In [1]:
```
import json
import uuid

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

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='Theodore',
    last_name='Thompson'
)

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

config
```

Out[2]:
```
{'bootstrap_servers': ['kafka.kafka.svc.cluster.local:9092'],
 'first_name': 'Theodore',
 'last_name': 'Thompson',
 'client_id': 'ThompsonTheodore',
 'topic_prefix': 'ThompsonTheodore'}
```

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, replication_f
    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('locations')
```

```
Topic "ThompsonTheodore-locations" already exists
```

Kafka Producer

The following code creates a `KafkaProducer` object which you can use to send Python objects that are serialized as JSON.

Note: This producer serializes Python objects as JSON. This means that object must be JSON serializable. As an example, Python `DateTime` values are not JSON serializable and must be converted to a string (e.g. ISO 8601) or a numeric value (e.g. a Unix timestamp) before being sent.

In [4]:
```
producer = KafkaProducer(
  bootstrap_servers=config['bootstrap_servers'],
  value_serializer=lambda x: json.dumps(x).encode('utf-8')
)
```

Send Data Function

The `send_data` function sends a Python object to a Kafka topic. This function adds the `topic_prefix` to the topic so `send_data('locations', data)` sends a JSON serialized message to `DoeJohn-locations` . The function also registers callbacks to let you know if the message has been sent or if an error has occured.

In [5]:
```
def on_send_success(record_metadata):
    print('Message sent:\n    Topic: "{}"\n    Partition: {}\n    Offset: {}'.for
        record_metadata.topic,
        record_metadata.partition,
        record_metadata.offset
    ))

def on_send_error(excp):
    print('I am an errback', exc_info=excp)
    # handle exception

def send_data(topic, data, config=config, producer=producer, msg_key=None):
    topic_prefix = config['topic_prefix']
    topic_name = '{}-{}'.format(topic_prefix, topic)

    if msg_key is not None:
        key = msg_key
    else:
        key = uuid.uuid4().hex

    producer.send(
        topic_name,
        value=data,
        key=key.encode('utf-8')
    ).add_callback(on_send_success).add_errback(on_send_error)
```

In [6]:
```
import os
import json
import time
from collections import namedtuple
import heapq
import uuid
import pandas as pd
import s3fs
import pyarrow.parquet as pq

endpoint_url='https://storage.budsc.midwest-datascience.com'
# s3 = s3fs.S3FileSystem(
#     anon=True,
#     client_kwargs={
#         'endpoint_url': endpoint_url
#     }
# )

acceleration_columns = [
    'offset',
    'id',
    'ride_id',
    'uuid',
    'x',
    'y',
    'z',
#     't'
]
Acceleration = namedtuple('Acceleration', acceleration_columns)
def read_accelerations():
    df = pq.ParquetDataset(
        'bdd/accelerations'
    ).read_pandas().to_pandas()

    df = df[acceleration_columns].sort_values(by=['offset'])

    records = [Acceleration(*record) for record in df.to_records(index=False)]

    return records
accelerations = read_accelerations()

location_columns = [
    'offset',
    'id',
    'ride_id',
    'uuid',
    'course',
    'latitude',
    'longitude',
    'geohash',
    'speed',
    'accuracy',
#     't'
]
Location = namedtuple('Location', location_columns)
def read_locations():
    df = pq.ParquetDataset(
        'bdd/locations').read_pandas().to_pandas()

    df = df[location_columns].sort_values(by=['offset'])

    records = [Location(*record) for record in df.to_records(index=False)]

    return records

locations = read_locations()
```

In [7]:
```
example_data = dict(
    key1='value1',
    key2='value2'
)

send_data('locations', locations)
```

In [8]:
```
import sched, time
s = sched.scheduler(time.time, time.sleep)
def do_something(sc):
    print("Doing stuff...")
    # do your stuff
    s.enter(0, 1, send_data, ('locations', locations))
    s.enter(04.5, 1, send_data, ('locations', locations))
    s.enter(3.3, 1, send_data, ('locations', locations))
    s.enter(3.8, 1, send_data, ('locations', locations))
    s.enter(4.3, 1, send_data, ('locations', locations))
    s.enter(3, 1, send_data, ('locations', locations))
    s.enter(3.4, 1, send_data, ('locations', locations))
    s.enter(4.8, 1, send_data, ('locations', locations))
    s.enter(4.3, 1, send_data, ('locations', locations))
    s.enter(3.4, 1, send_data, ('locations', locations))
    s.enter(4, 1, send_data, ('locations', locations))
    s.enter(4.8, 1, send_data, ('locations', locations))
    s.enter(3.9, 1, send_data, ('locations', locations))
    s.enter(4.1, 1, send_data, ('locations', locations))
    s.enter(3, 1, send_data, ('locations', locations))
    print("Done doing stuff...")
#     s.enter(60, 1, do_something, (s,))
#     s.enter(60, 1, do_something, (s,))
#     s.enter(60, 1, do_something, (s,))

s.enter(60, 1, do_something, (s,))
s.run()
```

```
Message sent:
    Topic: "ThompsonTheodore-locations"
    Partition: 0
    Offset: 2
Doing stuff...
Done doing stuff...
Message sent:
    Topic: "ThompsonTheodore-locations"
    Partition: 0
    Offset: 3
Message sent:
    Topic: "ThompsonTheodore-locations"
    Partition: 0
    Offset: 4
Message sent:
    Topic: "ThompsonTheodore-locations"
    Partition: 0
    Offset: 5
Message sent:
    Topic: "ThompsonTheodore-locations"
    Partition: 0
    Offset: 6
Message sent:
    Topic: "ThompsonTheodore-locations"
    Partition: 0
    Offset: 7
Message sent:
    Topic: "ThompsonTheodore-locations"
```

```
        Partition: 0
        Offset: 8
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 9
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 10
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 11
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 12
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 13
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 14
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 15
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 16
    Message sent:
        Topic: "ThompsonTheodore-locations"
        Partition: 0
        Offset: 17
```