Apache Kafka 3

# Developing Kafka Producers and Consumers

**Date of Publish:** 2019-03-08


**https://docs.hortonworks.com**

# Contents

# Developing Kafka Producers and Consumers

The examples in this chapter contain code for a basic Kafka producer and consumer, and similar examples for an SSL-enabled cluster.

Basic Producer Example

```
package com.hortonworks.example.kafka.producer;

import org.apache.kafka.clients.producer.Callback;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.clients.producer.RecordMetadata;

import java.util.Properties;
import java.util.Random;

public class BasicProducerExample {

   public static void main(String[] args){

        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
 "kafka.example.com:6667");
        props.put(ProducerConfig.ACKS_CONFIG, "all");
        props.put(ProducerConfig.RETRIES_CONFIG, 0);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringSerializer");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringSerializer");

        Producer<String, String> producer = new KafkaProducer<String,
 String>(props);
        TestCallback callback = new TestCallback();
        Random rnd = new Random();
        for (long i = 0; i < 100 ; i++) {
            ProducerRecord<String, String> data = new ProducerRecord<String,
 String>(
                    "test-topic", "key-" + i, "message-"+i );
            producer.send(data, callback);
        }

        producer.close();
    }


    private static class TestCallback implements Callback {
        @Override
        public void onCompletion(RecordMetadata recordMetadata, Exception e)
 {
            if (e != null) {
                System.out.println("Error while producing message to topic :"
 + recordMetadata);
                e.printStackTrace();
            } else {
                String message = String.format("sent message
 to topic:%s partition:%s  offset:%s", recordMetadata.topic(),
 recordMetadata.partition(), recordMetadata.offset());
                System.out.println(message);
```

```
            }
        }
    }

}
```

To run the producer example, use the following command:

$ java com.hortonworks.example.kafka.producer.BasicProducerExample

Producer Example for an SSL-Enabled Cluster

The following example adds three important configuration settings for SSL encryption and three for SSL authentication. The two sets of configuration settings are prefaced by comments.

```
package com.hortonworks.example.kafka.producer;

import org.apache.kafka.clients.CommonClientConfigs;
import org.apache.kafka.clients.producer.Callback;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.clients.producer.RecordMetadata;
import org.apache.kafka.common.config.SslConfigs;

import java.util.Properties;
import java.util.Random;

public class BasicProducerExample {

   public static void main(String[] args){

        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
 "kafka.example.com:6667");

        //configure the following three settings for SSL Encryption
        props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SSL");
        props.put(SslConfigs.SSL_TRUSTSTORE_LOCATION_CONFIG, "/var/private/
ssl/kafka.client.truststore.jks");
        props.put(SslConfigs.SSL_TRUSTSTORE_PASSWORD_CONFIG,  "test1234");

        // configure the following three settings for SSL Authentication
        props.put(SslConfigs.SSL_KEYSTORE_LOCATION_CONFIG, "/var/private/ssl/
kafka.client.keystore.jks");
        props.put(SslConfigs.SSL_KEYSTORE_PASSWORD_CONFIG, "test1234");
        props.put(SslConfigs.SSL_KEY_PASSWORD_CONFIG, "test1234");

        props.put(ProducerConfig.ACKS_CONFIG, "all");
        props.put(ProducerConfig.RETRIES_CONFIG, 0);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringSerializer");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringSerializer");

        Producer<String, String> producer = new KafkaProducer<String,
 String>(props);
        TestCallback callback = new TestCallback();
        Random rnd = new Random();
        for (long i = 0; i < 100 ; i++) {
            ProducerRecord<String, String> data = new ProducerRecord<String,
 String>(
                    "test-topic", "key-" + i, "message-"+i );
```

```
            producer.send(data, callback);
        }

        producer.close();
    }


    private static class TestCallback implements Callback {
        @Override
        public void onCompletion(RecordMetadata recordMetadata, Exception e)
 {
            if (e != null) {
                System.out.println("Error while producing message to topic :"
 + recordMetadata);
                e.printStackTrace();
            } else {
                String message = String.format("sent message
 to topic:%s partition:%s  offset:%s", recordMetadata.topic(),
 recordMetadata.partition(), recordMetadata.offset());
                System.out.println(message);
            }
        }
    }

}
```

To run the producer example, use the following command:

$ java com.hortonworks.example.kafka.producer.BasicProducerExample

Basic Consumer Example

```
package com.hortonworks.example.kafka.consumer;

import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRebalanceListener;
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.TopicPartition;

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

public class BasicConsumerExample {

    public static void main(String[] args) {

        Properties consumerConfig = new Properties();
        consumerConfig.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
 "kafka.example.com:6667");
        consumerConfig.put(ConsumerConfig.GROUP_ID_CONFIG, "my-group");
        consumerConfig.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
 "earliest");
        consumerConfig.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringDeserializer");
        consumerConfig.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringDeserializer");
        KafkaConsumer<byte[], byte[]> consumer = new
 KafkaConsumer<>(consumerConfig);
        TestConsumerRebalanceListener rebalanceListener = new
 TestConsumerRebalanceListener();
```

```
        consumer.subscribe(Collections.singletonList("test-topic"),
 rebalanceListener);

        while (true) {
            ConsumerRecords<byte[], byte[]> records = consumer.poll(1000);
            for (ConsumerRecord<byte[], byte[]> record : records) {
                System.out.printf("Received Message topic =%s, partition =%s,
 offset = %d, key = %s, value = %s\n", record.topic(), record.partition(),
 record.offset(), record.key(), record.value());
            }

            consumer.commitSync();
        }

    }

    private static class  TestConsumerRebalanceListener implements
 ConsumerRebalanceListener {
        @Override
        public void onPartitionsRevoked(Collection<TopicPartition>
 partitions) {
            System.out.println("Called onPartitionsRevoked with partitions:"
 + partitions);
        }

        @Override
        public void onPartitionsAssigned(Collection<TopicPartition>
 partitions) {
            System.out.println("Called onPartitionsAssigned with partitions:"
 + partitions);
        }
    }

}
```

To run the consumer example, use the following command:

# java com.hortonworks.example.kafka.consumer.BasicConsumerExample

Consumer Example for an SSL-Enabled Cluster

The following example adds three important configuration settings for SSL encryption and three for SSL authentication. The two sets of configuration settings are prefaced by comments.

```
package com.hortonworks.example.kafka.consumer;

import org.apache.kafka.clients.CommonClientConfigs;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRebalanceListener;
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.TopicPartition;
import org.apache.kafka.common.config.SslConfigs;

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

public class BasicConsumerExample {

   public static void main(String[] args) {

        Properties props = new Properties();
```

```
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
 "kafka.example.com:6667");

        //configure the following three settings for SSL Encryption
        props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SSL");
        props.put(SslConfigs.SSL_TRUSTSTORE_LOCATION_CONFIG, "/var/private/
ssl/kafka.client.truststore.jks");
        props.put(SslConfigs.SSL_TRUSTSTORE_PASSWORD_CONFIG,  "test1234");

        //configure the following three settings for SSL Authentication
        props.put(SslConfigs.SSL_KEYSTORE_LOCATION_CONFIG, "/var/private/ssl/
kafka.client.keystore.jks");
        props.put(SslConfigs.SSL_KEYSTORE_PASSWORD_CONFIG, "test1234");
        props.put(SslConfigs.SSL_KEY_PASSWORD_CONFIG, "test1234");

        props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-group");
        props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringDeserializer");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 "org.apache.kafka.common.serialization.StringDeserializer");
        KafkaConsumer<byte[], byte[]> consumer = new KafkaConsumer<>(props);
        TestConsumerRebalanceListener rebalanceListener = new
 TestConsumerRebalanceListener();
        consumer.subscribe(Collections.singletonList("test-topic"),
 rebalanceListener);

        while (true) {
            ConsumerRecords<byte[], byte[]> records = consumer.poll(1000);
            for (ConsumerRecord<byte[], byte[]> record : records) {
                System.out.printf("Received Message topic =%s, partition =%s,
 offset = %d, key = %s, value = %s\n", record.topic(), record.partition(),
 record.offset(), record.key(), record.value());
            }

            consumer.commitSync();
        }

    }

    private static class  TestConsumerRebalanceListener implements
 ConsumerRebalanceListener {
        @Override
        public void onPartitionsRevoked(Collection<TopicPartition>
 partitions) {
            System.out.println("Called onPartitionsRevoked with partitions:"
 + partitions);
        }

        @Override
        public void onPartitionsAssigned(Collection<TopicPartition>
 partitions) {
            System.out.println("Called onPartitionsAssigned with partitions:"
 + partitions);
        }
    }

}
```

To run the consumer example, use the following command:

$ java com.hortonworks.example.kafka.producer.BasicProducerExample