BIG DATA HADOOP AND SPARK DEVLOPMENT

ASSIGNMENT 24

Table of Contents:

BIG DATA HADOOPAND SPARK DEVELOPMENT

1. Introduction

In this assignment, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

2. Objective

This Assignment consolidates the deeper understanding of the Session – 24 Apache Kafka II

3. Problem Statement

- 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 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.
  - LINK: https://drive.google.com/file/d/0B_Qjau8wv1KoSnR5eHpKOF9rTFU/view?usp=sharing
- 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.

4. Expected Output

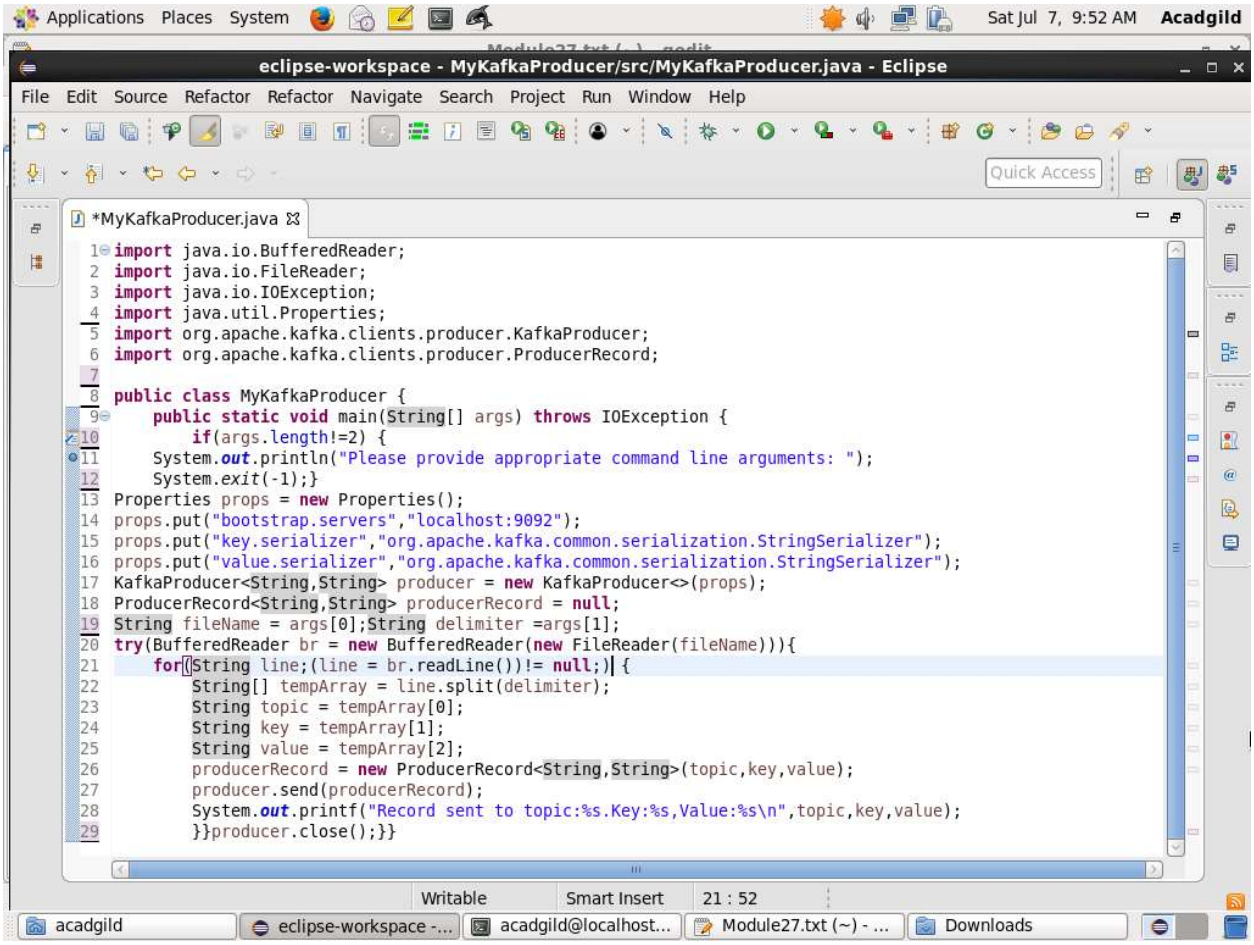
- Task 1
    - Create a java program MyKafkaProducer.java that takes a file name and delimiter as input arguments.

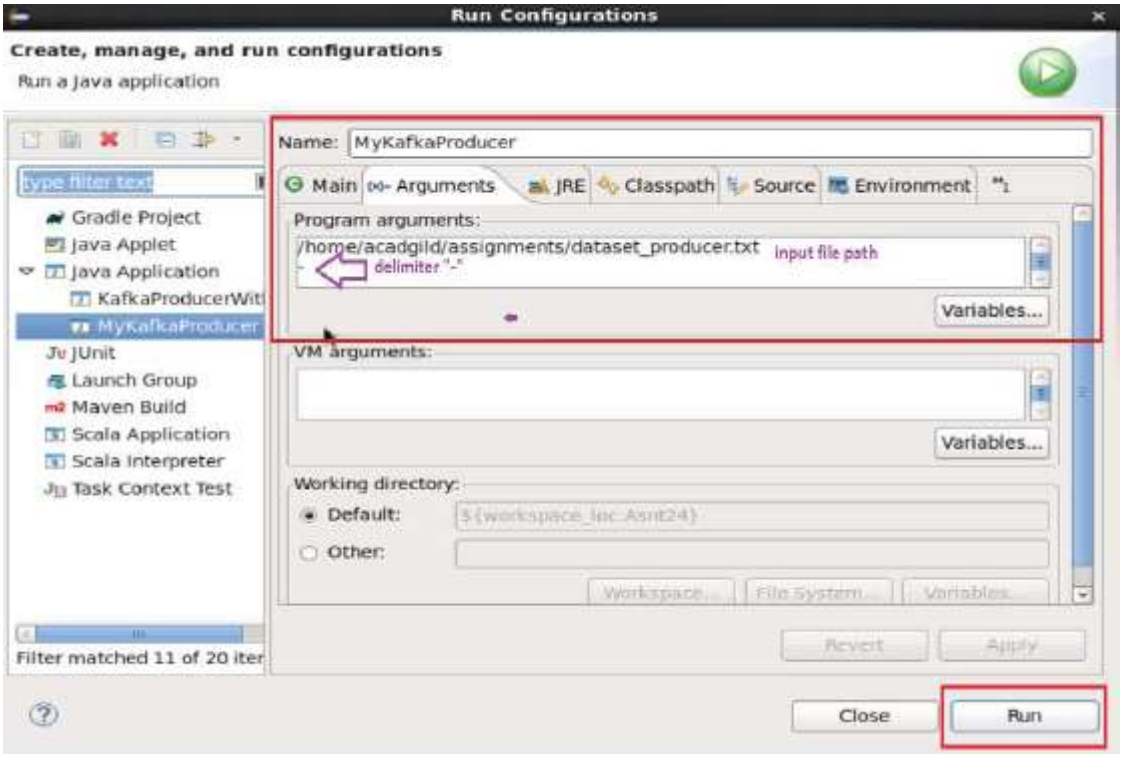
      It should read the content of file line by line.
      Fields in the file are in following order
      1. Kafka Topic Name
      2. Key
      3. value
      For every line, insert the key and value to the repsective Kafka broker in a fire and forget mode.
      After record is sent, it should print appropriate message on screen.
      Pass **dataset_producer.txt** as the input file and -as delimiter.
    - LINK: https://drive.google.com/file/d/0B_Qjau8wv1KoSnR5eHpKOF9rTFU/view?usp=sharing

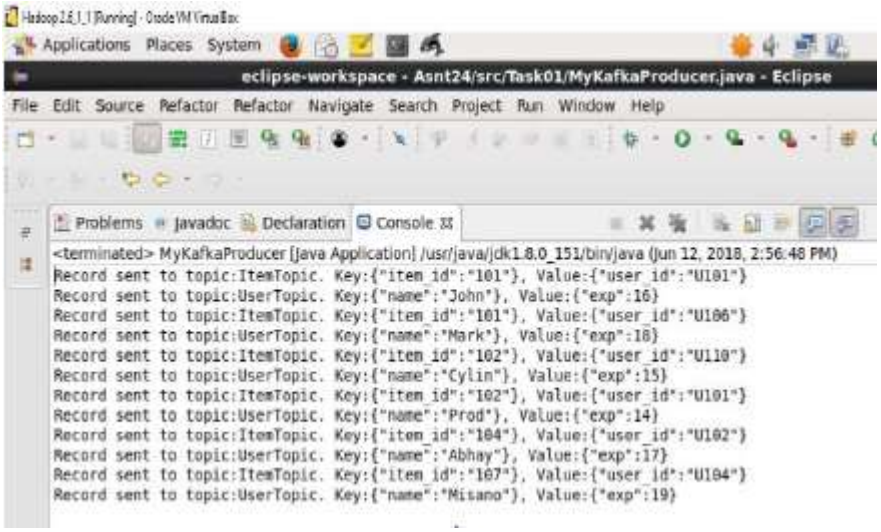
code for MyKafkaProducer.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class MyKafkaProducer {

    public static void main(String[] args) throws IOException {
        if (args.length != 2) {
            System.out.println("Please provide appropriate command line
arguments");
            System.exit(-1);
        }
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        KafkaProducer<String, String> producer = new
KafkaProducer<>(props);
        ProducerRecord<String, String> producerRecord = null;
        String fileName = args[0];
        String delimiter = args[1];
        try(BufferedReader br = new BufferedReader(new
FileReader(fileName))) {
            for(String line; (line = br.readLine()) != null; ) {
                String[] tempArray = line.split(delimiter);
                String topic = tempArray[0];
                String key = tempArray[1];
                String value = tempArray[2];
```

```
                producerRecord = new ProducerRecord<String, String>(topic,
key, value);
                producer.send(producerRecord);
                System.out.printf("Record sent to topic:%s. Key:%s,
Value:%s\n", topic, key, value);
            } } producer.close(); } }
```







Now, we run the console consumer commands on terminal to view the output of the program, using the below command:

o To read contents of ItemTopic

```
./bin/kafka-console-consumer.sh --topic ItemTopic --from-beginning \
--zookeeper localhost:2181 --property print.key=true
```

o To read contents of UserTopic

```
./bin/kafka-console-consumer.sh --topic UserTopic --from-beginning \
--zookeeper localhost:2181 \
--property print.key=true
```

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic ItemTopic --from-beginning \
> --zookeeper localhost:2181 \
> --property print.key=true
```

```
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"item_id":"107"}       {"user_id":"U104"}
^CProcessed a total of 6 messages
```

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic UserTopic --from-beginning \
> --zookeeper localhost:2181 \
> --property print.key=true
```

```
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Cylin"}        {"exp":15}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
```

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.

- Imports required for the program is given below:

```
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import java.util.concurrent.ExecutionException;
```

- This is the class called “KafkaProducerWithAck” which takes two arguments (Input File name and delimiter) in the command line.

```
public class KafkaProducerWithAck {
public static void main(String[] args) throws IOException, InterruptedException,
ExecutionException{
if (args.length != 2) {
System.out.println("Please provide appropriate command line arguments");
System.exit(-1);
}
```

- We configure the properties for KafkaProducer:

o We create a new instance of Properties called props.
o Using this instance we add properties to kafkaProducer like, bootstrapserver/meta-data-brokerlist, key and value serializers,acks and retries.

- Acks “all”- this means that the producer will receive a success response from the broker once all in-sync replicas received the message.
- Retries 3- When the producer receives an error message from the server, the error could be transient (e.g., a lack of leader for a partition). In this case, the value of the retries parameter will control how many times the producer will retry sending the message before giving up and notifying the client of an issue.

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("acks", "all");
```

```
props.put("retries", 3);
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
```

- We then instantiate the KafkaProducer class called producer, we have mentioned string in <> because both key and value are String.
- We add the properties instance (props)to KafkaProducer instance.
- We also instantiate ProducerRecord as producerRecord

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

- Now we take the data provided in the command line i.e. file name and delimiter and save them in the array of string variables called filename and delimiter

```
String fileName = args[0];
String delimiter = args[1];
```

- We read the contents of the input file, and save their contents arrays in different variables:

o We save the topic name i.e. first part of array(0th index elements) in String variable topic and similarly we save key and value variables too.

```
try(BufferedReader br = new BufferedReader(new FileReader(fileName))) {
for(String line; (line = br.readLine()) != null; ) {
String[] tempArray = line.split(delimiter);
String topic = tempArray[0];
String key = tempArray[1];
String value = tempArray[2];
```

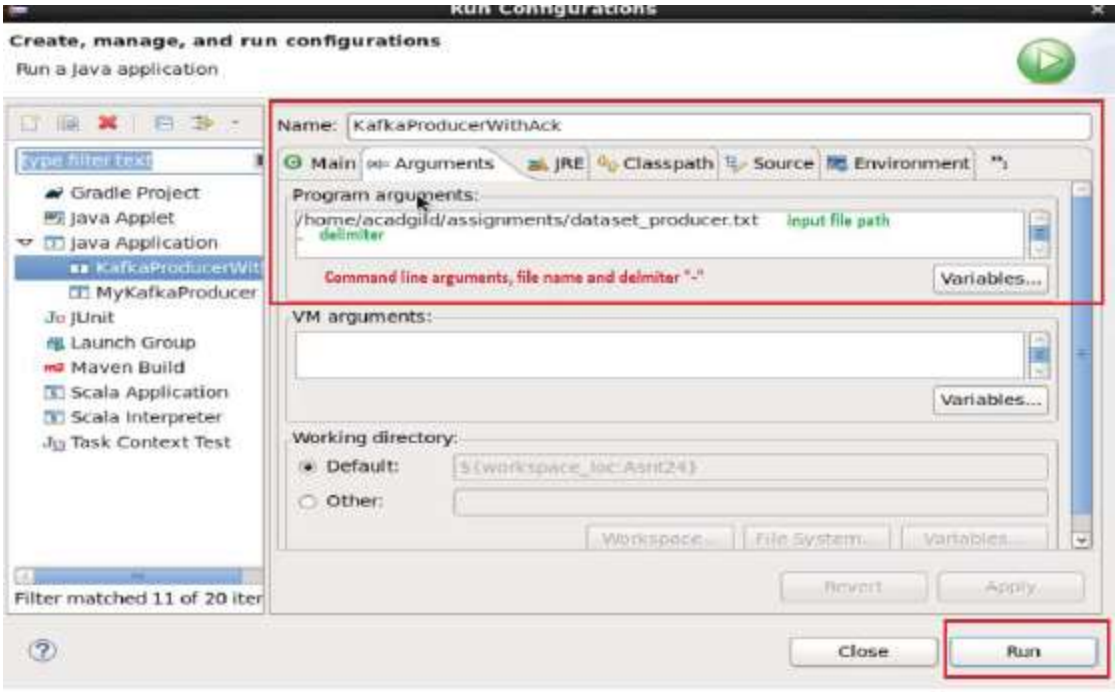
- Now, we pass the variables topic,key and value to producer record.
- We also print appropriate message which shows the topics,key and value contents.
- We finally, close the producer

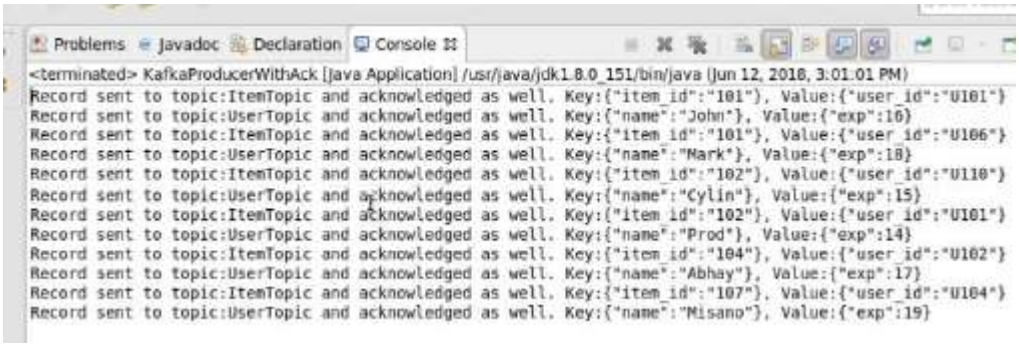
```
producerRecord = new ProducerRecord<String, String>(topic, key, value);
producer.send(producerRecord).get();
System.out.printf("Record sent to topic:%s and acknowledged as well. Key:%s,
Value:%s\n", topic, key, value);
} } producer.close(); } }
```

- We start zookeeper and kafka server using the below commands:

```
o ## starting zookeeper
bin/zookeeper-server-start.sh config/zookeeper.properties
o ## starting kafka server
bin/kafka-server-start.sh config/server.properties
```

- Now, we run this program in eclipse, by giving the arguments in “Run Configurations” as shown below:





· Now, we run the console consumer commands on terminal to view the output of the program, using the below command:

o To read contents of ItemTopic

```
./bin/kafka-console-consumer.sh --topic ItemTopic --from-beginning \
--zookeeper localhost:2181 --property print.key=true
```

```
{"item_id":"101"}	{"user_id":"U101"}
{"item_id":"101"}	{"user_id":"U106"}
{"item_id":"102"}	{"user_id":"U110"}
{"item_id":"102"}	{"user_id":"U101"}
{"item_id":"104"}	{"user_id":"U102"}
{"item_id":"107"}	{"user_id":"U104"}
{"item_id":"101"}	{"user_id":"U101"}
{"item_id":"101"}	{"user_id":"U106"}
{"item_id":"102"}	{"user_id":"U110"}
{"item_id":"102"}	{"user_id":"U101"}
{"item_id":"104"}	{"user_id":"U102"}
{"item_id":"107"}	{"user_id":"U104"}
{"item_id":"101"}	{"user_id":"U101"}
{"item_id":"101"}	{"user_id":"U106"}
{"item_id":"102"}	{"user_id":"U110"}
{"item_id":"102"}	{"user_id":"U101"}
{"item_id":"104"}	{"user_id":"U102"}
{"item_id":"107"}	{"user_id":"U104"}
{"item_id":"101"}	{"user_id":"U101"}
{"item_id":"101"}	{"user_id":"U106"}
{"item_id":"102"}	{"user_id":"U110"}
{"item_id":"102"}	{"user_id":"U101"}
{"item_id":"104"}	{"user_id":"U102"}
{"item_id":"107"}	{"user_id":"U104"}
{"item_id":"101"}	{"user_id":"U101"}
{"item_id":"101"}	{"user_id":"U106"}
{"item_id":"102"}	{"user_id":"U110"}
{"item_id":"102"}	{"user_id":"U101"}
{"item_id":"104"}	{"user_id":"U102"}
{"item_id":"107"}	{"user_id":"U104"}
```

o To read contents of UserTopic

./bin/kafka-console-consumer.sh --topic UserTopic --from-beginning \
--zookeeper localhost:2181 \
--property print.key=true

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic UserTopic --from-beginning \
> --zookeeper localhost:2181 \
> --property print.key=true
```

```
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Cylin"}	{"exp":15}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}	{"exp":17}
{"name":"Misano"}	{"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Cylin"}	{"exp":15}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}	{"exp":17}
{"name":"Misano"}	{"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Cylin"}	{"exp":15}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}	{"exp":17}
{"name":"Misano"}	{"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
```