Demo1:

1. Locate the directory which has all bat files. Example :
   D:\vidavid\Kafka_ElasticSearch\software\kafka_2.11-0.9.0.0\kafka_2.11-0.9.0.0\bin\windows>
2. Open console for consumer using below command:
   kafka-console-consumer.bat --zookeeper localhost:2181 --topic viji-topic
3. Run the below program

KafkaProducerApp.java

```
import java.util.*;

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

public class KafkaProducerApp {

	public static void main(String[] args) {
			Properties props = new Properties();
		props.put("bootstrap.servers",
"127.0.0.1:9092,127.0.0.1:9093");
		props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
		props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
		/*props.put("buffer.memory", 33554432);
		props.put("batch.size", 16384);
		*/
		KafkaProducer<String, String> myProducer = new
KafkaProducer<String, String>(props);

		try {
			for(int i=0;i<100;i++) {

				myProducer.send(new ProducerRecord<String,
String>("viji-topic", Integer.toString(i),
					"My Message :"+ Integer.toString(i)));

				//RecordMetadata future= myProducer.send(new
ProducerRecord<String, String>("viji-topic", Integer.toString(i),
				//		"My Message :"+
Integer.toString(i))).get();
				//System.out.println(future);
				//myProducer.flush();
			}
		}catch (Exception e) {
			System.out.println(e.getMessage());
				e.printStackTrace();
			}finally {
				myProducer.close();
			}

	}
```

```
}
```

Conclusion:

Check the output in the consumer terminal which has been opened.

Demo2:

```
Single Consumer in Java
- Same setup as before
Cluster setup:
- Single broker
- Two topics
- Three partitions per topic
- Single replication factor
Look for:
- kafka-producer-perf-test.sh
- subscribe() and assign()
- Add new partition
- Compare Consumer output
```

1. Locate the directory which has all bat files. Example : D:\vidavid\Kafka_ElasticSearch\software\kafka_2.11-0.9.0.0\kafka_2.11-0.9.0.0\bin\windows>
2. **Run the below command:**
3. **Create topic by using below command:**
   kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic "viji-topic"

   kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic "my-testTopic2"

   kafka-producer-perf-test.bat --topic **viji-topic** --num-record 50 --record-size 1 --throughput 10 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringDeserializer

   kafka-producer-perf-test.bat --topic **my-testTopic2** --num-record 50 --record-size 1 --throughput 10 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringDeserializer

4. Now run the below Programs:

   KafkaConsumerAppSubscribe.java

```
package org.cap.demo;
```

```
import org.apache.kafka.common.*;

import java.util.ArrayList;
import java.util.Properties;

import org.apache.kafka.clients.consumer.*;
public class KafkaConsumerAppSubscribe {

	public static void main(String[] args) {
	    // Create the Properties class to instantiate the Consumer with the desired
settings:
		Properties props = new Properties();
		props.put("bootstrap.servers", "localhost:9092, localhost:9093");
		props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
		props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

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

		// Create a KafkaConsumer instance and configure it with properties.
		KafkaConsumer<String, String> myConsumer = new KafkaConsumer<String,
String>(props);

		// Create a topic subscription list:
		ArrayList<String> topics = new ArrayList<String>();
		topics.add("viji-topic"); // Adds a TopicPartition instance representing a
topic and a partition.
		topics.add("my-testTopic2"); // Adds an additional TopicPartition instance
representing a different partition within the topic. Change as desired.
		// Assign partitions to the Consumer:
		myConsumer.subscribe(topics);

		try {
			while(true) {
				ConsumerRecords<String, String> records=myConsumer.poll(10);
				for(ConsumerRecord<String, String> record:records) {
					System.out.println(
							String.format("Topics: %s, Partition: %d,
offset: %d, Key:%s , Value: %s",

	record.topic(),record.partition(),record.offset(),record.key(),record.value()));
				}
			}
		}catch (Exception e) {
				e.printStackTrace();
			}finally {
				myConsumer.close();
			}
		}

}
```

KafkaConsumerAppAssign.java

```
package org.cap.demo;
import org.apache.kafka.common.*;

import java.util.ArrayList;
import java.util.Properties;
```

```
import org.apache.kafka.clients.consumer.*;
public class KafkaConsumerAppAssign {

	public static void main(String[] args) {
		// Create the Properties class to instantiate the Consumer with the
desired settings:
		Properties props = new Properties();
		props.put("bootstrap.servers", "localhost:9092, localhost:9093");
		props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
		props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

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

		// Create a KafkaConsumer instance and configure it with properties.
		KafkaConsumer<String, String> myConsumer = new KafkaConsumer<String,
String>(props);

		// Create a topic subscription list:
		ArrayList<TopicPartition> partitions = new ArrayList<TopicPartition>();
		partitions.add(new TopicPartition("viji-topic", 0)); // Adds a TopicPartition
instance representing a topic and a partition.
		partitions.add(new TopicPartition("my-testTopic2",0)); // Adds an additional
TopicPartition instance representing a different partition within the topic. Change as
desired.
		// Assign partitions to the Consumer:
		myConsumer.assign(partitions);


		try {
			while(true) {
				ConsumerRecords<String, String> records=myConsumer.poll(10);
				for(ConsumerRecord<String, String> record:records) {
					System.out.println(
							String.format("Topics: %s, Partition: %d,
offset: %d, Key:%s , Value: %s",

	record.topic(),record.partition(),record.offset(),record.key(),record.value()));
				}
			}
		}catch (Exception e) {
				e.printStackTrace();
			}finally {
				myConsumer.close();
			}
		}


}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.cap.kafkademo</groupId>
```

```
  <artifactId>KafkaDemo_Java</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>


            <dependency>
<groupId>org.apache.kafka</groupId>
<artifactId>kafka-clients</artifactId>
<version>0.9.0.0</version>
</dependency>
        <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-api</artifactId>
       <version>1.7.5</version>
   </dependency>
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.5</version>
   </dependency>
        </dependencies>
</project>
```

You can alter the topic by using below command:

a. kafka-topics.bat --zookeeper localhost:2181 --alter --topic viji-topic --partitions 4
b. kafka-topics.bat --describe --zookeeper localhost:2181 --topic viji-topic
c. Run the below command Again and check the output:
kafka-producer-perf-test.bat --topic viji-topic --num-record 50 --record-size 1 --throughput 10 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringDeserializer

Conclusion:

Execute the above code and see the result.

Consumer Group Demo:

Consumer Group comprising of Javabased
Consumer applications
Setup:
- Three Consumers with same group id
- Consuming a single topic with three partitions

Look for:
- Shared topic consumption
- Adding an additional Consumer
- Adding an additional topic
- Forcing a rebalance

1. Create new topic called “my-big-topic” using the below command:

kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic "my-big-topic"

2. Describe using the following command:
   kafka-topics.bat --zookeeper localhost:2181 --alter --topic my-big-topic --partitions 4

3. Now create below Java programs 3 for consumer which is under one group call test-group and one producer app. Names are as mentioned below:
   a. ProducerApp.java
   b. ConsumerGroupApp01.java
   c. ConsumerGroupApp02.java
   d. ConsumerGroupApp03.java

ConsumerGroupApp01.java

```
package org.cap.consumergroup;
import java.util.ArrayList;
import java.util.Properties;

import org.apache.kafka.clients.consumer.*;
public class ConsumerGroupApp01 {

	public static void main(String[] args) {
		// Create the Properties class to instantiate the Consumer with the
desired settings:
		Properties props = new Properties();
		props.put("bootstrap.servers", "localhost:9092, localhost:9093");
		props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
		props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

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

		// Create a KafkaConsumer instance and configure it with properties.
		KafkaConsumer<String, String> myConsumer = new KafkaConsumer<String,
String>(props);

		// Create a topic subscription list:
		ArrayList<String> topics = new ArrayList<String>();
		topics.add("my-big-topic");
		myConsumer.subscribe(topics);

		try {
			while(true) {
				ConsumerRecords<String, String> records=myConsumer.poll(10);
				for(ConsumerRecord<String, String> record:records) {
					System.out.println(
							String.format("Topics: %s, Partition: %d,
offset: %d, Key:%s , Value: %s",
```

```
        record.topic(),record.partition(),record.offset(),record.key(),record.value().to
UpperCase()));
                        }
                }
         }catch (Exception e) {
                        e.printStackTrace();
                }finally {
                        myConsumer.close();
                }
        }

}
```

ConsumerGroupApp02.java

```
package org.cap.consumergroup;

import java.util.ArrayList;
import java.util.Properties;

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

public class ConsumerGroupApp02 {

        public static void main(String[] args) {
                // Create the Properties class to instantiate the Consumer with the
desired settings:
      Properties props = new Properties();
      props.put("bootstrap.servers", "localhost:9092, localhost:9093");
      props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
      props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

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

      // Create a KafkaConsumer instance and configure it with properties.
      KafkaConsumer<String, String> myConsumer = new KafkaConsumer<String,
String>(props);

      // Create a topic subscription list:
      ArrayList<String> topics = new ArrayList<String>();
      topics.add("my-big-topic");
      myConsumer.subscribe(topics);

      try {
       while(true) {
                ConsumerRecords<String, String> records=myConsumer.poll(10);
                for(ConsumerRecord<String, String> record:records) {
                        System.out.println(
                                        String.format("Topics: %s, Partition: %d, offset:
%d, Key:%s , Value: %s",

        record.topic(),record.partition(),record.offset(),record.key(),record.value().to
UpperCase()));
                }
        }
```

```
      }catch (Exception e) {
                        e.printStackTrace();
                }finally {
                        myConsumer.close();
                }
        }
}
```

ConsumerGroupApp03.java

```
package org.cap.consumergroup;

import java.util.ArrayList;
import java.util.Properties;

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

public class ConsumerGroupApp03 {

        public static void main(String[] args) {
                  // Create the Properties class to instantiate the Consumer with the
desired settings:
    Properties props = new Properties();
    props.put("bootstrap.servers", "localhost:9092, localhost:9093");
    props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
    props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

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

    // Create a KafkaConsumer instance and configure it with properties.
    KafkaConsumer<String, String> myConsumer = new KafkaConsumer<String,
String>(props);

    // Create a topic subscription list:
    ArrayList<String> topics = new ArrayList<String>();
    topics.add("my-big-topic");
    myConsumer.subscribe(topics);

    try {
       while(true) {
                ConsumerRecords<String, String> records=myConsumer.poll(10);
                for(ConsumerRecord<String, String> record:records) {
                        System.out.println(
                                        String.format("Topics: %s, Partition: %d, offset:
%d, Key:%s , Value: %s",

        record.topic(),record.partition(),record.offset(),record.key(),record.value().to
UpperCase()));
                }
        }
    }catch (Exception e) {
                        e.printStackTrace();
                }finally {
                        myConsumer.close();
                }
        }
}
```

ProducerApp.java

```
package org.cap.consumergroup;
import java.util.*;

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

public class ProducerApp {

	public static void main(String[] args) {

				Properties props = new Properties();
		props.put("bootstrap.servers", "127.0.0.1:9092,127.0.0.1:9093");
		props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
		props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
		/*props.put("buffer.memory", 33554432);
		props.put("batch.size", 16384);
		*/
		KafkaProducer<String, String> myProducer = new KafkaProducer<String,
String>(props);

		try {
		    for(int i=0;i<100;i++) {

				myProducer.send(new ProducerRecord<String, String>("my-big-topic",
"abcdefghijklmnopqrstuvwxyz"));


		    }
		}catch (Exception e) {
		    System.out.println(e.getMessage());
				e.printStackTrace();
			}finally {
				myProducer.close();
			}

	}
	}
```

4. Now start the program once by one in the below order

a. ConsumerGroupApp01.java
b. ConsumerGroupApp02.java
c. ConsumerGroupApp03.java
d. ProducerApp.java

You can see all the consumer terminal the partition 0,1,2, results individually.

c

5. Now increate the no of partition in my-big-topic using below command:

   kafka-topics.bat --zookeeper localhost:2181 --alter --topic my-big-topic --partitions 4

   kafka-topics.bat --describe --zookeeper localhost:2181 --topic my-big-topic
6. Execute all the application the below order:
   a. ConsumerGroupApp01.java
   b. ConsumerGroupApp02.java
   c. ConsumerGroupApp03.java
   d. ConsumerGroupApp04.java
   e. ProducerApp.java

   This time you can see the newly added partition will appear in one of the terminal.

7. Now kill 2 consumer window (ConsumerGroupApp02 and ConsumerGroupApp04) and execute the same. This time partition will be automatically load balanced in the remaining 2 terminals.

Conclusion:

From the above demo we learnt how to create consumer group.