Assignment 9.1

Author: Anjani Bonda

Date: 5/13/2023

In [13]:
```
import os
import shutil
import json
from pathlib import Path

import pandas as pd

from kafka import KafkaProducer, KafkaAdminClient
from kafka.admin.new_topic import NewTopic
from kafka.errors import TopicAlreadyExistsError
from kafka import KafkaConsumer

from pyspark.sql import SparkSession
from pyspark.streaming import StreamingContext
from pyspark import SparkConf
from pyspark.sql.functions import window, from_json, col
from pyspark.sql.types import StringType, TimestampType, DoubleType, StructFiel
from pyspark.sql.functions import udf

current_dir = Path(os.getcwd()).absolute()
checkpoint_dir = current_dir.joinpath('checkpoints')
locations_checkpoint_dir = checkpoint_dir.joinpath('locations')
accelerations_checkpoint_dir = checkpoint_dir.joinpath('accelerations')

if locations_checkpoint_dir.exists():
    shutil.rmtree(locations_checkpoint_dir)

if accelerations_checkpoint_dir.exists():
    shutil.rmtree(accelerations_checkpoint_dir)

locations_checkpoint_dir.mkdir(parents=True, exist_ok=True)
accelerations_checkpoint_dir.mkdir(parents=True, exist_ok=True)
```

Configuration Parameters

In [14]:
```
config = dict(
    bootstrap_servers=['kafka.kafka.svc.cluster.local:9092'],
    first_name='Anjani',
    last_name='Bonda'
)

config['client_id'] = '{}{}'.format(
    config['last_name'],
    config['first_name']
)
config['topic_prefix'] = '{}{}'.format(
    config['last_name'],
    config['first_name']
```

```
)

config['locations_topic'] = '{}-locations'.format(config['topic_prefix'])
config['accelerations_topic'] = '{}-accelerations'.format(config['topic_prefix'
config['simple_topic'] = '{}-simple'.format(config['topic_prefix'])

config
```

Out[14]:
```
{'bootstrap_servers': ['kafka.kafka.svc.cluster.local:9092'],
 'first_name': 'Anjani',
 'last_name': 'Bonda',
 'client_id': 'BondaAnjani',
 'topic_prefix': 'BondaAnjani',
 'locations_topic': 'BondaAnjani-locations',
 'accelerations_topic': 'BondaAnjani-accelerations',
 'simple_topic': 'BondaAnjani-simple'}
```

Create Topic Utility Function

The `create_kafka_topic` helps create a Kafka topic based on your configuration settings. For instance, if your first name is *John* and your last name is *Doe*, `create_kafka_topic('locations')` will create a topic with the name `DoeJohn-locations`. The function will not create the topic if it already exists.

In [15]:
```
def create_kafka_topic(topic_name, config=config, num_partitions=1, replicatior
    bootstrap_servers = config['bootstrap_servers']
    client_id = config['client_id']
    topic_prefix = config['topic_prefix']
    name = '{}-{}'.format(topic_prefix, topic_name)

    admin_client = KafkaAdminClient(
        bootstrap_servers=bootstrap_servers,
        client_id=client_id
    )

    topic = NewTopic(
        name=name,
        num_partitions=num_partitions,
        replication_factor=replication_factor
    )

    topic_list = [topic]
    try:
        admin_client.create_topics(new_topics=topic_list)
        print('Created topic "{}"'.format(name))
    except TopicAlreadyExistsError as e:
        print('Topic "{}" already exists'.format(name))

create_kafka_topic('simple')
```

```
Topic "BondaAnjani-simple" already exists
```

In [16]:
```
spark = SparkSession\
    .builder\
    .appName("Assignment09")\
    .getOrCreate()

df_locations = spark \
```

```
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "kafka.kafka.svc.cluster.local:9092") \
  .option("subscribe", config['locations_topic']) \
  .load()
```

TODO: Create a data frame called `df_accelerations` that reads from the accelerations topic you published to in assignment 8. In order to read data from this topic, make sure that you are running the notebook you created in assignment 8 that publishes acceleration and location data to the `LastnameFirstname-simple` topic.

In [17]:
```
df_accelerations = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "kafka.kafka.svc.cluster.local:9092") \
  .option("subscribe", config['accelerations_topic']) \
  .load()
```

TODO: Create two streaming queries, `ds_locations` and `ds_accelerations` that publish to the `LastnameFirstname-simple` topic. See http://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#starting-streaming-queries and http://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html for more information.

In [18]:
```
ds_locations = df_locations \
    .writeStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka.kafka.svc.cluster.local:9092") \
    .option("topic", config['simple_topic']) \
    .option("checkpointLocation", "/tmp/venkidusamykesavadithya/checkpoint") \
    .start()

ds_accelerations = df_accelerations \
    .writeStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka.kafka.svc.cluster.local:9092") \
    .option("topic", config['simple_topic']) \
    .option("checkpointLocation", "/tmp/venkidusamykesavadithya/checkpoint") \
    .start()

try:
    ds_locations.awaitTermination()
    ds_accelerations.awaitTermination()
except KeyboardInterrupt:
    print("STOPPING STREAMING DATA")
```

```
23/05/15 04:13:05 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not
supported in streaming DataFrames/Datasets and will be disabled.
23/05/15 04:13:05 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not
supported in streaming DataFrames/Datasets and will be disabled.
23/05/15 04:13:05 WARN StreamingQueryManager: Stopping existing streaming quer
y [id=74bda713-7682-402c-8bd3-304823b652c1, runId=e8cea867-cc01-449e-ad22-4219
02d15ad7], as a new run is being started.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'key.deserializer'
was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'value.deserialize
r' was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'enable.auto.commi
t' was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'max.poll.records'
was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'auto.offset.rese
t' was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'key.deserializer'
was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'value.deserialize
r' was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'enable.auto.commi
t' was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'max.poll.records'
was supplied but isn't a known config.
23/05/15 04:13:05 WARN AdminClientConfig: The configuration 'auto.offset.rese
t' was supplied but isn't a known config.
23/05/15 04:13:05 ERROR MicroBatchExecution: Query [id = 74bda713-7682-402c-8b
d3-304823b652c1, runId = 58ad248c-ee6e-480b-abd7-be480c1836f5] terminated with
error
java.lang.NoClassDefFoundError: org/apache/kafka/clients/admin/OffsetSpec
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.$anonfun$fetch
LatestOffsets$2(KafkaOffsetReaderAdmin.scala:298)
        at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.sca
la:286)
        at scala.collection.Iterator.foreach(Iterator.scala:943)
        at scala.collection.Iterator.foreach$(Iterator.scala:943)
        at scala.collection.AbstractIterator.foreach(Iterator.scala:1431)
        at scala.collection.IterableLike.foreach(IterableLike.scala:74)
        at scala.collection.IterableLike.foreach$(IterableLike.scala:73)
        at scala.collection.AbstractIterable.foreach(Iterable.scala:56)
        at scala.collection.TraversableLike.map(TraversableLike.scala:286)
        at scala.collection.TraversableLike.map$(TraversableLike.scala:279)
        at scala.collection.mutable.AbstractSet.scala$collection$SetLike$$supe
r$map(Set.scala:50)
        at scala.collection.SetLike.map(SetLike.scala:105)
        at scala.collection.SetLike.map$(SetLike.scala:105)
        at scala.collection.mutable.AbstractSet.map(Set.scala:50)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.$anonfun$fetch
LatestOffsets$1(KafkaOffsetReaderAdmin.scala:298)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.$anonfun$parti
tionsAssignedToAdmin$1(KafkaOffsetReaderAdmin.scala:501)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.withRetries(Ka
fkaOffsetReaderAdmin.scala:518)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.partitionsAssi
gnedToAdmin(KafkaOffsetReaderAdmin.scala:498)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.fetchLatestOff
sets(KafkaOffsetReaderAdmin.scala:297)
        at org.apache.spark.sql.kafka010.KafkaMicroBatchStream.$anonfun$getOrC
reateInitialPartitionOffsets$1(KafkaMicroBatchStream.scala:251)
```

```
        at scala.Option.getOrElse(Option.scala:189)
        at org.apache.spark.sql.kafka010.KafkaMicroBatchStream.getOrCreateInit
ialPartitionOffsets(KafkaMicroBatchStream.scala:246)
        at org.apache.spark.sql.kafka010.KafkaMicroBatchStream.initialOffset(K
afkaMicroBatchStream.scala:98)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$getStartOffset$2(MicroBatchExecution.scala:455)
        at scala.Option.getOrElse(Option.scala:189)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.getSta
rtOffset(MicroBatchExecution.scala:455)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$constructNextBatch$4(MicroBatchExecution.scala:489)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken(ProgressReporter.scala:411)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken$(ProgressReporter.scala:409)
        at org.apache.spark.sql.execution.streaming.StreamExecution.reportTime
Taken(StreamExecution.scala:67)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$constructNextBatch$2(MicroBatchExecution.scala:488)
        at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.sca
la:286)
        at scala.collection.Iterator.foreach(Iterator.scala:943)
        at scala.collection.Iterator.foreach$(Iterator.scala:943)
        at scala.collection.AbstractIterator.foreach(Iterator.scala:1431)
        at scala.collection.IterableLike.foreach(IterableLike.scala:74)
        at scala.collection.IterableLike.foreach$(IterableLike.scala:73)
        at scala.collection.AbstractIterable.foreach(Iterable.scala:56)
        at scala.collection.TraversableLike.map(TraversableLike.scala:286)
        at scala.collection.TraversableLike.map$(TraversableLike.scala:279)
        at scala.collection.AbstractTraversable.map(Traversable.scala:108)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$constructNextBatch$1(MicroBatchExecution.scala:477)
        at scala.runtime.java8.JFunction0$mcZ$sp.apply(JFunction0$mcZ$sp.java:
23)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.withPr
ogressLocked(MicroBatchExecution.scala:802)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.constr
uctNextBatch(MicroBatchExecution.scala:473)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$runActivatedStream$2(MicroBatchExecution.scala:266)
        at scala.runtime.java8.JFunction0$mcV$sp.apply(JFunction0$mcV$sp.java:
23)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken(ProgressReporter.scala:411)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken$(ProgressReporter.scala:409)
        at org.apache.spark.sql.execution.streaming.StreamExecution.reportTime
Taken(StreamExecution.scala:67)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$runActivatedStream$1(MicroBatchExecution.scala:247)
        at org.apache.spark.sql.execution.streaming.ProcessingTimeExecutor.exe
cute(TriggerExecutor.scala:67)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.runAct
ivatedStream(MicroBatchExecution.scala:237)
        at org.apache.spark.sql.execution.streaming.StreamExecution.$anonfun$r
unStream$1(StreamExecution.scala:306)
        at scala.runtime.java8.JFunction0$mcV$sp.apply(JFunction0$mcV$sp.java:
23)
        at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:82
```

```
7)
        at org.apache.spark.sql.execution.streaming.StreamExecution.org$apache
$spark$sql$execution$streaming$StreamExecution$$runStream(StreamExecution.scal
a:284)
        at org.apache.spark.sql.execution.streaming.StreamExecution$$anon$1.ru
n(StreamExecution.scala:207)
Caused by: java.lang.ClassNotFoundException: org.apache.kafka.clients.admin.Of
fsetSpec
        ... 58 more
Exception in thread "stream execution thread for [id = 74bda713-7682-402c-8bd3
-304823b652c1, runId = 58ad248c-ee6e-480b-abd7-be480c1836f5]" java.lang.NoClas
sDefFoundError: org/apache/kafka/clients/admin/OffsetSpec
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.$anonfun$fetch
LatestOffsets$2(KafkaOffsetReaderAdmin.scala:298)
        at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.sca
la:286)
        at scala.collection.Iterator.foreach(Iterator.scala:943)
        at scala.collection.Iterator.foreach$(Iterator.scala:943)
        at scala.collection.AbstractIterator.foreach(Iterator.scala:1431)
        at scala.collection.IterableLike.foreach(IterableLike.scala:74)
        at scala.collection.IterableLike.foreach$(IterableLike.scala:73)
        at scala.collection.AbstractIterable.foreach(Iterable.scala:56)
        at scala.collection.TraversableLike.map(TraversableLike.scala:286)
        at scala.collection.TraversableLike.map$(TraversableLike.scala:279)
        at scala.collection.mutable.AbstractSet.scala$collection$SetLike$$supe
r$map(Set.scala:50)
        at scala.collection.SetLike.map(SetLike.scala:105)
        at scala.collection.SetLike.map$(SetLike.scala:105)
        at scala.collection.mutable.AbstractSet.map(Set.scala:50)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.$anonfun$fetch
LatestOffsets$1(KafkaOffsetReaderAdmin.scala:298)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.$anonfun$parti
tionsAssignedToAdmin$1(KafkaOffsetReaderAdmin.scala:501)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.withRetries(Ka
fkaOffsetReaderAdmin.scala:518)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.partitionsAssi
gnedToAdmin(KafkaOffsetReaderAdmin.scala:498)
        at org.apache.spark.sql.kafka010.KafkaOffsetReaderAdmin.fetchLatestOff
sets(KafkaOffsetReaderAdmin.scala:297)
        at org.apache.spark.sql.kafka010.KafkaMicroBatchStream.$anonfun$getOrC
reateInitialPartitionOffsets$1(KafkaMicroBatchStream.scala:251)
        at scala.Option.getOrElse(Option.scala:189)
        at org.apache.spark.sql.kafka010.KafkaMicroBatchStream.getOrCreateInit
ialPartitionOffsets(KafkaMicroBatchStream.scala:246)
        at org.apache.spark.sql.kafka010.KafkaMicroBatchStream.initialOffset(K
afkaMicroBatchStream.scala:98)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$getStartOffset$2(MicroBatchExecution.scala:455)
        at scala.Option.getOrElse(Option.scala:189)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.getSta
rtOffset(MicroBatchExecution.scala:455)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$constructNextBatch$4(MicroBatchExecution.scala:489)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken(ProgressReporter.scala:411)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken$(ProgressReporter.scala:409)
        at org.apache.spark.sql.execution.streaming.StreamExecution.reportTime
Taken(StreamExecution.scala:67)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
```

```
un$constructNextBatch$2(MicroBatchExecution.scala:488)
        at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.sca
la:286)
        at scala.collection.Iterator.foreach(Iterator.scala:943)
        at scala.collection.Iterator.foreach$(Iterator.scala:943)
        at scala.collection.AbstractIterator.foreach(Iterator.scala:1431)
        at scala.collection.IterableLike.foreach(IterableLike.scala:74)
        at scala.collection.IterableLike.foreach$(IterableLike.scala:73)
        at scala.collection.AbstractIterable.foreach(Iterable.scala:56)
        at scala.collection.TraversableLike.map(TraversableLike.scala:286)
        at scala.collection.TraversableLike.map$(TraversableLike.scala:279)
        at scala.collection.AbstractTraversable.map(Traversable.scala:108)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$constructNextBatch$1(MicroBatchExecution.scala:477)
        at scala.runtime.java8.JFunction0$mcZ$sp.apply(JFunction0$mcZ$sp.java:
23)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.withPr
ogressLocked(MicroBatchExecution.scala:802)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.constr
uctNextBatch(MicroBatchExecution.scala:473)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$runActivatedStream$2(MicroBatchExecution.scala:266)
        at scala.runtime.java8.JFunction0$mcV$sp.apply(JFunction0$mcV$sp.java:
23)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken(ProgressReporter.scala:411)
        at org.apache.spark.sql.execution.streaming.ProgressReporter.reportTim
eTaken$(ProgressReporter.scala:409)
        at org.apache.spark.sql.execution.streaming.StreamExecution.reportTime
Taken(StreamExecution.scala:67)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.$anonf
un$runActivatedStream$1(MicroBatchExecution.scala:247)
        at org.apache.spark.sql.execution.streaming.ProcessingTimeExecutor.exe
cute(TriggerExecutor.scala:67)
        at org.apache.spark.sql.execution.streaming.MicroBatchExecution.runAct
ivatedStream(MicroBatchExecution.scala:237)
        at org.apache.spark.sql.execution.streaming.StreamExecution.$anonfun$r
unStream$1(StreamExecution.scala:306)
        at scala.runtime.java8.JFunction0$mcV$sp.apply(JFunction0$mcV$sp.java:
23)
        at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:82
7)
        at org.apache.spark.sql.execution.streaming.StreamExecution.org$apache
$spark$sql$execution$streaming$StreamExecution$$runStream(StreamExecution.scal
a:284)
        at org.apache.spark.sql.execution.streaming.StreamExecution$$anon$1.ru
n(StreamExecution.scala:207)
Caused by: java.lang.ClassNotFoundException: org.apache.kafka.clients.admin.Of
fsetSpec
        ... 58 more
```

```
---------------------------------------------------------------------------
StreamingQueryException                   Traceback (most recent call last)
Cell In[18], line 19
     17 try:
     18     ds_locations.awaitTermination()
---> 19     ds_accelerations.awaitTermination()
     20 except KeyboardInterrupt:
     21     print("STOPPING STREAMING DATA")

File /opt/conda/lib/python3.10/site-packages/pyspark/sql/streaming/query.py:201, in StreamingQuery.awaitTermination(self, timeout)
    199     return self._jsq.awaitTermination(int(timeout * 1000))
    200 else:
--> 201     return self._jsq.awaitTermination()

File /opt/conda/lib/python3.10/site-packages/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 /opt/conda/lib/python3.10/site-packages/pyspark/errors/exceptions/captured.py:175, in capture_sql_exception.<locals>.deco(*a, **kw)
    171 converted = convert_exception(e.java_exception)
    172 if not isinstance(converted, UnknownException):
    173     # Hide where the exception came from that shows a non-Pythonic
    174     # JVM exception message.
--> 175     raise converted from None
    176 else:
    177     raise

StreamingQueryException: [STREAM_FAILED] Query [id = 74bda713-7682-402c-8bd3-304823b652c1, runId = 58ad248c-ee6e-480b-abd7-be480c1836f5] terminated with exception: org/apache/kafka/clients/admin/OffsetSpec
```

In [ ]: