

# Assignment 9 - Consumer

November 5, 2020

## 0.1 Assignment 9 Consumer

The following notebook should help you with debugging and testing assignment 9. It creates a `KafkaConsumer` that subscribes to the `LastnameFirstname-simple`, `LastnameFirstname-windowed`, and `LastnameFirstname-joined` topics and prints any messages.

```
[1]: import json

from kafka import KafkaConsumer
```

### 0.1.1 Configuration Parameters

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

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

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

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

config['simple_topic'] = '{}-simple'.format(config['topic_prefix'])
config['joined_topic'] = '{}-joined'.format(config['topic_prefix'])
config['windowed_topic'] = '{}-windowed'.format(config['topic_prefix'])

config
```

```
[2]: {'bootstrap_servers': ['kafka.kafka.svc.cluster.local:9092'],
      'first_name': 'Brandon',
      'last_name': 'Sams',
```

```

'client_id': 'SamsBrandon',
'topic_prefix': 'SamsBrandon',
'simple_topic': 'SamsBrandon-simple',
'joined_topic': 'SamsBrandon-joined',
'windowed_topic': 'SamsBrandon-windowed'}

```

Close the consumer, waiting indefinitely for any needed cleanup.

```

[3]: def create_kafka_consumer(topics, config=config):
    bootstrap_servers = config['bootstrap_servers']
    client_id = config['client_id']
    topic_prefix = config['topic_prefix']
    topic_list = ['{}-{}'.format(topic_prefix, topic) for topic in topics]

    return KafkaConsumer(
        *topic_list,
        client_id=client_id,
        bootstrap_servers=bootstrap_servers,
        value_deserializer=lambda x: json.loads(x)
    )

consumer = create_kafka_consumer(['simple', 'windowed', 'joined'])

```

Gets a list of this consumer's current subscriptions

```

[4]: consumer.subscription()

```

```

[4]: {'SamsBrandon-joined', 'SamsBrandon-simple', 'SamsBrandon-windowed'}

```

The following function prints messages from the current consumer subscriptions. It will continue until manually stopped.

```

[5]: def print_messages(consumer=consumer):
    try:
        for message in consumer:
            msg_metadata = 'Message metadata: {}:{}:{}'.format(
                message.topic, message.partition, message.offset
            )

            if message.key is not None:
                msg_key = message.key.decode('utf-8')
            else:
                msg_key = ''
            msg_value = json.dumps(message.value, indent=2)
            msg_value = '\n'.join([' {} '.format(value) for value in
↪msg_value.split('\n')])

            print('Message metadata:')

```

```
        print('  Topic: {}'.format(message.topic))
        print('  Partition: {}'.format(message.partition))
        print('  Offset: {}'.format(message.offset))
        print('Message Key: {}'.format(msg_key))
        print('Message Value:')
        print(msg_value)
        print()
    except KeyboardInterrupt:
        print("STOPPING MESSAGE CONSUMER")

print_messages()
```

STOPPING MESSAGE CONSUMER

Close the consumer, waiting indefinitely for any needed cleanup.

```
[6]: consumer.close()
```