**Session 24:**

# KAFKA INTRODUCTION

**Assignment 2**

# Assignment 24: Apache Kafka II Assignment Problems

## Problem Statement

### Data Sets Present:

**1. dataset_producer.txt :**

**Terminal Execution :**

```
[acadgild@localhost ~]$ cat /home/acadgild/Desktop/dataset_producer.txt
ItemTopic-{"item_id":"101"}-{"user_id":"U101"}
UserTopic-{"name":"John"}-{"exp":16}
ItemTopic-{"item_id":"101"}-{"user_id":"U106"}
UserTopic-{"name":"Mark"}-{"exp":18}
ItemTopic-{"item_id":"102"}-{"user_id":"U110"}
UserTopic-{"name":"Cylin"}-{"exp":15}
ItemTopic-{"item_id":"102"}-{"user_id":"U101"}
UserTopic-{"name":"Prod"}-{"exp":14}
ItemTopic-{"item_id":"104"}-{"user_id":"U102"}
UserTopic-{"name":"Abhay"}-{"exp":17}
ItemTopic-{"item_id":"107"}-{"user_id":"U104"}
UserTopic-{"name":"Misano"}-{"exp":19}[acadgild@localhost ~]$
```

## Task 1

**Create a java program MyKafkaProducer.java that takes a file name and delimiter as input arguments.
It should read the content of file line by line.
Fields in the file are in following order
1. Kafka Topic Name
2. Key
3. value
For every line Created, insert the key and value to the repsective Kafka broker in a fire and forget mode.**

**After record is sent, it should print appropriate message on screen.**
**Pass dataset_producer.txt as the input file and -as delimiter.**

**Solution :**

**1. MyKafkaProducer.java :**

```
package com.acadgild.kafka.api;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.util.Scanner;

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

public class MyKafkaProducer {

	public static void main(String[] args){

		List<String> lines = new ArrayList<String>();

		try {
			Scanner scanner = new Scanner(new
File("/home/acadgild/Desktop/dataset_producer.txt"));
			while (scanner.hasNextLine()) {
			//	System.out.println(scanner.nextLine());
			lines.add(scanner.nextLine());
			}
			scanner.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
```

```
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("acks", "0");
        props.put("retries", 0);
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("partitioner.class", "com.acadgild.kafka.api.RangePartitioner");

        KafkaProducer<String, String> producer = new KafkaProducer<>(props);
        ProducerRecord<String, String> producerRecord = null;

        for (String line : lines) {
            String[] token = line.split("-");
            producerRecord = new ProducerRecord<String, String>(token[0],
token[1],token[2]);
            producer.send(producerRecord);
            System.out.println(token[0] +" " +token[1]+" "+token[2]+" Sent!");
        }
        producer.close();
       }


}
```

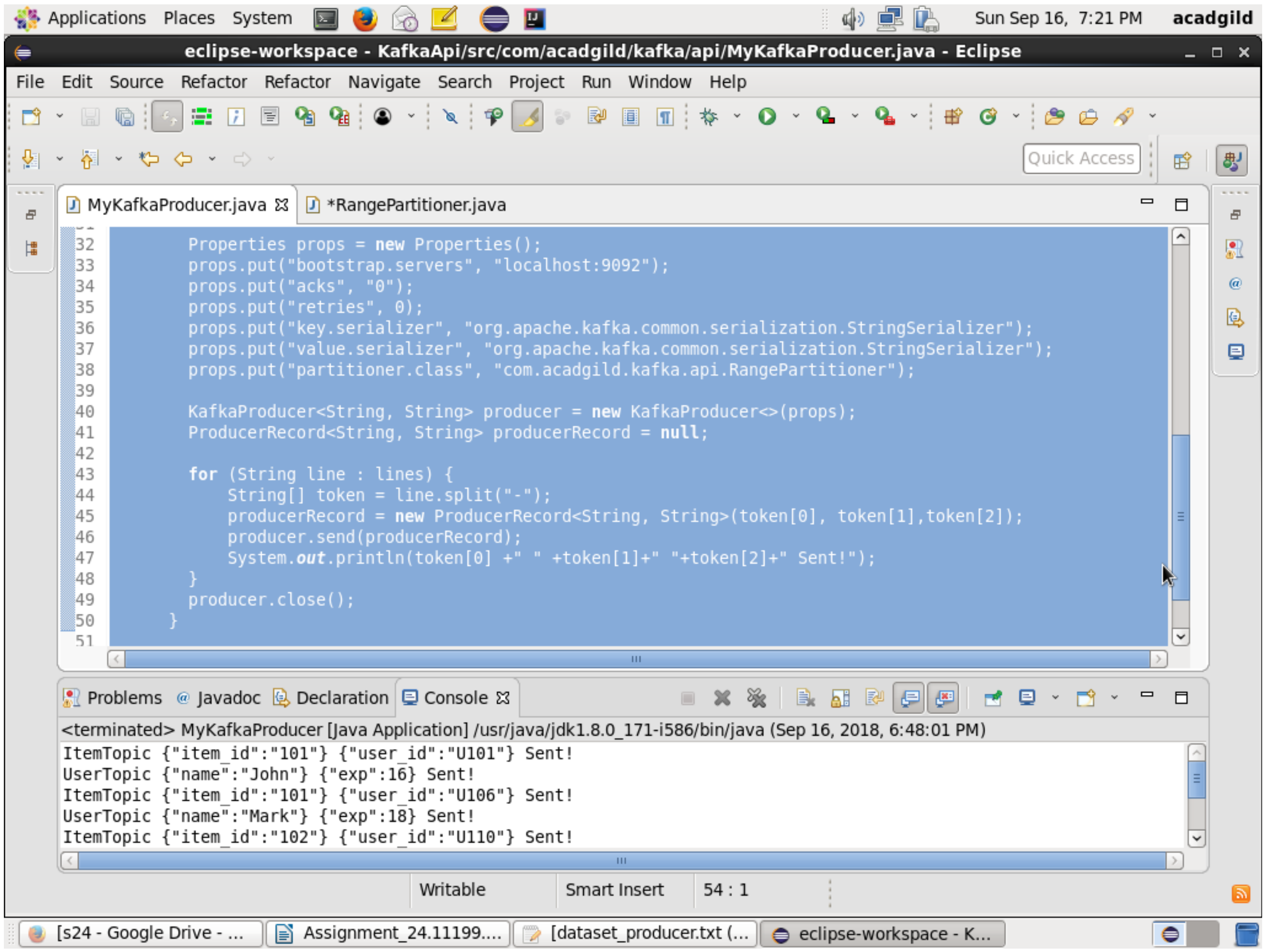
**2.** RangePartitioner.java

```
package com.acadgild.kafka.api;

import java.util.List;
import java.util.Map;
import org.apache.kafka.clients.producer.Partitioner;
import org.apache.kafka.common.Cluster;
import org.apache.kafka.common.PartitionInfo;


public class RangePartitioner implements Partitioner{

      public void configure(Map<String, ?> configs) {}

      public int partition(String topic, Object key, byte[] keyBytes,
            Object value, byte[] valueBytes, Cluster cluster) {
                  List<PartitionInfo> partitions =
cluster.partitionsForTopic(topic);

                  // Getting the number of partitions for the topic
                  int numPartitions = partitions.size();
                        System.out.println("Topic: "+ topic + "The key: " +
keyBytes.toString() + " value: " + valueBytes.toString() +"\n");
                        return numPartitions - 1;

            }

      public void close() {}


}
```

## 3. Result :

```
ItemTopic {"item_id":"101"} {"user_id":"U101"} Sent!
UserTopic {"name":"John"} {"exp":16} Sent!
ItemTopic {"item_id":"101"} {"user_id":"U106"} Sent!
UserTopic {"name":"Mark"} {"exp":18} Sent!
ItemTopic {"item_id":"102"} {"user_id":"U110"} Sent!
UserTopic {"name":"Cylin"} {"exp":15} Sent!
ItemTopic {"item_id":"102"} {"user_id":"U101"} Sent!
UserTopic {"name":"Prod"} {"exp":14} Sent!
ItemTopic {"item_id":"104"} {"user_id":"U102"} Sent!
UserTopic {"name":"Abhay"} {"exp":17} Sent!
ItemTopic {"item_id":"107"} {"user_id":"U104"} Sent!
UserTopic {"name":"Misano"} {"exp":19} Sent!
```

## 4. Output :



Applications Places System
Sun Sep 16, 7:21 PM acadgild
eclipse-workspace - KafkaApi/src/com/acadgild/kafka/api/MyKafkaProducer.java - Eclipse
File Edit Source Refactor Refactor Navigate Search Project Run Window Help
Quick Access
MyKafkaProducer.java
*RangePartitioner.java
Problems
Javadoc
Declaration
Console
Writable
Smart Insert
54 : 1
[s24 - Google Drive - ...
Assignment_24.11199....
[dataset_producer.txt (...
eclipse-workspace - K...

```
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("acks", "0");
        props.put("retries", 0);
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("partitioner.class", "com.acadgild.kafka.api.RangePartitioner");

        KafkaProducer<String, String> producer = new KafkaProducer<>(props);
        ProducerRecord<String, String> producerRecord = null;

        for (String line : lines) {
            String[] token = line.split("-");
            producerRecord = new ProducerRecord<String, String>(token[0], token[1],token[2]);
            producer.send(producerRecord);
            System.out.println(token[0] +" " +token[1]+" "+token[2]+" Sent!");
        }
        producer.close();
    }
```

```
<terminated> MyKafkaProducer [Java Application] /usr/java/jdk1.8.0_171-i586/bin/java (Sep 16, 2018, 6:48:01 PM)
ItemTopic {"item_id":"101"} {"user_id":"U101"} Sent!
UserTopic {"name":"John"} {"exp":16} Sent!
ItemTopic {"item_id":"101"} {"user_id":"U106"} Sent!
UserTopic {"name":"Mark"} {"exp":18} Sent!
ItemTopic {"item_id":"102"} {"user_id":"U110"} Sent!
```

## Task 2

**Modify the previous program MyKafkaProducer.java and create a new Java program KafkaProducerWithAck.java.**

**This should perform the same task as of KafkaProducer.java with some modification. When passing any data to a topic, it should wait for acknowledgement.**

**After acknowledgement is received from the broker, it should print the key and value which has been written to a specified topic.**

**The application should attempt for 3 retries before giving any exception. Pass dataset_producer.txt as the input file and -as delimiter.**

## Solution :

### 1. MyKafkaProducerWithAck.java :

```
package com.acadgild.kafka.api;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.util.Scanner;
import java.util.concurrent.ExecutionException;

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

public class MyKafkaProducerWithAck {

        public static void main(String[] args){

                List<String> lines = new ArrayList<String>();

                try {
                        Scanner scanner = new Scanner(new
File("/home/acadgild/Desktop/dataset_producer.txt"));
                        while (scanner.hasNextLine()) {
                        //      System.out.println(scanner.nextLine());
```

```
                    lines.add(scanner.nextLine());
                    }
                    scanner.close();
            } catch (FileNotFoundException e) {
                    e.printStackTrace();
            }



    Properties props = new Properties();
    props.put("bootstrap.servers", "localhost:9092");
    props.put("acks", "1");
    props.put("retries", 3);
    props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
    props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
    props.put("partitioner.class", "com.acadgild.kafka.api.RangePartitioner");

    KafkaProducer<String, String> producer = new KafkaProducer<>(props);
    ProducerRecord<String, String> producerRecord = null;

    for (String line : lines) {
            String[] token = line.split("-");
            producerRecord = new ProducerRecord<String, String>(token[0], token[1],token[2]);
            int retries = 0;
            try {
                    retries++;
                            producer.send(producerRecord).get();
                            System.out.println(token[0] +" " +token[1]+" "+token[2]+" Sent!");
                    } catch (InterruptedException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                            if(retries >= 3)
                            System.out.println("Error :" + e.getMessage());
                    } catch (ExecutionException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                            if(retries >= 3)
                            System.out.println("Error :" + e.getMessage());
                    }

    }
    producer.close();
   }


}
```

## 2. RangePartitioner.java

```
package com.acadgild.kafka.api;

import java.util.List;
import java.util.Map;
import org.apache.kafka.clients.producer.Partitioner;
import org.apache.kafka.common.Cluster;
import org.apache.kafka.common.PartitionInfo;


public class RangePartitioner implements Partitioner{

      public void configure(Map<String, ?> configs) {}

      public int partition(String topic, Object key, byte[] keyBytes,
            Object value, byte[] valueBytes, Cluster cluster) {
                  List<PartitionInfo> partitions =
cluster.partitionsForTopic(topic);

                  // Getting the number of partitions for the topic
                  int numPartitions = partitions.size();
                        System.out.println("Topic: "+ topic + "The key: " +
keyBytes.toString() + " value: " + valueBytes.toString() +"\n");
                        return numPartitions - 1;

            }

      public void close() {}


}
```

## 3. Result :

```
ItemTopic {"item_id":"101"} {"user_id":"U101"} Sent!
UserTopic {"name":"John"} {"exp":16} Sent!
ItemTopic {"item_id":"101"} {"user_id":"U106"} Sent!
UserTopic {"name":"Mark"} {"exp":18} Sent!
ItemTopic {"item_id":"102"} {"user_id":"U110"} Sent!
UserTopic {"name":"Cylin"} {"exp":15} Sent!
ItemTopic {"item_id":"102"} {"user_id":"U101"} Sent!
UserTopic {"name":"Prod"} {"exp":14} Sent!
ItemTopic {"item_id":"104"} {"user_id":"U102"} Sent!
UserTopic {"name":"Abhay"} {"exp":17} Sent!
ItemTopic {"item_id":"107"} {"user_id":"U104"} Sent!
UserTopic {"name":"Misano"} {"exp":19} Sent!
```