APIAnalysis

June 21, 2025

[2]:
```
!pip install kafka-python graphviz
```

```
Requirement already satisfied: kafka-python in /opt/conda/lib/python3.11/site-
packages (2.2.13)
Requirement already satisfied: graphviz in /opt/conda/lib/python3.11/site-
packages (0.21)
```

[3]:
```
import pandas as pd
import json
import time
import os
import matplotlib.pyplot as plt
from graphviz import Digraph
from kafka import KafkaProducer, KafkaConsumer
from kafka import KafkaAdminClient
from kafka.admin import KafkaAdminClient, NewTopic
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col, avg
from pyspark.sql.types import StructType, StringType, FloatType
```

[4]:
```
admin = KafkaAdminClient(bootstrap_servers="172.29.16.101:9092")
print("Aktuelle Topics:", admin.list_topics())
```

```
Aktuelle Topics: ['big-data-test', 'github-trending-all', 'github-scraped-
trending', 'hello-world', 'steam-pc-prices', 'flightsTest', 'srp-2022-data',
'energy-consumption2', 'stocks', 'eu_energy_data', 'music', 'roulette', 'github-
trending-all-v1', 'g3-hello-world', 'github-trending-all-v3', 'geizhals-ssd',
'github-trending-all-v2', 'titanic-stream', 'steam-hwsurvey-summary', 'gpu-
topic', 'amadeus_flights', 'current-weather-api', 'wikimedia-changes',
'intraday-data', 'geizhals-gpu', 'srp-data', 'energy-sustainability',
'taxi_samples', 'energy-consumption', 'g3-raw-html-test', 'weather-report',
'geizhals-cpu', 'finanzdaten', 'flights', 'geizhals-ram', '__consumer_offsets',
'music_data']
```

[21]:
```
airports = ['BER', 'CDG', 'IST', 'LHR']
df_all = pd.DataFrame()

for airport in airports:
```

```
    df = pd.read_csv(f"Kafka_Spark/CSVs/amadeus_prices_{airport}.csv")
    df["Abflug"] = airport
    df_all = pd.concat([df_all, df], ignore_index=True)

df_all['FetchedAt'] = pd.to_datetime(df_all['FetchedAt'], errors='coerce')
df_all.dropna(inplace=True)
df_all['FetchedAt'] = df_all['FetchedAt'].dt.strftime('%Y-%m-%d %H:%M')
```

[6]:
```
producer = KafkaProducer(
    bootstrap_servers="172.29.16.101:9092",
    value_serializer=lambda v: json.dumps(v).encode("utf-8")
)
for _, row in df_all.iterrows():
    producer.send('amadeus_flights', row.to_dict())

producer.flush()
print("Daten wurden an Kafka gesendet.")
```

```
Daten wurden an Kafka gesendet.
```

[ ]:
```
consumer = KafkaConsumer( "amadeus_flights",
bootstrap_servers="172.29.16.101:9092", auto_offset_reset="earliest",
value_deserializer=lambda v: json.loads(v.decode("utf-8"))
)
msg = next(iter(consumer))
print("Erste empfangene Nachricht:", msg.value)
```

[10]:
```
spark = SparkSession.builder \
    .appName("AmadeusFlights") \
    .config("spark.master", "local[*]") \
    .getOrCreate()
```

[15]:
```
kafka_bootstrap = "172.29.16.101:9092"
topic_name = "amadeus_flights"

df_kafka = (
    spark.readStream
    .format("kafka")
    .option("kafka.bootstrap.servers", kafka_bootstrap)
    .option("subscribe", topic_name) \
    .option("startingOffsets", "earliest") \
    .load()
)


schema = StructType() \
    .add("FetchedAt", StringType()) \
```

```
    .add("Destination", StringType()) \
    .add("MinPrice", FloatType()) \
    .add("Abflug", StringType())

df_json = df_kafka.selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("data")) \
    .select("data.*")

df_fco = df_json.filter(col("Destination") == "FCO")

df_avg = df_fco.groupBy("Abflug").agg(avg("MinPrice").
 ↪alias("Durchschnittspreis"))

output_path = "output/amadeus_fco_avg.csv"
query = df_avg.writeStream \
    .outputMode("complete") \
    .format("csv") \
    .option("path", "output/amadeus_fco_avg/") \
    .option("checkpointLocation", "output/checkpoints/") \
    .start()

query.awaitTermination()
```

```
---------------------------------------------------------------------------
AnalysisException                         Traceback (most recent call last)
Cell In[15], line 10
      1 kafka_bootstrap = "172.29.16.101:9092"
      2 topic_name = "amadeus_flights"
      4 df_kafka = (
      5     spark.readStream
      6     .format("kafka")
      7     .option("kafka.bootstrap.servers", kafka_bootstrap)
      8     .option("subscribe", topic_name) \
      9     .option("startingOffsets", "earliest") \
---> 10     .load()
     11 )
     14 schema = StructType() \
     15     .add("FetchedAt", StringType()) \
     16     .add("Destination", StringType()) \
     17     .add("MinPrice", FloatType()) \
     18     .add("Abflug", StringType())
     20 df_json = df_kafka.selectExpr("CAST(value AS STRING)") \
     21     .select(from_json(col("value"), schema).alias("data")) \
     22     .select("data.*")

File /usr/local/spark/python/pyspark/sql/streaming/readwriter.py:304, in␣
 ↪DataStreamReader.load(self, path, format, schema, **options)
```

```
    302     return self._df(self._jreader.load(path))
    303 else:
--> 304     return self._df(self._jreader.load())

File /usr/local/spark/python/lib/py4j-0.10.9.7-src.zip/py4j/java_gateway.py:
 ↪1322, in JavaMember.__call__(self, *args)
   1316 command = proto.CALL_COMMAND_NAME +\
   1317     self.command_header +\
   1318     args_command +\
   1319     proto.END_COMMAND_PART
   1321 answer = self.gateway_client.send_command(command)
-> 1322 return_value = get_return_value(
   1323     answer, self.gateway_client, self.target_id, self.name)
   1325 for temp_arg in temp_args:
   1326     if hasattr(temp_arg, "_detach"):

File /usr/local/spark/python/pyspark/errors/exceptions/captured.py:185, in␣
 ↪capture_sql_exception.<locals>.deco(*a, **kw)
    181 converted = convert_exception(e.java_exception)
    182 if not isinstance(converted, UnknownException):
    183     # Hide where the exception came from that shows a non-Pythonic
    184     # JVM exception message.
--> 185     raise converted from None
    186 else:
    187     raise

AnalysisException: Failed to find data source: kafka. Please deploy the␣
 ↪application as per the deployment section of Structured Streaming + Kafka␣
 ↪Integration Guide.
```

[ ]:
```
df_spark_result = pd.read_csv("output/amadeus_fco_avg.csv")

plt.figure(figsize=(8, 5))
bars = plt.bar(df_spark_result['Abflug'],␣
 ↪df_spark_result['Durchschnittspreis'], color='skyblue')
plt.ylabel("Durchschnittspreis (€)")
plt.title("Durchschnittspreise nach FCO - Amadeus (Spark-Auswertung)")

for bar, preis in zip(bars, df_spark_result['Durchschnittspreis']):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 1, f"{preis:.
 ↪0f} €",
             ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()
```

[ ]:
```
dot = Digraph()
dot.node('A', 'Amadeus CSVs (BER, CDG, IST, LHR)')
dot.node('B', 'Kafka Producer')
dot.node('C', 'Kafka Topic: amadeus_flights')
dot.node('D', 'Spark Consumer & ETL')
dot.node('E', 'Durchschnittspreise (CSV)')
dot.node('F', 'Visualisierung')

dot.edges(['AB', 'BC', 'CD', 'DE', 'EF'])
dot.render('data_flow_amadeus', format='png', cleanup=False)
dot
```