**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.

Diagram

Description automatically generated

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.

Diagram

Description automatically generated

**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.

**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:**

Text

Description automatically generated

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

On second terminal enter the command:

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

**OUTPUT:**

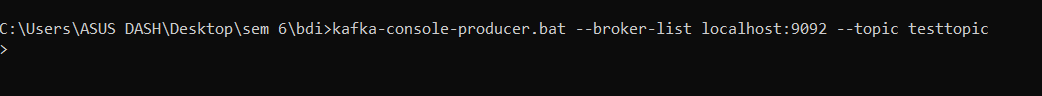
Text

Description automatically generated

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**

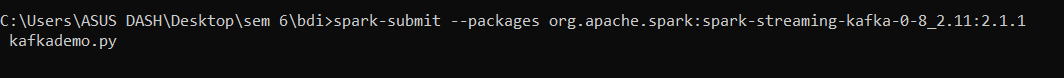


5) On fourth terminal we run the command:

**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.

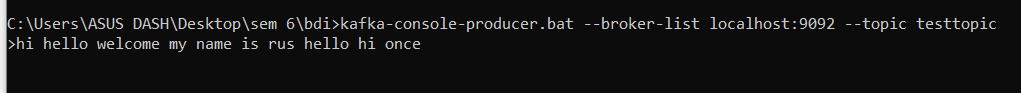
**OUTPUT:**



Text

Description automatically generated

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



7) And the output is obtained at terminal 4:

A picture containing diagram

Description automatically generated

**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.

.