Kafka Consumer - Java

❖ Create simple example that creates a **Kafka Consumer**

    ❖ that consumes messages form the **Kafka Producer** we wrote

❖ **Create Consumer** that uses topic from first example to receive messages

❖ **Process messages** from Kafka with **Consumer**

❖ Demonstrate how Consumer Groups work

- ❖ **Specify bootstrap servers**

- ❖ **Specify Consumer Group**

- ❖ **Specify Record Key deserializer**

- ❖ **Specify Record Value deserializer**

- ❖ **Subscribe to Topic from last session**

KafkaConsumerExample.java ✕

KafkaConsumerExample

```java
package com.cloudurable.kafka;
import org.apache.kafka.clients.consumer.*;
import org.apache.kafka.clients.consumer.Consumer;
import org.apache.kafka.common.serialization.LongDeserializer;
import org.apache.kafka.common.serialization.StringDeserializer;


import java.util.Collections;
import java.util.Properties;


public class KafkaConsumerExample {


    private final static String TOPIC = "my-example-topic";
    private final static String BOOTSTRAP_SERVERS =
            "localhost:9092,localhost:9093,localhost:9094";

```

Create Consumer using Topic to Receive Records

```java
KafkaConsumerExample.java  ×

  KafkaConsumerExample   createConsumer()

16      private static Consumer<Long, String> createConsumer() {
17          final Properties props = new Properties();
18          props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
19                                  BOOTSTRAP_SERVERS);
20          props.put(ConsumerConfig.GROUP_ID_CONFIG,
21                                  "KafkaExampleConsumer");
22          props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
23                  LongDeserializer.class.getName());
24          props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
25                  StringDeserializer.class.getName());
26
27          // Create the consumer using props.
28          final Consumer<Long, String> consumer =
29                                  new KafkaConsumer<>(props);
30
31          // Subscribe to the topic.
32          consumer.subscribe(Collections.singletonList(TOPIC));
33          return consumer;
34      }
```

KafkaConsumerExample.java ×

KafkaConsumerExample

```java
40      static void runConsumer() throws InterruptedException {
41          final Consumer<Long, String> consumer = createConsumer();
42
43          final int giveUp = 100;    int noRecordsCount = 0;
44
45          while (true) {
46              final ConsumerRecords<Long, String> consumerRecords =
47                      consumer.poll( timeout: 1000);
48
49              if (consumerRecords.count()==0) {
50                  noRecordsCount++;
51                  if (noRecordsCount > giveUp) break;
52                  else continue;
53              }
54
55              consumerRecords.forEach(record -> {
56                  System.out.printf("Consumer Record:(%d, %s, %d, %d)\n",
57                          record.key(), record.value(),
58                          record.partition(), record.offset());
59              });
60
61              consumer.commitAsync();
62          }
63          consumer.close();
64          System.out.println("DONE");
```

Consumer poll

- poll() method returns fetched records based on current partition offset

- Blocking method waiting for specified time if no records available

- When/If records available, method returns straight away

- Control the maximum records returned by the poll() with

  - **props.put(ConsumerConfig.*MAX_POLL_RECORDS_CONFIG*, 100);**

- poll() is not meant to be called from multiple threads

```
67
68 ▶    public static void main(String... args) throws Exception {
69          runConsumer();
70      }
71
```

Run ⧉ KafkaConsumerExample

```
ssl.protocol = TLS
ssl.provider = null
ssl.secure.random.implementation = null
ssl.trustmanager.algorithm = PKIX
ssl.truststore.location = null
ssl.truststore.password = null
ssl.truststore.type = JKS
value.deserializer = class org.apache.kafka.common.serialization.StringDeserializer

15:17:35.267 [main] INFO  o.a.kafka.common.utils.AppInfoParser – Kafka version : 0.10.2.0
15:17:35.267 [main] INFO  o.a.kafka.common.utils.AppInfoParser – Kafka commitId : 576d93a8dc0cf421
15:17:35.384 [main] INFO  o.a.k.c.c.i.AbstractCoordinator – Discovered coordinator 10.0.0.115:9093
15:17:35.391 [main] INFO  o.a.k.c.c.i.ConsumerCoordinator – Revoking previously assigned partitions
15:17:35.391 [main] INFO  o.a.k.c.c.i.AbstractCoordinator – (Re-)joining group KafkaExampleConsumer
15:17:42.257 [main] INFO  o.a.k.c.c.i.AbstractCoordinator – Successfully joined group KafkaExampleC
15:17:42.259 [main] INFO  o.a.k.c.c.i.ConsumerCoordinator – Setting newly assigned partitions [my-e
Consumer Record:(1494973064716, Hello Mom 1494973064716, 6, 4)
Consumer Record:(1494973064719, Hello Mom 1494973064719, 10, 6)
Consumer Record:(1494973064718, Hello Mom 1494973064718, 9, 9)
Consumer Record:(1494973064717, Hello Mom 1494973064717, 12, 9)
Consumer Record:(1494973064720, Hello Mom 1494973064720, 4, 8)
```

```xml
logback.xml  ×

1  <configuration>
2      <appender name="STDOUT"
3              class="ch.qos.logback.core.ConsoleAppender">
4          <encoder>
5              <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level
6                  %logger{36} - %msg%n</pattern>
7          </encoder>
8      </appender>
9
10     <logger name="org.apache.kafka" level="INFO"/>
11     <logger name="org.apache.kafka.common.metrics" level="INFO"/>
12
13     <root level="debug">
14         <appender-ref ref="STDOUT" />
15     </root>
16  </configuration>
```

- ❖ Kafka uses sl4j

- ❖ Set level to DEBUG to see what is going on

Try this: Consumers in Same Group

- ❖ Three consumers and one producer sending 25 records

- ❖ Run three consumers processes

- ❖ Same Group ID

Consumer 0 (key, value, partition, offset)

```
Consumer Record:(1495042369488, Hello Mom 1495042369488, 0, 9)
Consumer Record:(1495042369490, Hello Mom 1495042369490, 3, 9)
Consumer Record:(1495042369498, Hello Mom 1495042369498, 3, 10)
Consumer Record:(1495042369504, Hello Mom 1495042369504, 3, 11)
Consumer Record:(1495042369508, Hello Mom 1495042369508, 3, 12)
Consumer Record:(1495042369491, Hello Mom 1495042369491, 4, 9)
Consumer Record:(1495042369503, Hello Mom 1495042369503, 4, 10)
Consumer Record:(1495042369505, Hello Mom 1495042369505, 4, 11)
Consumer Record:(1495042369494, Hello Mom 1495042369494, 2, 9)
Consumer Record:(1495042369499, Hello Mom 1495042369499, 2, 10)
```

Consumer 1 (key, value, partition, offset)

```
10:52:59.099 [main] INFO  o.a.k.c.c.i.AbstractCoordinator - Discov
Consumer Record:(1495042369486, Hello Mom 1495042369486, 12, 10)
Consumer Record:(1495042369493, Hello Mom 1495042369493, 12, 11)
Consumer Record:(1495042369507, Hello Mom 1495042369507, 12, 12)
Consumer Record:(1495042369487, Hello Mom 1495042369487, 9, 12)
Consumer Record:(1495042369492, Hello Mom 1495042369492, 10, 7)
Consumer Record:(1495042369495, Hello Mom 1495042369495, 10, 8)
Consumer Record:(1495042369501, Hello Mom 1495042369501, 11, 8)
Consumer Record:(1495042369506, Hello Mom 1495042369506, 11, 9)
```

Consumer 2 (key, value, partition, offset)

```
Consumer Record:(1495042369509, Hello Mom 1495042369509, 6, 6)
Consumer Record:(1495042369497, Hello Mom 1495042369497, 7, 11)
Consumer Record:(1495042369500, Hello Mom 1495042369500, 7, 12)
Consumer Record:(1495042369496, Hello Mom 1495042369496, 5, 7)
Consumer Record:(1495042369510, Hello Mom 1495042369510, 5, 8)
Consumer Record:(1495042369489, Hello Mom 1495042369489, 8, 9)
Consumer Record:(1495042369502, Hello Mom 1495042369502, 8, 10)
```

Producer

```
sent record(key=1495042369509 value=Hello Mom 1495042369509) meta(partition=6, offset=6) t
sent record(key=1495042369487 value=Hello Mom 1495042369487) meta(partition=9, offset=12)
sent record(key=1495042369486 value=Hello Mom 1495042369486) meta(partition=12, offset=10)
sent record(key=1495042369493 value=Hello Mom 1495042369493) meta(partition=12, offset=11)
sent record(key=1495042369507 value=Hello Mom 1495042369507) meta(partition=12, offset=12)
sent record(key=1495042369488 value=Hello Mom 1495042369488) meta(partition=0, offset=9) t
sent record(key=1495042369490 value=Hello Mom 1495042369490) meta(partition=3, offset=9) t
sent record(key=1495042369498 value=Hello Mom 1495042369498) meta(partition=3, offset=10)
sent record(key=1495042369504 value=Hello Mom 1495042369504) meta(partition=3, offset=11)
sent record(key=1495042369508 value=Hello Mom 1495042369508) meta(partition=3, offset=12)
sent record(key=1495042369497 value=Hello Mom 1495042369497) meta(partition=7, offset=11)
sent record(key=1495042369500 value=Hello Mom 1495042369500) meta(partition=7, offset=12)
sent record(key=1495042369492 value=Hello Mom 1495042369492) meta(partition=10, offset=7)
sent record(key=1495042369495 value=Hello Mom 1495042369495) meta(partition=10, offset=8)
sent record(key=1495042369491 value=Hello Mom 1495042369491) meta(partition=4, offset=9) t
sent record(key=1495042369503 value=Hello Mom 1495042369503) meta(partition=4, offset=10)
sent record(key=1495042369505 value=Hello Mom 1495042369505) meta(partition=4, offset=11)
sent record(key=1495042369496 value=Hello Mom 1495042369496) meta(partition=5, offset=7) t
sent record(key=1495042369510 value=Hello Mom 1495042369510) meta(partition=5, offset=8) t
sent record(key=1495042369489 value=Hello Mom 1495042369489) meta(partition=8, offset=9) t
sent record(key=1495042369502 value=Hello Mom 1495042369502) meta(partition=8, offset=10)
sent record(key=1495042369501 value=Hello Mom 1495042369501) meta(partition=11, offset=8)
sent record(key=1495042369506 value=Hello Mom 1495042369506) meta(partition=11, offset=9)
sent record(key=1495042369494 value=Hello Mom 1495042369494) meta(partition=2, offset=9) t
sent record(key=1495042369499 value=Hello Mom 1495042369499) meta(partition=2, offset=10)
```

Which consumer owns partition 10? How many Consumer Records objects did Consumer 0 get?

What is the next offset from Partition 5 that Consumer 2 should get?

Why does each consumer get unique

Try this: Consumers in Different Groups

- Three consumers with unique group and one producer sending 5 records

- Modify Consumer to have unique group id

- Run three consumers processes

- Run one producer

- What happens?

Consumer 0(key, value, partition, offset)

```
Consumer Record:(1495043607696, Hello Mom 1495043607696, 0, 10)
Consumer Record:(1495043607699, Hello Mom 1495043607699, 7, 13)
Consumer Record:(1495043607700, Hello Mom 1495043607700, 2, 11)
Consumer Record:(1495043607697, Hello Mom 1495043607697, 10, 9)
Consumer Record:(1495043607698, Hello Mom 1495043607698, 10, 10)
```

Consumer 1(key, value, partition, offset)

```
Consumer Record:(1495043607696, Hello Mom 1495043607696, 0, 10)
Consumer Record:(1495043607699, Hello Mom 1495043607699, 7, 13)
Consumer Record:(1495043607700, Hello Mom 1495043607700, 2, 11)
Consumer Record:(1495043607697, Hello Mom 1495043607697, 10, 9)
Consumer Record:(1495043607698, Hello Mom 1495043607698, 10, 10)
```

Consumer 2(key, value, partition, offset)

```
Consumer Record:(1495043607696, Hello Mom 1495043607696, 0, 10)
Consumer Record:(1495043607699, Hello Mom 1495043607699, 7, 13)
Consumer Record:(1495043607700, Hello Mom 1495043607700, 2, 11)
Consumer Record:(1495043607697, Hello Mom 1495043607697, 10, 9)
Consumer Record:(1495043607698, Hello Mom 1495043607698, 10, 10)
```

Producer

```
sent (key=1495043832401 ) meta(partition=7, offset=14)
sent (key=1495043832400 ) meta(partition=1, offset=11)
sent (key=1495043832404 ) meta(partition=6, offset=7)
sent (key=1495043832402 ) meta(partition=0, offset=11)
sent (key=1495043832403 ) meta(partition=3, offset=13)
```

Which consumer(s) owns partition 10?

How many ConsumerRecords objects did Consumer 0 get?

What is the next offset from Partition 2 that Consumer 2 get?

Why does each Consumer get the same messages?

Lab : Consumer – Java API – 45 Minutes