NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

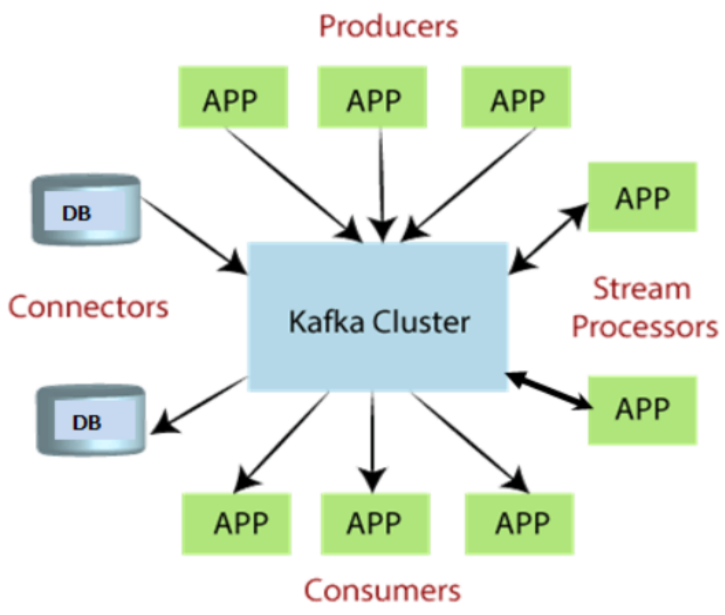
EXPERIMENT 10

AIM: Read streaming data using Kafka.

THEORY:

What is Apache Kafka?

Apache Kafka is a software platform which is based on a distributed streaming process. It is a publish-subscribe messaging system which let exchanging of data between applications, servers, and processors as well. Apache Kafka was originally developed by LinkedIn, and later it was donated to the Apache Software Foundation. Currently, it is maintained by Confluent under Apache Software Foundation. Apache Kafka has resolved the lethargic trouble of data communication between a sender and a receiver.



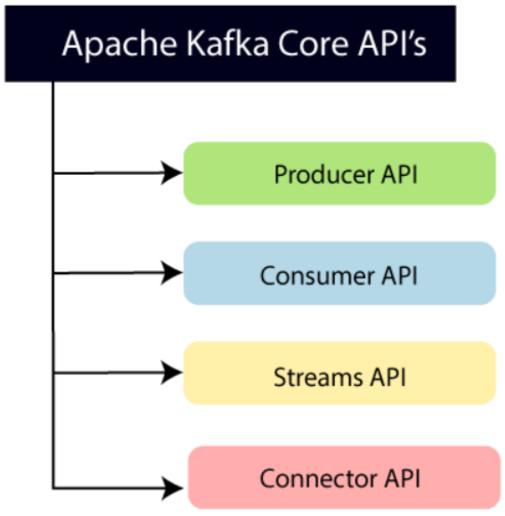
A publish-subscribe messaging system allows a sender to send/write the message and a receiver to read that message. In Apache Kafka, a sender is known as a producer who publishes messages, and a receiver is known as a consumer who consumes that message by subscribing it.

A streaming platform in Kafka has the following key capabilities:

1. As soon as the streams of records occur, it processes it.
2. It works similar to an enterprise messaging system where it publishes and subscribes streams of records.
3. It stores the streams of records in a fault-tolerant durable way.

Apache Kafka Core API:

1. **Producer API:** This API allows/permits an application to publish streams of records to one or more topics. (discussed in later section)
2. **Consumer API:** This API allows an application to subscribe one or more topics and process the stream of records produced to them.
3. **Streams API:** This API allows an application to effectively transform the input streams to the output streams. It permits an application to act as a stream processor which consumes an input stream from one or more topics, and produce an output stream to one or more output topics.
4. **Connector API:** This API executes the reusable producer and consumer APIs with the existing data systems or applications.



Topics and Logs

A topic is considered a key abstraction in Kafka. A topic can be considered a feed name or category where the messages will be published. Each topic is further subdivided into partitions. Partitions split data across different nodes by Kafka brokers. In each of the messages within the partition, there is an ID number assigned, which is technically known as the offset.

Guarantees

In a Kafka messaging system, there are essentially three guarantees:

1. A message sent by a producer to a topic partition is appended to the Kafka log based on the order sent.

2. Consumers see messages in the order of their log storage.

3. For topics with replication factor N, Kafka will tolerate N-1 number of failures without losing the messages previously committed to the log.

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

CODE:

```
import findspark

findspark.init('')

import pyspark

from pyspark import RDD

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

from pyspark.streaming.kafka import KafkaUtils


if name=="main":

   sc = SparkContext(appName="PythonSparkStreamingKafka")

   ssc = StreamingContext(sc,60)

   #Creating Kafka direct stream

   message= KafkaUtils.createDirectStream(ssc, ["testtopic"], {"metadata.broker.list":"|replace
with your Kafka private address|:9092"})

   words=message.map(lambda x:x[1].flatmap(lambda x:x.split(" ")))

   wordcount=words.map(lambda x: (x,1).reduceByKey(lambda a,b:a+b))

   wordcount.ppr int()

   #Starting Spark context

   ssc.start()

   ssc.awaitTermination()
```

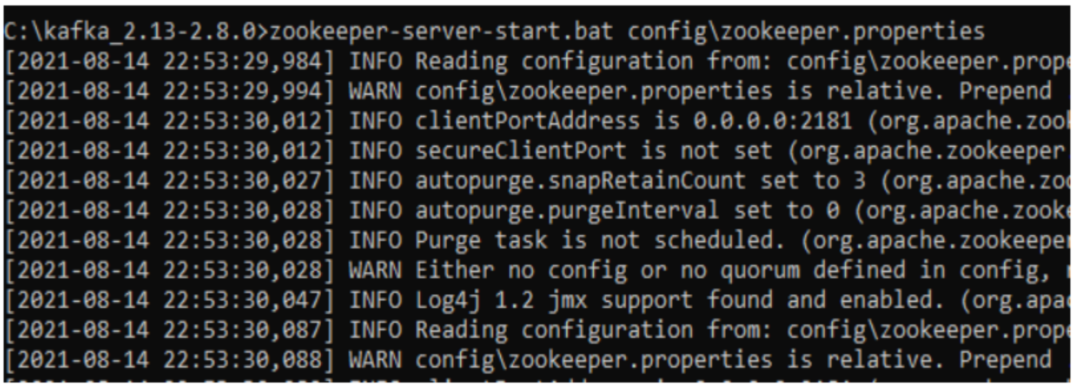
Commands:

1) Open 4 command line terminals

2) On first terminal enter the command:

```
zookeeper-server-start.bat .\config\zookeeper.properties
```

OUTPUT:



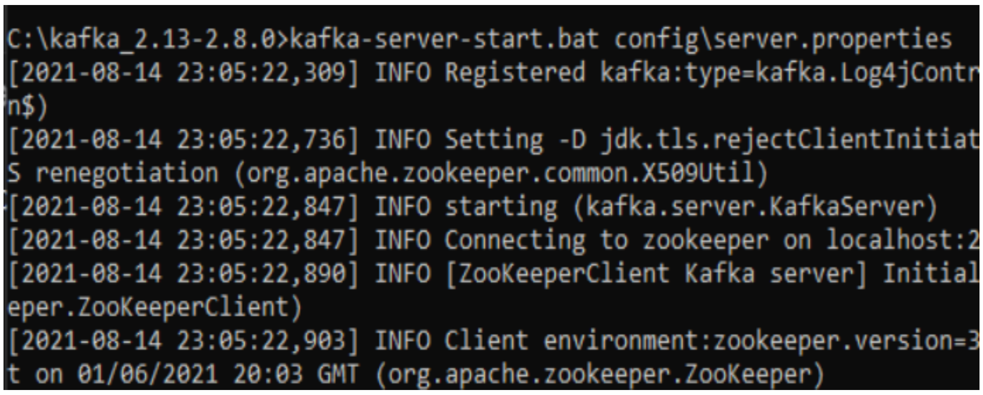
```
C:\kafka_2.13-2.8.0>zookeeper-server-start.bat config\zookeeper.properties
[2021-08-14 22:53:29,984] INFO Reading configuration from: config\zookeeper.prope
[2021-08-14 22:53:29,994] WARN config\zookeeper.properties is relative. Prepend
[2021-08-14 22:53:30,012] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zoo
[2021-08-14 22:53:30,012] INFO secureClientPort is not set (org.apache.zookeeper
[2021-08-14 22:53:30,027] INFO autopurge.snapRetainCount set to 3 (org.apache.zo
[2021-08-14 22:53:30,028] INFO autopurge.purgeInterval set to 0 (org.apache.zook
[2021-08-14 22:53:30,028] INFO Purge task is not scheduled. (org.apache.zookeepe
[2021-08-14 22:53:30,028] WARN Either no config or no quorum defined in config,
[2021-08-14 22:53:30,047] INFO Log4j 1.2 jmx support found and enabled. (org.apa
[2021-08-14 22:53:30,087] INFO Reading configuration from: config\zookeeper.prope
[2021-08-14 22:53:30,088] WARN config\zookeeper.properties is relative. Prepend
```

3) Zookeeper is up and running at the specified port.

On second terminal enter the command:

kafka-server-start.bat .\config\server.properties

OUTPUT:

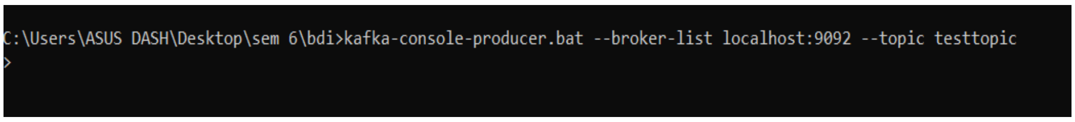


```
C:\kafka_2.13-2.8.0>kafka-server-start.bat config\server.properties
[2021-08-14 23:05:22,309] INFO Registered kafka:type=kafka.Log4jContr
n$)
[2021-08-14 23:05:22,736] INFO Setting -D jdk.tls.rejectClientInitiat
S renegotiation (org.apache.zookeeper.common.X509Util)
[2021-08-14 23:05:22,847] INFO starting (kafka.server.KafkaServer)
[2021-08-14 23:05:22,847] INFO Connecting to zookeeper on localhost:2
[2021-08-14 23:05:22,890] INFO [ZooKeeperClient Kafka server] Initial
eper.ZooKeeperClient)
[2021-08-14 23:05:22,903] INFO Client environment:zookeeper.version=3
t on 01/06/2021 20:03 GMT (org.apache.zookeeper.ZooKeeper)
```

Now, you have the kafka running in one command line window and zookeeper running in another. If you want to start working with them, you will have to open another command line window and start writing commands.

4) On third terminal we run the command

kafka-console-producer.bat --broker-list localhost:9092 --topic testtopic



```
C:\Users\ASUS DASH\Desktop\sem 6\bdi>kafka-console-producer.bat --broker-list localhost:9092 --topic testtopic
>
```

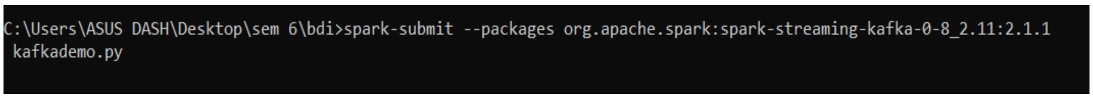
5) On fourth terminal we run the command:

spark-submit --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.1.1

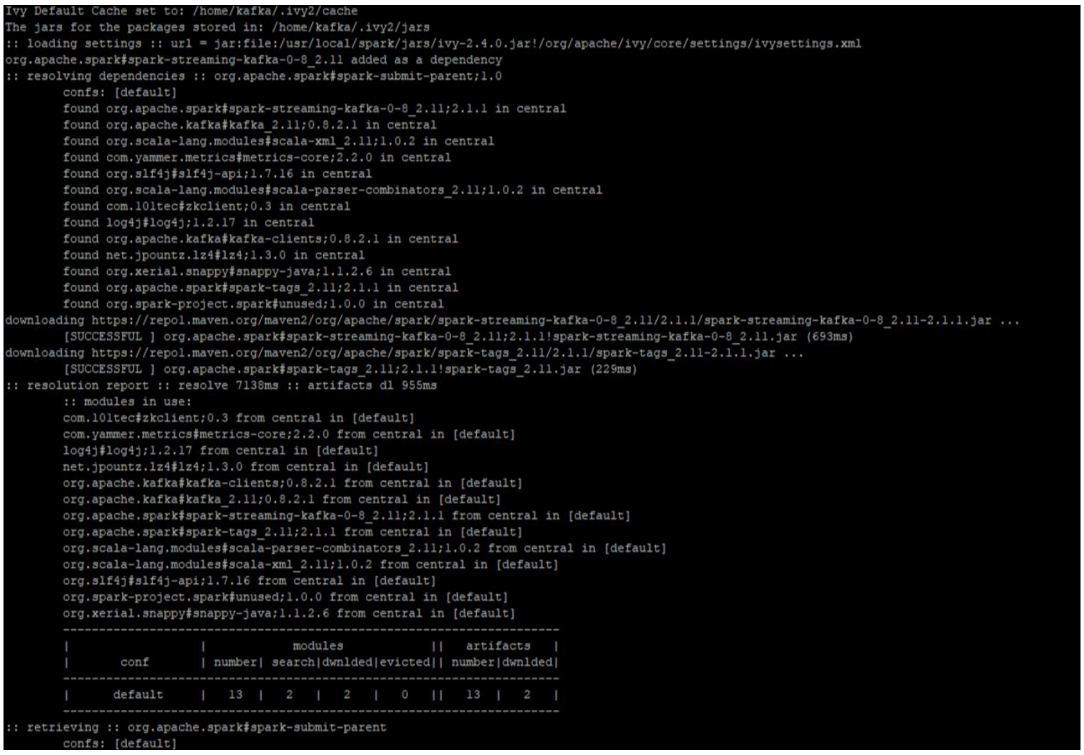
kafkademo is the python file which contains the code.

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

OUTPUT:

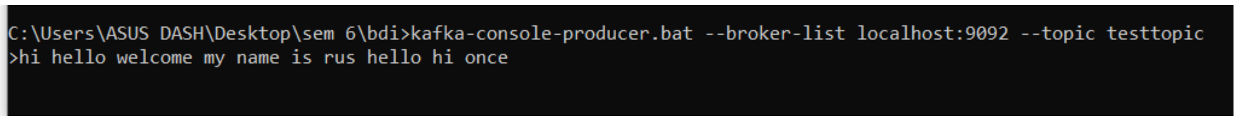


```
C:\Users\ASUS DASH\Desktop\sem 6\bdi>spark-submit --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.1.1
 kafkademo.py
```



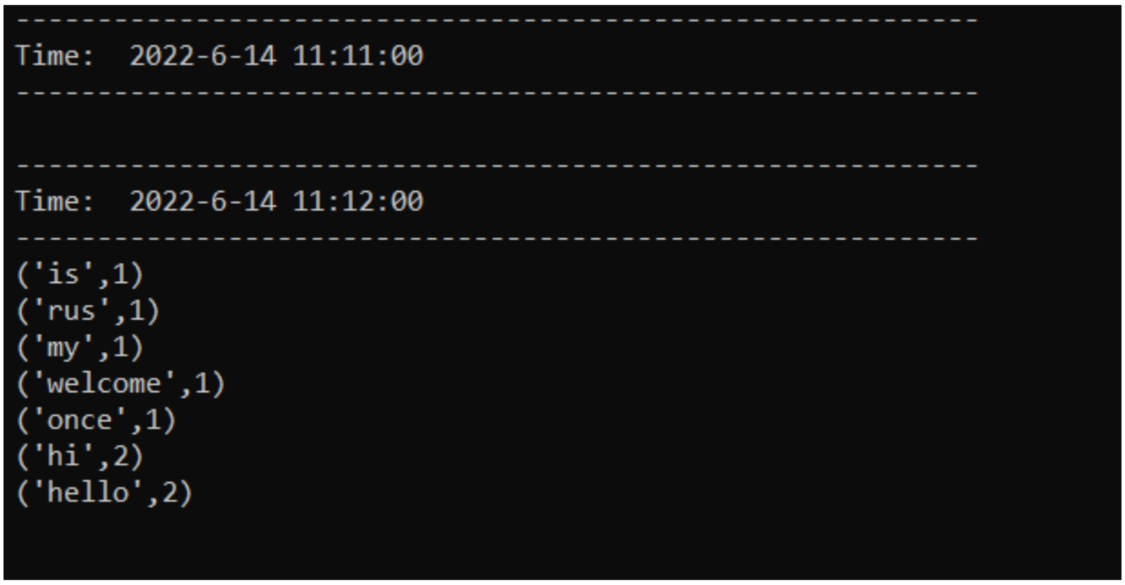
```
Ivy Default Cache set to: /home/kafka/.ivy2/cache
The jars for the packages stored in: /home/kafka/.ivy2/jars
:: loading settings :: url = jar:file:/usr/local/spark/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
org.apache.spark#spark-streaming-kafka-0-8_2.11 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
        confs: [default]
        found org.apache.spark#spark-streaming-kafka-0-8_2.11;2.1.1 in central
        found org.apache.kafka#kafka_2.11;0.8.2.1 in central
        found org.scala-lang.modules#scala-xml_2.11;1.0.2 in central
        found com.yammer.metrics#metrics-core;2.2.0 in central
        found org.slf4j#slf4j-api;1.7.16 in central
        found org.scala-lang.modules#scala-parser-combinators_2.11;1.0.2 in central
        found com.101tec#zkclient;0.3 in central
        found log4j#log4j;1.2.17 in central
        found org.apache.kafka#kafka-clients;0.8.2.1 in central
        found net.jpountz.lz4#lz4;1.3.0 in central
        found org.xerial.snappy#snappy-java;1.1.2.6 in central
        found org.apache.spark#spark-tags_2.11;2.1.1 in central
        found org.spark-project.spark#unused;1.0.0 in central
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8_2.11/2.1.1/spark-streaming-kafka-0-8_2.11-2.1.1.jar ...
        [SUCCESSFUL ] org.apache.spark#spark-streaming-kafka-0-8_2.11;2.1.1!spark-streaming-kafka-0-8_2.11.jar (693ms)
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-tags_2.11/2.1.1/spark-tags_2.11-2.1.1.jar ...
        [SUCCESSFUL ] org.apache.spark#spark-tags_2.11;2.1.1!spark-tags_2.11.jar (229ms)
:: resolution report :: resolve 7138ms :: artifacts dl 955ms
        :: modules in use:
        com.101tec#zkclient;0.3 from central in [default]
        com.yammer.metrics#metrics-core;2.2.0 from central in [default]
        log4j#log4j;1.2.17 from central in [default]
        net.jpountz.lz4#lz4;1.3.0 from central in [default]
        org.apache.kafka#kafka-clients;0.8.2.1 from central in [default]
        org.apache.kafka#kafka_2.11;0.8.2.1 from central in [default]
        org.apache.spark#spark-streaming-kafka-0-8_2.11;2.1.1 from central in [default]
        org.apache.spark#spark-tags_2.11;2.1.1 from central in [default]
        org.scala-lang.modules#scala-parser-combinators_2.11;1.0.2 from central in [default]
        org.scala-lang.modules#scala-xml_2.11;1.0.2 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.2.6 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   13  |   2   |   2   |   0   ||   13  |   2   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent
        confs: [default]
```

6) Now our command line is waiting for the data we enter the data in terminal 3:



```
C:\Users\ASUS DASH\Desktop\sem 6\bdi>kafka-console-producer.bat --broker-list localhost:9092 --topic testtopic
>hi hello welcome my name is rus hello hi once
```

7) And the output is obtained at terminal 4:



```
-------------------------------------------
Time:  2022-6-14 11:11:00
-------------------------------------------


-------------------------------------------
Time:  2022-6-14 11:12:00
-------------------------------------------
('is',1)
('rus',1)
('my',1)
('welcome',1)
('once',1)
('hi',2)
('hello',2)
```

NAME: Meith Navlakha
SAP: 60004190068
BDI Practical

CONCLUSION:

Thus in this experiment, we explored and learnt about Apache Kafka, a distributed stream processing application build over Hadoop. We learnt what is Apache, what is it's architecture and where we can apply it. In this experiment, we have integrated Kafka to pySpark and read the streaming data using Kafka. The code and output for the same have been observed and attached above. Thus, we have successfully performed our experiment of reading streaming data on Kafka.

.