SISTEMI ZA OBRADU I ANALIZU VELIKE KOLIČINE PODATAKA - Projekat 2

Filip Trajković 1574

KORIŠĆENE TEHNOLOGIJE

Apache Hadoop

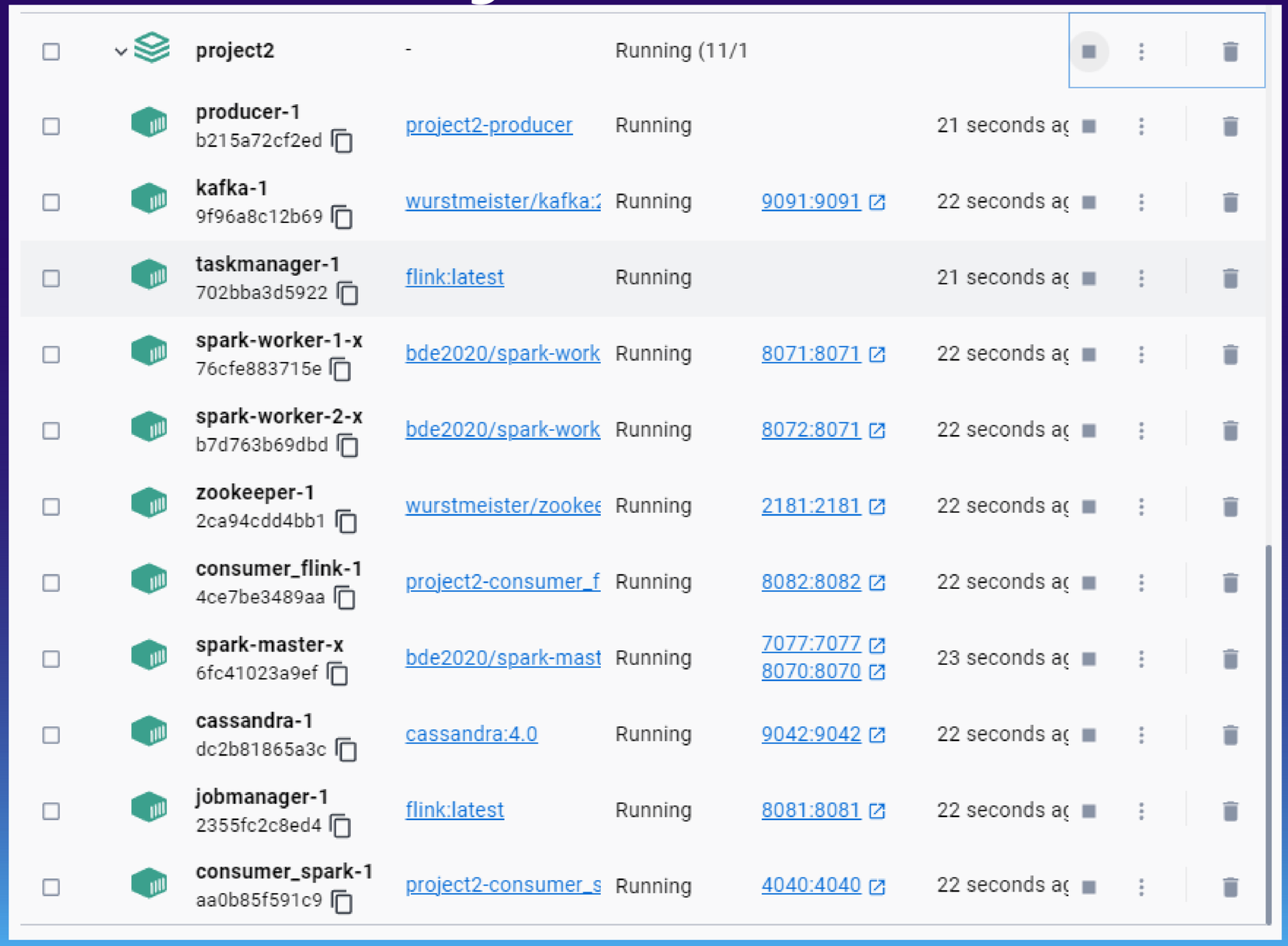
Apache Spark

Apache Flink

Flink

Cassandra

Spisak kontejnera



Izgled Producer skripte

```
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["lat"] = float(value["lat"])
            value["lon"] = float(value["lon"])
            value["alt"] = float(value["alt"])
            value["user"] = int(value["user"])
            value["ts"] = int(time.time())
            value["year"] = int(value["year"])
            value["month"] = int(value["month"])
            value["day"] = int(value["day"])
            value["hour"] = int(value["hour"])
            value["minute"] = int(value["minute"])
            value["second"] = int(value["second"])
            print(value)
            producer.send(os.environ["KAFKA_TOPIC"], value=value)
            time.sleep(float(os.environ["KAFKA_INTERVAL"]))

```

Izgled Cassandra tabela

```
# cqlsh
Connected to cloudinfra at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.7 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> DESC keyspaces;

locations_db  system_auth         system_schema  system_views
system        system_distributed  system_traces  system_virtual_schema

cqlsh> USE locations_db;
cqlsh:locations_db> DESC tables;

flinkgeolocation  geolocation

cqlsh:locations_db> SELECT * FROM geolocation;

 user | end                       | maxvalue  | meanvalue          | minvalue  | start                     | times_data_sent
------+---------------------------+-----------+--------------------+-----------+---------------------------+-----------------
  163 | 2023-03-30 22:56:00+0000 | 190.28871 | 190.28871154785156 | 190.28871 | 2023-03-30 22:55:50+0000 |               2
  154 | 2023-02-15 16:53:00+0000 |   16.4042 | 16.404199600219727 |   16.4042 | 2023-02-15 16:52:50+0000 |               8
   21 | 2023-02-15 16:38:00+0000 | 1036.7454 |  1033.464564732143 | 1030.1837 | 2023-02-15 16:37:50+0000 |               7
  111 | 2023-03-30 23:00:55+0000 | 160.76115 |  158.5739288330078 | 154.19948 | 2023-03-30 23:00:45+0000 |               3
   97 | 2023-03-24 17:16:40+0000 | 249.34383 |  242.7821502685547 | 236.22047 | 2023-03-24 17:16:30+0000 |               2
   86 | 2023-02-15 16:38:10+0000 | 187.00787 | 187.00787353515625 | 187.00787 | 2023-02-15 16:38:00+0000 |               2
  118 | 2023-02-15 17:33:30+0000 | 170.60367 |  158.7926483154297 | 141.07611 | 2023-02-15 17:33:20+0000 |               5
  161 | 2023-03-30 22:58:55+0000 | 269.02887 | 260.27996826171875 | 255.90552 | 2023-03-30 22:58:45+0000 |               3
  128 | 2023-03-30 23:02:45+0000 |  147.6378 | 142.71653747558594 | 137.79527 | 2023-03-30 23:02:35+0000 |               2

(9 rows)
cqlsh:locations_db> SELECT * FROM flinkgeolocation;

 user | count | maxvalue         | meanvalue        | minvalue
------+-------+------------------+------------------+------------------
  163 |   3.0 | 160.761154855643 | 160.761154855643 | 160.761154855643
  111 |   2.0 | 160.761154855643 | 160.761154855643 | 160.761154855643
  161 |   1.0 | 255.905511811024 | 255.905511811024 | 255.905511811024
  128 |   2.0 | 187.007874015748 | 180.446194225722 | 173.884514435696

(4 rows)
cqlsh:locations_db>
```

Streaming aplikacija

Shell skripta za pokretanje kontejnera

```
#! /bin/bash

docker container rm SparkStreamingApp

docker build --rm -t bde/spark-app .

docker run --name SparkStreamingApp --net project2_default -p 4042:4042 bde/spark-app

```


Streaming aplikacija

Dockerfile

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

ENV KAFKA_HOST=kafka:9092
ENV KAFKA_TOPIC=locations
ENV KAFKA_CONSUMER_GROUP=Spark-Group
ENV SPARK_APPLICATION_PYTHON_LOCATION /app/streaming_app.py
ENV SPARK_APPLICATION_ARGS "lat lon | 38.0 40.1 115.4 130.6"
ENV SPARK_SUBMIT_ARGS --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2,com.datastax.spark:spark-cassandra-connector_2.12:3.2.0
```

Streaming aplikacija

Setup Spark i Kafka konekcija

```
73  if __name__ == '__main__':
74      if len(sys.argv) < 2:
75          print("Usage: main.py <input folder> ")
76          exit(-1)
77
78      inputValuesDictionary = readInputValues()
79      print(inputValuesDictionary)
80
81
82      conf = SparkConf()
83      #conf.setMaster("spark://spark-master-x:7077")
84      conf.setMaster("local")
85      conf.set("spark.driver.memory","4g")
86      conf.set("spark.cassandra.connection.host", "cassandra")
87      conf.set("spark.cassandra.connection.port", 9042)
88
89      spark = SparkSession.builder.config(conf=conf).appName("AverageLatitude").getOrCreate()
90
91      spark.sparkContext.setLogLevel("ERROR")
92
93      df = (
94          spark.readStream.format("kafka")
95          .option("kafka.bootstrap.servers", "kafka:9092")
96          .option("subscribe", "locations")
97          .option("startingOffsets", "latest")
98          .load()
99      )
100
```


Streaming aplikacija

```
    parsed_valuesWithTimestamp = df.select(
        "timestamp", from_json(col("value").cast("string"), schema).alias("parsed_values")
    )


    parsed_valuesWithTimestamp.printSchema()
    #parsed_valuesWithTimestamp = parsed_valuesWithTimestamp.where(col("parsed_values.lat") > inputValuesDictionary["lat"][0]).where(col("parsed_values.lat")

    parsed_valuesWithTimestamp = findDfBasedOnInputValues(inputValuesDictionary, parsed_valuesWithTimestamp)

    timestampXaltitudes = parsed_valuesWithTimestamp.selectExpr("timestamp", "parsed_values.alt AS alt", "parsed_values.user AS user")

    parsed_values = parsed_valuesWithTimestamp.selectExpr("parsed_values.*")
    #parsed_values.printSchema()
    timestamps = parsed_values.selectExpr("ts AS ts")
    latitudes = parsed_values.selectExpr("lat AS lat")
    longitudes = parsed_values.selectExpr( "lon AS lon")
    altitudes = parsed_values.selectExpr("alt AS alt")
    labels = parsed_values.selectExpr("label AS label")
    users = parsed_values.selectExpr("user AS user")


    #timestampXaltitudes.printSchema()
    AltitudeValuesAvgWindows = timestampXaltitudes.groupBy(window(timestampXaltitudes.timestamp, "20 seconds", "10 seconds"), timestampXaltitudes.user).agg(
        mean("alt").alias("meanvalue"),
        max("alt").alias("maxvalue"),
        min("alt").alias("minvalue"),
        count("user").alias("times_data_sent")
    )


    #AltitudeValuesAvgWindows.printSchema()
    AltitudeValuesAvgWindowsForDB = AltitudeValuesAvgWindows.selectExpr("window.start as start",
                                                                        "window.end as end",
                                                                        "user",
                                                                        "meanvalue",
                                                                        "maxvalue",
                                                                        "minvalue",
                                                                        "times_data_sent" )
    #AltitudeValuesAvgWindowsForDB.printSchema()
```

Konverzija ulaznih vrednosti

→

Streaming aplikacija

Upis rezultata obrade na konzolu i u Cassandra bazu podataka

```
145     query = (
146         AltitudeValuesAvgWindows.writeStream.outputMode("update")
147         .queryName("average_latitude")
148         .format("console") # format("org.apache.spark.sql.cassandra").options(**load_options)
149         .trigger(processingTime="1 seconds")
150         .option("truncate", "false")
151         .start()
152         #.awaitTermination()
153     )
154 
155     ##########################################################################################
156     ###################         UPIS U CASSANDRA BAZU PODATAKA         ######################
157     ##########################################################################################
158 
159     def writeToCassandra(writeDF, _):
160         writeDF.write \
161             .format("org.apache.spark.sql.cassandra")\
162             .mode('append')\
163             .options(table="geolocation", keyspace="locations_db")\
164             .save()
165 
166     AltitudeValuesAvgWindowsForDB.writeStream \
167         .option("spark.cassandra.connection.host","cassandra:9042")\
168         .foreachBatch(writeToCassandra) \
169         .outputMode("update") \
170         .start()\
171         .awaitTermination()
172 
```


Streaming aplikacija

Prikaz batch obrade u konzoli

```
-------------------------------------------
Batch: 6
-------------------------------------------
+------------------------------------------+----+-----------------+---------+---------+---------------+
|window                                    |user|meanvalue        |maxvalue |minvalue |times_data_sent|
+------------------------------------------+----+-----------------+---------+---------+---------------+
|{2023-04-04 17:33:20, 2023-04-04 17:33:30}|128 |229.65879821777344|232.93964|226.37796|2              |
|{2023-04-04 17:33:25, 2023-04-04 17:33:35}|128 |229.65879821777344|232.93964|226.37796|2              |
+------------------------------------------+----+-----------------+---------+---------+---------------+

-------------------------------------------
Batch: 7
-------------------------------------------
+------------------------------------------+----+-----------------+---------+---------+---------------+
|window                                    |user|meanvalue        |maxvalue |minvalue |times_data_sent|
+------------------------------------------+----+-----------------+---------+---------+---------------+
|{2023-04-04 17:33:35, 2023-04-04 17:33:45}|128 |254.2650909423828 |282.15222|226.37796|2              |
|{2023-04-04 17:33:30, 2023-04-04 17:33:40}|128 |221.45669555664062|226.37796|216.53543|2              |
|{2023-04-04 17:33:25, 2023-04-04 17:33:35}|128 |225.28434244791666|232.93964|216.53543|3              |
|{2023-04-04 17:33:40, 2023-04-04 17:33:50}|128 |282.1522216796875 |282.15222|282.15222|1              |
+------------------------------------------+----+-----------------+---------+---------+---------------+

-------------------------------------------
Batch: 8
-------------------------------------------
+------------------------------------------+----+-----------------+---------+---------+---------------+
|window                                    |user|meanvalue        |maxvalue |minvalue |times_data_sent|
+------------------------------------------+----+-----------------+---------+---------+---------------+
|{2023-04-04 17:33:45, 2023-04-04 17:33:55}|128 |257.5459289550781 |259.18634|255.90552|2              |
|{2023-04-04 17:33:40, 2023-04-04 17:33:50}|128 |265.74802652994794|282.15222|255.90552|3              |
+------------------------------------------+----+-----------------+---------+---------+---------------+

-------------------------------------------
Batch: 9
-------------------------------------------
+------------------------------------------+----+-----------------+---------+---------+---------------+
|window                                    |user|meanvalue        |maxvalue |minvalue |times_data_sent|
+------------------------------------------+----+-----------------+---------+---------+---------------+
|{2023-04-04 17:33:45, 2023-04-04 17:33:55}|128 |252.62466939290366|259.18634|242.78215|3              |
|{2023-04-04 17:33:50, 2023-04-04 17:34:00}|128 |264.1076126098633 |285.43307|242.78215|2              |
|{2023-04-04 17:33:55, 2023-04-04 17:34:05}|128 |285.4330749511719 |285.43307|285.43307|1              |
+------------------------------------------+----+-----------------+---------+---------+---------------+
```

Flink Streaming aplikacija

Eksterni "Dependency" u Flink projektu

```
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.5</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.5</version>
        </dependency>

        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>RELEASE</version>
            <scope>compile</scope>
        </dependency>
    </dependencies>
```



```
<dependencies>

    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-scala_2.12</artifactId>
        <version>${flink.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-streaming-scala_2.12</artifactId>
        <version>${flink.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-connector-kafka</artifactId>
        <version>${flink.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-clients</artifactId>
        <version>${flink.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-streaming-java</artifactId>
        <version>${flink.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-connector-cassandra_2.12</artifactId>
        <version>${flink.version}</version>
    </dependency>
```

Flink Streaming aplikacija

Postavljanje stream-a

```
2 usages  MrFICAX *
public class DataStreamJob {

    no usages  MrFICAX *
    public static void main(String[] args) throws Exception {

        StreamExecutionEnvironment environment = StreamExecutionEnvironment.getExecutionEnvironment();
        String inputTopic = "locations";
        String server = "kafka:9092";

        DataStream<String> dataStream = StreamConsumer(inputTopic, server, environment);

        new *
        DataStream<Location> locationStream = ConvertStreamFromJsonToLocationType(dataStream).filter(new FilterFunction<Location>() {
            new *
            @Override
            public boolean filter(Location location) throws Exception {
                return (location.getLongitude() > 115.2 && location.getLongitude() < 117.5);           }
        });
        //locationStream.print();

        SingleOutputStreamOperator windowedStream = locationStream
                .keyBy(Location::getUser) KeyedStream<Location, String>
                .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5))) WindowedStream<Location, String, TimeWindow>
                .aggregate(new AverageAggregate());

        SingleOutputStreamOperator windowedStream2 = locationStream
                .keyBy(Location::getUser) KeyedStream<Location, String>
                .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5))) WindowedStream<Location, String, TimeWindow>
                .process(new MyProcessWindowFunction());
                //.aggregate(new AverageAggregate(), new MyProcessWindowFunction() );

        //windowedStream.print();
        //windowedStream2.print();

        CassandraService cassandraService = new CassandraService();
        cassandraService.sinkToCassandraDB(windowedStream);

        environment.execute();
    }
```

Flink Streaming aplikacija

Kreiranje Consumer-a sa Kafka Topic-a

```
1 usage  MrFICAX
public static DataStream<String> StreamConsumer(String inputTopic, String server, StreamExecutionEnvironment environment) throws Exception {
    FlinkKafkaConsumer<String> flinkKafkaConsumer = createStringConsumerForTopic(inputTopic, server);
    DataStream<String> stringInputStream = environment.addSource(flinkKafkaConsumer);


    MrFICAX
    return stringInputStream.map(new MapFunction<String, String>() {
        no usages
        private static final long serialVersionUID = -999736771747691234L;

        MrFICAX
        @Override
        public String map(String value) throws Exception {
            return value;
        }
    });
}

1 usage  MrFICAX
public static FlinkKafkaConsumer<String> createStringConsumerForTopic(
        String topic, String kafkaAddress) {

    Properties props = new Properties();
    props.setProperty("bootstrap.servers", kafkaAddress);
    //props.setProperty("group.id",kafkaGroup);
    FlinkKafkaConsumer<String> consumer = new FlinkKafkaConsumer<>(
            topic, new SimpleStringSchema(), props);

    return consumer;
}
```

Flink Streaming aplikacija

Location klasa

```java
@Builder
@Getter
@Setter
@ToString
@EqualsAndHashCode
@AllArgsConstructor
@NoArgsConstructor(force = true)
public class Location {
    no usages
    final Double Latitude;
    no usages
    final Double Longitude;
    no usages
    final Double Altitude;
    no usages
    final String   Label;
    no usages
    final String User;
    no usages
    final Integer Year;
    no usages
    final Integer Month;
    no usages
    final Integer Day;
    no usages
    final Integer Hour;
    no usages
    final Integer Minute;
    no usages
    final Integer Second;
    no usages
    final Integer   Timestamp;

}
```

Flink Streaming aplikacija

Konverzija Stream-a uz Json tipa u tip Location

```
1 usage   MrFICAX
91 @  public static DataStream<Location> ConvertStreamFromJsonToLocationType(DataStream<String> jsonStream) {
92        return jsonStream.map(kafkaMessage -> {
93            try {
94                JsonNode jsonNode = new ObjectMapper().readValue(kafkaMessage, JsonNode.class);
95                return Location.builder()
96                        .Latitude(jsonNode.get("lat").asDouble())
97                        .Longitude(jsonNode.get("lon").asDouble())
98                        .Altitude(jsonNode.get("alt").asDouble())
99                        .Label(jsonNode.get("label").asText())
100                       .User(jsonNode.get("user").asText())
101                       .Year(jsonNode.get("year").asInt())
102                       .Month(jsonNode.get("month").asInt())
103                       .Day(jsonNode.get("day").asInt())
104                       .Hour(jsonNode.get("hour").asInt())
105                       .Minute(jsonNode.get("minute").asInt())
106                       .Second(jsonNode.get("second").asInt())
107                       .Timestamp(jsonNode.get("ts").asInt())
108                       .build();
109
110           } catch (Exception e) {
111               return null;
112           }
113       }).filter(Objects::nonNull).forward();
114   }
115
```


Flink Streaming aplikacija

AggregateFunction za obradu svakog prozora

```
1 usage  MrFICAX
class AverageAggregate implements AggregateFunction<Location, Tuple5<String, Double, Double, Double, Double>, Tuple5<String, Double, Double, Double, Double>> {
    no usages  MrFICAX
    @Override
    public Tuple5<String, Double, Double, Double, Double> createAccumulator() {
        return new Tuple5<>("", 0D, Double.MAX_VALUE, Double.MIN_VALUE, 0D);
    }

    MrFICAX
    @Override
    public Tuple5<String, Double, Double, Double, Double> add(Location value, Tuple5<String, Double, Double, Double, Double> accumulator) {
        accumulator.f0 = value.getUser();
        accumulator.f1 += value.getAltitude();
        accumulator.f2 = accumulator.f2 > value.getAltitude() ? value.getAltitude() : accumulator.f2; //MINIMUM
        accumulator.f3 = accumulator.f3 < value.getAltitude() ? value.getAltitude() : accumulator.f3; //MAXIMUM

        return new Tuple5<>(accumulator.f0, accumulator.f1, accumulator.f2, accumulator.f3, accumulator.f4 + 1);
    }

    MrFICAX
    @Override
    public Tuple5<String, Double, Double, Double, Double> getResult(Tuple5<String, Double, Double, Double, Double> acc) {
        return new Tuple5<>(acc.f0, acc.f2, acc.f3, acc.f1 / acc.f4, acc.f4);
    }

    MrFICAX
    @Override
    public Tuple5<String, Double, Double, Double, Double> merge(Tuple5<String, Double, Double, Double, Double> acc1, Tuple5<String, Double, Double, Double, Double> acc2) {
        return new Tuple5<>(acc1.f0, acc1.f1 + acc2.f1, acc1.f2 < acc2.f2 ? acc1.f2 : acc2.f2, acc1.f2 > acc2.f2 ? acc1.f2 : acc2.f2, acc1.f4 + acc2.f4);
    }
```

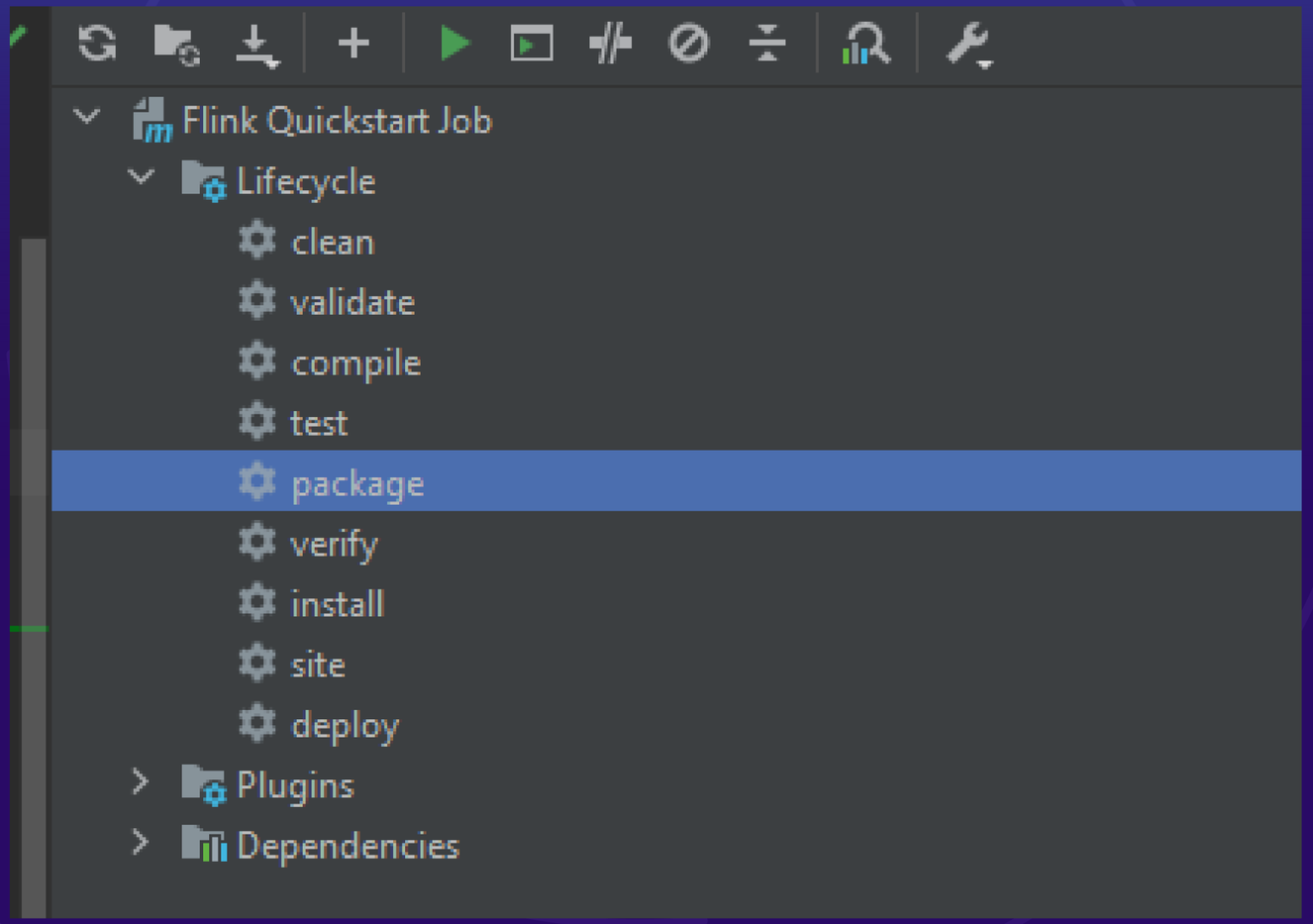

Flink Streaming aplikacija

Konverzija DataStream-a i upis u Cassandra bazu podataka

```
public final class   CassandraService {

    2 usages
    private static final Logger LOGGER = Logger.getLogger(DataStreamJob.class);

    1 usage   MrFICAX *
    public final void sinkToCassandraDB(final DataStream<Tuple5<String, Double, Double, Double, Double>> sinkFilteredLocationStream) throws Exception {

        LOGGER.info("Creating car data to sink into cassandraDB.");
        SingleOutputStreamOperator<Tuple5<String, String, String, String, String>> sinkLocationStream = sinkFilteredLocationStream.map(
                (MapFunction<Tuple5<String, Double, Double, Double, Double>, Tuple5<String, String, String, String, String>>) filteredData ->
                        new Tuple5<>(filteredData.f0, Double.toString(filteredData.f1), Double.toString(filteredData.f2), Double.toString(filteredData.f3), Double.toString(filteredData.f4)))
                .returns(new TupleTypeInfo<>(TypeInformation.of(String.class),
                        TypeInformation.of(String.class),
                        TypeInformation.of(String.class),
                        TypeInformation.of(String.class),
                        TypeInformation.of(String.class)));

        sinkLocationStream.print();
        LOGGER.info("Open Cassandra connection and Sinking locations data into cassandraDB.");
        CassandraSink
                .addSink(sinkLocationStream)
//...
                .setHost("cassandra")
                .setQuery("INSERT INTO locations_db.flinkgeolocation(user, minvalue, maxvalue, meanvalue, count) values (?, ?, ?, ?, ?);")
                .build();

    }
}
```

Flink Streaming aplikacija

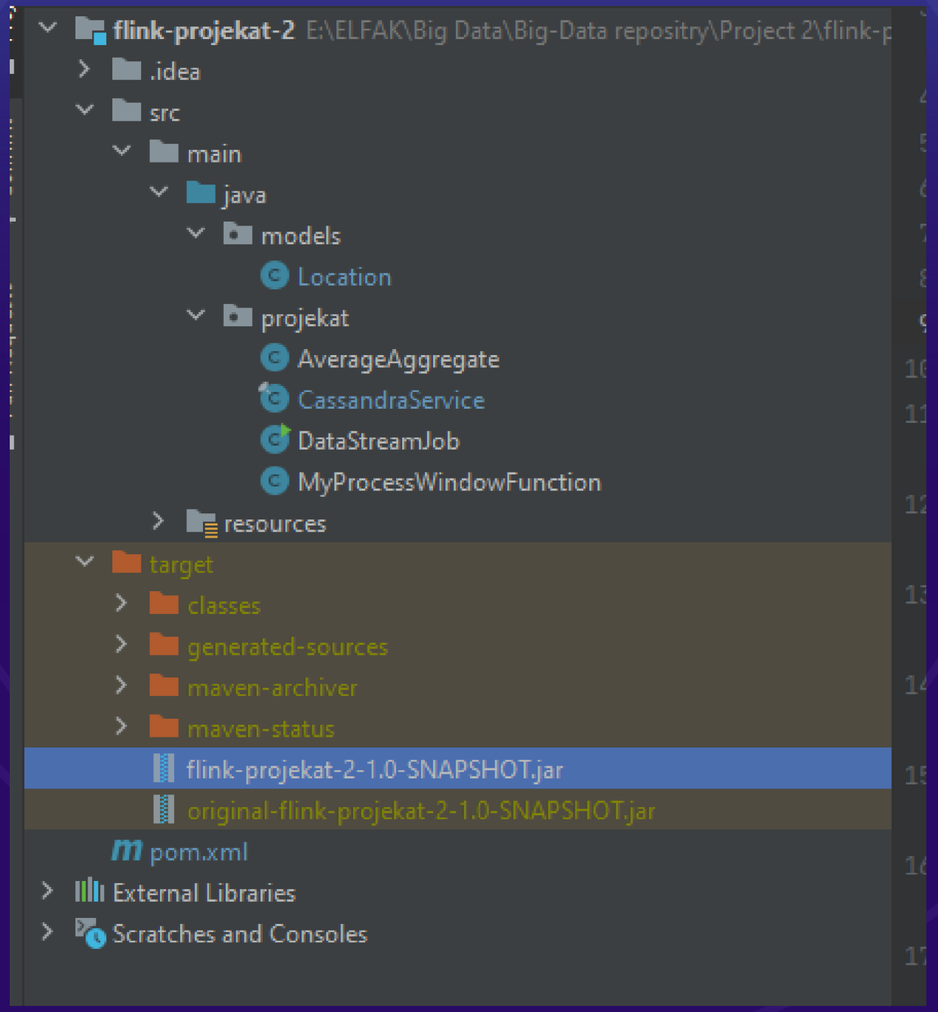
Kreiranje .jar fajla uz pomoć Maven package



→

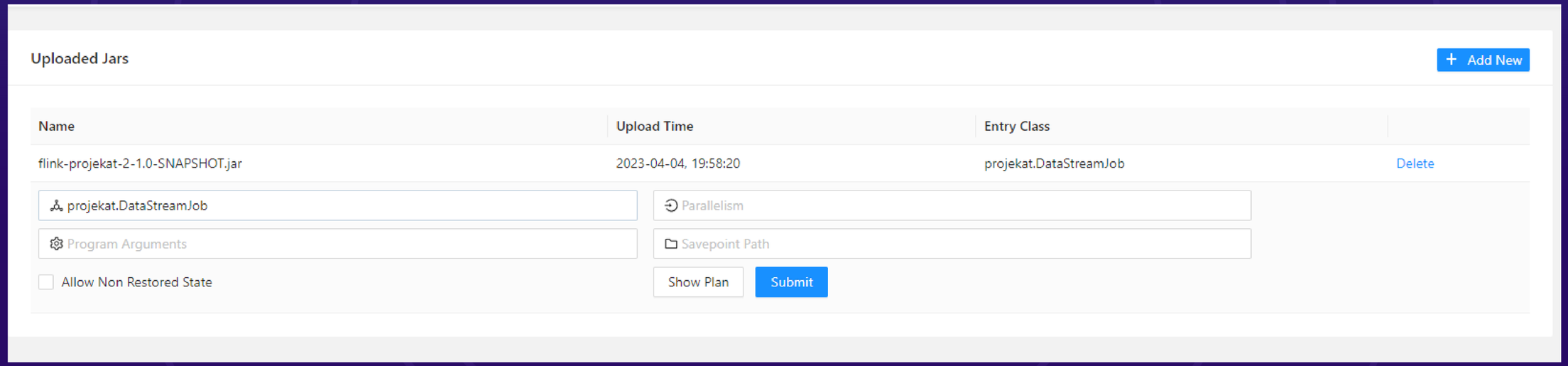
Flink Streaming aplikacija

Dobija se SNAPSHOT.jar fajl



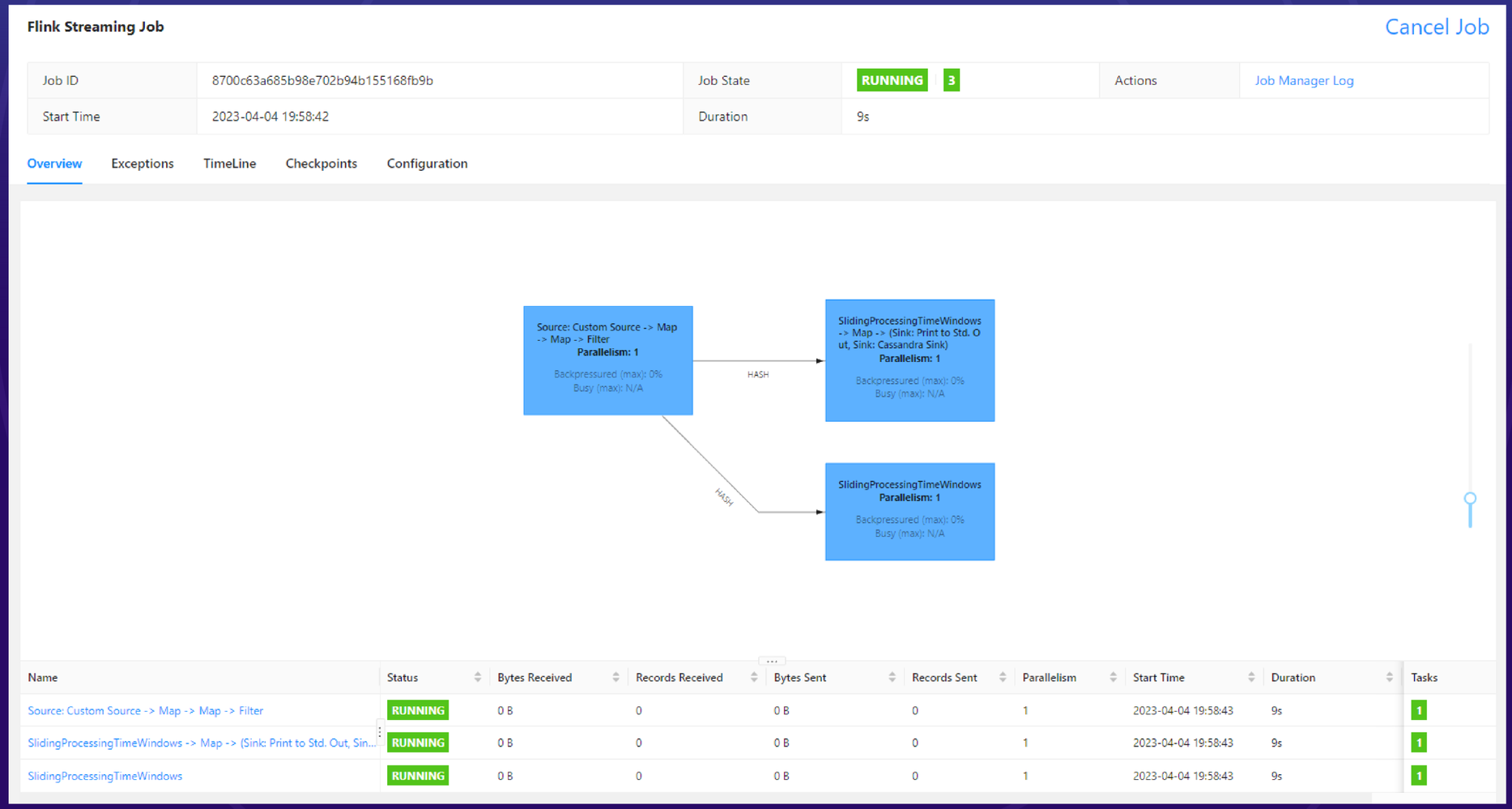
Flink Streaming aplikacija

Postavljanje .jar fajla na Flink kontejner

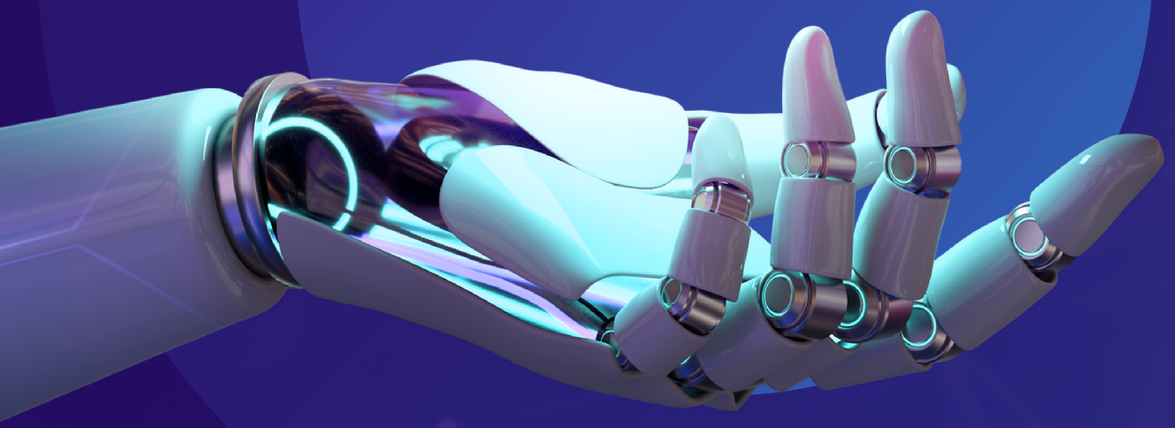


Flink Streaming aplikacija

Pregled pokrenutog posla na Flink kontejner-u



SCAN ME



HVALA NA PAŽNJI

ficax.al@elfak.rs