Reading data from Kafka is a bit different than reading data from other messaging systems and there are few unique concepts and ideas involved.

Refer scaling on Kafka.

**Creating a Kafka Consumer**

The first step to start consuming records is to create a KafkaConsumer instance.

To start we just need to use the 3 mandatory properties: bootstrap.servers, key.deserializer and value.deserializer.

There is a fourth property group.id, which is not mandatory. It specifies the Consumer Group the KafkaConsumer instance belongs to.

|  |
| --- |
| Properties props = new Properties(); |
| props.put("bootstrap.servers", "broker1:9092,broker2:9092"); |
| props.put("group.id", "CountryCounter"); |
| props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer"); |
| props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer"); |
| KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(props); |

# Subscribing to Topics Once we created a consumer, the next step is to subscribe to one or more topics. consumer.subscribe(Collections.singletonList("customerCountries"));

# It is also possible to call subscribe with a regular expression. consumer.subscribe("test.\*");

# The Poll Loop At the heart of the consumer API is a simple loop for polling the server for more data. Once the consumer subscribes to topics, the poll loop handles all details of coordination, partition rebalances, heartbeats and data fetching.

# The first time you poll() with a new consumer, poll() is responsible for finding the GroupCoordinator, joining the consumer group and receiving a partition assignment. If a rebalance is triggered, it will be handled inside the poll loop as well. And of course the heartbeats that keep consumers alive are sent from within the poll loop. For this reason, we try to make sure that whatever processing we do between iterations is fast and efficient.

# A consumer should be accessed by only 1 thread. Many threads accessing a consumer or single thread accessing multiple consumers are not recommended.