Task: A3 Streaming / Messaging with Apache Kafka
Name: Edgardo Andres Panza Penso
Date: 02/Jan/2020

Exercise 1

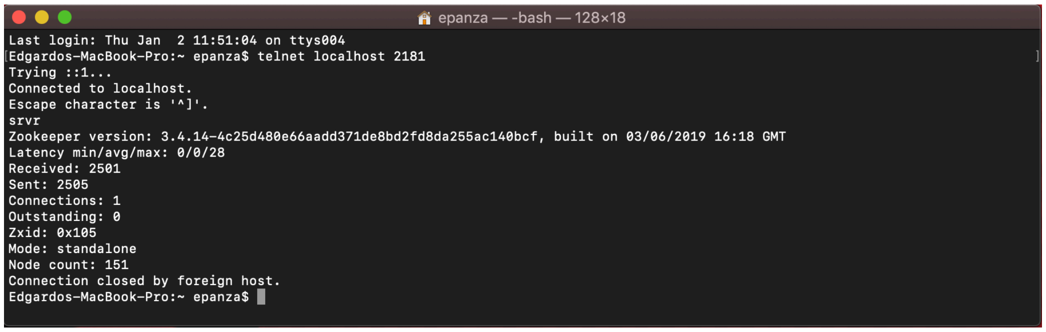
In the git repository you can find the 3 python scripts:

https://github.com/edpape007/Kafka.git

1. First, we have to install / start the zookeeper and kafka server

   $ brew install kafka
   $ brew install zookeeper
   $ zkServer start

   We can test that zookeeper is running with “telnet localhost 2181” and then sending “srvr”

   epanza — -bash — 128×18

   ```
   Last login: Thu Jan  2 11:51:04 on ttys004
   Edgardos-MacBook-Pro:~ epanza$ telnet localhost 2181
   Trying ::1...
   Connected to localhost.
   Escape character is '^]'.
   srvr
   Zookeeper version: 3.4.14-4c25d480e66aadd371de8bd2fd8da255ac140bcf, built on 03/06/2019 16:18 GMT
   Latency min/avg/max: 0/0/28
   Received: 2501
   Sent: 2505
   Connections: 1
   Outstanding: 0
   Zxid: 0x105
   Mode: standalone
   Node count: 151
   Connection closed by foreign host.
   Edgardos-MacBook-Pro:~ epanza$
   ```

   Then we can start the kafka server:

   $ kafka-server-start /usr/local/etc/kafka/server.properties

2. We need to create the topic “KafkaTask”

   $ kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic KafkaTask

3. Open a new terminal and run de script “producerA.py”

   ```
   from time import sleep
   from json import dumps
   from kafka import KafkaProducer
   ```

```
producer =
KafkaProducer(bootstrap_servers=['localhost:9092'],
                         value_serializer=lambda x:
                         dumps(x).encode('utf-8'))

for e in range(100000):
    if (e % 2) == 0:
        data = {'producer A - number': e}
        producer.send('KafkaTask', value=data)
    sleep(1)
```

4. Open a new terminal and run de script “producerB.py”

```
from time import sleep
from json import dumps
from kafka import KafkaProducer

producer =
KafkaProducer(bootstrap_servers=['localhost:9092'],
                         value_serializer=lambda x:
                         dumps(x).encode('utf-8'))

for e in range(100000):
    data = {'producer B - number' : e}
    producer.send('KafkaTask', value=data)
    sleep(2)
```

5. Open a new terminal and run de script “consumerC.py”

```
from kafka import KafkaConsumer
from pymongo import MongoClient
from json import loads

consumer = KafkaConsumer(
    'KafkaTask',
     bootstrap_servers=['localhost:9092'],
     auto_offset_reset='earliest',
     enable_auto_commit=True,
     group_id='my-group',
     value_deserializer=lambda x: loads(x.decode('utf-8')))

for message in consumer:
    message = message.value
    print(message)
```

Exercise 2

Apache Kafka acts as a buffer so your systems won't crash. Previously, data transformations from external source systems were done in batches often at night. Apache Kafka solves this slow, multi-step process by acting as an intermediary receiving data from source systems and then making this data available to target systems in real time. What's more, your systems won't crash because Apache Kafka is its own separate set of servers (called an Apache Kafka cluster).

Kafka Reduces the need for multiple integrations. Essentially, Apache Kafka reduces the need for multiple integrations–as all your data goes through Apache Kafka. Rather than your developers coding multiple integrations so you can harvest data from different systems, you only have to create one integration with Apache Kafka for each producing system and each consuming system.

Low latency and high throughput. By decoupling your data streams, Apache Kafka lets you consume data when you want it. Without the need for slow integrations, Apache Kafka decreases latency to a mere 10 milliseconds (~10x decrease or more compared to other integrations). This means you can deliver data quickly and in real time. Apache Kafka can also horizontally scale to hundreds of brokers (or servers) within a cluster to manage big data.

Exercise 3

YES