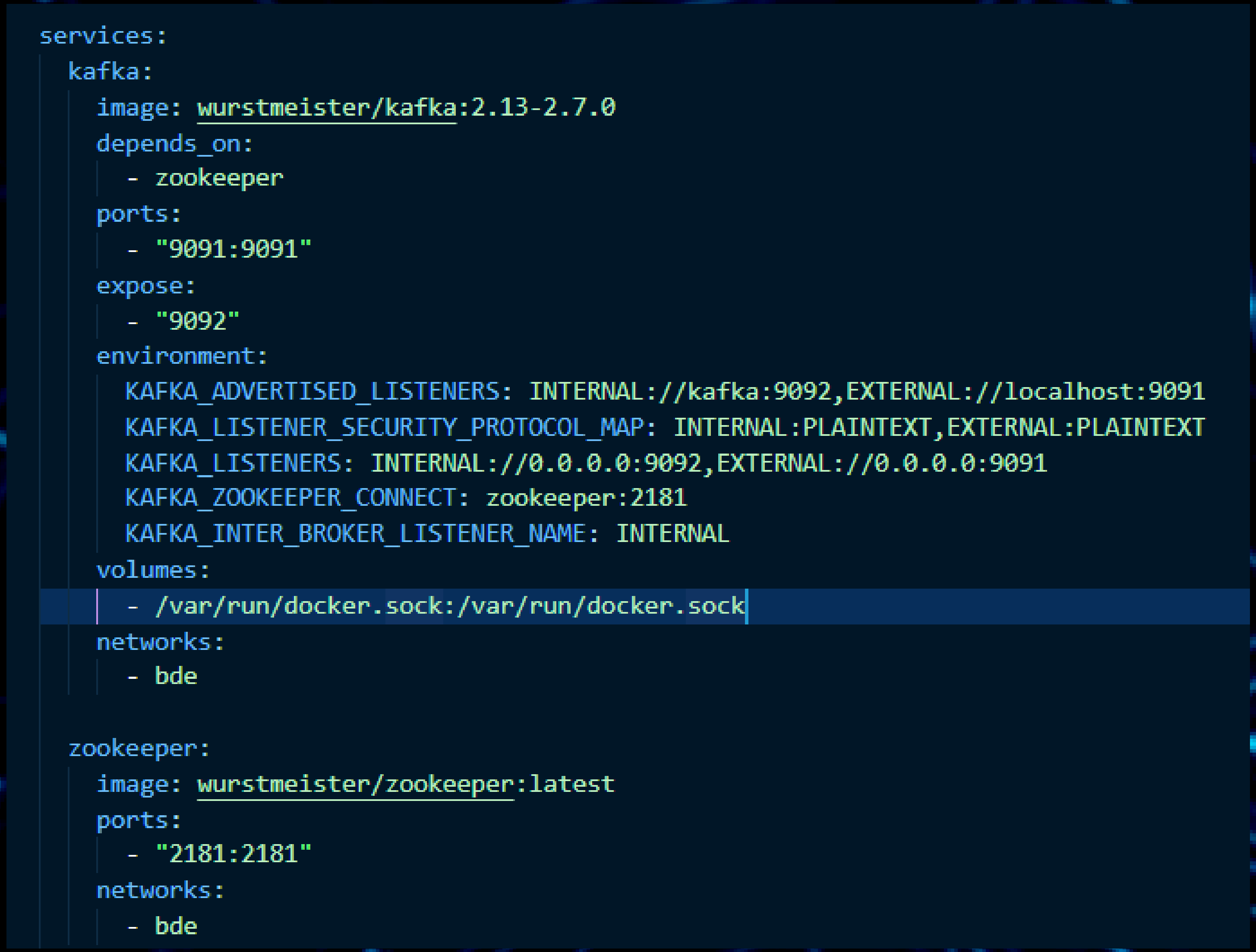
Big Data Systems

Mentor:
Prof. dr Dragan H. Stojanovic
Student:
Petrovic Nikola 1466

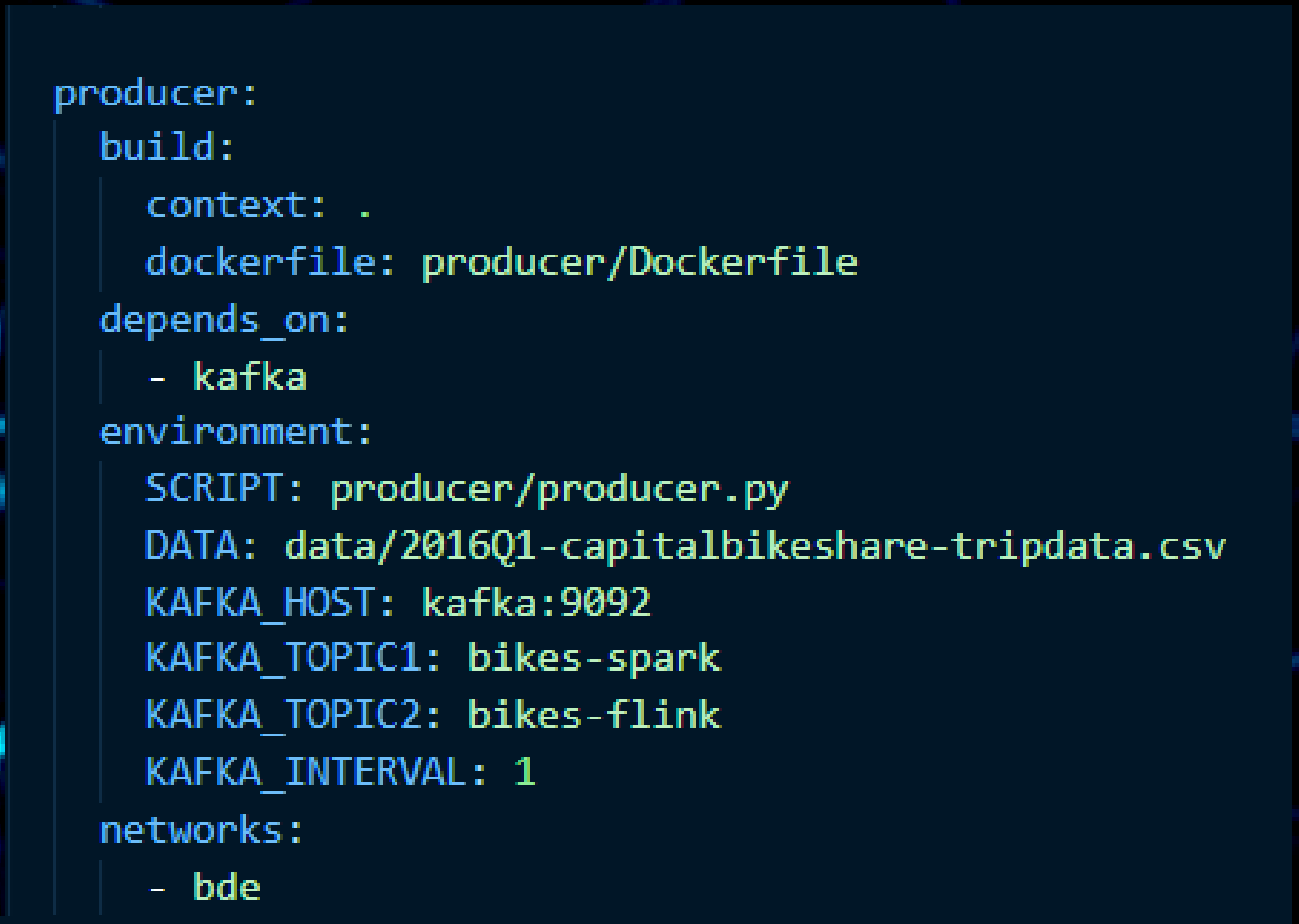
Kreiranje docker-compose fajla



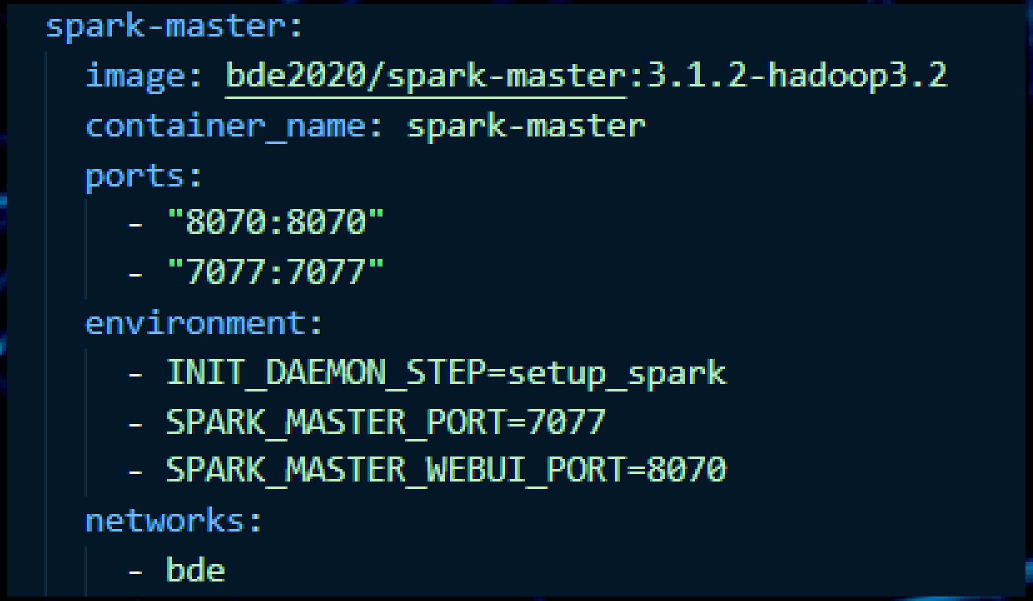
```yaml
version: "3.9"


networks:
  bde:
    external: true
```



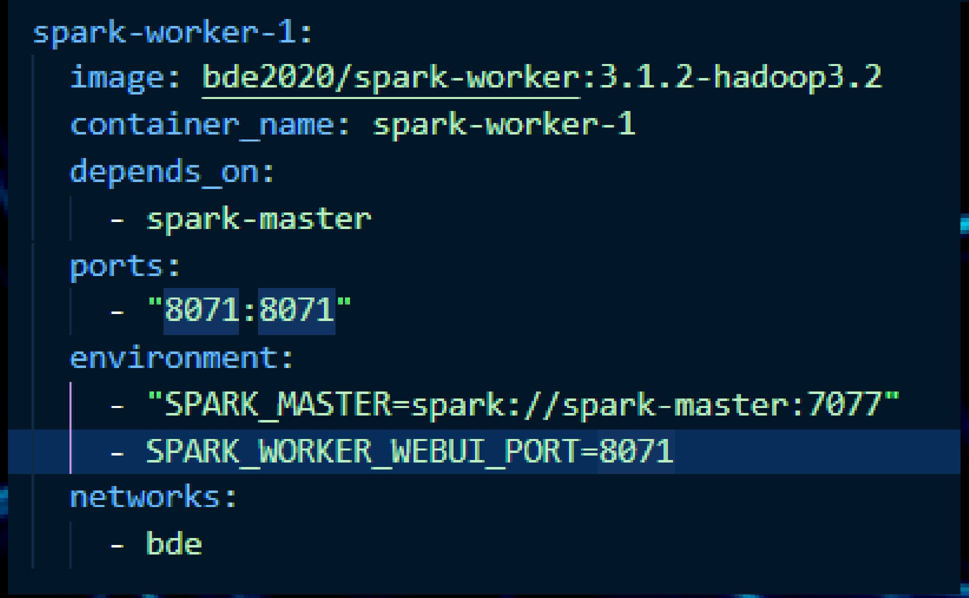
```yaml
services:
  kafka:
    image: wurstmeister/kafka:2.13-2.7.0
    depends_on:
      - zookeeper
    ports:
      - "9091:9091"
    expose:
      - "9092"
    environment:
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka:9092,EXTERNAL://localhost:9091
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
      KAFKA_LISTENERS: INTERNAL://0.0.0.0:9092,EXTERNAL://0.0.0.0:9091
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - bde

  zookeeper:
    image: wurstmeister/zookeeper:latest
    ports:
      - "2181:2181"
    networks:
      - bde
```



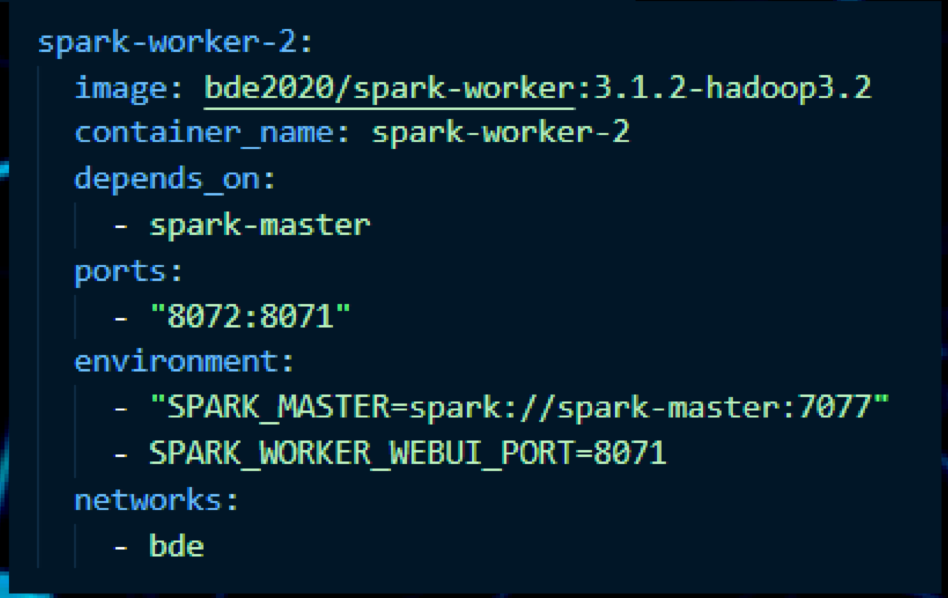
```yaml
producer:
  build:
    context: .
    dockerfile: producer/Dockerfile
  depends_on:
    - kafka
  environment:
    SCRIPT: producer/producer.py
    DATA: data/2016Q1-capitalbikeshare-tripdata.csv
    KAFKA_HOST: kafka:9092
    KAFKA_TOPIC1: bikes-spark
    KAFKA_TOPIC2: bikes-flink
    KAFKA_INTERVAL: 1
  networks:
    - bde
```



```yaml
spark-master:
  image: bde2020/spark-master:3.1.2-hadoop3.2
  container_name: spark-master
  ports:
    - "8070:8070"
    - "7077:7077"
  environment:
    - INIT_DAEMON_STEP=setup_spark
    - SPARK_MASTER_PORT=7077
    - SPARK_MASTER_WEBUI_PORT=8070
  networks:
    - bde
```

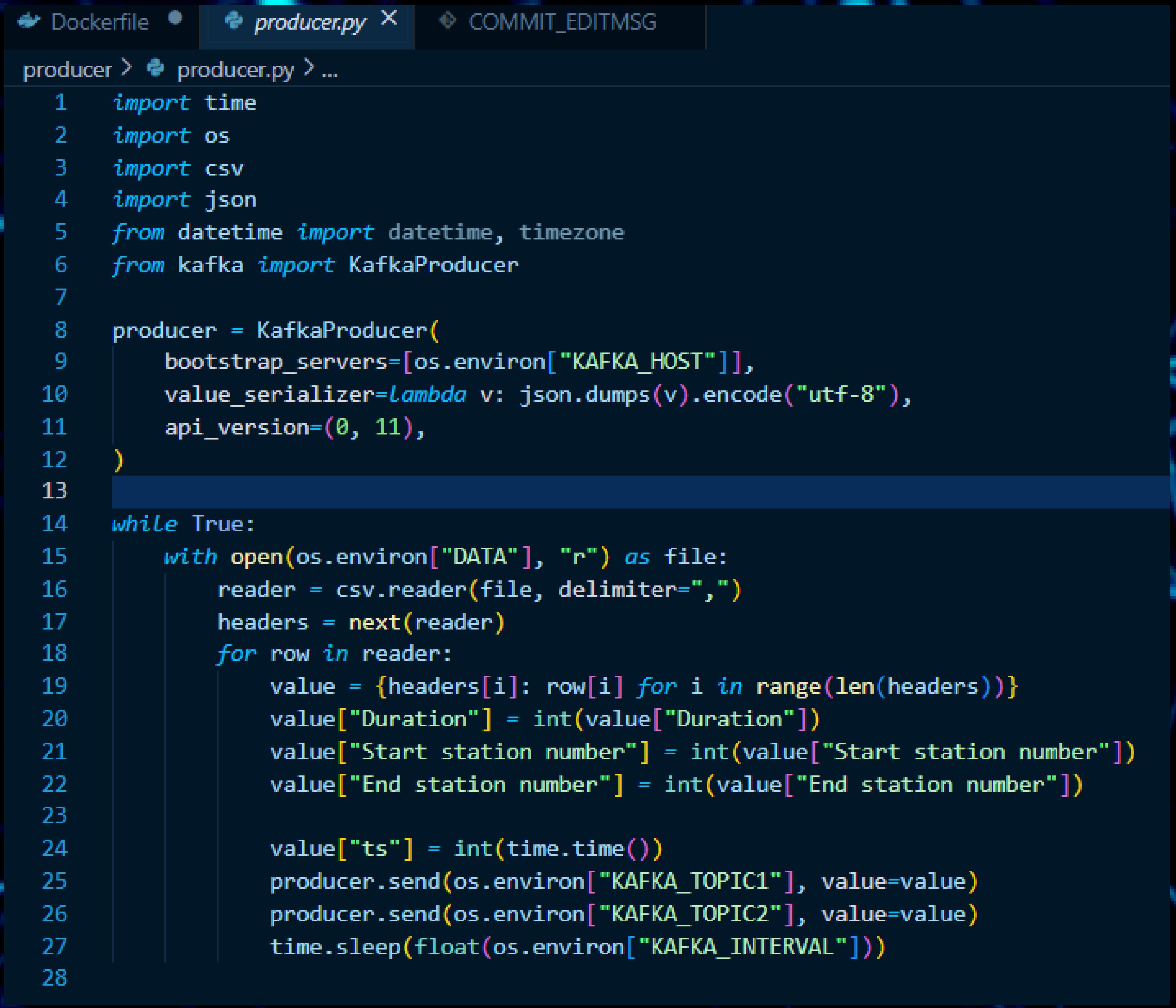


```yaml
spark-worker-1:
  image: bde2020/spark-worker:3.1.2-hadoop3.2
  container_name: spark-worker-1
  depends_on:
    - spark-master
  ports:
    - "8071:8071"
  environment:
    - "SPARK_MASTER=spark://spark-master:7077"
    - SPARK_WORKER_WEBUI_PORT=8071
  networks:
    - bde
```



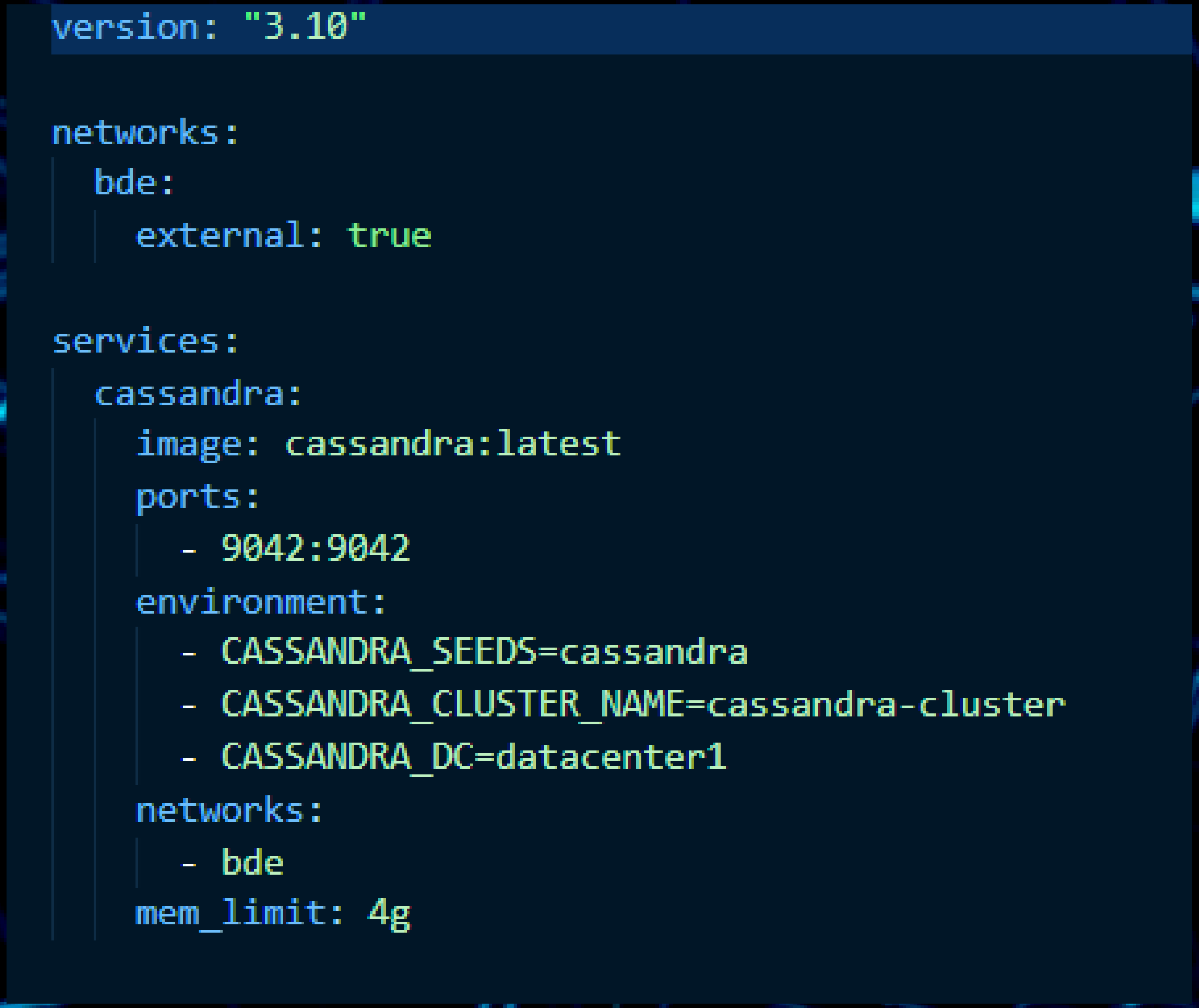
```yaml
spark-worker-2:
  image: bde2020/spark-worker:3.1.2-hadoop3.2
  container_name: spark-worker-2
  depends_on:
    - spark-master
  ports:
    - "8072:8071"
  environment:
    - "SPARK_MASTER=spark://spark-master:7077"
    - SPARK_WORKER_WEBUI_PORT=8071
  networks:
    - bde
```

Kafka producer



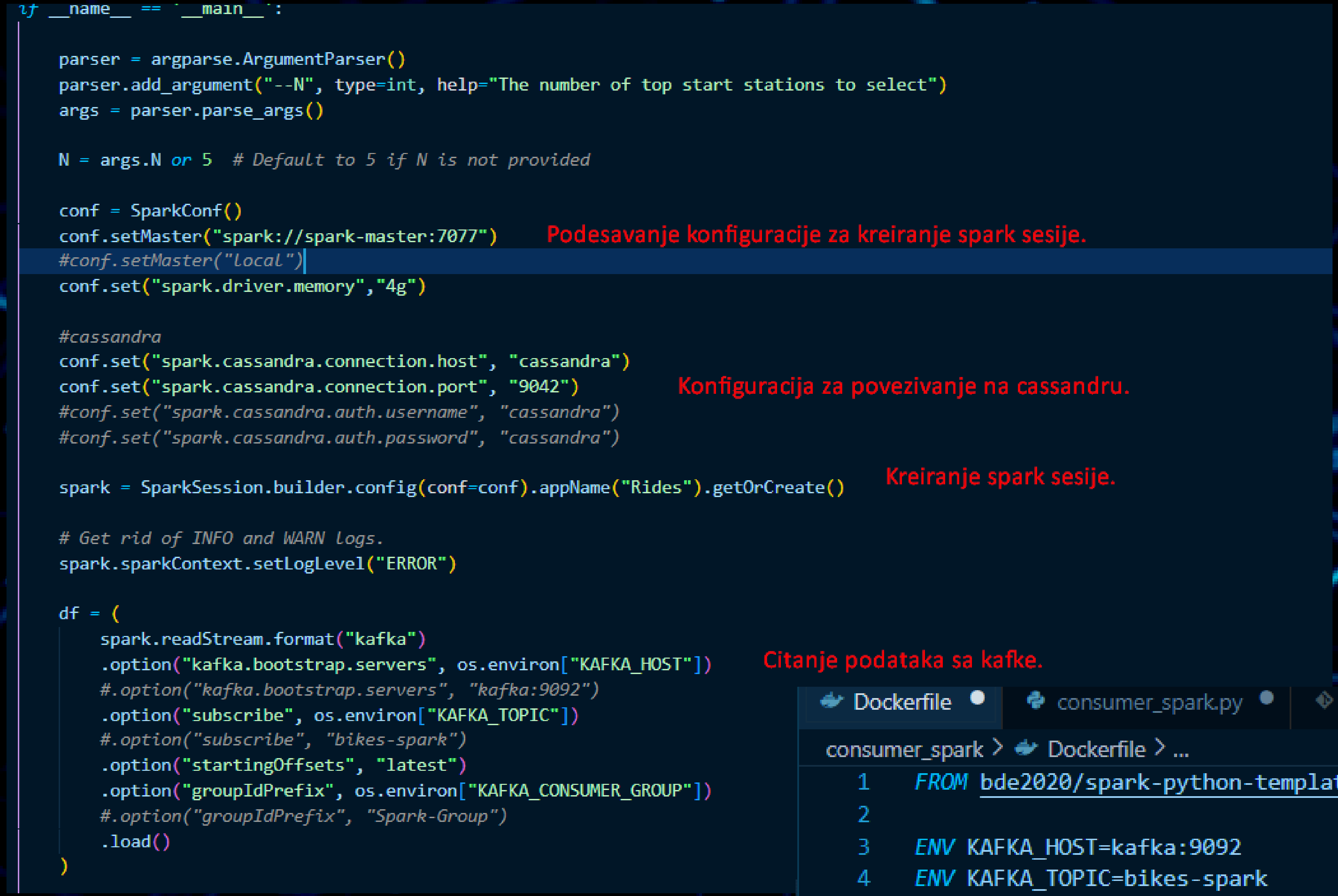
```python
import time
import os
import csv
import json
from datetime import datetime, timezone
from kafka import KafkaProducer

producer = KafkaProducer(
    bootstrap_servers=[os.environ["KAFKA_HOST"]],
    value_serializer=lambda v: json.dumps(v).encode("utf-8"),
    api_version=(0, 11),
)

while True:
    with open(os.environ["DATA"], "r") as file:
        reader = csv.reader(file, delimiter=",")
        headers = next(reader)
        for row in reader:
            value = {headers[i]: row[i] for i in range(len(headers))}
            value["Duration"] = int(value["Duration"])
            value["Start station number"] = int(value["Start station number"])
            value["End station number"] = int(value["End station number"])

            value["ts"] = int(time.time())
            producer.send(os.environ["KAFKA_TOPIC1"], value=value)
            producer.send(os.environ["KAFKA_TOPIC2"], value=value)
            time.sleep(float(os.environ["KAFKA_INTERVAL"]))

```

Cassandra docker-compose



```yaml
version: "3.10"

networks:
  bde:
    external: true

services:
  cassandra:
    image: cassandra:latest
    ports:
      - 9042:9042
    environment:
      - CASSANDRA_SEEDS=cassandra
      - CASSANDRA_CLUSTER_NAME=cassandra-cluster
      - CASSANDRA_DC=datacenter1
    networks:
      - bde
    mem_limit: 4g
```

kafka-spark-cassandra povezivanje



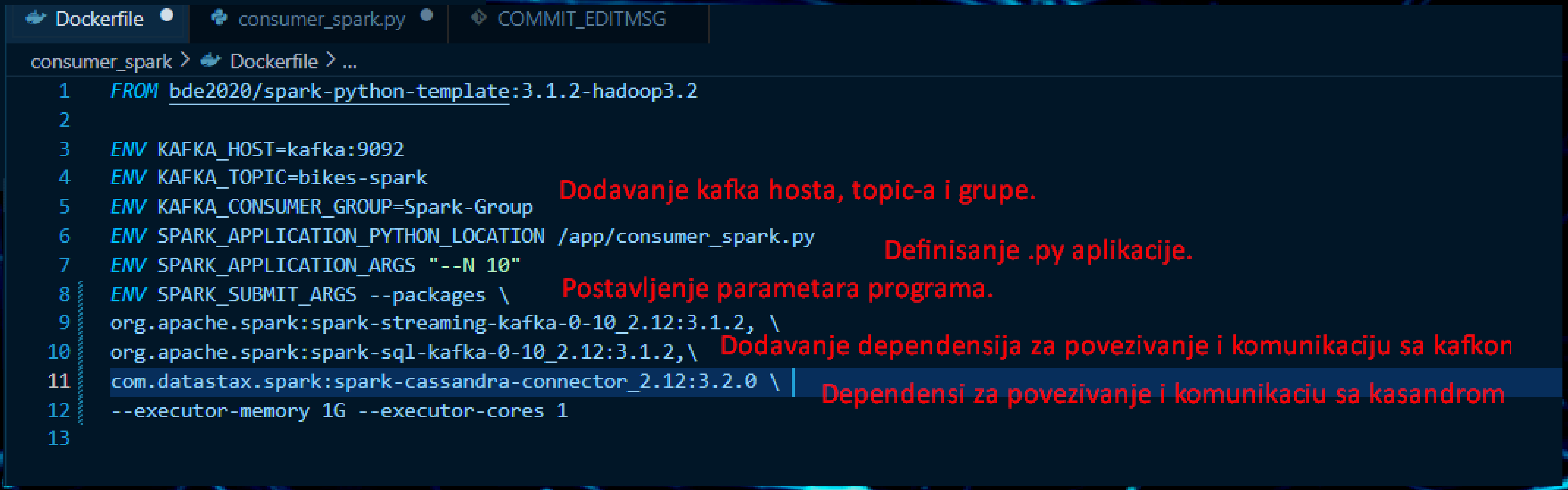
```python
if __name__ == '__main__':

    parser = argparse.ArgumentParser()
    parser.add_argument("--N", type=int, help="The number of top start stations to select")
    args = parser.parse_args()

    N = args.N or 5  # Default to 5 if N is not provided

    conf = SparkConf()
    conf.setMaster("spark://spark-master:7077")
    #conf.setMaster("local")
    conf.set("spark.driver.memory","4g")

    #cassandra
    conf.set("spark.cassandra.connection.host", "cassandra")
    conf.set("spark.cassandra.connection.port", "9042")
    #conf.set("spark.cassandra.auth.username", "cassandra")
    #conf.set("spark.cassandra.auth.password", "cassandra")

    spark = SparkSession.builder.config(conf=conf).appName("Rides").getOrCreate()

    # Get rid of INFO and WARN logs.
    spark.sparkContext.setLogLevel("ERROR")

    df = (
        spark.readStream.format("kafka")
        .option("kafka.bootstrap.servers", os.environ["KAFKA_HOST"])
        #.option("kafka.bootstrap.servers", "kafka:9092")
        .option("subscribe", os.environ["KAFKA_TOPIC"])
        #.option("subscribe", "bikes-spark")
        .option("startingOffsets", "latest")
        .option("groupIdPrefix", os.environ["KAFKA_CONSUMER_GROUP"])
        #.option("groupIdPrefix", "Spark-Group")
        .load()
    )
```

Podesavanje konfiguracije za kreiranje spark sesije.

Konfiguracija za povezivanje na cassandru.

Kreiranje spark sesije.

Citanje podataka sa kafke.



Dockerfile ● | consumer_spark.py ● | COMMIT_EDITMSG

consumer_spark > Dockerfile > ...

```dockerfile
FROM bde2020/spark-python-template:3.1.2-hadoop3.2

ENV KAFKA_HOST=kafka:9092
ENV KAFKA_TOPIC=bikes-spark
ENV KAFKA_CONSUMER_GROUP=Spark-Group
ENV SPARK_APPLICATION_PYTHON_LOCATION /app/consumer_spark.py
ENV SPARK_APPLICATION_ARGS "--N 10"
ENV SPARK_SUBMIT_ARGS --packages \
org.apache.spark:spark-streaming-kafka-0-10_2.12:3.1.2, \
org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2,\
com.datastax.spark:spark-cassandra-connector_2.12:3.2.0 \
--executor-memory 1G --executor-cores 1

```

Dodavanje kafka hosta, topic-a i grupe.

Definisanje .py aplikacije.

Postavljenje parametara programa.

Dodavanje dependensija za povezivanje i komunikaciju sa kafkon

Dependensi za povezivanje i komunikaciu sa kasandrom

Spark aplikacija

```
schema = StructType([
    StructField("Duration", StringType(), False),
    StructField("Start date", TimestampType(), False),
    StructField("End date", TimestampType(), False),
    StructField("Start station number", StringType(), False),
    StructField("Start station", StringType(), False),
    StructField("End station number", StringType(), False),
    StructField("End station", StringType(), False),
    StructField("Bike number", StringType(), False),
    StructField("Member type", StringType(), False),
    StructField("timestamp", TimestampType(), False)
])
```

Priprema podataka za deserijalizaciju is JSON formata.

```
# Parse the "value" field as JSON format and cast the columns to the appropriate data types
parsed_values = df.select("timestamp", from_json(col("value").cast("string"), schema).alias("parsed_values"))
```

Deserijalizacija.

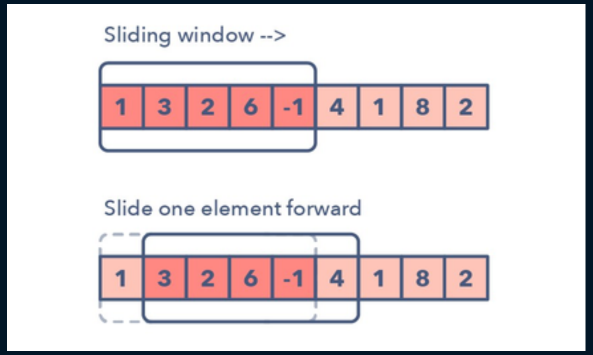
```
durations = parsed_values.selectExpr("timestamp", "parsed_values.Duration AS Duration", "parsed_values[\"Start station\"] as start_station")
```

Izdvajanje atributa od interesa.

```
windowDuration = "60 seconds"  # The length of the window
slideDuration = "5 seconds"  # The sliding interval
```

Kreiranje klizajuceg prozora i grupisanje podataka po polaznim stanicam

```
durationInfo = durations.groupBy(durations.start_station, window(durations.timestamp, windowDuration, slideDuration)).agg(
    avg("Duration").alias("avg_duration"),
    min("Duration").alias("min_duration"),
    mean("Duration").alias("mean_duration"),
    max("Duration").alias("max_duration"),
    count("Start_station").alias("start_station_count"),
    col("window.start").alias("start_date"),
    col("window.end").alias("end_date")
).drop("window")#.dropDuplicates()

durationInfo.printSchema()
```

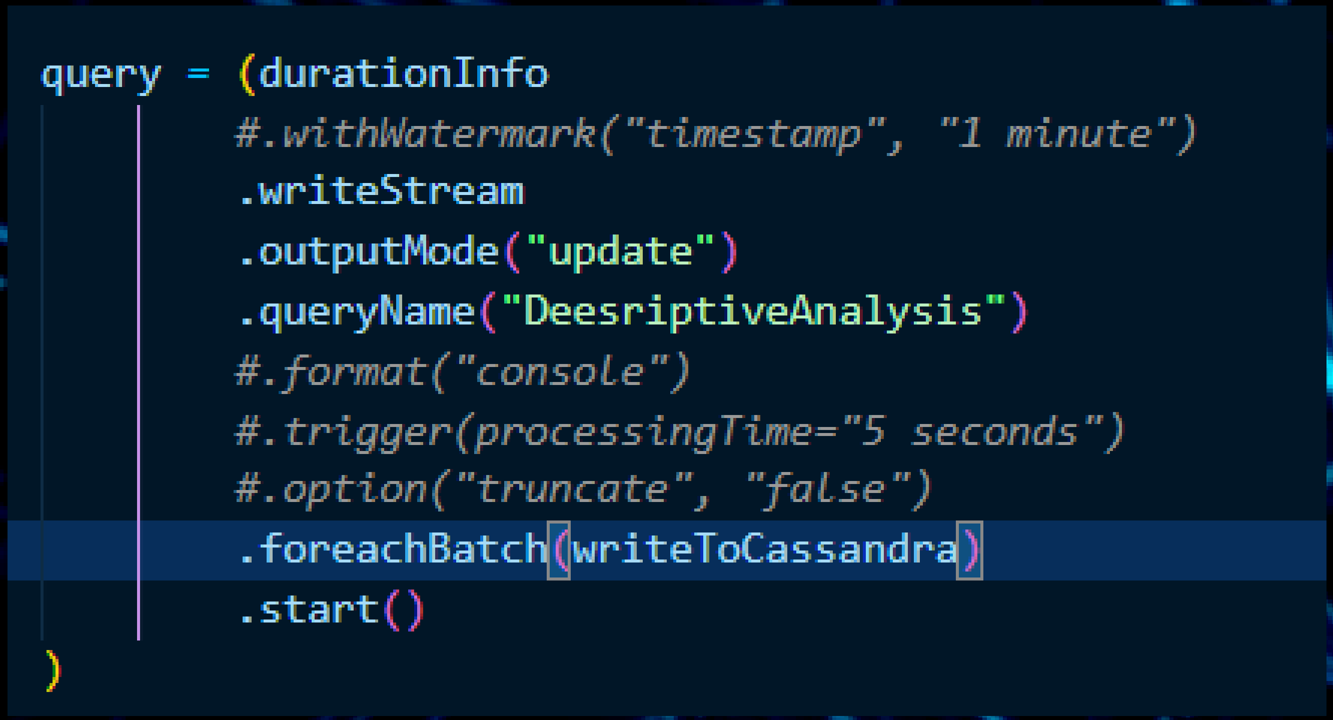


```
popular_start_stations = (durations
    .groupBy(durations.start_station, window(durations.timestamp, windowDuration, windowDuration))
    .agg(count("*").alias("popularity_count"))
    .orderBy(desc("popularity_count"))
    .select(col("start_station"), col("popularity_count"), col("window.start").alias("start_date"), col("window.end").alias("end_date"))
)
```

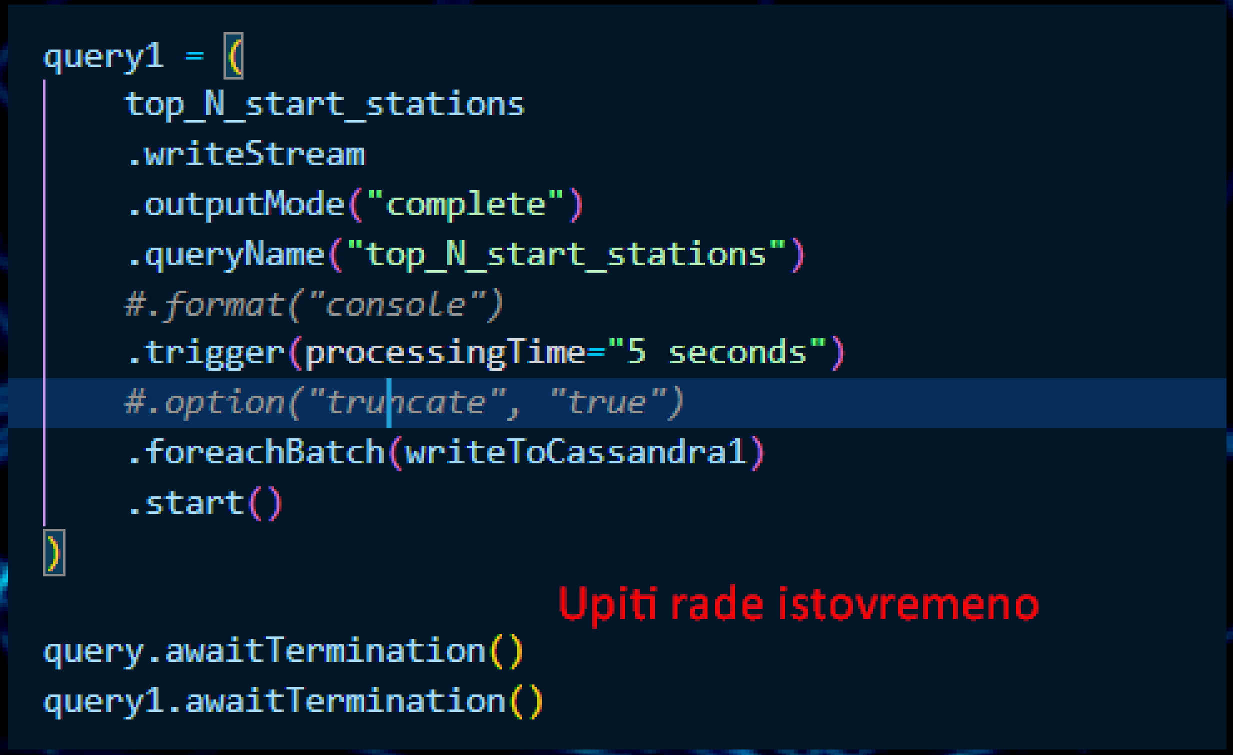
Pronalazanje top N najpopularnijih polaznih stanic

```
top_N_start_stations = popular_start_stations.limit(N)
top_N_start_stations.printSchema();
```

Upis podataka u bazu

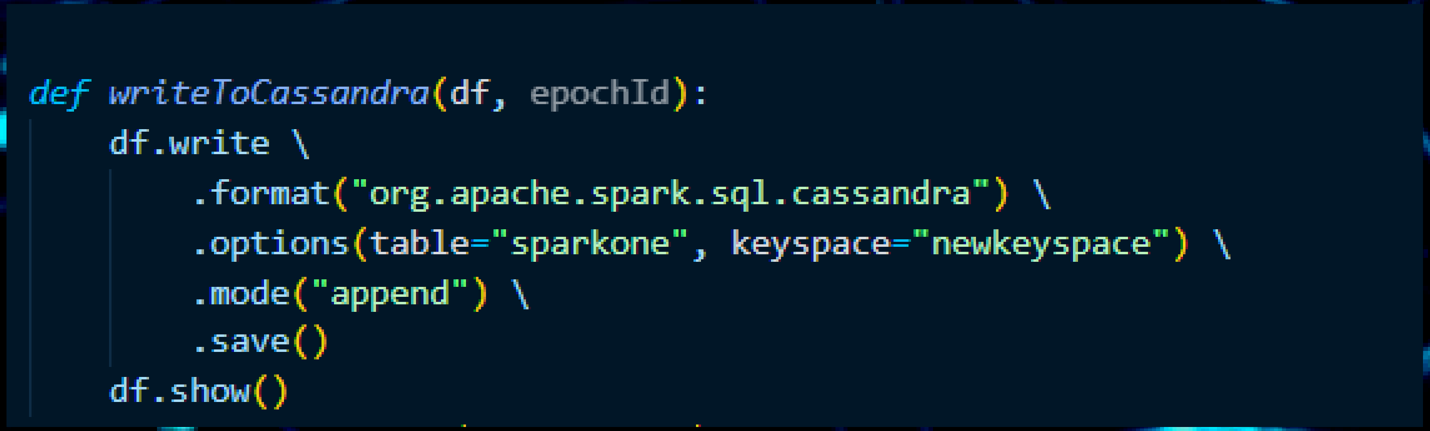


```python
query = (durationInfo
        #.withWatermark("timestamp", "1 minute")
        .writeStream
        .outputMode("update")
        .queryName("DeesriptiveAnalysis")
        #.format("console")
        #.trigger(processingTime="5 seconds")
        #.option("truncate", "false")
        .foreachBatch(writeToCassandra)
        .start()
)
```

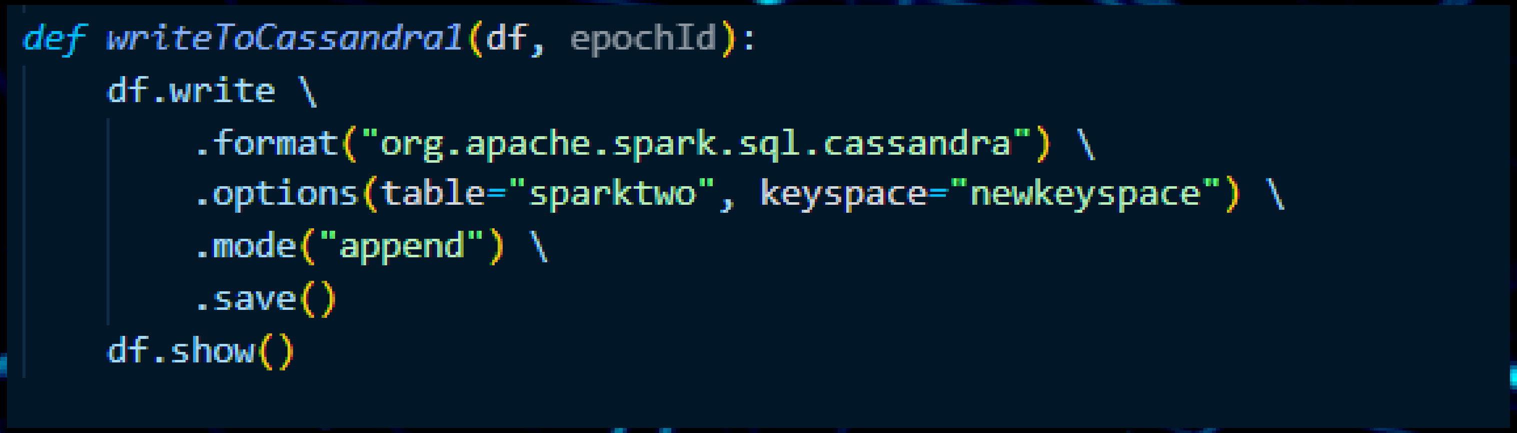


```python
query1 = (
    top_N_start_stations
    .writeStream
    .outputMode("complete")
    .queryName("top_N_start_stations")
    #.format("console")
    .trigger(processingTime="5 seconds")
    #.option("truncate", "true")
    .foreachBatch(writeToCassandra1)
    .start()
)

query.awaitTermination()
query1.awaitTermination()
```

Upiti rade istovremeno

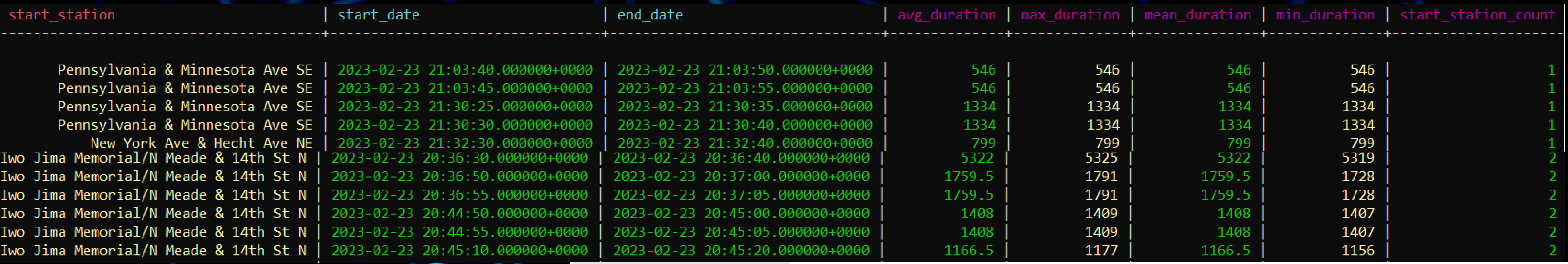


```python
def writeToCassandra(df, epochId):
    df.write \
        .format("org.apache.spark.sql.cassandra") \
        .options(table="sparkone", keyspace="newkeyspace") \
        .mode("append") \
        .save()
    df.show()
```

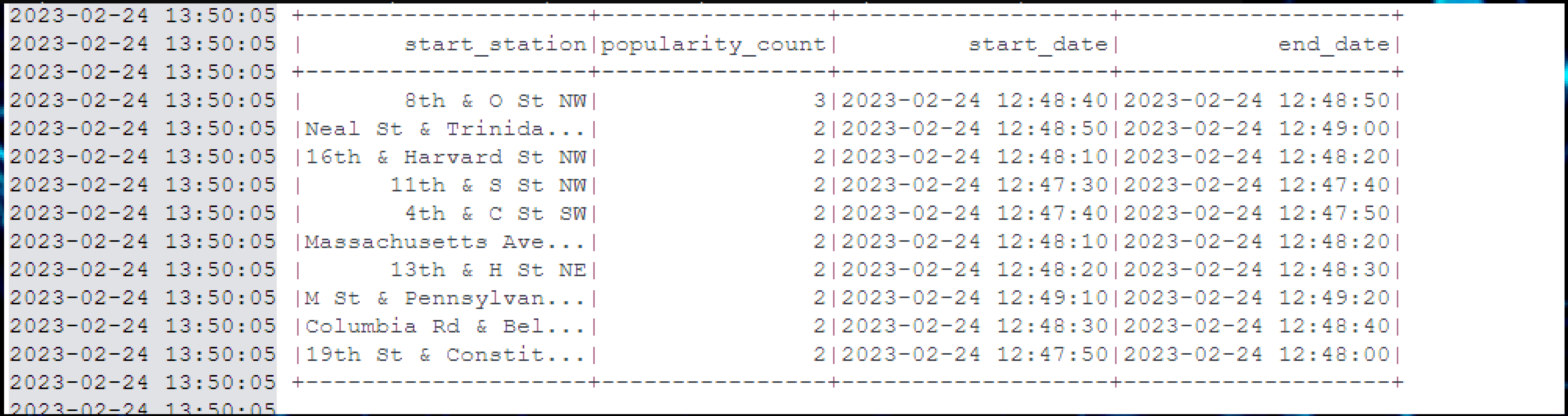


```python
def writeToCassandra1(df, epochId):
    df.write \
        .format("org.apache.spark.sql.cassandra") \
        .options(table="sparktwo", keyspace="newkeyspace") \
        .mode("append") \
        .save()
    df.show()
```

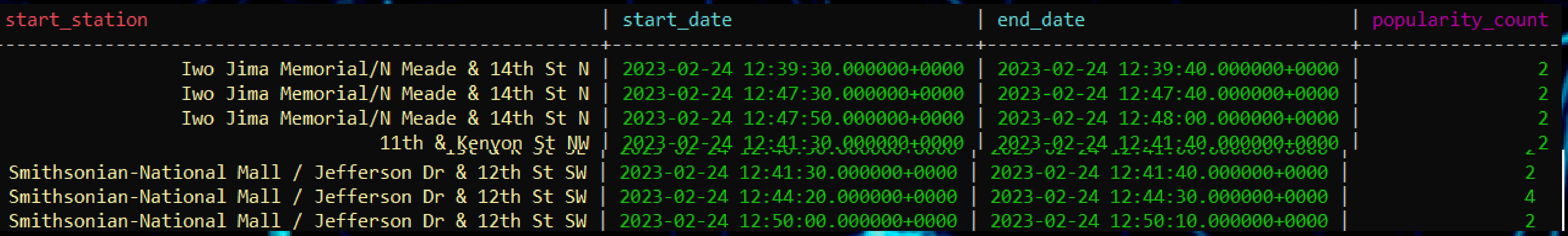
Prikaz rezultata



```
start_station                           | start_date                        | end_date                          | avg_duration | max_duration | mean_duration | min_duration | start_station_count
----------------------------------------+-----------------------------------+-----------------------------------+--------------+--------------+---------------+--------------+--------------------


      Pennsylvania & Minnesota Ave SE | 2023-02-23 21:03:40.000000+0000 | 2023-02-23 21:03:50.000000+0000 |          546 |          546 |           546 |          546 |                   1
      Pennsylvania & Minnesota Ave SE | 2023-02-23 21:03:45.000000+0000 | 2023-02-23 21:03:55.000000+0000 |          546 |          546 |           546 |          546 |                   1
      Pennsylvania & Minnesota Ave SE | 2023-02-23 21:30:25.000000+0000 | 2023-02-23 21:30:35.000000+0000 |         1334 |         1334 |          1334 |         1334 |                   1
      Pennsylvania & Minnesota Ave SE | 2023-02-23 21:30:30.000000+0000 | 2023-02-23 21:30:40.000000+0000 |         1334 |         1334 |          1334 |         1334 |                   1
           New York Ave & Hecht Ave NE | 2023-02-23 21:32:30.000000+0000 | 2023-02-23 21:32:40.000000+0000 |          799 |          799 |           799 |          799 |                   1
Iwo Jima Memorial/N Meade & 14th St N | 2023-02-23 20:36:30.000000+0000 | 2023-02-23 20:36:40.000000+0000 |         5322 |         5325 |          5322 |         5319 |                   2
Iwo Jima Memorial/N Meade & 14th St N | 2023-02-23 20:36:50.000000+0000 | 2023-02-23 20:37:00.000000+0000 |       1759.5 |         1791 |        1759.5 |         1728 |                   2
Iwo Jima Memorial/N Meade & 14th St N | 2023-02-23 20:36:55.000000+0000 | 2023-02-23 20:37:05.000000+0000 |       1759.5 |         1791 |        1759.5 |         1728 |                   2
Iwo Jima Memorial/N Meade & 14th St N | 2023-02-23 20:44:50.000000+0000 | 2023-02-23 20:45:00.000000+0000 |         1408 |         1409 |          1408 |         1407 |                   2
Iwo Jima Memorial/N Meade & 14th St N | 2023-02-23 20:44:55.000000+0000 | 2023-02-23 20:45:05.000000+0000 |         1408 |         1409 |          1408 |         1407 |                   2
Iwo Jima Memorial/N Meade & 14th St N | 2023-02-23 20:45:10.000000+0000 | 2023-02-23 20:45:20.000000+0000 |       1166.5 |         1177 |        1166.5 |         1156 |                   2
```



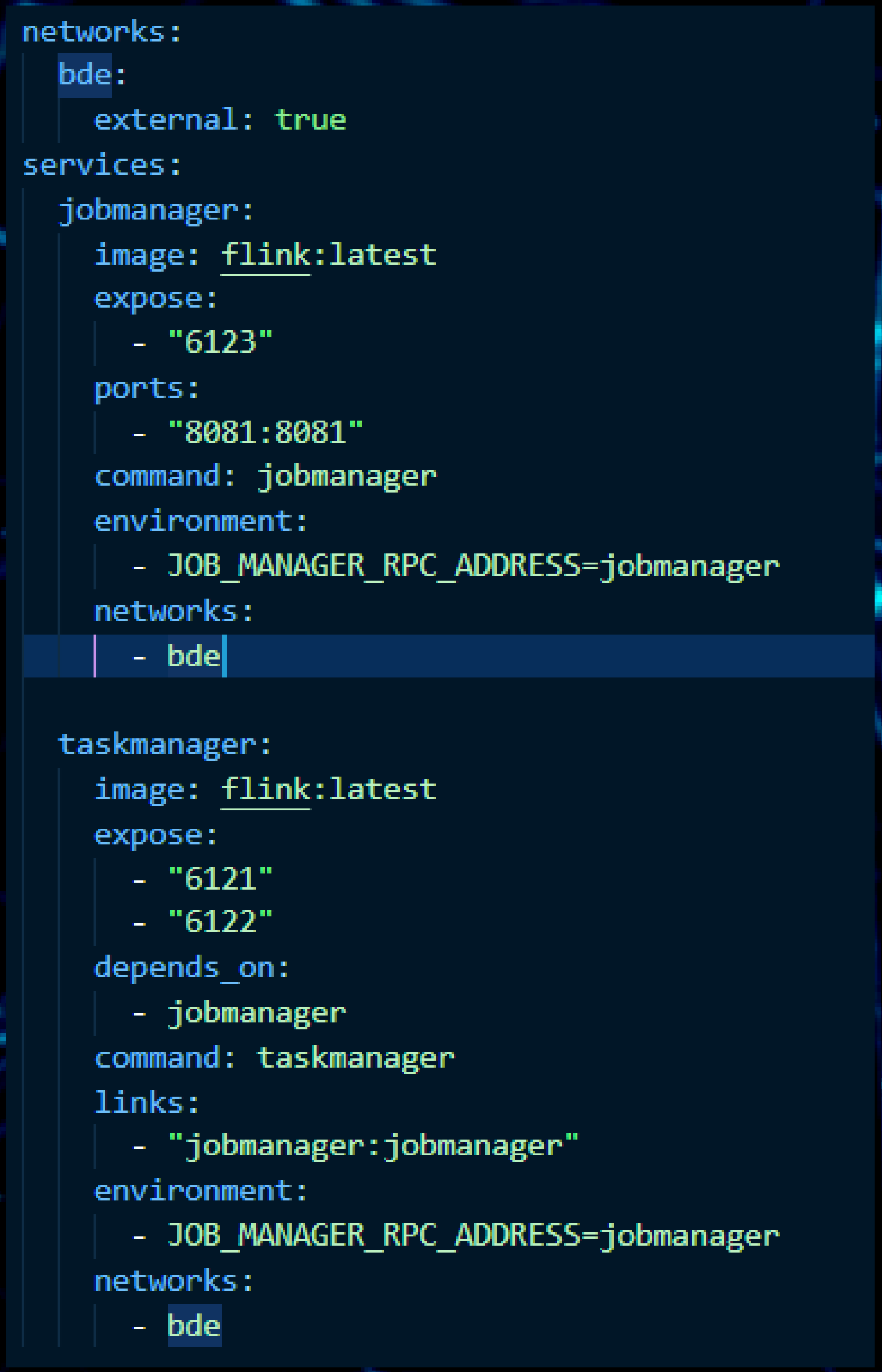
```
2023-02-24 13:50:05 +--------------------+----------------+-------------------+-------------------+
2023-02-24 13:50:05 |       start_station|popularity_count|         start_date|           end_date|
2023-02-24 13:50:05 +--------------------+----------------+-------------------+-------------------+
2023-02-24 13:50:05 |       8th & O St NW|               3|2023-02-24 12:48:40|2023-02-24 12:48:50|
2023-02-24 13:50:05 |Neal St & Trinida...|               2|2023-02-24 12:48:50|2023-02-24 12:49:00|
2023-02-24 13:50:05 |16th & Harvard St NW|               2|2023-02-24 12:48:10|2023-02-24 12:48:20|
2023-02-24 13:50:05 |      11th & S St NW|               2|2023-02-24 12:47:30|2023-02-24 12:47:40|
2023-02-24 13:50:05 |       4th & C St SW|               2|2023-02-24 12:47:40|2023-02-24 12:47:50|
2023-02-24 13:50:05 |Massachusetts Ave...|               2|2023-02-24 12:48:10|2023-02-24 12:48:20|
2023-02-24 13:50:05 |      13th & H St NE|               2|2023-02-24 12:48:20|2023-02-24 12:48:30|
2023-02-24 13:50:05 |M St & Pennsylvan...|               2|2023-02-24 12:49:10|2023-02-24 12:49:20|
2023-02-24 13:50:05 |Columbia Rd & Bel...|               2|2023-02-24 12:48:30|2023-02-24 12:48:40|
2023-02-24 13:50:05 |19th St & Constit...|               2|2023-02-24 12:47:50|2023-02-24 12:48:00|
2023-02-24 13:50:05 +--------------------+----------------+-------------------+-------------------+
2023-02-24 13:50:05
```



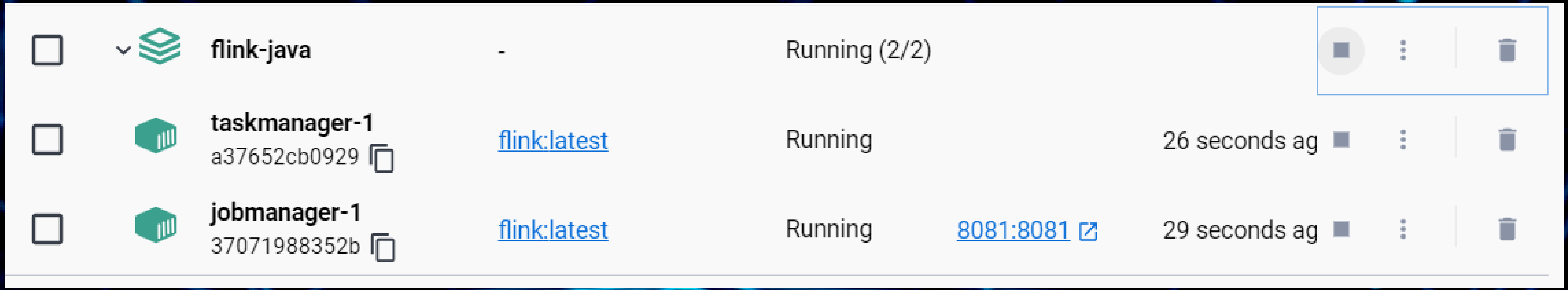
```
start_station                                     | start_date                      | end_date                        | popularity_count
--------------------------------------------------+---------------------------------+---------------------------------+-----------------

             Iwo Jima Memorial/N Meade & 14th St N | 2023-02-24 12:39:30.000000+0000 | 2023-02-24 12:39:40.000000+0000 |                2
             Iwo Jima Memorial/N Meade & 14th St N | 2023-02-24 12:47:30.000000+0000 | 2023-02-24 12:47:40.000000+0000 |                2
             Iwo Jima Memorial/N Meade & 14th St N | 2023-02-24 12:47:50.000000+0000 | 2023-02-24 12:48:00.000000+0000 |                2
                               11th & Kenyon St NW | 2023-02-24 12:41:30.000000+0000 | 2023-02-24 12:41:40.000000+0000 |                2
Smithsonian-National Mall / Jefferson Dr & 12th St SW | 2023-02-24 12:41:30.000000+0000 | 2023-02-24 12:41:40.000000+0000 |              2
Smithsonian-National Mall / Jefferson Dr & 12th St SW | 2023-02-24 12:44:20.000000+0000 | 2023-02-24 12:44:30.000000+0000 |              4
Smithsonian-National Mall / Jefferson Dr & 12th St SW | 2023-02-24 12:50:00.000000+0000 | 2023-02-24 12:50:10.000000+0000 |              2
```

Flink aplikacija

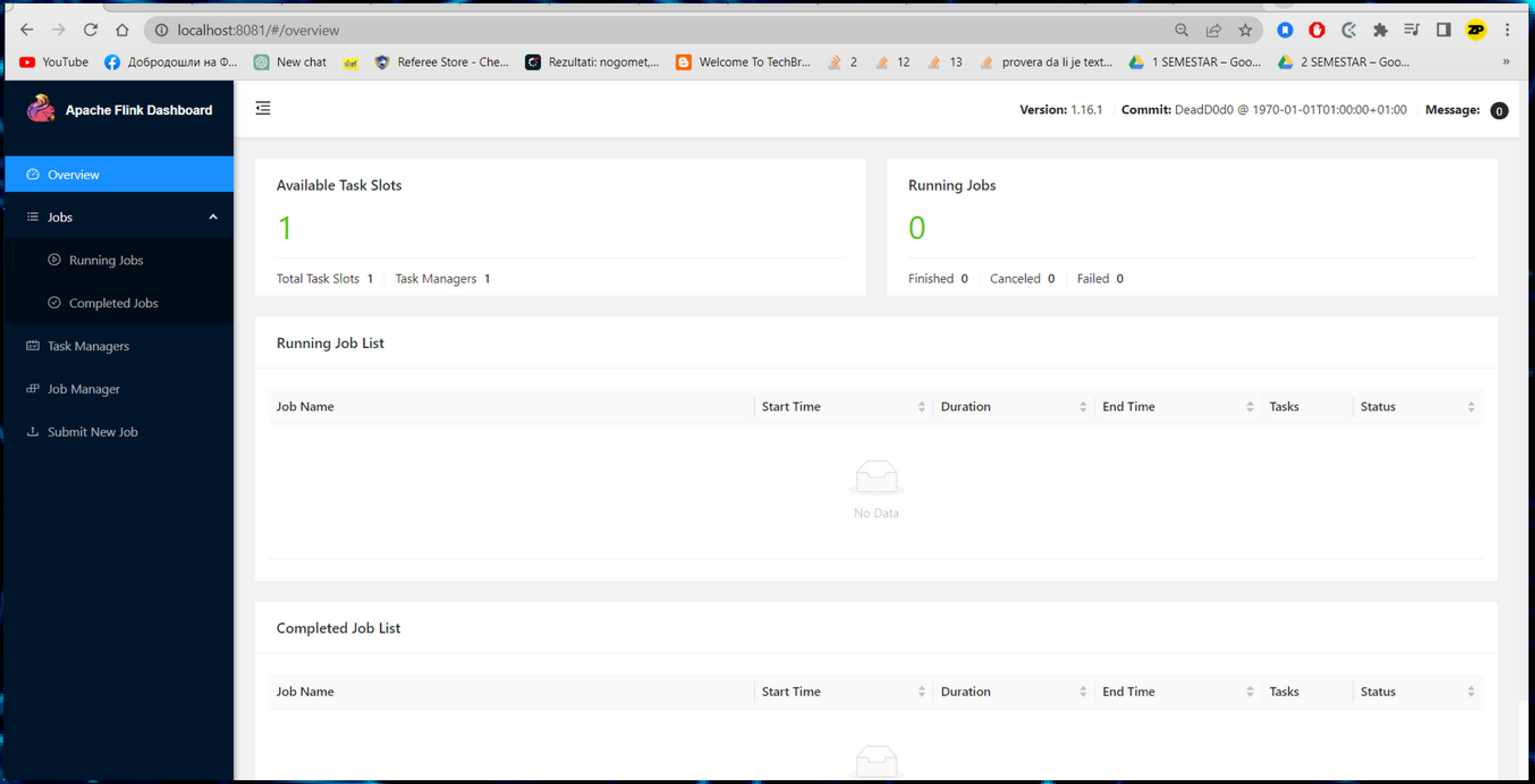
- Java 1.8



```
networks:
  bde:
    external: true
services:
  jobmanager:
    image: flink:latest
    expose:
      - "6123"
    ports:
      - "8081:8081"
    command: jobmanager
    environment:
      - JOB_MANAGER_RPC_ADDRESS=jobmanager
    networks:
      - bde

  taskmanager:
    image: flink:latest
    expose:
      - "6121"
      - "6122"
    depends_on:
      - jobmanager
    command: taskmanager
    links:
      - "jobmanager:jobmanager"
    environment:
      - JOB_MANAGER_RPC_ADDRESS=jobmanager
    networks:
      - bde
```



```
flink-java                          -             Running (2/2)
  taskmanager-1     flink:latest                  Running                 26 seconds ag
  a37652cb0929
  jobmanager-1      flink:latest                  Running   8081:8081     29 seconds ag
  37071988352b
```



localhost:8081/#/overview

YouTube | Добродошли на Ф... | New chat | Referee Store - Che... | Rezultati: nogomet,... | Welcome To TechBr... | 2 | 12 | 13 | provera da li je text... | 1 SEMESTAR – Goo... | 2 SEMESTAR – Goo...

Apache Flink Dashboard

Version: 1.16.1 Commit: DeadD0d0 @ 1970-01-01T01:00:00+01:00 Message: 0

Available Task Slots
1
Total Task Slots 1 Task Managers 1

Running Jobs
0
Finished 0 Canceled 0 Failed 0

Running Job List

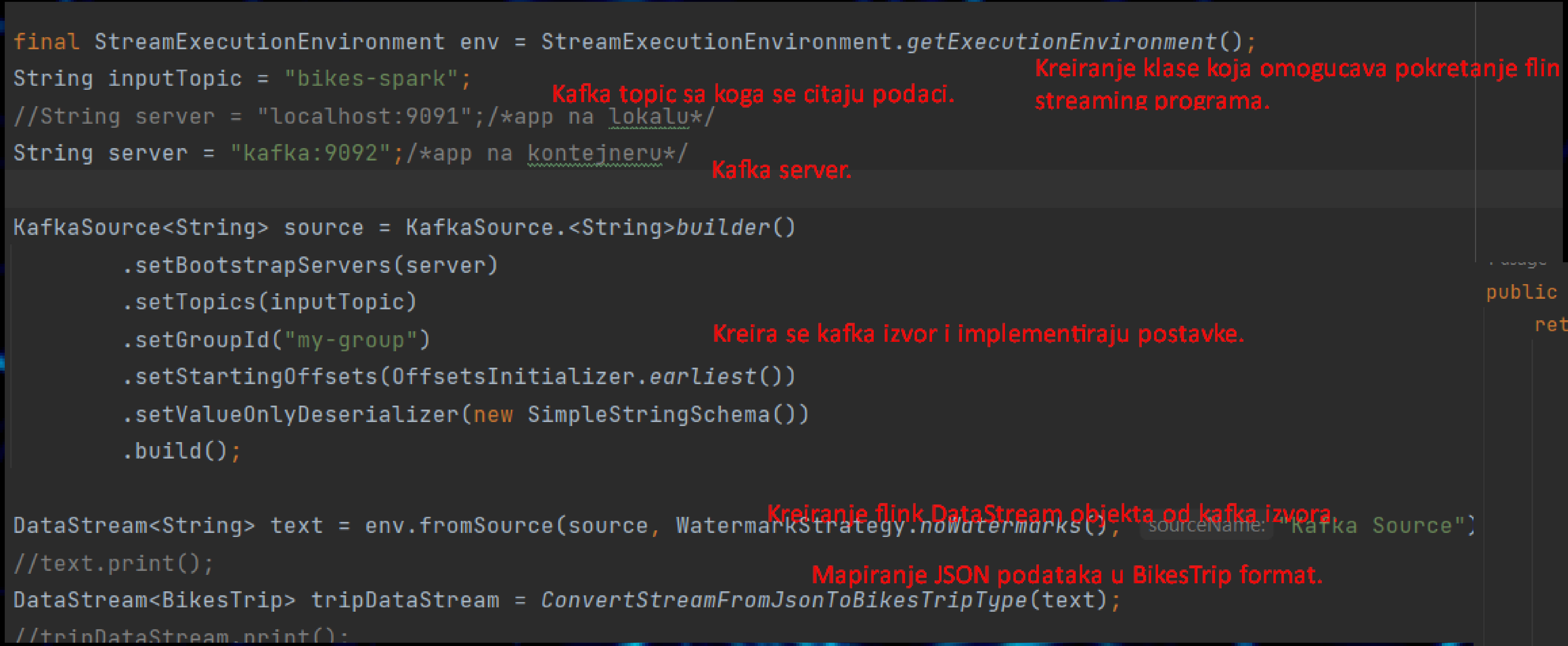
Job Name	Start Time	Duration	End Time	Tasks	Status

No Data

Completed Job List

Job Name	Start Time	Duration	End Time	Tasks	Status

Postavke



```
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
String inputTopic = "bikes-spark";
//String server = "localhost:9091";/*app na lokalu*/
String server = "kafka:9092";/*app na kontejneru*/

KafkaSource<String> source = KafkaSource.<String>builder()
        .setBootstrapServers(server)
        .setTopics(inputTopic)
        .setGroupId("my-group")
        .setStartingOffsets(OffsetsInitializer.earliest())
        .setValueOnlyDeserializer(new SimpleStringSchema())
        .build();

DataStream<String> text = env.fromSource(source, WatermarkStrategy.noWatermarks(), sourceName: "Kafka Source")
//text.print();
DataStream<BikesTrip> tripDataStream = ConvertStreamFromJsonToBikesTripType(text);
//tripDataStream.print();
```

Kreiranje klase koja omogucava pokretanje flin streaming programa.

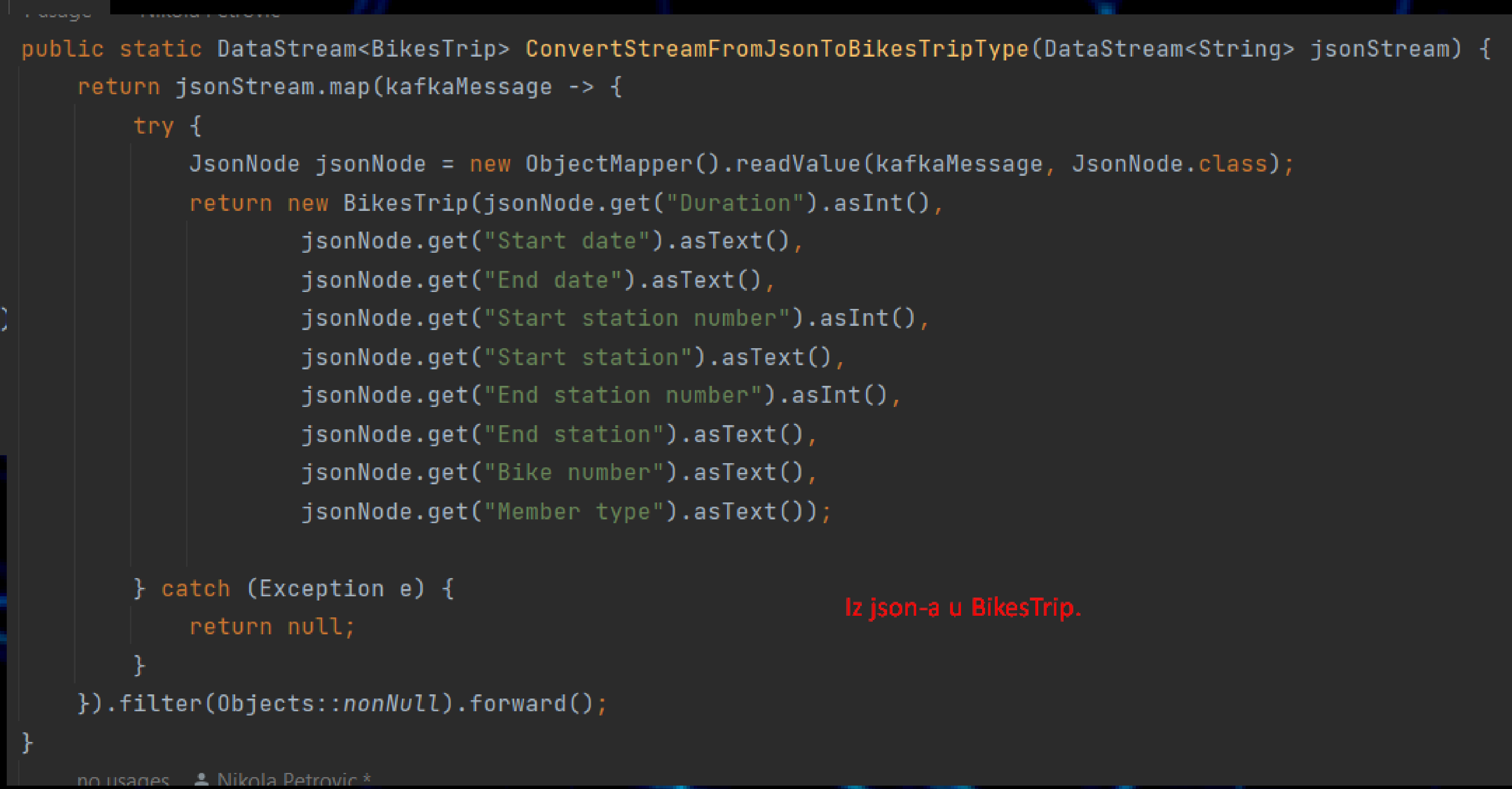
Kafka topic sa koga se citaju podaci.

Kafka server.

Kreira se kafka izvor i implementiraju postavke.

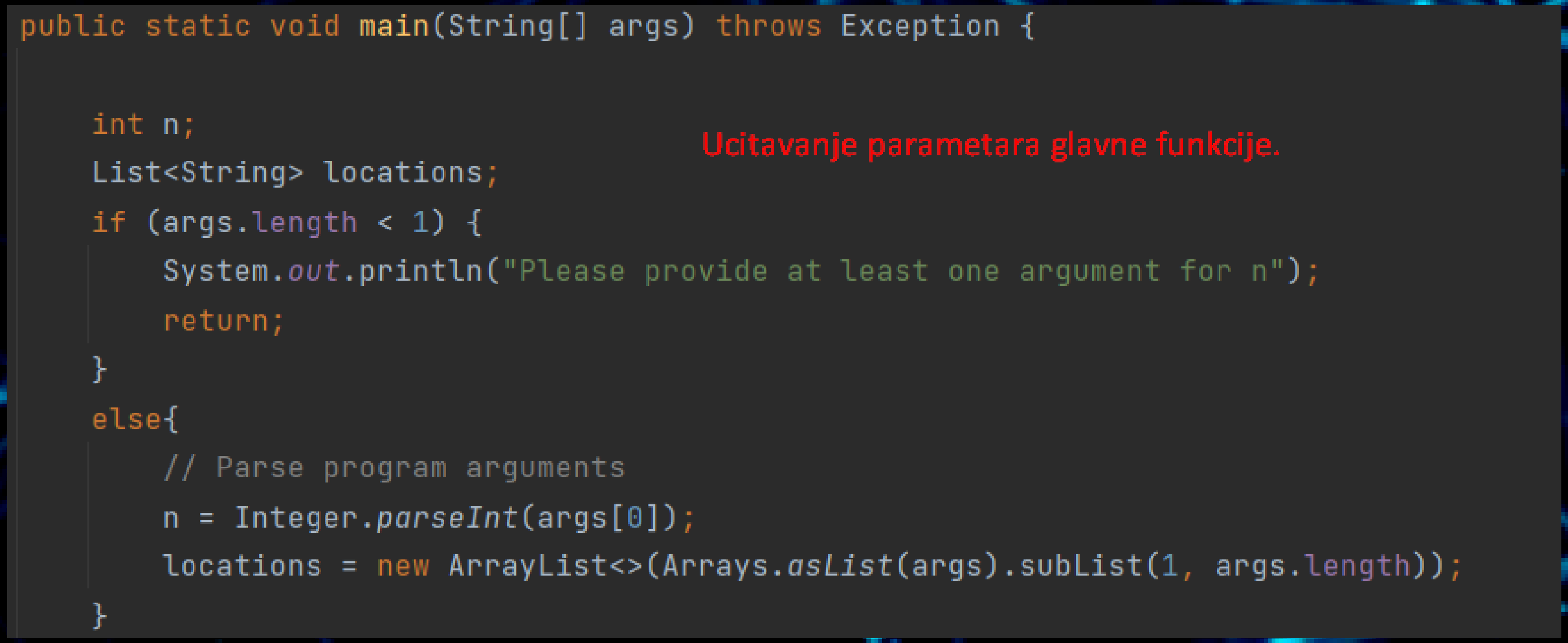
Kreiranje flink DataStream objekta od kafka izvora.

Mapiranje JSON podataka u BikesTrip format.



```
public static DataStream<BikesTrip> ConvertStreamFromJsonToBikesTripType(DataStream<String> jsonStream) {
    return jsonStream.map(kafkaMessage -> {
        try {
            JsonNode jsonNode = new ObjectMapper().readValue(kafkaMessage, JsonNode.class);
            return new BikesTrip(jsonNode.get("Duration").asInt(),
                    jsonNode.get("Start date").asText(),
                    jsonNode.get("End date").asText(),
                    jsonNode.get("Start station number").asInt(),
                    jsonNode.get("Start station").asText(),
                    jsonNode.get("End station number").asInt(),
                    jsonNode.get("End station").asText(),
                    jsonNode.get("Bike number").asText(),
                    jsonNode.get("Member type").asText());

        } catch (Exception e) {
            return null;
        }
    }).filter(Objects::nonNull).forward();
}
no usages   Nikola Petrovic *
```

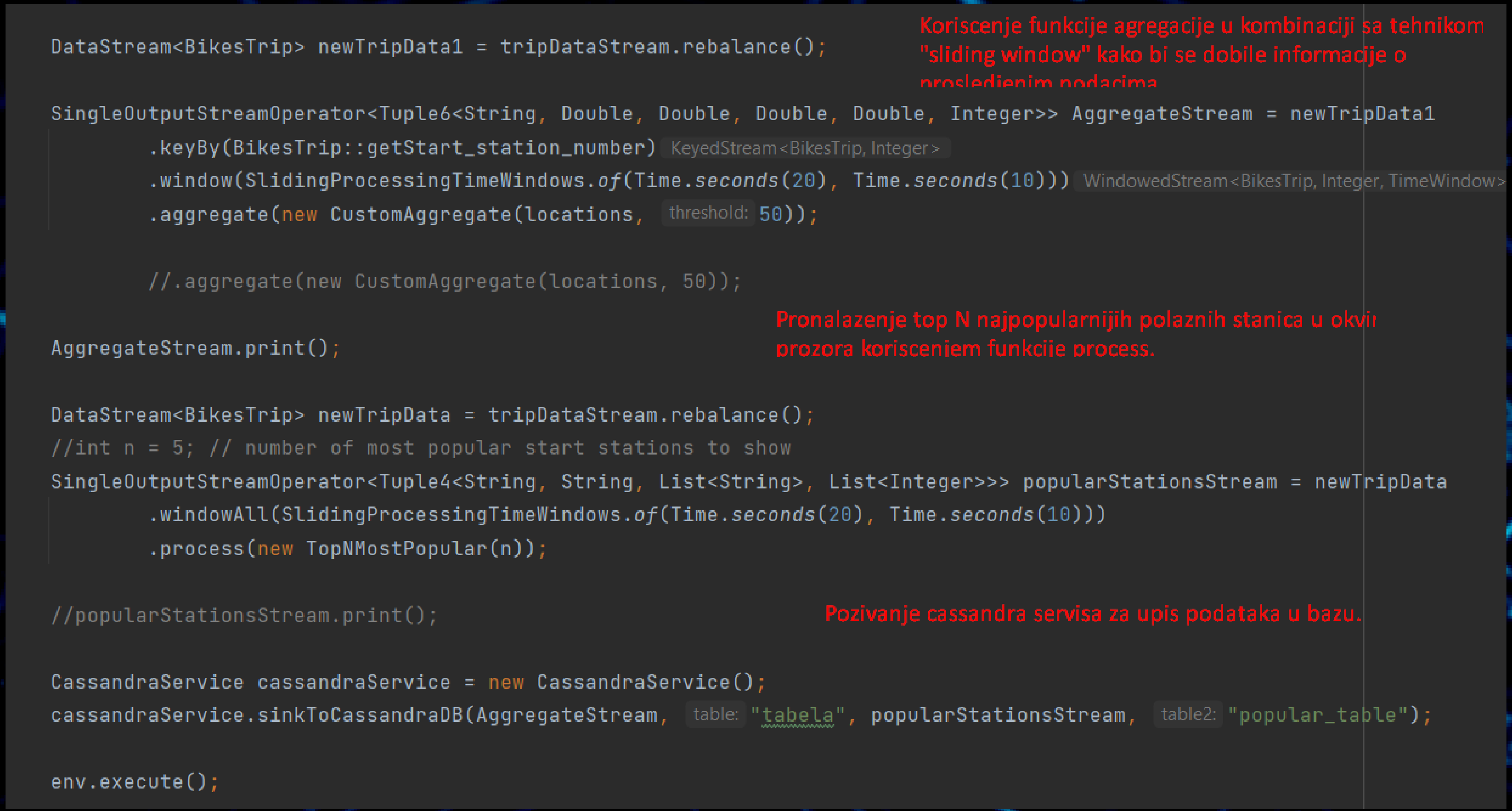
Iz json-a u BikesTrip.



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

    int n;
    List<String> locations;
    if (args.length < 1) {
        System.out.println("Please provide at least one argument for n");
        return;
    }
    else{
        // Parse program arguments
        n = Integer.parseInt(args[0]);
        locations = new ArrayList<>(Arrays.asList(args).subList(1, args.length));
    }
```

Ucitavanje parametara glavne funkcije.

Funkcionalnost



Koriscenje funkcije agregacije u kombinaciji sa tehnikom "sliding window" kako bi se dobile informacije o prosledjenim podacima

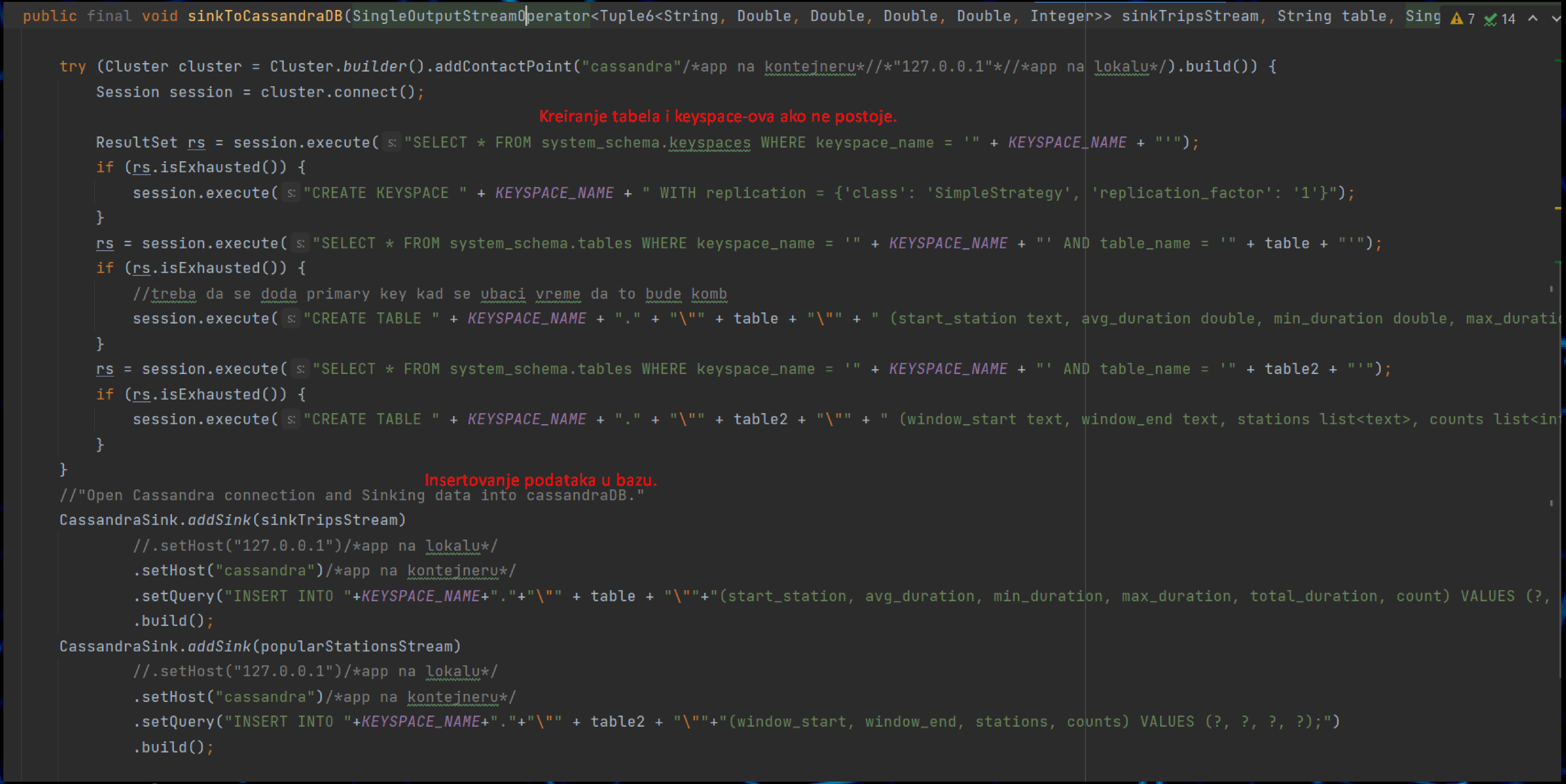
Pronalazenje top N najpopularnijih polaznih stanica u okvir prozora koriscenjem funkcije process.

Pozivanje cassandra servisa za upis podataka u bazu.

```
DataStream<BikesTrip> newTripData1 = tripDataStream.rebalance();

SingleOutputStreamOperator<Tuple6<String, Double, Double, Double, Double, Integer>> AggregateStream = newTripData1
        .keyBy(BikesTrip::getStart_station_number) KeyedStream<BikesTrip, Integer>
        .window(SlidingProcessingTimeWindows.of(Time.seconds(20), Time.seconds(10))) WindowedStream<BikesTrip, Integer, TimeWindow>
        .aggregate(new CustomAggregate(locations,  threshold: 50));

        //.aggregate(new CustomAggregate(locations, 50));

AggregateStream.print();

DataStream<BikesTrip> newTripData = tripDataStream.rebalance();
//int n = 5; // number of most popular start stations to show
SingleOutputStreamOperator<Tuple4<String, String, List<String>, List<Integer>>> popularStationsStream = newTripData
        .windowAll(SlidingProcessingTimeWindows.of(Time.seconds(20), Time.seconds(10)))
        .process(new TopNMostPopular(n));

//popularStationsStream.print();

CassandraService cassandraService = new CassandraService();
cassandraService.sinkToCassandraDB(AggregateStream,  table: "tabela", popularStationsStream,  table2: "popular_table");

env.execute();
```

public class CustomAggregate implements AggregateFunction<BikesTrip, Tuple5<String, Double, Integer, Double, Integer>, Tuple6<String, Double, Double, Double, Double, Integer>>
public class TopNMostPopular extends ProcessAllWindowFunction<BikesTrip, Tuple4<String, String, List<String>, List<Integer>>, TimeWindow>

Upis podataka u bazu



```
public final void sinkToCassandraDB(SingleOutputStreamOperator<Tuple6<String, Double, Double, Double, Double, Integer>> sinkTripsStream, String table, Sing

    try (Cluster cluster = Cluster.builder().addContactPoint("cassandra"/*app na kontejneru*//*"127.0.0.1"*//*app na lokalu*/).build()) {
        Session session = cluster.connect();
```

Kreiranje tabela i keyspace-ova ako ne postoje.

```
        ResultSet rs = session.execute( s: "SELECT * FROM system_schema.keyspaces WHERE keyspace_name = '" + KEYSPACE_NAME + "'");
        if (rs.isExhausted()) {
            session.execute( s: "CREATE KEYSPACE " + KEYSPACE_NAME + " WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}");
        }
        rs = session.execute( s: "SELECT * FROM system_schema.tables WHERE keyspace_name = '" + KEYSPACE_NAME + "' AND table_name = '" + table + "'");
        if (rs.isExhausted()) {
            //treba da se doda primary key kad se ubaci vreme da to bude komb
            session.execute( s: "CREATE TABLE " + KEYSPACE_NAME + "." + "\"" + table + "\"" + " (start_station text, avg_duration double, min_duration double, max_durati
        }
        rs = session.execute( s: "SELECT * FROM system_schema.tables WHERE keyspace_name = '" + KEYSPACE_NAME + "' AND table_name = '" + table2 + "'");
        if (rs.isExhausted()) {
            session.execute( s: "CREATE TABLE " + KEYSPACE_NAME + "." + "\"" + table2 + "\"" + " (window_start text, window_end text, stations list<text>, counts list<in
        }
    }
```

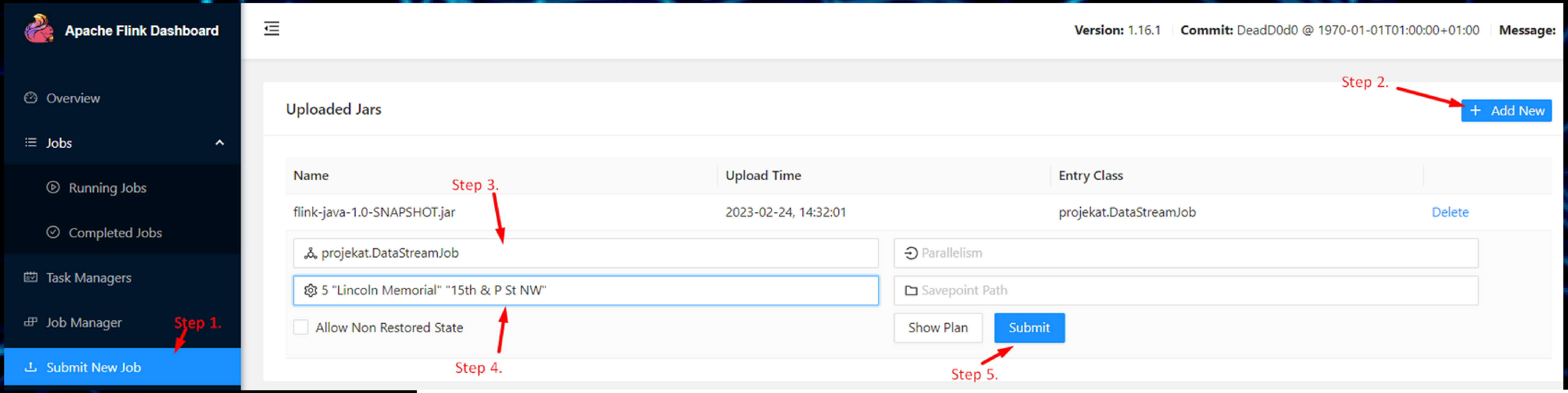
Insertovanje podataka u bazu.

```
    //"Open Cassandra connection and Sinking data into cassandraDB."
    CassandraSink.addSink(sinkTripsStream)
            //.setHost("127.0.0.1")/*app na lokalu*/
            .setHost("cassandra")/*app na kontejneru*/
            .setQuery("INSERT INTO "+KEYSPACE_NAME+"."+"\"" + table + "\""+"(start_station, avg_duration, min_duration, max_duration, total_duration, count) VALUES (?,
            .build();
    CassandraSink.addSink(popularStationsStream)
            //.setHost("127.0.0.1")/*app na lokalu*/
            .setHost("cassandra")/*app na kontejneru*/
            .setQuery("INSERT INTO "+KEYSPACE_NAME+"."+"\"" + table2 + "\""+"(window_start, window_end, stations, counts) VALUES (?, ?, ?, ?);")
            .build();
```

Pokretanje aplikacije

Aplikacija moze biti pokrenuta na lokalu i na kontejnerima.
Prvo je potrebno proslediti parametre glavne funkcije, sto su u ovom slucaju broj N i polazna stanice od interesa.
Da bi aplikacija bila pokrenuta na kontejnerima, potrebno je prvo izvrsiti komandu **mvn clean package** kako bi kreirali **.jar** fajl aplikacije koji se postavlja na web UI jobmanager-a.

```
C:\Users\Nikola Petrovic\Desktop\bigdata\kafka\docker-kafka-spark-flink-py-java\flink-java>mvn clean package
```



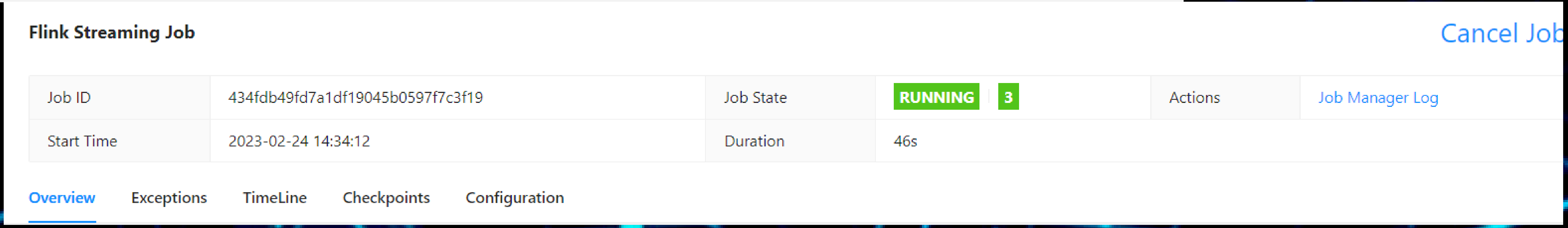
Version: 1.16.1 Commit: DeadD0d0 @ 1970-01-01T01:00:00+01:00 Message:

Uploaded Jars

Name	Upload Time	Entry Class	
flink-java-1.0-SNAPSHOT.jar	2023-02-24, 14:32:01	projekat.DataStreamJob	Delete

projekat.DataStreamJob

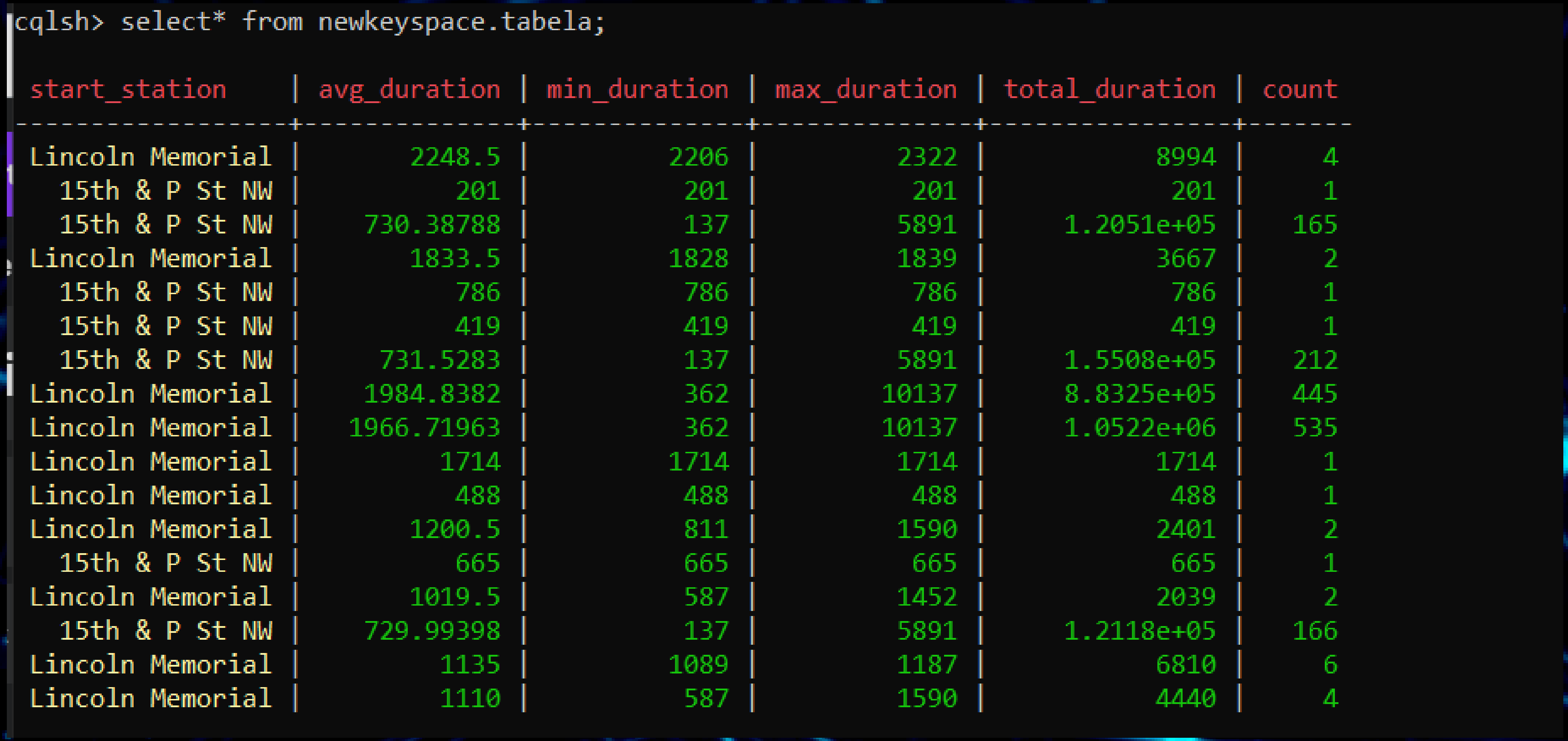
5 "Lincoln Memorial" "15th & P St NW"



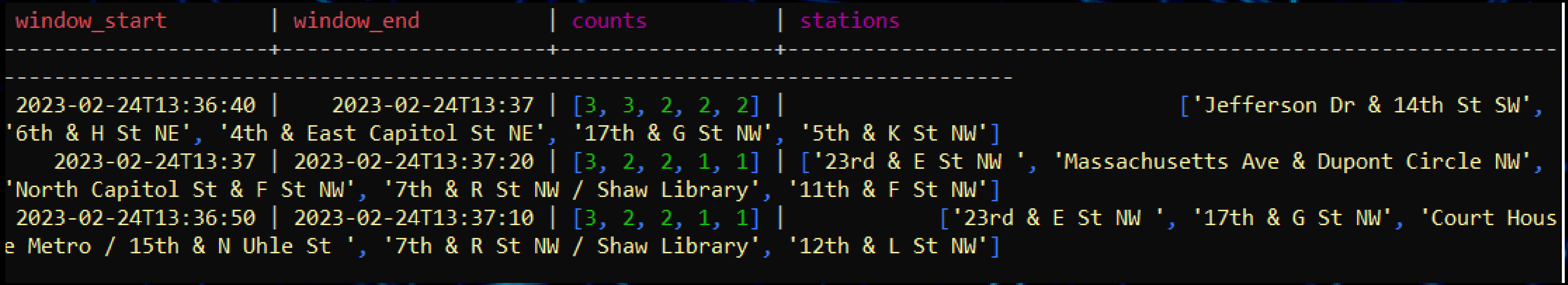
Flink Streaming Job

Job ID	434fdb49fd7a1df19045b0597f7c3f19	Job State	RUNNING 3	Actions	Job Manager Log
Start Time	2023-02-24 14:34:12	Duration	46s		

Rezultati



```
cqlsh> select* from newkeyspace.tabela;

 start_station   | avg_duration | min_duration | max_duration | total_duration | count
-----------------+--------------+--------------+--------------+----------------+-------
 Lincoln Memorial |       2248.5 |         2206 |         2322 |           8994 |     4
   15th & P St NW |          201 |          201 |          201 |            201 |     1
   15th & P St NW |    730.38788 |          137 |         5891 |     1.2051e+05 |   165
 Lincoln Memorial |       1833.5 |         1828 |         1839 |           3667 |     2
   15th & P St NW |          786 |          786 |          786 |            786 |     1
   15th & P St NW |          419 |          419 |          419 |            419 |     1
   15th & P St NW |     731.5283 |          137 |         5891 |     1.5508e+05 |   212
 Lincoln Memorial |    1984.8382 |          362 |        10137 |     8.8325e+05 |   445
 Lincoln Memorial |   1966.71963 |          362 |        10137 |     1.0522e+06 |   535
 Lincoln Memorial |         1714 |         1714 |         1714 |           1714 |     1
 Lincoln Memorial |          488 |          488 |          488 |            488 |     1
 Lincoln Memorial |       1200.5 |          811 |         1590 |           2401 |     2
   15th & P St NW |          665 |          665 |          665 |            665 |     1
 Lincoln Memorial |       1019.5 |          587 |         1452 |           2039 |     2
   15th & P St NW |    729.99398 |          137 |         5891 |     1.2118e+05 |   166
 Lincoln Memorial |         1135 |         1089 |         1187 |           6810 |     6
 Lincoln Memorial |         1110 |          587 |         1590 |           4440 |     4
```



```
 window_start        | window_end          | counts          | stations
---------------------+---------------------+-----------------+-------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
 2023-02-24T13:36:40 |    2023-02-24T13:37 | [3, 3, 2, 2, 2] |                             ['Jefferson Dr & 14th St SW',
'6th & H St NE', '4th & East Capitol St NE', '17th & G St NW', '5th & K St NW']
    2023-02-24T13:37 | 2023-02-24T13:37:20 | [3, 2, 2, 1, 1] | ['23rd & E St NW ', 'Massachusetts Ave & Dupont Circle NW',
'North Capitol St & F St NW', '7th & R St NW / Shaw Library', '11th & F St NW']
 2023-02-24T13:36:50 | 2023-02-24T13:37:10 | [3, 2, 2, 1, 1] |            ['23rd & E St NW ', '17th & G St NW', 'Court Hous
e Metro / 15th & N Uhle St ', '7th & R St NW / Shaw Library', '12th & L St NW']
```

Hvala na Paznji!