Session 24 – Apache Kafka - II

Assignment 1

The dataset provided for this task: dataset_producer.txt contains of a '-' delimited list of elements specifying the topic name, key and value.

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, insert the key and value to the respective 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.

The two topics specified in the text file are created beforehand so that the data from file can be inserted into the topic.

> *bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 2 --topic UserTopic*
> *bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 2 --topic ItemTopic*

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --pa
rtitions 2 --topic ItemTopic
Created topic "ItemTopic".
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --pa
rtitions 2 --topic UserTopic
Created topic "UserTopic".
[acadgild@localhost kafka_2.12-0.10.1.1]$
```

The topics are created and now the data can be inserted into these topics.

Below is the code for the above task

MyKafkaProducer.java

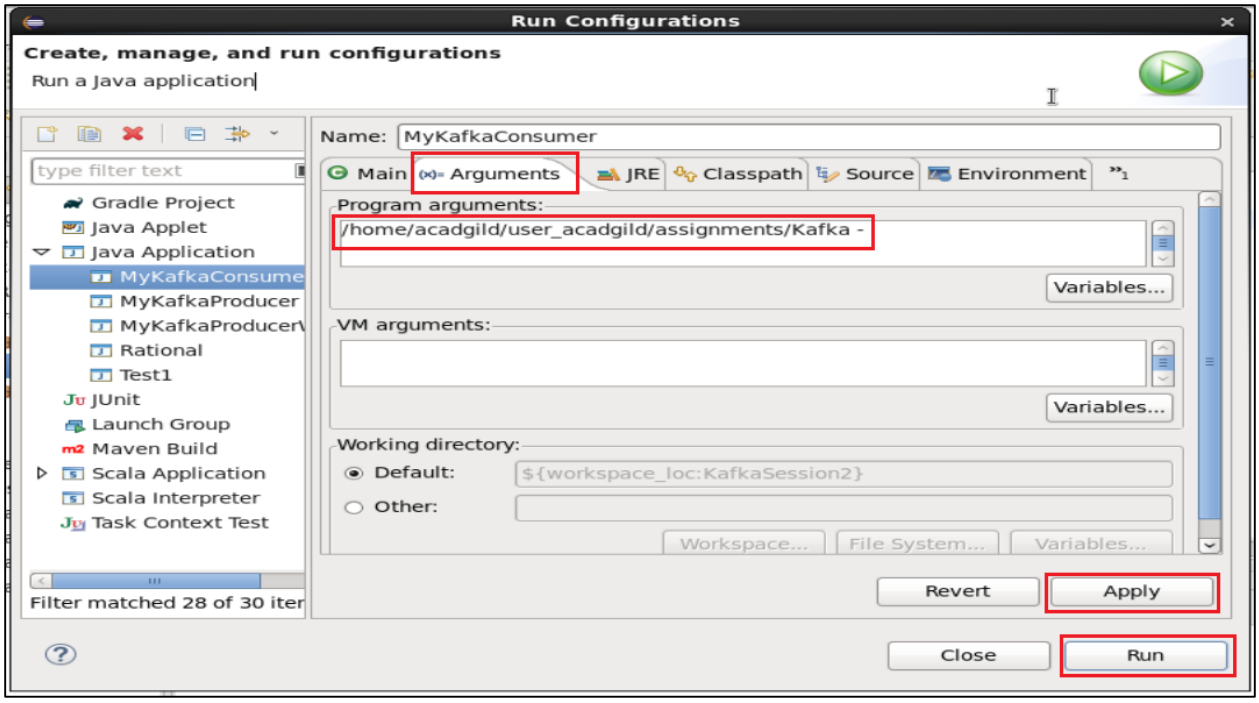
```
package com.acadgild.kafka;

import java.io.File;
import java.util.Properties;
import java.util.Scanner;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class MyKafkaProducer {
        public static void main(String[] args) throws Exception {
                // Check arguments length value
                if (args.length == 0) {
                        System.out.println("Inckude the topic name in the command");
                        return;
                }
                // create instance for properties to access producer configs
                Properties props = new Properties();
                // Assign localhost id
                props.put("bootstrap.servers", "localhost:9092");
                // 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
// 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");
Scanner scanner = new Scanner(new File(args[0].toString()));
Scanner datascanner = null;
int index = 0;
String topicname = "";
String key = "";
String value = "";
while (scanner.hasNextLine()) {
        datascanner = new Scanner(scanner.nextLine());
        datascanner.useDelimiter(args[1].toString());
        while (datascanner.hasNext()) {
                String data = datascanner.next();
                if (index == 0)
                        topicname = data;
                else if (index == 1)
                        key = data;
                else if (index == 2)
                        value = data;
                else
                        System.out.println("Invalid data");
```

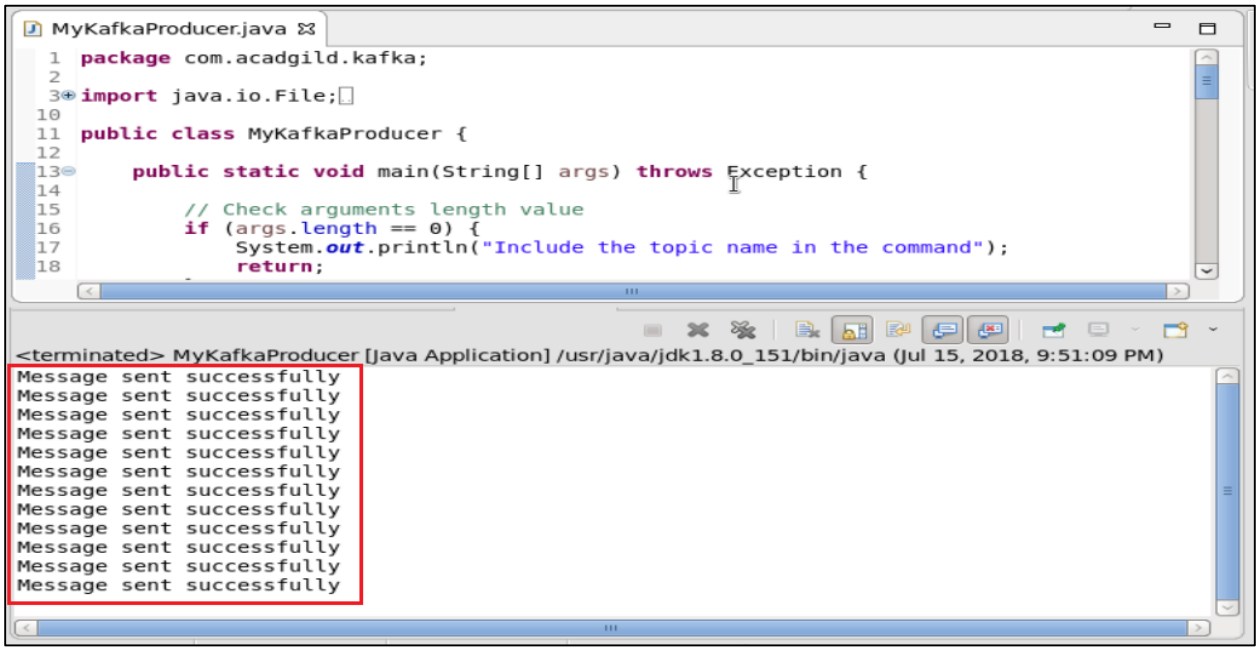
```
                    index++;
                }
                try {
                    Producer<String, String> producer = new
        KafkaProducer<String, String>(props);
                    producer.send(new ProducerRecord<String, String>(topicname,
                    key, value));
                    System.out.println("Message sent successfully");
                    producer.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
                index = 0;
            }
            scanner.close();
        }
    }
```

The arguments passed to this program are the file location of the dataset_producer.txt file and the delimiter.

In Eclipse, under Run Configurations, go to Arguments Tab and the arguments can be passed under Program Arguements. Then, click on Apply and then Run.



The program is run and according to the requirement, a success message is displayed on screen after every line from the file is inserted.



Since there are 12 lines in the text file and each one is inserted into the respective topic, there are 12 success messages in the console output.

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic UserTopic
--property print.key=true --property key.separator=',' --from-beginning
{"name":"John"},{"exp":16}
{"name":"Mark"},{"exp":18}
{"name":"Prod"},{"exp":14}
{"name":"Abhay"},{"exp":17}
{"name":"Misano"},{"exp":19}
{"name":"Cylin"},{"exp":15}
```

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic ItemTopic
--property print.key=true --property key.separator=',' --from-beginning
{"item_id":"102"},{"user_id":"U110"}
{"item_id":"102"},{"user_id":"U101"}
{"item_id":"104"},{"user_id":"U102"}
{"item_id":"101"},{"user_id":"U101"}
{"item_id":"101"},{"user_id":"U106"}
{"item_id":"107"},{"user_id":"U104"}
```

When we create a consumer from the console to verify whether the data has been pushed or not, we can see that the data has been inserted into the topic. 6 records belong to the UserTopic which got inserted accordingly and 6 belong to the ItemTopic and they too got inserted accordingly.

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.

Below is the code for the above task

MyKafkaProducerWithAck.java

```
package com.acadgild.kafka.task2;


import java.io.File;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.Properties;
```

```
import java.util.Scanner;
import java.util.Set;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class MyKafkaProducerWithAck {
	public static void main(String[] args) throws Exception {
		// Check arguments length value
		if (args.length == 0) {
			System.out.println("Include the topic name in the command");
			return;
		}
		// 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", 3);
		// 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");
                Scanner scanner = new Scanner(new File(args[0].toString()));
                Scanner datascanner = null;
                int index = 0;
                String topicname = "";
                String key = "";
                String value = "";
                ArrayList<String> topics = new ArrayList<String>();
                while (scanner.hasNextLine()) {
                        datascanner = new Scanner(scanner.nextLine());
                        datascanner.useDelimiter(args[1].toString());
                        while (datascanner.hasNext()) {
                                String data = datascanner.next();
                                if (index == 0)
                                        topicname = data;
                                else if (index == 1)
                                        key = data;
                                else if (index == 2)
                                        value = data;
                                else
                                        System.out.println("Invalid data");
                                index++;
                        }
                        try {
```

```
                    Producer<String, String> producer = new KafkaProducer<String,
String>(props);

                    //get method waits for the acknowledgement from the topic and
                    //returns a recordMetadata object.

                    producer.send(new ProducerRecord<String, String>(topicname,
                    key, value)).get();

                    System.out.println("Message sent successfully");

                    //adding all the topics to a list to retrieve data from these topics

                    topics.add(topicname);

                    producer.close();

                } catch (Exception e) {

                    e.printStackTrace();

                }

                index = 0;

            }

            //finding out the unique topics from the list of topics

            Set<String> uniquetopics = new HashSet<String>(topics);

            //sending the unique topics as a list to the consumer so that the consumer
            //consumes from them

            com.acadgild.kafka.task2.MyKafkaConsumer.consumeData(new
            ArrayList(uniquetopics));

            scanner.close();

        }


}
```

The arguments passed to this program are the file location of the dataset_producer.txt file and the delimiter.

MyKafkaConsumer.java

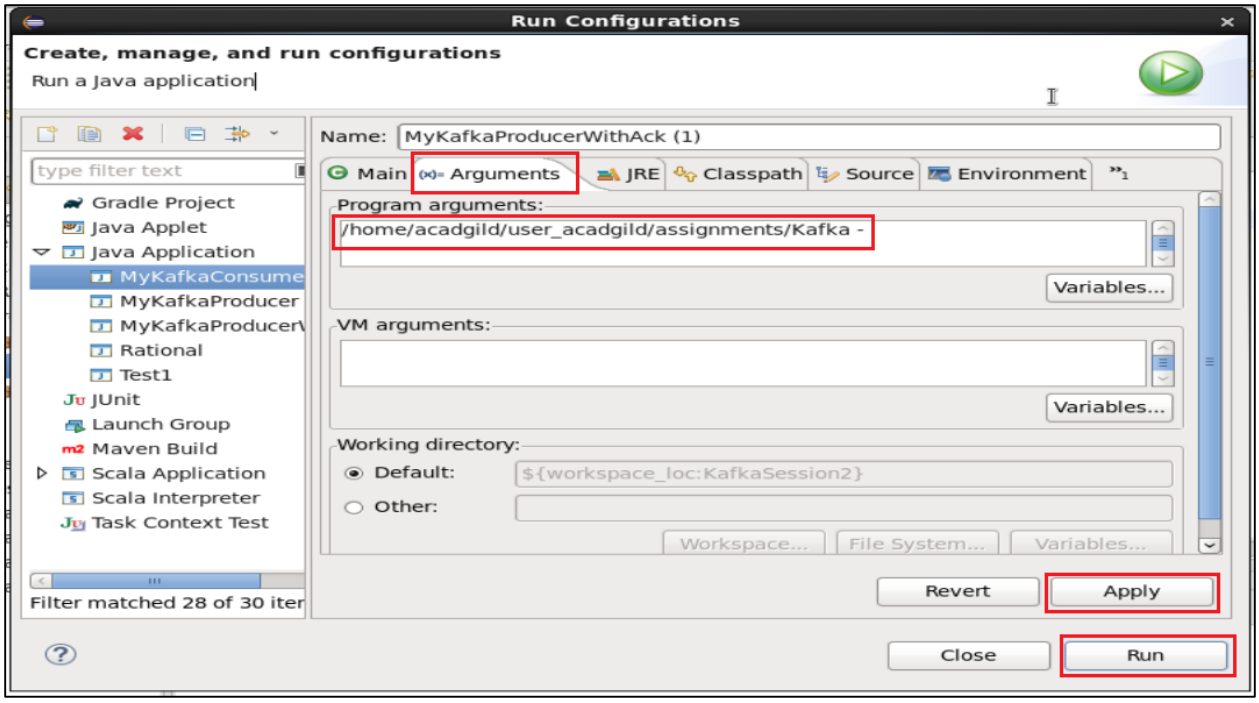
```
package com.acadgild.kafka.task2;
```

```
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 MyKafkaConsumer {
	public static void consumeData(ArrayList topics) {
		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", "10000");
		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(topics);
		System.out.println("Subscribed to topic(s) " + topics);
		try {
			while (true) {
				consumer.poll(0);
```

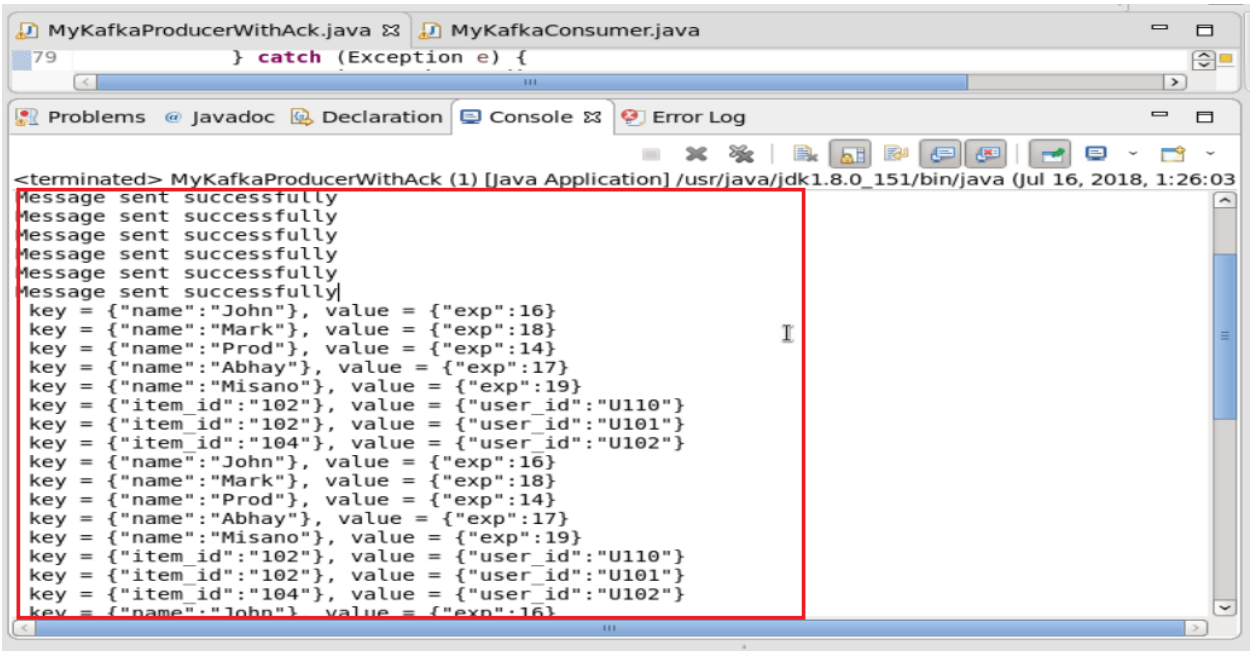
```
                    consumer.seekToBeginning(consumer.assignment());
                    ConsumerRecords<String, String> records = consumer.poll(0);
                    for (ConsumerRecord<String, String> record : records)
                        // print the key and value for the consumer records.
                        System.out.printf(" key = %s, value = %s\n", record.key(),
record.value());
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                consumer.close();
            }
        }
}
```

In the consumer, the list of topics is set as subscriptions for the consumer object and the consumer consumes and displays the contents of the topics.

In Eclipse, under Run Configurations, go to Arguments Tab and the arguments can be passed under Program Arguments. Then, click on Apply and then Run.



The program is run and a success message and the details that were inserted are also displayed on the screen.



As seen in the screenshot above,

1. The success message upon inserting the data from the text file is shown,
2. Then, the data consumed from the consumer is also shown.