Assignment

Week17: Apache Kafka - Distributed Event Streaming Platform

IMPORTANT

Self-assessment enables students to develop:

1. A sense of responsibility for their own learning and the ability & desire to continue learning,
2. Self-knowledge & capacity to assess their own performance critically & accurately, and
3. An understanding of how to apply their knowledge and abilities in different contexts.

All assignments are for self-assessment. Solutions will be released on every subsequent week. Once the solution is out, evaluate yourself.

No discussions/queries allowed on assignment questions in slack channel.

***Note**: You can raise your doubts in the subsequent week once the solution is released*

Solution1:

Step 1: open one terminal and start Kafka server

./kafka-server-start.sh ../config/server.properties

Step 2: open one terminal and create a topic

./kafka-topics.sh –create –topic topic1 –bootstrap-server localhost:9092 –partitions 1 –replication-factor1

```
[cloudera@quickstart bin]$ ./kafka-topics.sh --create --topic topic1 --bootstrap
-server localhost:9092 --partitions 1 --replication-factor 1
```

Step 3: open one terminal and start producer

./kafka-console-producer.sh –broker-list localhost:9092 –topic topic1

Step 4: open one terminal and start consumer

./kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic topic1 –from-beginning

Step 5: Go to producer terminal →Send messages from producer one by one as one, two, three

```
[cloudera@quickstart bin]$ ./kafka-console-producer.sh --broker-list localhost:9
092 --topic topic1
>one
>two
>Three
```

Step 6: go to consumer terminal → check if messages are received

```
[cloudera@quickstart bin]$ ./kafka-console-producer.sh --broker-list localhost:9
092 --topic topic1
>one
>two
>Three
```

Solution 2:

Step 1: create a topic named topic1

./kafka-topics.sh –create –topic topic1 –bootstrap-server localhost:9092 –partitions 1 –replication-factor1

Step 2: create Java Producer

```
import java.util.Properties;
import java.util.Scanner;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.IntegerSerializer;
import org.apache.kafka.common.serialization.StringSerializer;
```

```
public class MyProducer1 {
    public static void main(String[] args){

    //Step 1- set the properties
    Properties props = new Properties();
    props.put(ProducerConfig.CLIENT_ID_CONFIG,"producer_id1");
    props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,"localhost:9092");
    props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,IntegerSerializer.class.getName());
    props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,IntegerSerializer.class.getName());

    //Step 2- Create Object of KafkaProducer
    KafkaProducer<Integer,Integer> producer = new KafkaProducer<Integer,Integer>(props);

    try{
    while (true){
            Scanner scan = new Scanner(System.in);
        System.out.println("Enter new number of complaints received: ");
          int num = scan.nextInt();

    //Step 3- Calling the send method on this producer object
    producer.send(new ProducerRecord<Integer,Integer>("topic1",1,num));
    }
```

```
    }
    catch(Exception e){
            System.out.println(e.getMessage());
    }
    finally{
    //Step 4- Close the producer object
    producer.close();
    }

            }
}
```

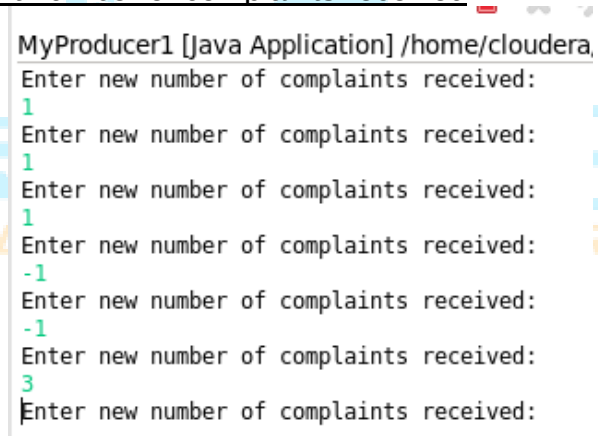
Step 3: Create Java Consumer

```
import java.util.Collections;
import java.util.Properties;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.serialization.IntegerDeserializer;
import org.apache.kafka.common.serialization.StringDeserializer;
public class MyConsumer {
```

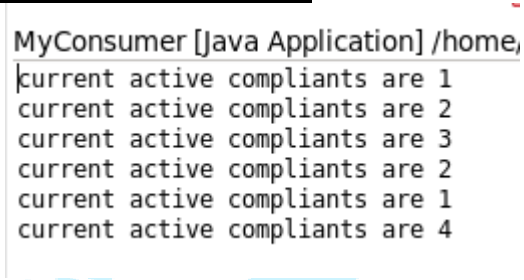
```
public static void main(String[] args){
//Step 1- set the properties
        Properties props = new Properties();
        props.put(ConsumerConfig.CLIENT_ID_CONFIG,"id1");
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,"localhost:9092");
        props.put(ConsumerConfig.GROUP_ID_CONFIG,"group1");

props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,IntegerDeserializer.class.getName());

props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,IntegerDeserializer.class.getName());
                        props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,"earliest");
        //step 2 : Create Object of KafkaConsumer and subscribe to the topic
        KafkaConsumer<Integer,Integer> con = new KafkaConsumer<Integer,Integer>(props);

        con.subscribe(Collections.singletonList("topic1"));
        Integer sum =new Integer(0);

        //step 3 : consume the message and process it as you want
while (true){
 ConsumerRecords<Integer,Integer> records=  con.poll(1);
 for(ConsumerRecord<Integer,Integer> record : records){
        //sum =sum+Integer.parseInt( record.value());
```

```
            sum =sum+ record.value();
           System.out.println("current active compliants are " + sum);

    }
   }
   //step 4 : close
   }
}
```

Step 4: Run your producer and insert number of complaints received



Step 5: Run your consumer and check the active complaints



Note: you can try executing multiple instances of producer and consumer and check the numbers

Solution 3:

Step 1: create topic

./kafka-topics.sh –create –topic topic2 –bootstrap-server localhost:9092 –partitions 2 –replication-factor1

Step 2: Create Java Producer

Same as above just change highlighted number (key- that represents state)1 and then 2 and topic name

producer.send(new ProducerRecord<Integer,Integer>("topic2",2,num));

Step 3: Create Java Consumer

Add below lines in existing consumer

```
con.subscribe(Collections.singletonList("topic2"));

Integer sum2 =new Integer(0);

        //step 3 : consume the message and process it as you want
  while (true){
   ConsumerRecords<Integer,Integer> records=  con.poll(1);
   for(ConsumerRecord<Integer,Integer> record : records){
        //sum =sum+Integer.parseInt( record.value());
         if(record.key()==1){
                sum1 =sum1+ record.value();
                                System.out.printf("partition = %d offset = %d, key = %d, Active complaints =
%d\n", record.partition(),record.offset(), record.key(), sum1);
         }

         else{
                sum2 =sum2+ record.value();
                System.out.printf("partition = %d offset = %d, key = %d, Active complaints = %d\n",
record.partition(),record.offset(), record.key(), sum2);
```

```
        }
    }
}
```

Step 4: In producer, Change the key to 1 → Run your producer and insert number of complaints received

```
MyProducer1 [Java Application] /home/clouder
Enter new number of complaints received:
1
Enter new number of complaints received:
3
Enter new number of complaints received:
1
Enter new number of complaints received:
3
```


Step 5: In producer, Change the key to 2 → Run your producer and insert number of complaints received

```
MyProducer1 [Java Application] /home/cloude
Enter new number of complaints received:
2
Enter new number of complaints received:
4
Enter new number of complaints received:
```

Step 6 : Run your consumer and check the active complaints

```
MyConsumer [Java Application] /home/cloudera/Desktop/Softwa
partition = 0 offset = 2, key = 1, Active complaints = 1
partition = 0 offset = 3, key = 1, Active complaints = 4
partition = 0 offset = 4, key = 1, Active complaints = 5
partition = 0 offset = 5, key = 1, Active complaints = 8
partition = 1 offset = 0, key = 2, Active complaints = 2
partition = 1 offset = 1, key = 2, Active complaints = 6
```

Note: unique key records go to same partition. Key can be taken as string to store actual state and not represent by number.

Google REVIEWS

5 Star Google Rated
Big Data Course

LEARN FROM THE EXPERT

9108179578

Call for more details