Assignment 9.1

In [1]:
```
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 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, StructFie
ld, StructType
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

> **TODO:** Change the configuration prameters to the appropriate values for your setup.

In [2]:
```
config = dict(
    bootstrap_servers=['kafka.kafka.svc.cluster.local:9092'],
    first_name='Kyle',
    last_name='Morris'
)

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_prefi
x'])
config['simple_topic'] = '{}-simple'.format(config['topic_prefix'])

config
```

Out[2]:
```
{'bootstrap_servers': ['kafka.kafka.svc.cluster.local:9092'],
 'first_name': 'Kyle',
 'last_name': 'Morris',
 'client_id': 'MorrisKyle',
 'topic_prefix': 'MorrisKyle',
 'locations_topic': 'MorrisKyle-locations',
 'accelerations_topic': 'MorrisKyle-accelerations',
 'simple_topic': 'MorrisKyle-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 [3]:
```
def create_kafka_topic(topic_name, config=config, num_partitions=1, replicatio
n_factor=1):
    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 "MorrisKyle-simple" already exists
```

In [4]:
```
spark = SparkSession\
    .builder\
    .appName("Assignment09")\
    .getOrCreate()

df_locations = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", config['bootstrap_servers']) \
  .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 [5]:
```
df_accelerations = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", config['bootstrap_servers']) \
  .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 (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 (http://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html) for more information.

In [6]:
```
ds_locations = df_locations \
    .writeStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka.kafka.svc.cluster.local:9092") \
    .option("topic", config['simple_topic']) \
    .option('checkpointLocation', str(accelerations_checkpoint_dir)) \
    .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', str(locations_checkpoint_dir)) \
    .start()

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

```
---------------------------------------------------------------------------
StreamingQueryException                   Traceback (most recent call last)
<ipython-input-6-e73da87421a1> in <module>
     17
     18 try:
---> 19     ds_locations.awaitTermination()
     20     ds_accelerations.awaitTermination()
     21 except KeyboardInterrupt:

/usr/local/spark/python/pyspark/sql/streaming.py in awaitTermination(self, ti
meout)
    101             return self._jsq.awaitTermination(int(timeout * 1000))
    102         else:
--> 103             return self._jsq.awaitTermination()
    104
    105     @property

/usr/local/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in __cal
l__(self, *args)
   1302
   1303         answer = self.gateway_client.send_command(command)
-> 1304         return_value = get_return_value(
   1305             answer, self.gateway_client, self.target_id, self.name)
   1306

/usr/local/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
    135                 # Hide where the exception came from that shows a non
-Pythonic
    136                 # JVM exception message.
--> 137                 raise_from(converted)
    138             else:
    139                 raise

/usr/local/spark/python/pyspark/sql/utils.py in raise_from(e)

StreamingQueryException: Failed to construct kafka consumer
=== Streaming Query ===
Identifier: [id = ace3dee2-d4a5-4175-b6b3-7664ed194b6e, runId = a1b434dd-0ee0
-4bbf-845c-c773de406dd5]
Current Committed Offsets: {}
Current Available Offsets: {}

Current State: ACTIVE
Thread State: RUNNABLE

Logical Plan:
WriteToMicroBatchDataSource org.apache.spark.sql.kafka010.KafkaStreamingWrite
@1d70f360
+- StreamingDataSourceV2Relation [key#7, value#8, topic#9, partition#10, offs
et#11L, timestamp#12, timestampType#13], org.apache.spark.sql.kafka010.KafkaS
ourceProvider$KafkaScan@6e4a114c, KafkaV2[Subscribe[MorrisKyle-locations]]
```

In [ ]:
```
# This code worked for me once, and then every time since it fails
# to create a kafka consumer. I'm not sure why.
```