## **Exercise 4: Simple Producer and Simple Consumer**

## **Introduction**

In this exercise, you will run a Java client application that produces messages to and consumes messages from an Apache Kafka® cluster.

## **Prerequisites**

This document assumes that you already have:

[Gradle](https://gradle.org/install/) installed

[Java 11](https://www.oracle.com/java/technologies/javase-downloads.html) installed and configured as the current java version for the environment

Kafka cluster is up and running.

## **Create Project**

Create a new directory anywhere you'd like for this project:

mkdir kafka-java-getting-started && cd kafka-java-getting-started

Create the following Gradle build file for the project, named build.gradle:

buildscript {

repositories {

mavenCentral()

}

dependencies {

classpath "com.github.jengelman.gradle.plugins:shadow:4.0.3"

}

}

plugins {

id "java"

id "idea"

id "eclipse"

}

sourceCompatibility = "1.11"

targetCompatibility = "1.11"

version = "0.0.1"

repositories {

mavenCentral()

maven {

url "https://packages.confluent.io/maven"

}

}

apply plugin: "com.github.johnrengelman.shadow"

dependencies {

implementation group: 'org.slf4j', name: 'slf4j-nop', version: '1.7.36'

implementation group: 'org.apache.kafka', name: 'kafka-clients', version: '3.1.0'

}

jar {

manifest {

attributes('Main-Class': 'examples.SimpleProducer')

}

}

## **Build Producer**

Create a directory for the Java files in this project:

mkdir -p src/main/java/examples

Paste the following Java code into a file located at src/main/java/examples/SimpleProducer.java

Package examples;

//import util.properties packages

import java.util.Properties;

//import simple producer packages

import org.apache.kafka.clients.producer.Producer;

//import KafkaProducer packages

import org.apache.kafka.clients.producer.KafkaProducer;

//import ProducerRecord packages

import org.apache.kafka.clients.producer.ProducerRecord;

//Create java class named “SimpleProducer”

public class SimpleProducer {

public static void main(String[] args) throws Exception{

// Check arguments length value

if(args.length == 0){

System.out.println("Enter topic name”);

return;

}

//Assign topicName to string variable

String topicName = args[0].toString();

// create instance for properties to access producer configs

Properties props = new Properties();

//Assign localhost id

props.put("bootstrap.servers", “localhost:9092");

//Set acknowledgements for producer requests.

props.put("acks", “all");

//If the request fails, the producer can automatically retry,

props.put("retries", 0);

//Specify buffer size in config

props.put("batch.size", 16384);

//Reduce the no of requests less than 0

props.put("linger.ms", 1);

//The buffer.memory controls the total amount of memory available to the producer for buffering.

props.put("buffer.memory", 33554432);

props.put("key.serializer",

"org.apache.kafka.common.serialization.StringSerializer");

props.put("value.serializer",

"org.apache.kafka.common.serialization.StringSerializer");

Producer<String, String> producer = new KafkaProducer

<String, String>(props);

for(int i = 0; i < 10; i++)

producer.send(new ProducerRecord<String, String>(topicName,

Integer.toString(i), Integer.toString(i)));

System.out.println(“Message sent successfully”);

producer.close();

}

}

You can test the code before preceding by compiling with:

gradle build

And you should see:

BUILD SUCCESSFUL

## **Produce Messages**

To build a JAR that we can run from the command line, first run:

gradle shadowJar

And you should see:

BUILD SUCCESSFUL

Run the following command to build and execute the producer application, which will produce some random data events to the Hello-Kafka topic.

java -cp “build/libs/kafka-java-getting-started-0.0.1.jar; /path/to/kafka/kafka\_2.12-3.1.0/libs/\*” examples.ProducerExample Hello-Kafka

You should see output that resembles:

Message sent successfully

To check the above output open new terminal and type Consumer CLI command to receive messages.

>> bin/kafka-console-consumer.sh --zookeeper localhost:2181 —topic <topic-name> —from-beginning

1

2

3

4

5

6

7

8

9

10

## **SimpleConsumer Application**

Paste the following Java code into a file located at src/main/java/examples/SimpleConsumer.java

Package examples;

import java.util.Properties;

import java.util.Arrays;

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

import org.apache.kafka.clients.consumer.ConsumerRecords;

import org.apache.kafka.clients.consumer.ConsumerRecord;

public class SimpleConsumer {

public static void main(String[] args) throws Exception {

if(args.length == 0){

System.out.println("Enter topic name");

return;

}

//Kafka consumer configuration settings

String topicName = args[0].toString();

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("group.id", "test");

props.put("enable.auto.commit", "true");

props.put("auto.commit.interval.ms", "1000");

props.put("session.timeout.ms", "30000");

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);

//Kafka Consumer subscribes list of topics here.

consumer.subscribe(Arrays.asList(topicName))

//print the topic name

System.out.println("Subscribed to topic " + topicName);

int i = 0;

while (true) {

ConsumerRecords<String, String> records = consumer.poll(100);

for (ConsumerRecord<String, String> record : records)

// print the offset,key and value for the consumer records.

System.out.printf("offset = %d, key = %s, value = %s\n",

record.offset(), record.key(), record.value());

}

}

}

Once again, you can compile the code before preceding by with:

gradle build

And you should see:

BUILD SUCCESSFUL

## **Consume Messages**

From another terminal, run the following command to run the consumer application which will read the messages from the Hello-Kafka topic and write the information to the terminal.

java -cp “build/libs/kafka-java-getting-started-0.0.1.jar; /path/to/kafka/kafka\_2.12-3.1.0/libs/\*” examples.ProducerExample Hello-Kafka

**Input** − Open the producer CLI and send some messages to the topic. You can put the smple input as ‘Hello Consumer’.

**Output** − Following will be the output.

Subscribed to topic Hello-Kafka

offset = 3, key = null, value = Hello Consumer