KAFKA

Example for PRODUCER Program:

```
package com.tech.kafka

object kafkaproducer extends App {

 import java.util.Properties

 import org.apache.kafka.clients.producer._

 /*
 The acks=0 -> means the Producer does not wait for any ack from Kafka
broker at all. (fire-and-forget)
 The acks=1 is leader acknowledgment. (A-sync)
 The acks=all or acks=-1 (Synch) is all acknowledgment which means the
leader gets write confirmation from
   the full set of ISRs before sending an ack back to the producer.
 */

 val  props = new Properties()
 props.put("bootstrap.servers",
"192.168.138.145:9092,192.168.138.145:9093")
//first property is the ip addr of broker and its port no.
//multiple brokers/ports can be mentioned by comma separated.
 props.put("acks","0") // → here the ack mode is its fire  and forget.
 props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer")
 props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer")
/*here the kind of serialization has to be mentioned. In this example
we use string serializer. While consuming this message we need to used
the same serializer. */

 val producer = new KafkaProducer[String, String](props)
/*while creating instance for the class KafkaProducer, we need to pass
parameters which has the properties which we specified using PUT().
Here we pass the key value pair input which is defined in  the map
variable props */
 val record = new ProducerRecord("tk3", "100", "hello")
 producer.send(record)
/* the created object "producer" has the config info..
Now the above created object "record" has the below info.
 This consists of a topic name to which the record is being sent, an
optional partition number, and an optional key and value. */

 println(s"End")
 producer.close()
}

/* here we hard core some value to the ProducerRecord class ("hello")
and we have producing it to the topic "tk3" which is asked to write in
the broker specified in the IP address/port no. */
```

Example for CONSUMER Program:

```
package com.tech.kafka

import java.util

import org.apache.kafka.clients.consumer.KafkaConsumer

import scala.collection.JavaConverters._

object kafkaconsumer extends App {

  import java.util.Properties

  val TOPIC="tk3"

  val  props = new Properties()
  props.put("bootstrap.servers", "localhost:9092")

  props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer")
  props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer")
  /*
   Auto.offset.reset is a property to specify whether you want to
consume the records
   from the beginning (earliest) or the last committed offset (latest).
   */
  props.put("auto.offset.reset", "latest")
  props.put("group.id", "g123")

  val consumer = new KafkaConsumer[String, String](props)
/*similarly to producer program here we use the class KafkaConsumer()
along with the properties */
  consumer.subscribe(util.Collections.singletonList(TOPIC))
//here we subscribe to a topic
  while(true){
    val records=consumer.poll(100)
    for (record<-records.asScala){
     println(record)
// we get 100 records from consumer() and loop to process it.
    }
  }
}
```