NagarajVinay - Producer 8.1

May 9, 2021

[8]:
```
import json
import uuid

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

[9]:
```
config = dict(
    bootstrap_servers=['kafka.kafka.svc.cluster.local:9092'],
    first_name='Vinay',
    last_name='Nagaraj'
)

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

config
```

[9]:
```
{'bootstrap_servers': ['kafka.kafka.svc.cluster.local:9092'],
 'first_name': 'Vinay',
 'last_name': 'Nagaraj',
 'client_id': 'NagarajVinay',
 'topic_prefix': 'NagarajVinay'}
```

0.0.1 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 *Vinay* and your last name is *Nagaraj*, `create_kafka_topic('locations')` will create a topic with the name `DoeJohn-locations`. The function will not create the topic if it already exists.

[10]:
```
def loadParquet(parq_path):
    pqr = spark.read.parquet(parq_path)
    # Convert from spark dataframe to pandas dataframe
    pqr = pqr.toPandas()
    return pqr

def splitstr(std):
    before, after = str(std).split('.')
    return before, after

def startTimer(results_dir):
    # Loop on time
    print("call function here")
    retval = startTimedParquetStreamUpdateLoop(results_dir)
    # Stop if time is over and there are no more partitions.
    if ((time.time() - start_time) < 70 and retval == 0):
        t = threading.Timer(interval, startTimer(results_dir))
```

[11]:
```
def create_kafka_topic(topic_name, config=config, num_partitions=1,␣
 ↪replication_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('locations')
```

```
Topic "NagarajVinay-locations" already exists
```

0.0.2 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.

[12]:
```
producer = KafkaProducer(
  bootstrap_servers=config['bootstrap_servers'],
  value_serializer=lambda x: json.dumps(x).encode('utf-8')
)
```

0.0.3 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 `NagarajVinay-locations`. The function also registers callbacks to let you know if the message has been sent or if an error has occured.

[13]:
```
def on_send_success(record_metadata):
    print('Message sent:\n    Topic: "{}"\n    Partition: {}\n    Offset: {}'.
 ↪format(
        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)
```

[14]:
```
example_data = dict(
    key1='value1',
    key2='value2'
)

send_data('locations', example_data)
```

```
Message sent:
    Topic: "NagarajVinay-locations"
    Partition: 0
    Offset: 1
```

[ ]:
```

```