Upravljanje i analiza velikih skupova podataka

Projekat 2 - Spark Structured Streaming

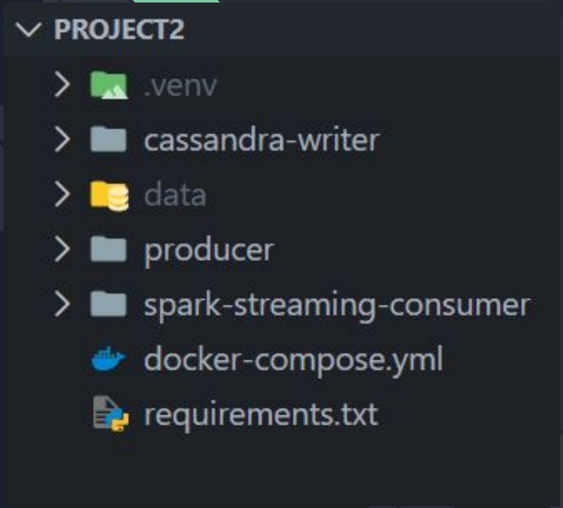
Luka Gavrilović 1823

Projekat 2

- Real-time analiza kašnjenja letova korišćenjem Spark Structured Streaming-a:
  - Stream obrada podataka sa Kafka topic-a
  - PySpark Structured Streaming + Docker Spark klaster (BDE)
  - Rezultati snimljeni u Cassandra NoSQL bazu

Struktura projekta

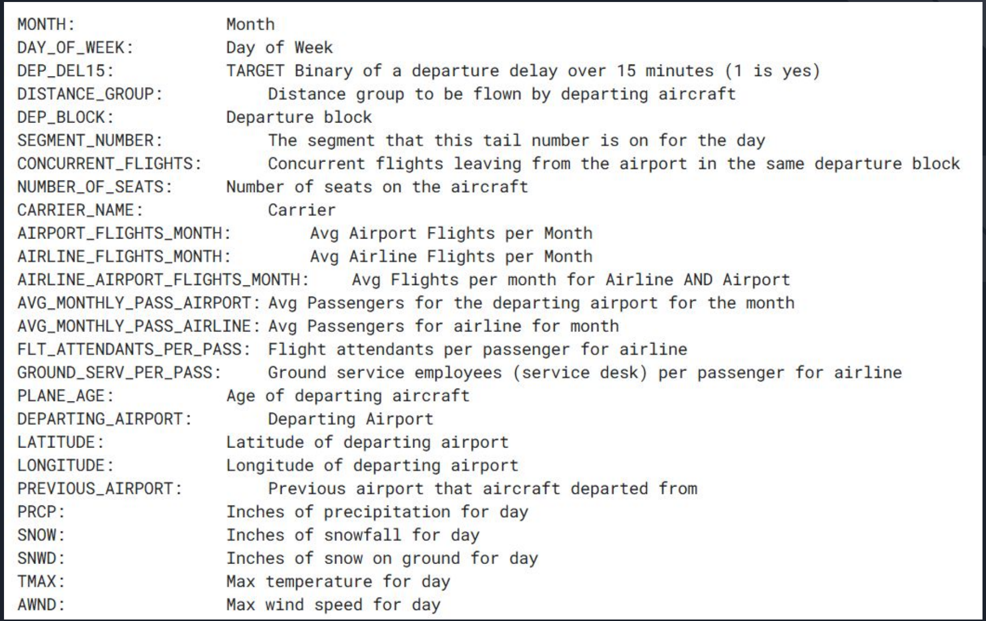
- Pomoćna aplikacija (producer) koja čita velike CSV datoteke slog po slog i šalje ih na Kafka topic
- Spark Streaming aplikacija (consumer) koja:
  - Prima podatke sa Kafka topic-a u realnom vremenu
  - Primenjuje vremenske prozore (tumbling/sliding)
  - Računa statistiku odabranih atributa: broj, min, max, avg, stddev
  - Šalje rezultate na drugi Kafka topic
- Cassandra writer aplikacija koja konzumira Kafka topic i upisuje rezultate u bazu



```
PROJECT2
  .venv
  cassandra-writer
  data
  producer
  spark-streaming-consumer
  docker-compose.yml
  requirements.txt
```

Skup podataka

- 2019 Airline Delays w/Weather and Airport Detail (≈1.37GB)
- https://www.kaggle.com/datasets/threnjen/2019-airline-delays-and-cancellations?select=full_data_flightdelay.csv
- Skup podataka sa detaljnim informacijama o avio-kompanijama, aerodromima i vremenskim uslovima



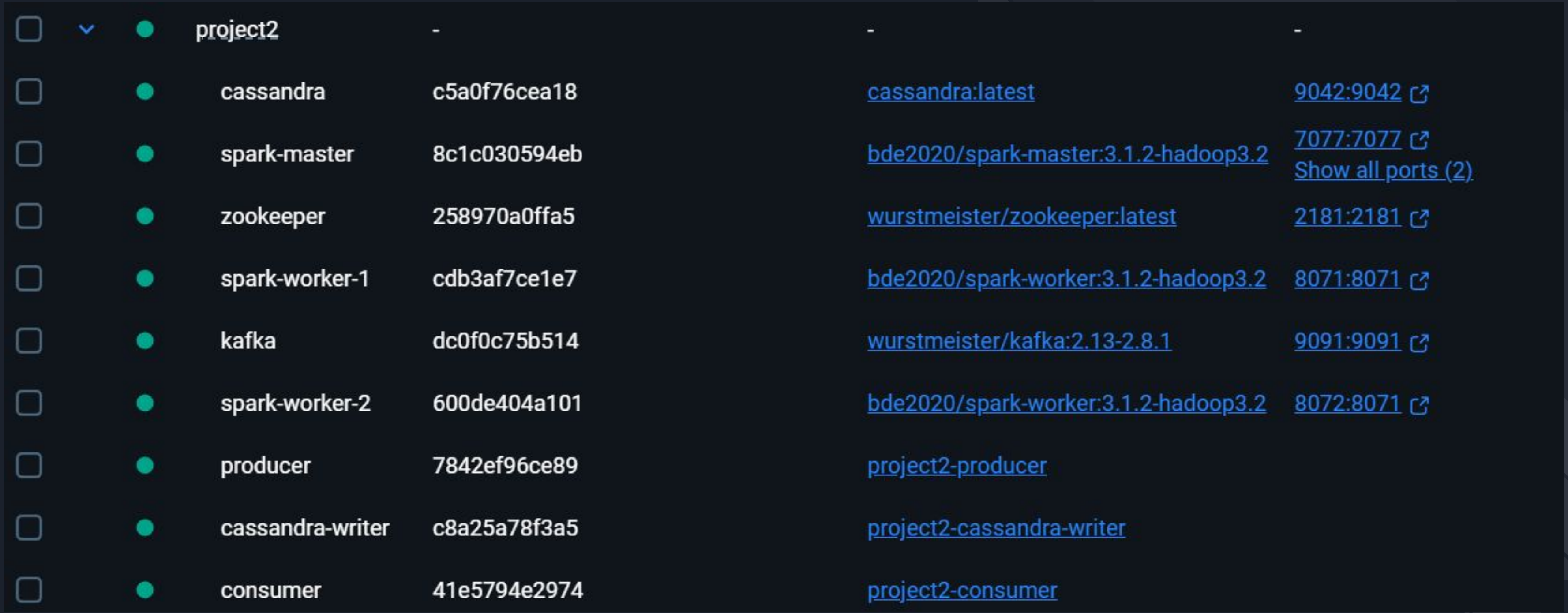
MONTH:	Month
DAY_OF_WEEK:	Day of Week
DEP_DEL15:	TARGET Binary of a departure delay over 15 minutes (1 is yes)
DISTANCE_GROUP:	Distance group to be flown by departing aircraft
DEP_BLOCK:	Departure block
SEGMENT_NUMBER:	The segment that this tail number is on for the day
CONCURRENT_FLIGHTS:	Concurrent flights leaving from the airport in the same departure block
NUMBER_OF_SEATS:	Number of seats on the aircraft
CARRIER_NAME:	Carrier
AIRPORT_FLIGHTS_MONTH:	Avg Airport Flights per Month
AIRLINE_FLIGHTS_MONTH:	Avg Airline Flights per Month
AIRLINE_AIRPORT_FLIGHTS_MONTH:	Avg Flights per month for Airline AND Airport
AVG_MONTHLY_PASS_AIRPORT:	Avg Passengers for the departing airport for the month
AVG_MONTHLY_PASS_AIRLINE:	Avg Passengers for airline for month
FLT_ATTENDANTS_PER_PASS:	Flight attendants per passenger for airline
GROUND_SERV_PER_PASS:	Ground service employees (service desk) per passenger for airline
PLANE_AGE:	Age of departing aircraft
DEPARTING_AIRPORT:	Departing Airport
LATITUDE:	Latitude of departing airport
LONGITUDE:	Longitude of departing airport
PREVIOUS_AIRPORT:	Previous airport that aircraft departed from
PRCP:	Inches of precipitation for day
SNOW:	Inches of snowfall for day
SNWD:	Inches of snow on ground for day
TMAX:	Max temperature for day
AWND:	Max wind speed for day

Docker i klaster kontejnera

- `docker-compose.yml`klaster konfiguracija:
  - Spark master + 2 workera
  - Kafka & Zookeeper
  - Cassandra
  - Producer
  - Consumer
  - Cassandra Writer
- Aplikacija pokreće se unutar mreže `bde`

- Pokretanje infrastrukture:
  - `docker network create bde`
  - pozicioniranje u folder gde se nalazi `docker-compose.yml`
  - `docker-compose up --build -d`

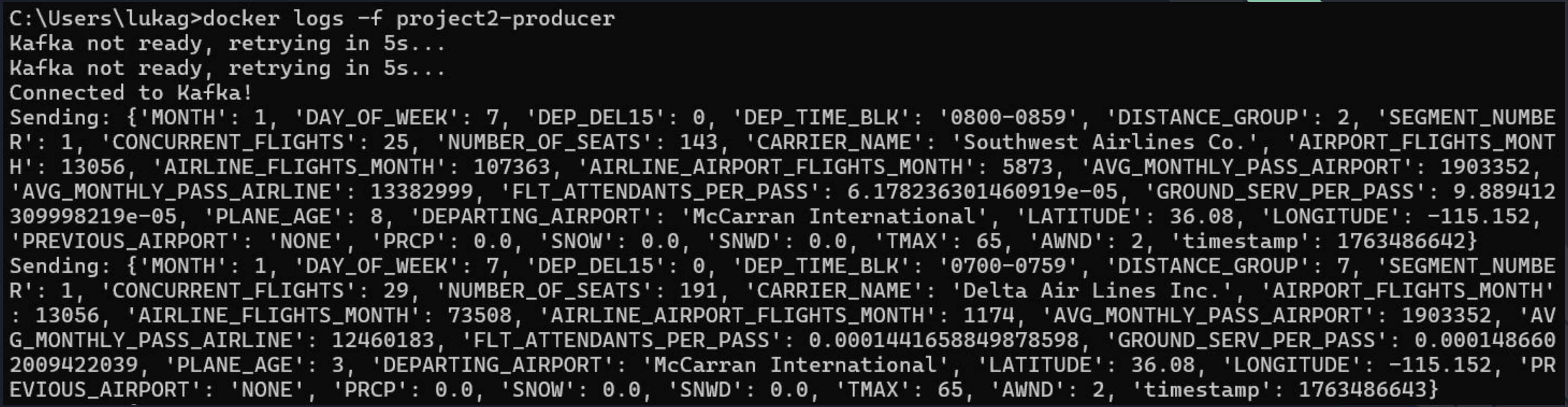
Docker i klaster kontejnera



●	project2	-	-	-
●	cassandra	c5a0f76cea18	cassandra:latest	9042:9042
●	spark-master	8c1c030594eb	bde2020/spark-master:3.1.2-hadoop3.2	7077:7077 Show all ports (2)
●	zookeeper	258970a0ffa5	wurstmeister/zookeeper:latest	2181:2181
●	spark-worker-1	cdb3af7ce1e7	bde2020/spark-worker:3.1.2-hadoop3.2	8071:8071
●	kafka	dc0f0c75b514	wurstmeister/kafka:2.13-2.8.1	9091:9091
●	spark-worker-2	600de404a101	bde2020/spark-worker:3.1.2-hadoop3.2	8072:8071
●	producer	7842ef96ce89	project2-producer	
●	cassandra-writer	c8a25a78f3a5	project2-cassandra-writer	
●	consumer	41e5794e2974	project2-consumer	

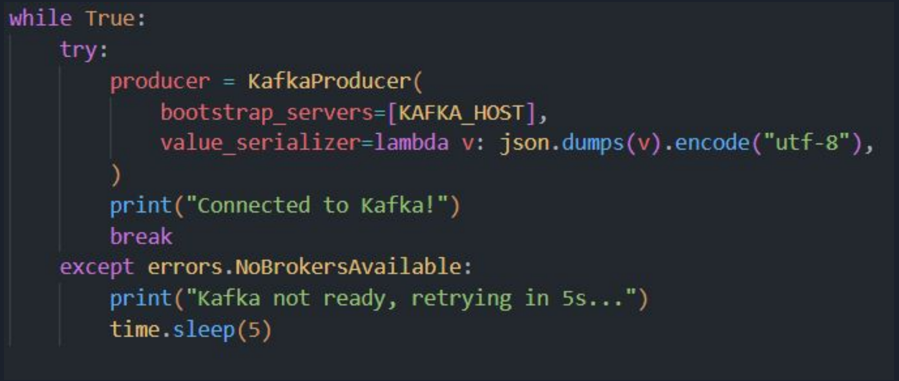
Kafka producer

- Čita podatke iz dataseta i upisuje ih na topic koji je definisan kao promenljiva okruženja
- Sadrži .py fajl, Dockerfile i requirements.txt
- Praćenje učitavanja i slanja podataka na Kafka topic:
    - `docker logs -f project2-producer`

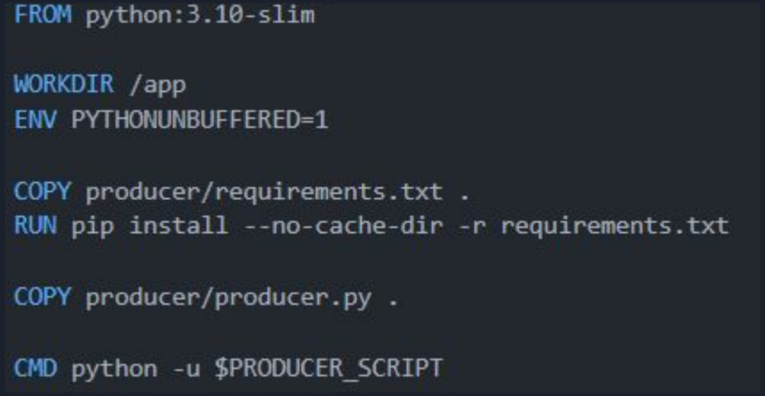


```
C:\Users\lukag>docker logs -f project2-producer
Kafka not ready, retrying in 5s...
Kafka not ready, retrying in 5s...
Connected to Kafka!
Sending: {'MONTH': 1, 'DAY_OF_WEEK': 7, 'DEP_DEL15': 0, 'DEP_TIME_BLK': '0800-0859', 'DISTANCE_GROUP': 2, 'SEGMENT_NUMBE
R': 1, 'CONCURRENT_FLIGHTS': 25, 'NUMBER_OF_SEATS': 143, 'CARRIER_NAME': 'Southwest Airlines Co.', 'AIRPORT_FLIGHTS_MONT
H': 13056, 'AIRLINE_FLIGHTS_MONTH': 107363, 'AIRLINE_AIRPORT_FLIGHTS_MONTH': 5873, 'AVG_MONTHLY_PASS_AIRPORT': 1903352,
'AVG_MONTHLY_PASS_AIRLINE': 13382999, 'FLT_ATTENDANTS_PER_PASS': 6.178236301460919e-05, 'GROUND_SERV_PER_PASS': 9.889412
309998219e-05, 'PLANE_AGE': 8, 'DEPARTING_AIRPORT': 'McCarran International', 'LATITUDE': 36.08, 'LONGITUDE': -115.152,
'PREVIOUS_AIRPORT': 'NONE', 'PRCP': 0.0, 'SNOW': 0.0, 'SNWD': 0.0, 'TMAX': 65, 'AWND': 2, 'timestamp': 1763486642}
Sending: {'MONTH': 1, 'DAY_OF_WEEK': 7, 'DEP_DEL15': 0, 'DEP_TIME_BLK': '0700-0759', 'DISTANCE_GROUP': 7, 'SEGMENT_NUMBE
R': 1, 'CONCURRENT_FLIGHTS': 29, 'NUMBER_OF_SEATS': 191, 'CARRIER_NAME': 'Delta Air Lines Inc.', 'AIRPORT_FLIGHTS_MONTH'
: 13056, 'AIRLINE_FLIGHTS_MONTH': 73508, 'AIRLINE_AIRPORT_FLIGHTS_MONTH': 1174, 'AVG_MONTHLY_PASS_AIRPORT': 1903352, 'AV
G_MONTHLY_PASS_AIRLINE': 12460183, 'FLT_ATTENDANTS_PER_PASS': 0.00014416588498785598, 'GROUND_SERV_PER_PASS': 0.000148660
2009422039, 'PLANE_AGE': 3, 'DEPARTING_AIRPORT': 'McCarran International', 'LATITUDE': 36.08, 'LONGITUDE': -115.152, 'PR
EVIOUS_AIRPORT': 'NONE', 'PRCP': 0.0, 'SNOW': 0.0, 'SNWD': 0.0, 'TMAX': 65, 'AWND': 2, 'timestamp': 1763486643}
```

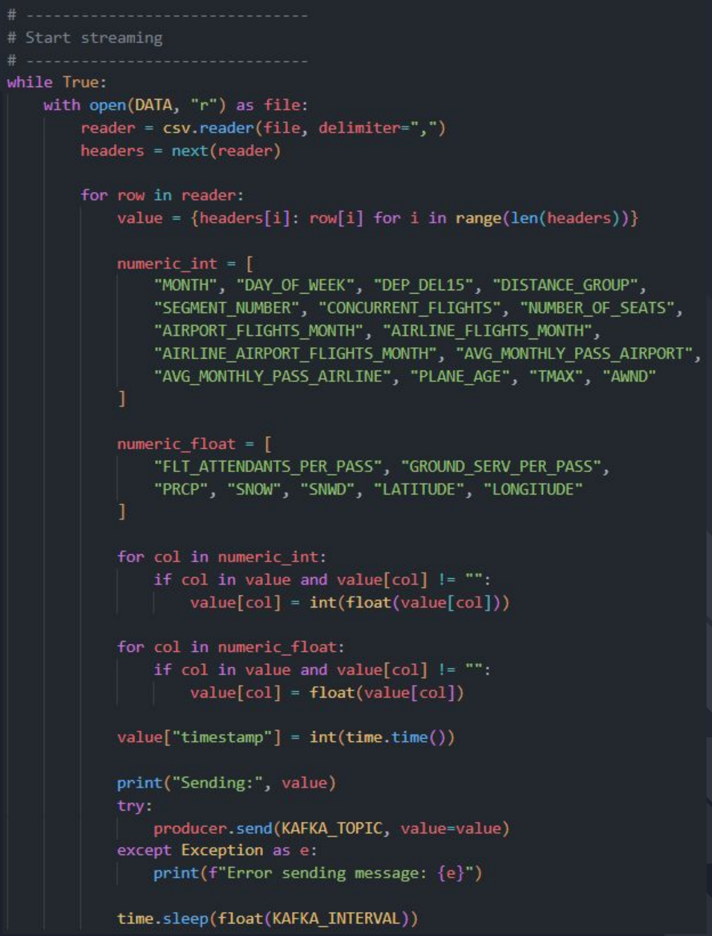
Kafka producer



```python
while True:
    try:
        producer = KafkaProducer(
            bootstrap_servers=[KAFKA_HOST],
            value_serializer=lambda v: json.dumps(v).encode("utf-8"),
        )
        print("Connected to Kafka!")
        break
    except errors.NoBrokersAvailable:
        print("Kafka not ready, retrying in 5s...")
        time.sleep(5)
```



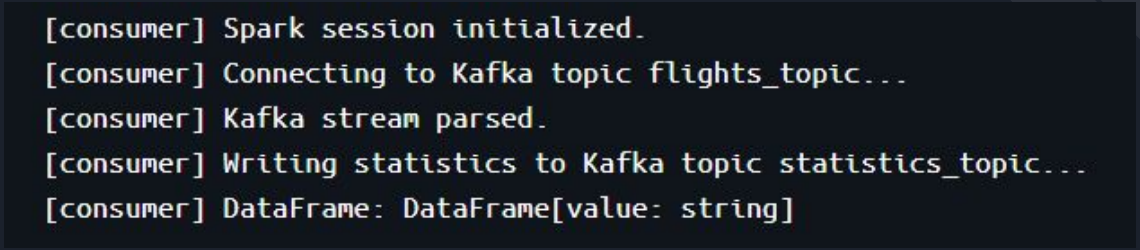
```dockerfile
FROM python:3.10-slim

WORKDIR /app
ENV PYTHONUNBUFFERED=1

COPY producer/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY producer/producer.py .

CMD python -u $PRODUCER_SCRIPT
```



```python
# ------------------------------
# Start streaming
# ------------------------------
while True:
    with open(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))}

            numeric_int = [
                "MONTH", "DAY_OF_WEEK", "DEP_DEL15", "DISTANCE_GROUP",
                "SEGMENT_NUMBER", "CONCURRENT_FLIGHTS", "NUMBER_OF_SEATS",
                "AIRPORT_FLIGHTS_MONTH", "AIRLINE_FLIGHTS_MONTH",
                "AIRLINE_AIRPORT_FLIGHTS_MONTH", "AVG_MONTHLY_PASS_AIRPORT",
                "AVG_MONTHLY_PASS_AIRLINE", "PLANE_AGE", "TMAX", "AWND"
            ]

            numeric_float = [
                "FLT_ATTENDANTS_PER_PASS", "GROUND_SERV_PER_PASS",
                "PRCP", "SNOW", "SNWD", "LATITUDE", "LONGITUDE"
            ]

            for col in numeric_int:
                if col in value and value[col] != "":
                    value[col] = int(float(value[col]))

            for col in numeric_float:
                if col in value and value[col] != "":
                    value[col] = float(value[col])

            value["timestamp"] = int(time.time())

            print("Sending:", value)
            try:
                producer.send(KAFKA_TOPIC, value=value)
            except Exception as e:
                print(f"Error sending message: {e}")

            time.sleep(float(KAFKA_INTERVAL))
```

Spark streaming consumer

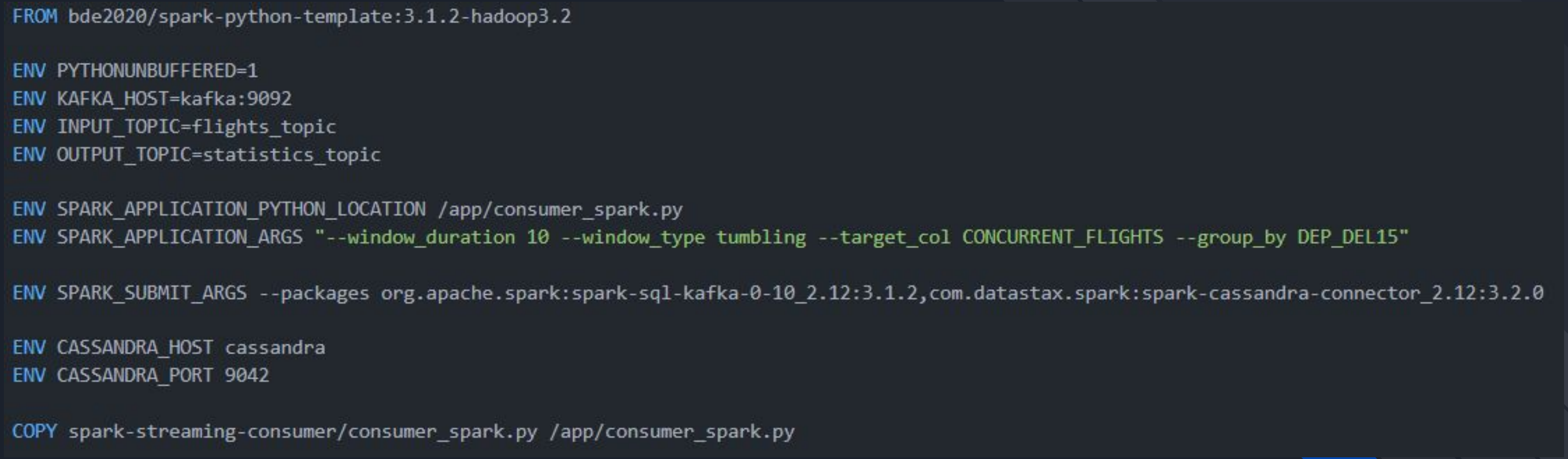
- Spark session konfigurisan za Kafka i Cassandra konekciju
- Čita stream sa Kafka topic-a, parsira JSON u Spark DataFrame
- Podržani vremenski prozori: tumbling i sliding
- Grupisanje i statistika po odabranim kolonama (npr. DEP_DEL15, CARRIER_NAME)
- Sadrži .py fajl, Dockerfile i requirements.txt
- Praćenje učitavanja i slanja podataka na Kafka topic:
    - `docker logs -f project2-consumer`



```
[consumer] Spark session initialized.
[consumer] Connecting to Kafka topic flights_topic...
[consumer] Kafka stream parsed.
[consumer] Writing statistics to Kafka topic statistics_topic...
[consumer] DataFrame: DataFrame[value: string]
```

Spark streaming consumer

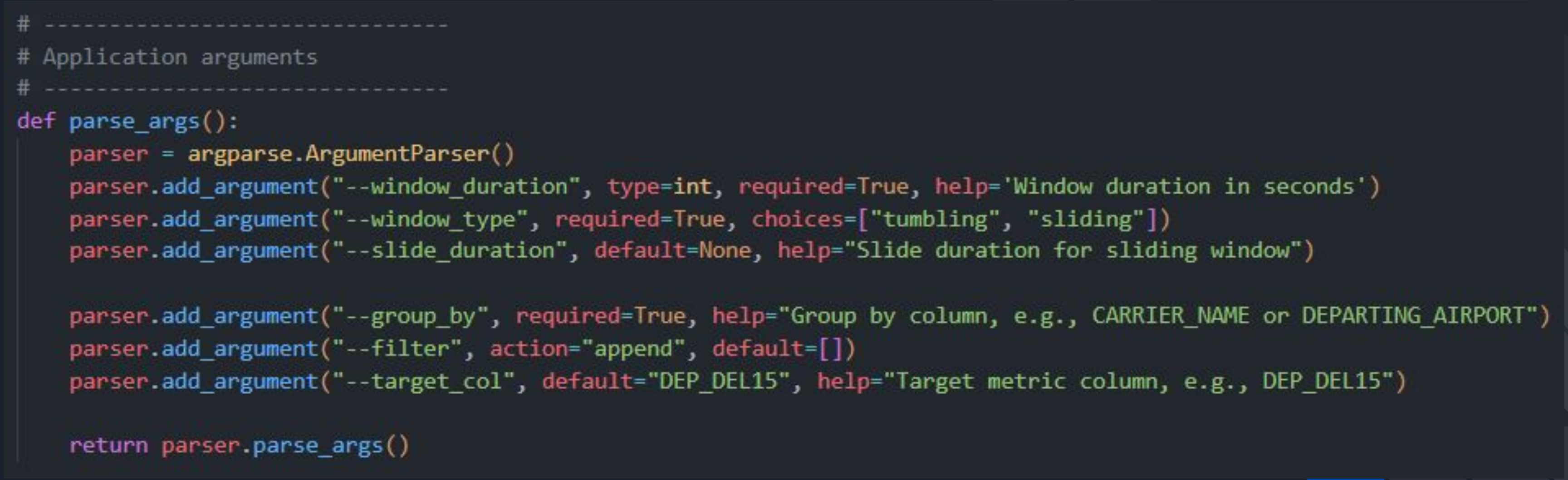
- Dockerfile



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

ENV PYTHONUNBUFFERED=1
ENV KAFKA_HOST=kafka:9092
ENV INPUT_TOPIC=flights_topic
ENV OUTPUT_TOPIC=statistics_topic

ENV SPARK_APPLICATION_PYTHON_LOCATION /app/consumer_spark.py
ENV SPARK_APPLICATION_ARGS "--window_duration 10 --window_type tumbling --target_col CONCURRENT_FLIGHTS --group_by DEP_DEL15"

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

ENV CASSANDRA_HOST cassandra
ENV CASSANDRA_PORT 9042

COPY spark-streaming-consumer/consumer_spark.py /app/consumer_spark.py
```

Spark streaming consumer

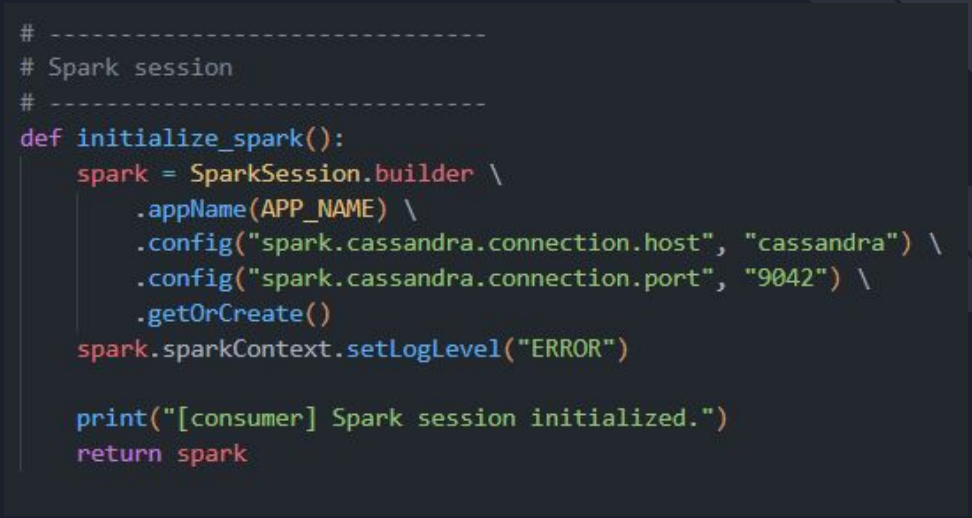
- Argumenti aplikacije



```
# ------------------------------
# Application arguments
# ------------------------------
def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("--window_duration", type=int, required=True, help='Window duration in seconds')
    parser.add_argument("--window_type", required=True, choices=["tumbling", "sliding"])
    parser.add_argument("--slide_duration", default=None, help="Slide duration for sliding window")

    parser.add_argument("--group_by", required=True, help="Group by column, e.g., CARRIER_NAME or DEPARTING_AIRPORT")
    parser.add_argument("--filter", action="append", default=[])
    parser.add_argument("--target_col", default="DEP_DEL15", help="Target metric column, e.g., DEP_DEL15")

    return parser.parse_args()
```

Spark streaming consumer

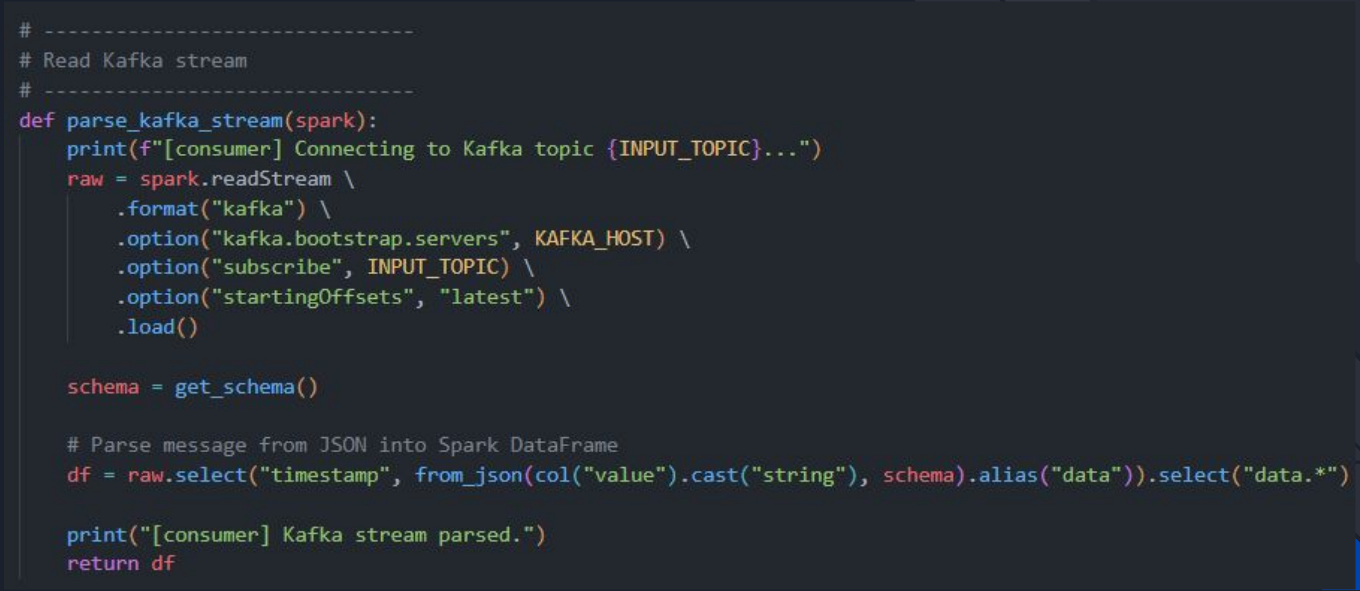
- Spark inicijalizacija



```python
# ------------------------------
# Spark session
# ------------------------------
def initialize_spark():
    spark = SparkSession.builder \
        .appName(APP_NAME) \
        .config("spark.cassandra.connection.host", "cassandra") \
        .config("spark.cassandra.connection.port", "9042") \
        .getOrCreate()
    spark.sparkContext.setLogLevel("ERROR")

    print("[consumer] Spark session initialized.")
    return spark
```

Spark streaming consumer

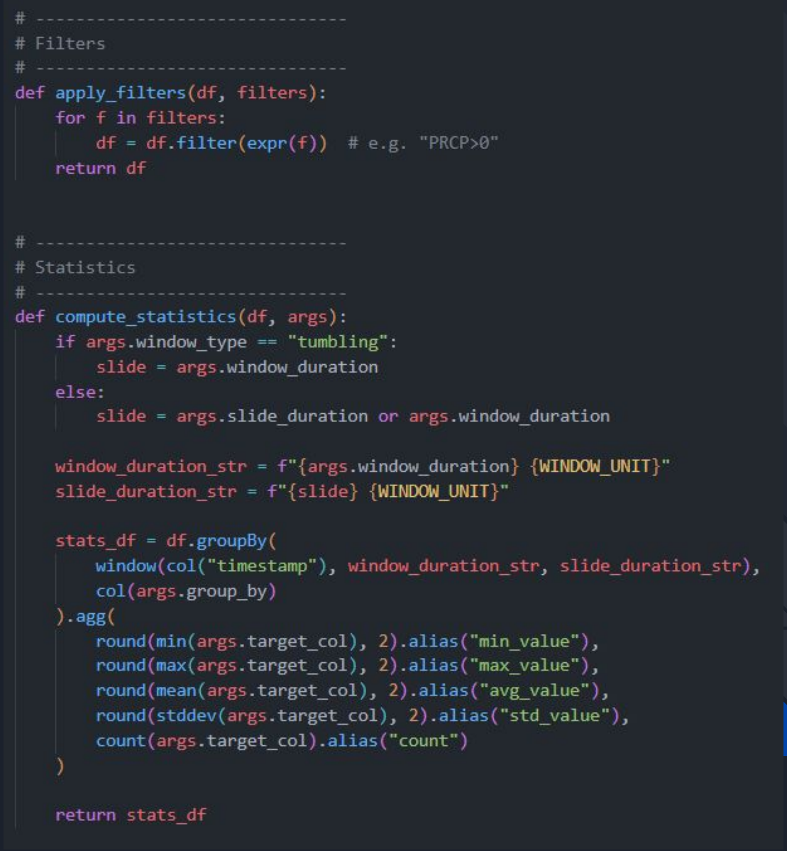
- Čitanje Kafka stream-a



```python
# ------------------------------
# Read Kafka stream
# ------------------------------
def parse_kafka_stream(spark):
    print(f"[consumer] Connecting to Kafka topic {INPUT_TOPIC}...")
    raw = spark.readStream \
        .format("kafka") \
        .option("kafka.bootstrap.servers", KAFKA_HOST) \
        .option("subscribe", INPUT_TOPIC) \
        .option("startingOffsets", "latest") \
        .load()

    schema = get_schema()

    # Parse message from JSON into Spark DataFrame
    df = raw.select("timestamp", from_json(col("value").cast("string"), schema).alias("data")).select("data.*")

    print("[consumer] Kafka stream parsed.")
    return df
```

Spark streaming consumer

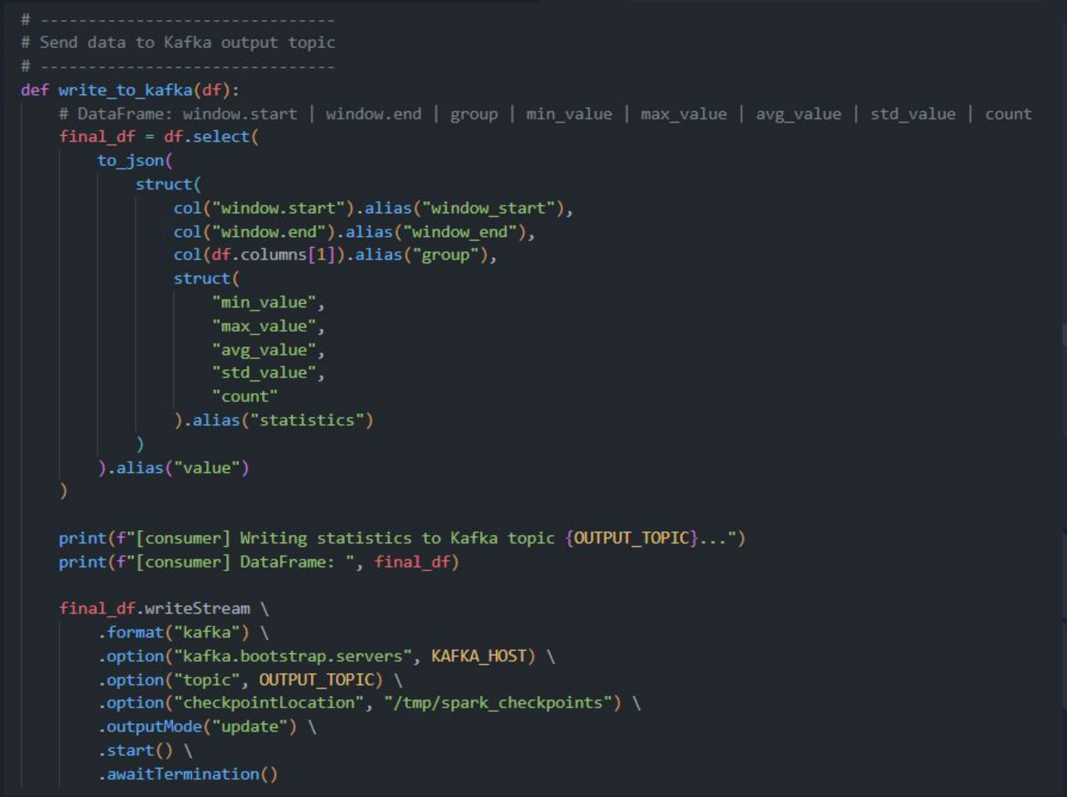
- Primena filtera i računanje statistike



```
# ------------------------------
# Filters
# ------------------------------
def apply_filters(df, filters):
    for f in filters:
        df = df.filter(expr(f))  # e.g. "PRCP>0"
    return df


# ------------------------------
# Statistics
# ------------------------------
def compute_statistics(df, args):
    if args.window_type == "tumbling":
        slide = args.window_duration
    else:
        slide = args.slide_duration or args.window_duration

    window_duration_str = f"{args.window_duration} {WINDOW_UNIT}"
    slide_duration_str = f"{slide} {WINDOW_UNIT}"

    stats_df = df.groupBy(
        window(col("timestamp"), window_duration_str, slide_duration_str),
        col(args.group_by)
    ).agg(
        round(min(args.target_col), 2).alias("min_value"),
        round(max(args.target_col), 2).alias("max_value"),
        round(mean(args.target_col), 2).alias("avg_value"),
        round(stddev(args.target_col), 2).alias("std_value"),
        count(args.target_col).alias("count")
    )

    return stats_df
```

Spark streaming consumer

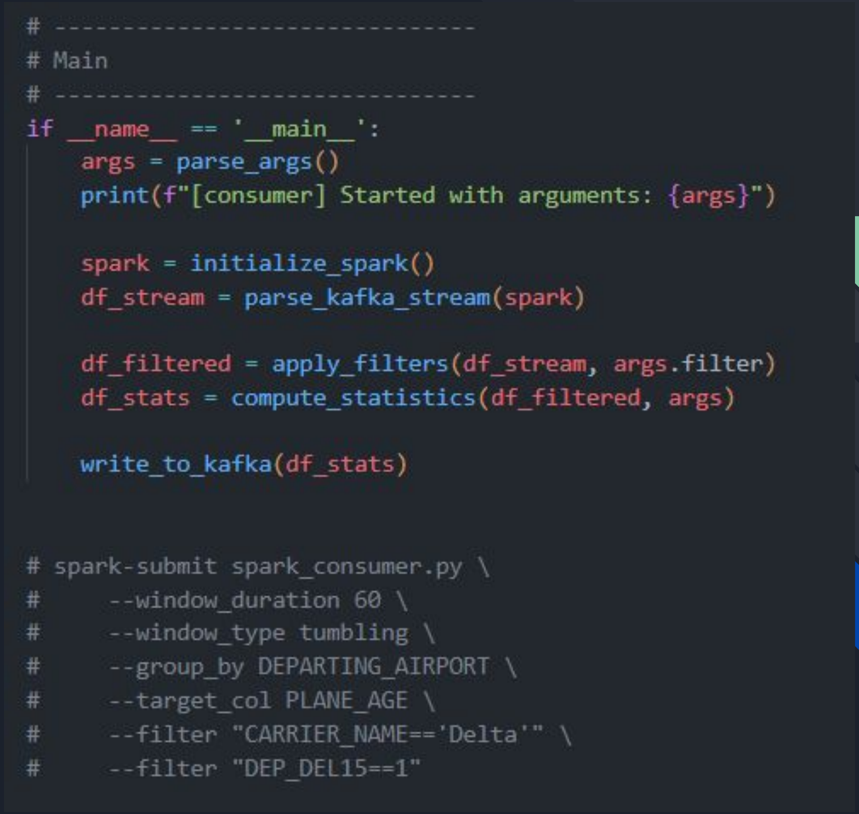
- Upis na novi Kafka topic



```python
# ------------------------------
# Send data to Kafka output topic
# ------------------------------
def write_to_kafka(df):
    # DataFrame: window.start | window.end | group | min_value | max_value | avg_value | std_value | count
    final_df = df.select(
        to_json(
            struct(
                col("window.start").alias("window_start"),
                col("window.end").alias("window_end"),
                col(df.columns[1]).alias("group"),
                struct(
                    "min_value",
                    "max_value",
                    "avg_value",
                    "std_value",
                    "count"
                ).alias("statistics")
            )
        ).alias("value")
    )

    print(f"[consumer] Writing statistics to Kafka topic {OUTPUT_TOPIC}...")
    print(f"[consumer] DataFrame: ", final_df)

    final_df.writeStream \
        .format("kafka") \
        .option("kafka.bootstrap.servers", KAFKA_HOST) \
        .option("topic", OUTPUT_TOPIC) \
        .option("checkpointLocation", "/tmp/spark_checkpoints") \
        .outputMode("update") \
        .start() \
        .awaitTermination()
```

Spark streaming consumer

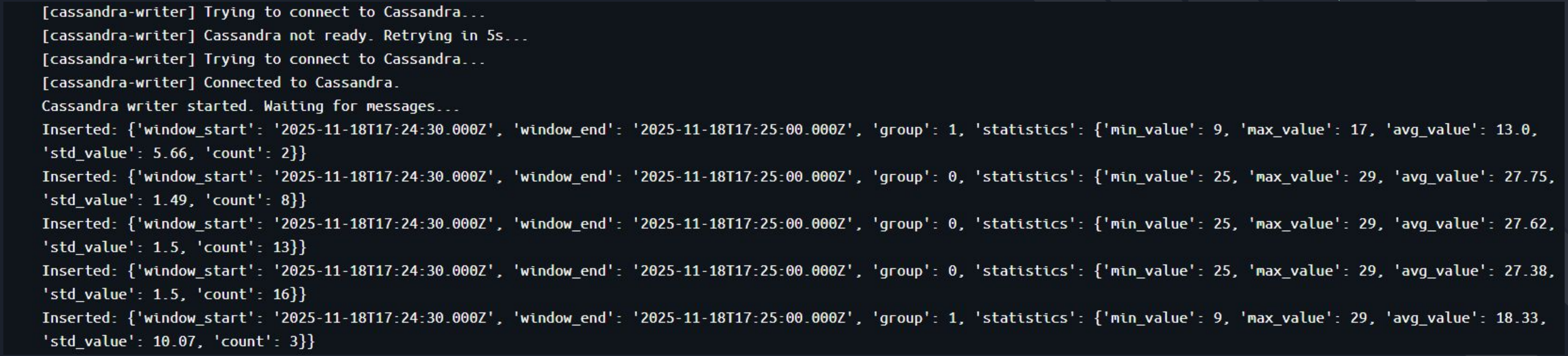
- Main i primer pokretanja



```
# ------------------------------
# Main
# ------------------------------
if __name__ == '__main__':
    args = parse_args()
    print(f"[consumer] Started with arguments: {args}")

    spark = initialize_spark()
    df_stream = parse_kafka_stream(spark)

    df_filtered = apply_filters(df_stream, args.filter)
    df_stats = compute_statistics(df_filtered, args)

    write_to_kafka(df_stats)


# spark-submit spark_consumer.py \
#     --window_duration 60 \
#     --window_type tumbling \
#     --group_by DEPARTING_AIRPORT \
#     --target_col PLANE_AGE \
#     --filter "CARRIER_NAME=='Delta'" \
#     --filter "DEP_DEL15==1"
```

Cassandra writer

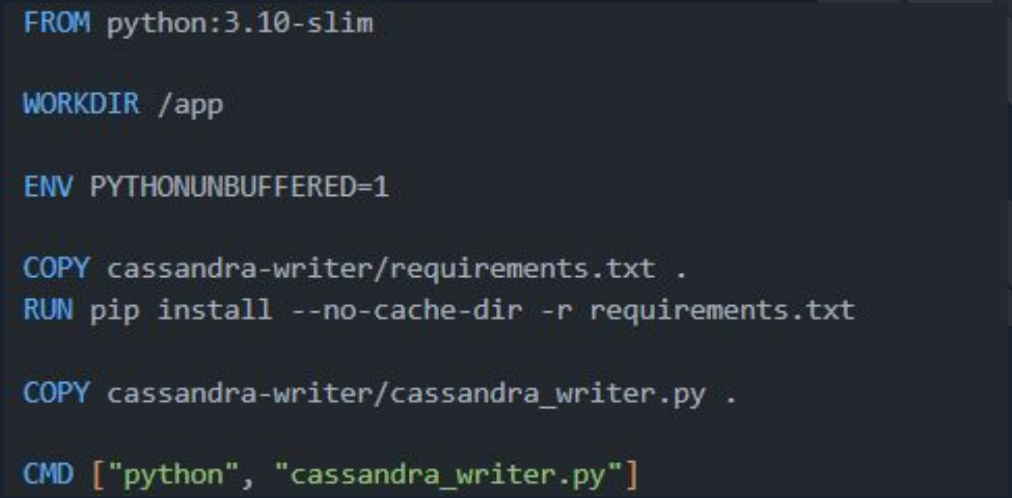
- Čita poruke sa OUTPUT_TOPIC
- Kreira keyspace i tabelu u Cassandri ako ne postoje
- Ubacuje podatke u tabelu TABLE
- Sadrži .py fajl, Dockerfile i requirements.txt
- Praćenje učitavanja i slanja podataka na Kafka topic:
  - `docker logs -f project2-cassandra-writer`



```
[cassandra-writer] Trying to connect to Cassandra...
[cassandra-writer] Cassandra not ready. Retrying in 5s...
[cassandra-writer] Trying to connect to Cassandra...
[cassandra-writer] Connected to Cassandra.
Cassandra writer started. Waiting for messages...
Inserted: {'window_start': '2025-11-18T17:24:30.000Z', 'window_end': '2025-11-18T17:25:00.000Z', 'group': 1, 'statistics': {'min_value': 9, 'max_value': 17, 'avg_value': 13.0,
'std_value': 5.66, 'count': 2}}
Inserted: {'window_start': '2025-11-18T17:24:30.000Z', 'window_end': '2025-11-18T17:25:00.000Z', 'group': 0, 'statistics': {'min_value': 25, 'max_value': 29, 'avg_value': 27.75,
'std_value': 1.49, 'count': 8}}
Inserted: {'window_start': '2025-11-18T17:24:30.000Z', 'window_end': '2025-11-18T17:25:00.000Z', 'group': 0, 'statistics': {'min_value': 25, 'max_value': 29, 'avg_value': 27.62,
'std_value': 1.5, 'count': 13}}
Inserted: {'window_start': '2025-11-18T17:24:30.000Z', 'window_end': '2025-11-18T17:25:00.000Z', 'group': 0, 'statistics': {'min_value': 25, 'max_value': 29, 'avg_value': 27.38,
'std_value': 1.5, 'count': 16}}
Inserted: {'window_start': '2025-11-18T17:24:30.000Z', 'window_end': '2025-11-18T17:25:00.000Z', 'group': 1, 'statistics': {'min_value': 9, 'max_value': 29, 'avg_value': 18.33,
'std_value': 10.07, 'count': 3}}
```

Cassandra writer

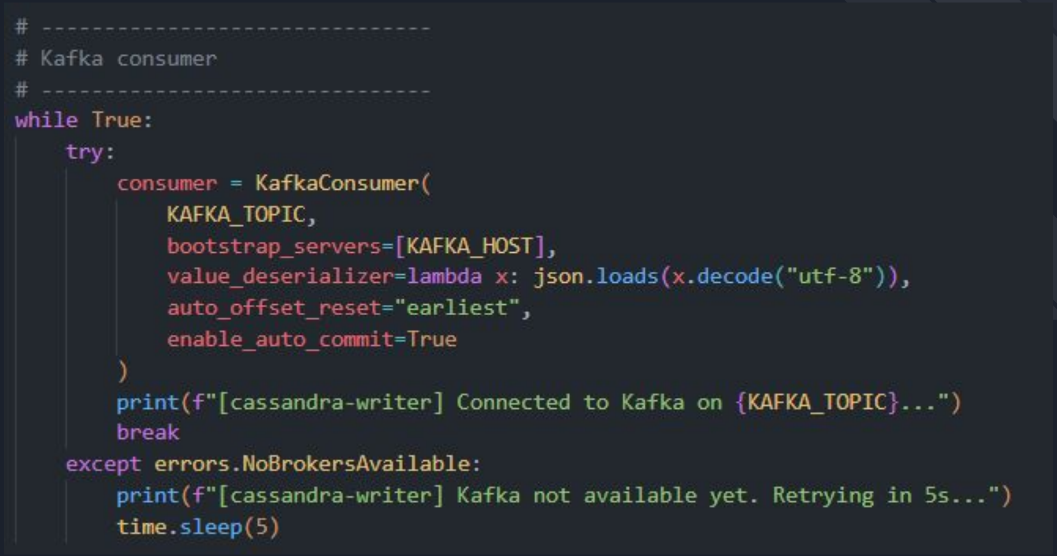
- Dockerfile



```
FROM python:3.10-slim

WORKDIR /app

ENV PYTHONUNBUFFERED=1

COPY cassandra-writer/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY cassandra-writer/cassandra_writer.py .

CMD ["python", "cassandra_writer.py"]
```

Cassandra writer

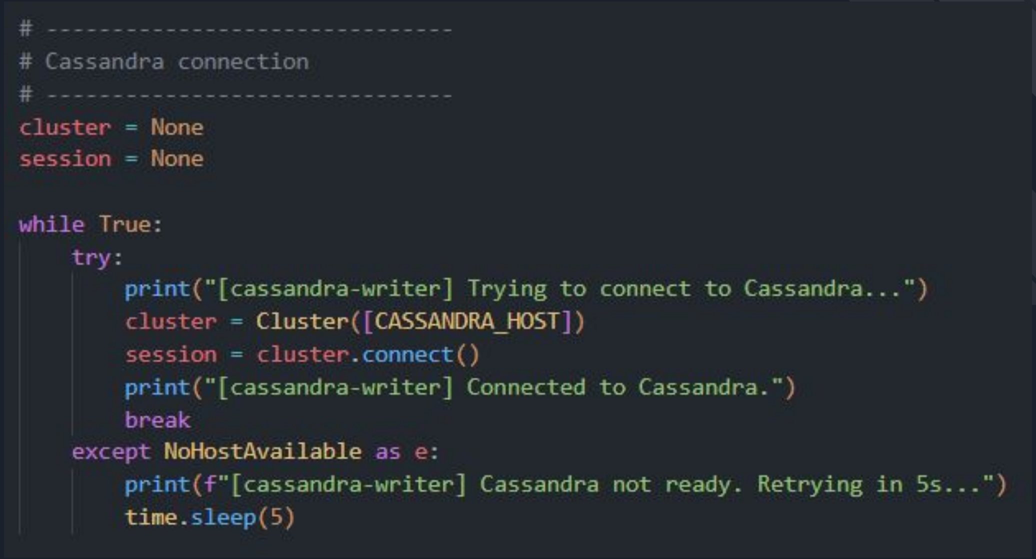
- Konektovanje na Kafka OUTPUT_TOPIC



```python
# ------------------------------
# Kafka consumer
# ------------------------------
while True:
    try:
        consumer = KafkaConsumer(
            KAFKA_TOPIC,
            bootstrap_servers=[KAFKA_HOST],
            value_deserializer=lambda x: json.loads(x.decode("utf-8")),
            auto_offset_reset="earliest",
            enable_auto_commit=True
        )
        print(f"[cassandra-writer] Connected to Kafka on {KAFKA_TOPIC}...")
        break
    except errors.NoBrokersAvailable:
        print(f"[cassandra-writer] Kafka not available yet. Retrying in 5s...")
        time.sleep(5)
```

Cassandra writer

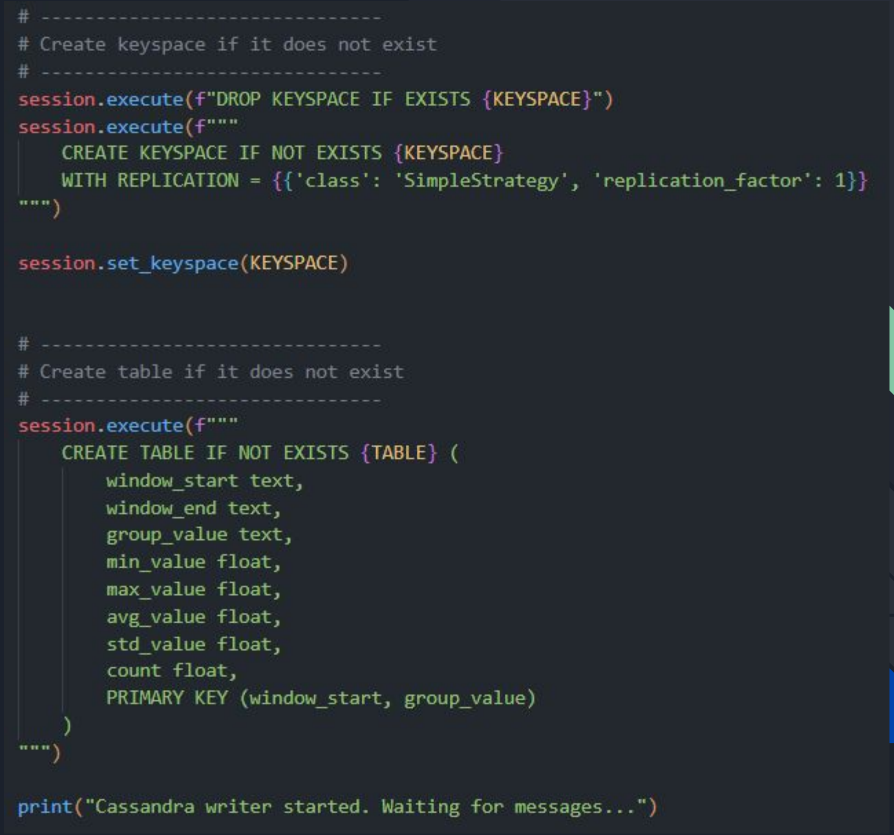
- Konektovanje na Cassandra bazu



```python
# ------------------------------
# Cassandra connection
# ------------------------------
cluster = None
session = None

while True:
    try:
        print("[cassandra-writer] Trying to connect to Cassandra...")
        cluster = Cluster([CASSANDRA_HOST])
        session = cluster.connect()
        print("[cassandra-writer] Connected to Cassandra.")
        break
    except NoHostAvailable as e:
        print(f"[cassandra-writer] Cassandra not ready. Retrying in 5s...")
        time.sleep(5)
```

Cassandra writer

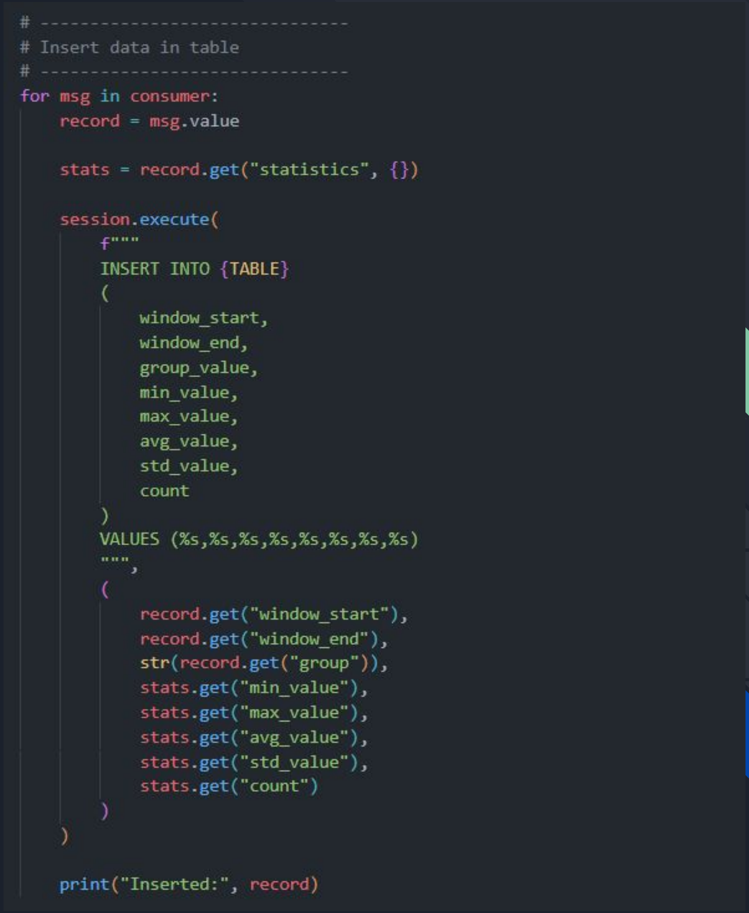
- Kreiranje keyspace-a i tabele



```
# ------------------------------
# Create keyspace if it does not exist
# ------------------------------
session.execute(f"DROP KEYSPACE IF EXISTS {KEYSPACE}")
session.execute(f"""
    CREATE KEYSPACE IF NOT EXISTS {KEYSPACE}
    WITH REPLICATION = {{'class': 'SimpleStrategy', 'replication_factor': 1}}
""")

session.set_keyspace(KEYSPACE)


# ------------------------------
# Create table if it does not exist
# ------------------------------
session.execute(f"""
    CREATE TABLE IF NOT EXISTS {TABLE} (
        window_start text,
        window_end text,
        group_value text,
        min_value float,
        max_value float,
        avg_value float,
        std_value float,
        count float,
        PRIMARY KEY (window_start, group_value)
    )
""")

print("Cassandra writer started. Waiting for messages...")
```

Cassandra writer

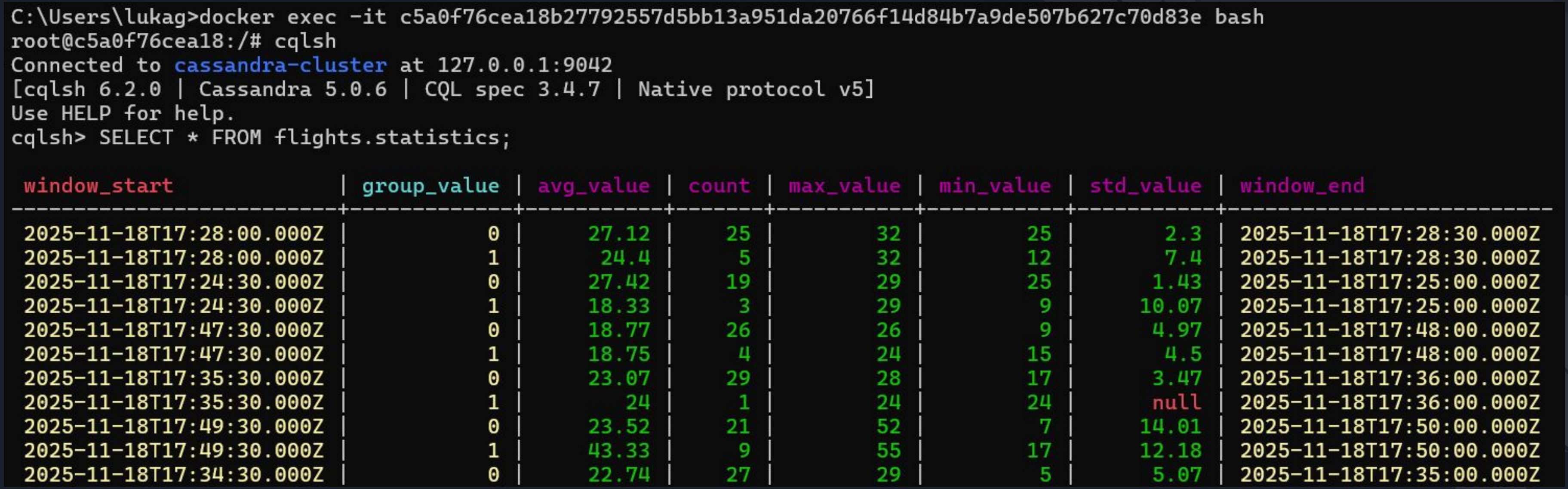
- Upis podataka



```python
# ------------------------------
# Insert data in table
# ------------------------------
for msg in consumer:
    record = msg.value

    stats = record.get("statistics", {})

    session.execute(
        f"""
        INSERT INTO {TABLE}
        (
            window_start,
            window_end,
            group_value,
            min_value,
            max_value,
            avg_value,
            std_value,
            count
        )
        VALUES (%s,%s,%s,%s,%s,%s,%s,%s)
        """,
        (
            record.get("window_start"),
            record.get("window_end"),
            str(record.get("group")),
            stats.get("min_value"),
            stats.get("max_value"),
            stats.get("avg_value"),
            stats.get("std_value"),
            stats.get("count")
        )
    )

    print("Inserted:", record)
```

Rezultati - Cassandra

- Pristup terminalu kontejnera: `docker exec -it <name> bash`
- Otvaranje cqlsh: `cqlsh`
- Pregled tabele: `SELECT * FROM flights.statistics;`



```
C:\Users\lukag>docker exec -it c5a0f76cea18b27792557d5bb13a951da20766f14d84b7a9de507b627c70d83e bash
root@c5a0f76cea18:/# cqlsh
Connected to cassandra-cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> SELECT * FROM flights.statistics;

 window_start             | group_value | avg_value | count | max_value | min_value | std_value | window_end
--------------------------+-------------+-----------+-------+-----------+-----------+-----------+--------------------------
 2025-11-18T17:28:00.000Z |           0 |     27.12 |    25 |        32 |        25 |       2.3 | 2025-11-18T17:28:30.000Z
 2025-11-18T17:28:00.000Z |           1 |      24.4 |     5 |        32 |        12 |       7.4 | 2025-11-18T17:28:30.000Z
 2025-11-18T17:24:30.000Z |           0 |     27.42 |    19 |        29 |        25 |      1.43 | 2025-11-18T17:25:00.000Z
 2025-11-18T17:24:30.000Z |           1 |     18.33 |     3 |        29 |         9 |     10.07 | 2025-11-18T17:25:00.000Z
 2025-11-18T17:47:30.000Z |           0 |     18.77 |    26 |        26 |         9 |      4.97 | 2025-11-18T17:48:00.000Z
 2025-11-18T17:47:30.000Z |           1 |     18.75 |     4 |        24 |        15 |       4.5 | 2025-11-18T17:48:00.000Z
 2025-11-18T17:35:30.000Z |           0 |     23.07 |    29 |        28 |        17 |      3.47 | 2025-11-18T17:36:00.000Z
 2025-11-18T17:35:30.000Z |           1 |        24 |     1 |        24 |        24 |      null | 2025-11-18T17:36:00.000Z
 2025-11-18T17:49:30.000Z |           0 |     23.52 |    21 |        52 |         7 |     14.01 | 2025-11-18T17:50:00.000Z
 2025-11-18T17:49:30.000Z |           1 |     43.33 |     9 |        55 |        17 |     12.18 | 2025-11-18T17:50:00.000Z
 2025-11-18T17:34:30.000Z |           0 |     22.74 |    27 |        29 |         5 |      5.07 | 2025-11-18T17:35:00.000Z
```

HVALA NA PAŽNJI!