**Kafka Exercise Manual 1**

**Exercise -1 - Topics Creation**

1. Verify the kafka installation by creating a sample topic from any of your kafka

**kafka-topics --create --zookeeper kafka-1:2181,kafka-2:2181,kafka-3:2181 --replication-factor 3 --partitions 6 --topic topic1 --config cleanup.policy=delete --config delete.retention.ms=60000**

1. Verify the topic creation by executing the below command

**kafka-topics --list --zookeeper kafka-1:2181**

The result will be the topic name which you have created in step 1

1. Detailed description of the topic can be viewed by the below command

**kafka-topics --describe --zookeeper kafka-1:2181 –topic topic1**

**Exercise – 2 – Producer and Consumer Test**

1. Verify the data flow from producer to consumer using the below steps
2. Open a kafka kafka and start producer using the below command

**kafka-console-producer --broker-list kafka-1:9092 --topic topic1**

* 1. Open another kafka kafka and start consumer using the below command

**kafka-console-consumer --bootstrap-server kafka-1:9092 --topic topic1**

* 1. Type something in your producer refer the below screenshot



* 1. You can see the message sent from producer here. Refer screenshot



**Testing the Kafka Logs**

1. Verify the Kafka Logs by navigating to the kafka log location

**cd /var/lib/kafka/**

1. Navigate to folder with topic-\*

**cd topic-1**

1. Open the \*.log file which will hold the topic based data log.

**cat 00000000000000000000.log**

Try with other machines

**Exercise 3 – Checking Kafka Failover**

1. Create a topic called **Check-failover**.
2. Execute the following command

**kafka-topics --create --zookeeper kafka-1:2181 --replication-factor 3 --partitions 12 --topic check-failover**

You are creating a topic named check-failover with 3 replications. For 3 replications you need to have 3 kafkas at least. Else error will occur while topic creation. Partitions for the topic are 12.

1. Open another instances of kafka-1 machine and become sudo user
2. Launch the consumer for checking the failover using the below command

**kafka-console-consumer --bootstrap-server kafka-1:9092,kafka-2:9092, kafka-3:9092 --topic check-failover --from-beginning**

1. Open kafka-2 and get into sudo access
2. Launch the producer using the below command

**kafka-console-producer --kafka-list kafka-1:9092,kafka-2:9092 --topic check-failover**

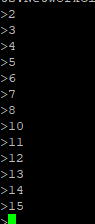
Ignore error messages if any.

1. Now start two more consumers on kafka-2 and kafka-3
2. You will see the same messages sent by producer due to the keyword –from-beginning
3. Stop all the producers and consumers now
4. Start all the consumers with the below command

**kafka-console-consumer --bootstrap-server kafka-1:9092,kafka-2:9092 --topic check-failover --consumer-property group.id=mygroup**

The property –consumer-property group.id will add all the consumers to the consumer group mygroup.

1. Start producer from any one of the kafka.
2. Enter ‘1’ in the producer and check which consumer captured it.
3. Enter the numbers in sequence from the producer and which how the consumers are capturing it.



1. Enter ctrl+C and kill the consumer in 3rd kafka
2. Check the messages in the two consumers
3. Kill all the three consumers and send 15 messages in producer
4. Track the consumer offset lag using the below command

kafka-consumer-groups --bootstrap-server kafka1:9092 --describe –group

**Exercise 4 - Write Java Producer**

* 1. Create a topic name “first” using the following command

kafka-topics --create --zookeeper kafka-1:2181 --replication-factor 3 --partitions 12 --topic java-topic

* 1. Create a directory in /root/kafka in name “code” in kafka-1

mkdir ~/kafka

cd ~/kafka

* 1. Create a file called first\_producer.java and paste the below content in the first\_Producer.java file

//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 “First\_Producer”

public class first\_Producer {

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", "kafka-1:9092");

//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();

}

}

* 1. Compile the First\_Producer.java using the below command

javac -cp "/usr/share/java/kafka/\*" first\_Producer.java

Now you will be seeing a file name first\_Producer.class.

* 1. Run the producer using the below command

java -cp "/usr/share/java/kafka/\*":. first\_Producer first

You will receive a text “Message sent successfully”

* 1. To verify the message sent by producer we have created. Execute the below command

kafka-console-consumer --bootstrap-server kafka-1:9092,kafka-2:9092 --topic first --from-beginning

You will see the random generated numbers sent to the topic.

**Exercise 5 - Write Java Consumer**

1. Create a file name vi First\_Consumer.java in the direrctory /root/kafka/kafka\_2.11\*/code

2. Paste the below code in the First Consumer.java file

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 First\_Consumer {

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", "kafka-1:9092");

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());

}

}

}

You can now able to see a file name First\_Consumer.java

1. Compile the java code

javac -cp "/usr/share/java/kafka/\*":. First\_Consumer.java

1. Execute the consumer class using the below command

java -cp "/usr/share/java/kafka/\*":. first\_Consumer first

1. To verify the data from the topic you can either run the producer In another instances of kafka-1 or you can test by the below command

kafka-console-producer --kafka-list kafka-1:9092,kafka-2:9093 --topic first

You can pass any message and the message will be displayed in the of offset, key and value.

**Exercise 6**

**Creating Advanced Producer – Message batching and Compression**

1. Create a file name Adv\_Producer.java and copy the contents of First\_Producer.java and place In it.

2. Change the below properties

Change the class name to Adv\_Producer

props.put("linger.ms", 1); to props.put("linger.ms", 10000);

Changing the batch interval

props.put("batch.size", 16358); to props.put("batch.size", 32000);

Changing the batch Size

3. Add the below compression parameter to the code

props.put("compression.codec", "snappy");

4. Create a topic named “Adv-Test” using the below command

kafka-topics --create --zookeeper kafka-1:2181 --replication-factor 3 --partitions 4 --topic Adv-Test

5. Compile the producer class using the below command

javac -cp "/usr/share/java/kafka/\*" Adv\_Producer.java

1. Execute the advanced producer by the below command

java -cp "/usr/share/java/kafka/\*":. Adv\_Producer Adv-Test

1. Open Kafka-2 and execute the consumer in command line

kafka-console-consumer --bootstrap-server kafka-1:9094,kafka-2:9092, kafka-3:9092 --topic Adv-Test --from-beginning

You could see the messages getting populated in consumer end.

**Exercise 7**

**Advanced Producer with custom Partitioner**

1. Create a topic name “Custom-Topic” using the below command

kafka-topics --describe --zookeeper kafka-1:2181 --topic Custom-Topic

1. Create a directory name “custom” in /root/kafka/code/

mkdir custom

1. Navigate to /root/kafka/code/custom directory by

cd /root/kafka/code/custom

1. Create a file named MyPartitioner.java and paste the contents of custom partitioner

import kafka.producer.Partitioner;

import kafka.utils.VerifiableProperties;

public class MyPartitioner implements Partitioner {

public MyPartitioner(VerifiableProperties props) {

}

public int partition(Object employeeIdStr, int numOfPartitions) {

int partition = 0;

String stringKey = (String) employeeIdStr;

Integer intKey = Integer.parseInt(stringKey);

if (intKey > 0) {

partition = intKey % numOfPartitions;

}

System.out.println("Returning partition number [" + partition + "] " +

"for key ["+employeeIdStr+"]");

return partition;

}

}

1. Create Custom producer name MyProducer.java and paste the below contents in it.

import java.util.Date;

import java.util.Properties;

import java.util.Random;

import kafka.javaapi.producer.Producer;

import kafka.producer.KeyedMessage;

import kafka.producer.ProducerConfig;

public class MyProducer {

private static Producer<String, String> producer;

public final static String kafkaList = "kafka-1:9092,kafka-2:9093";

public final static String PARTITIONER\_IMPLEMENTATION\_CLASS

= "MyPartitioner";

private static final String TOPIC = "Custom-Topic";

public void initialize() {

Properties props = new Properties();

props.put("metadata.kafka.list", kafkaList);

props.put("serializer.class", "kafka.serializer.StringEncoder");

props.put("partitioner.class", PARTITIONER\_IMPLEMENTATION\_CLASS);

props.put("request.required.acks", "1");

ProducerConfig config = new ProducerConfig(props);

producer = new Producer<String, String>(config);

}

public void publish(String key, String message) {

KeyedMessage<String, String> data = new KeyedMessage<String, String>(

TOPIC, key, message);

producer.send(data);

}

public void closeProducer() {

producer.close();

}

public static void main(String[] args) {

Myproducer Myproducer = new Myproducer();

// Initialize the producer with required properties

Myproducer.initialize();

// Publish message to kafkas

Random rnd = new Random();

for (long employeeLogInEvent = 0; employeeLogInEvent < 10;

employeeLogInEvent++) {

String employeeId = String.valueOf(rnd.nextInt(10));

String msg = "EmployeeID:" + employeeId + ", LoginTime:" + new Date();

Myproducer.publish(employeeId, msg);

}

// Close the connection between kafka and producer

Myproducer.closeProducer();

}

}

1. Create a kafka consumer which gets the data from the topic name KafkaConsumer.java

import java.util.\*;

import kafka.consumer.Consumer;

import kafka.consumer.ConsumerConfig;

import kafka.consumer.ConsumerIterator;

import kafka.consumer.KafkaStream;

import kafka.javaapi.consumer.ConsumerConnector;

public class KafkaConsumer {

private ConsumerConnector consumerConnector = null;

private final String topic = "Custom-Topic";

public void initialize() {

Properties props = new Properties();

props.put("zookeeper.connect", "kafka-1:2181");

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

props.put("zookeeper.session.timeout.ms", "400");

props.put("zookeeper.sync.time.ms", "300");

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

ConsumerConfig conConfig = new ConsumerConfig(props);

consumerConnector = Consumer.createJavaConsumerConnector(conConfig);

}

public void consume() {

//Key = topic name, Value = No. of threads for topic

Map<String, Integer> topicCount = new HashMap<String, Integer>();

topicCount.put(topic, new Integer(1));

//ConsumerConnector creates the message stream for each topic

Map<String, List<KafkaStream<byte[], byte[]>>> consumerStreams =

consumerConnector.createMessageStreams(topicCount);

// Get Kafka stream for topic 'mytopic'

List<KafkaStream<byte[], byte[]>> kStreamList =

consumerStreams.get(topic);

// Iterate stream using ConsumerIterator

for (final KafkaStream<byte[], byte[]> kStreams : kStreamList) {

ConsumerIterator<byte[], byte[]> consumerIte = kStreams.iterator();

while (consumerIte.hasNext())

System.out.println("Message consumed from topic[" + topic + "] : " + new String(consumerIte.next().message()));

}

//Shutdown the consumer connector

if (consumerConnector != null) consumerConnector.shutdown();

}

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

KafkaConsumer kafkaConsumer = new KafkaConsumer();

// Configure Kafka consumer

kafkaConsumer.initialize();

// Start consumption

kafkaConsumer.consume();

}

}

1. Compile all the three .java files using the below commands

javac -cp "/usr/share/java/kafka/\*" MyPartitioner.java

javac -cp "/usr/share/java/kafka/\*" MyProducer.java

javac -cp "/usr/share/java/kafka/\*" KafkaConsumer.java

1. Run the Producer

java -cp "/usr/share/java/kafka/\*":. MyProducer Custom-Topic

The message will be sent to the topic Custom-Topic

1. Run the KafkaConsumer

java -cp "/usr/share/java/kafka/":. KafkaConsumer

You could see the messages with partition details.