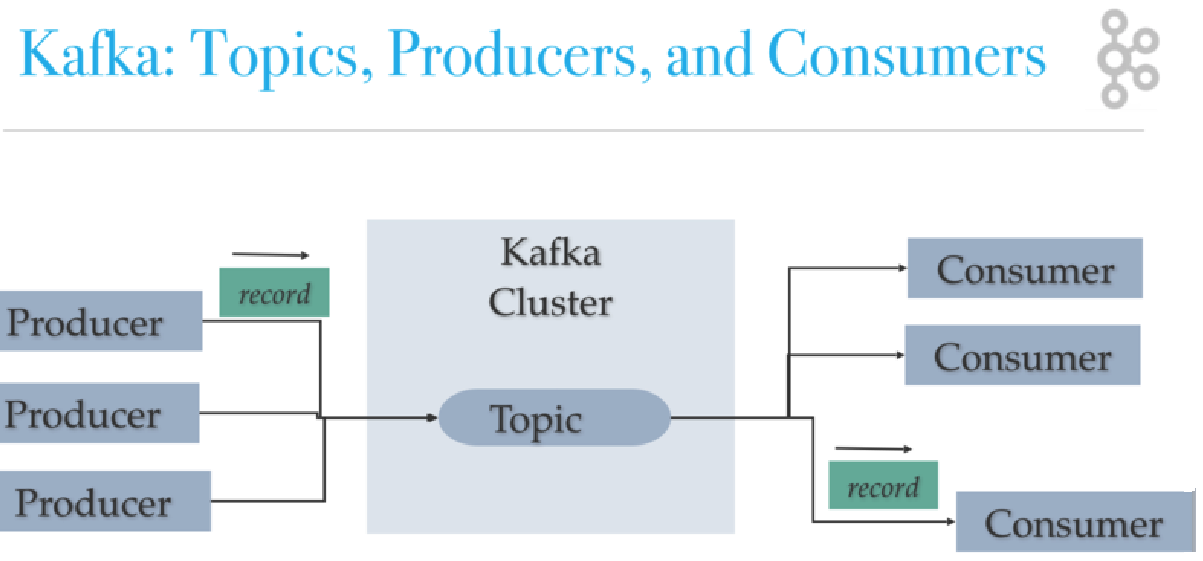
Session 24: More Kafka Assignment 1

- Prachi Mohite

Some Brief about Kafka

Kafka consists of records, topics, consumers, producers, brokers, logs, partitions, and clusters. Records can have keys (optional), values, and timestamps. Kafka records are immutable. A Kafka Topic is a stream of records ("/orders", "/user-signups"). You can think of a topic as a feed name.

The Kafka Producer API is used to produce streams of data records. The Kafka Consumer API is used to consume a stream of records from Kafka. A broker is a Kafka server that runs in a Kafka cluster. Kafka brokers form a cluster. The Kafka cluster consists of many Kafka brokers on many servers. Broker sometimes refer to more of a logical system or as Kafka as a whole.



Kafka Producers

Kafka producers send records to topics. The records are sometimes referred to as messages. The producer picks which partition to send a record to per topic. The producer can send records round-robin. The producer could implement priority systems based on sending records to certain partitions based on the priority of the record.

Generally speaking, producers send records to a partition based on the record's key. The default partitioner for Java uses a hash of the record's key to choose the partition or uses a round-robin strategy if the record has no key.
The important concept here is that the producer picks partition.

Kafka Consumer Groups

You group consumers into a consumer group by use case or function of the group. One consumer group might be responsible for delivering records to high-speed, in-memory microservices while another consumer group is streaming those same records to Hadoop. Consumer groups have names to identify them from other consumer groups.

A consumer group has a unique id. Each consumer group is a subscriber to one or more Kafka topics. Each consumer group maintains its offset per topic partition. If you need multiple subscribers, then you have multiple consumer groups. A record gets delivered to only one consumer in a consumer group.

Each consumer in a consumer group processes records and only one consumer in that group will get the same record. Consumers in a consumer group load balance record processing.

KafkaProducer API

- KafkaProducer class provides send method to send messages asynchronously to a topic. The signature of send() is as follows

```
producer.send(new ProducerRecord<byte[],byte[]>(topic,
partition, key1, value1) , callback);
```

- ProducerRecord − The producer manages a buffer of records waiting to be sent.
- Callback − A user-supplied callback to execute when the record has been acknowledged by the server (null indicates no callback).
- KafkaProducer class provides a flush method to ensure all previously sent messages have been actually completed. Syntax of the flush method is as follows −

```
public void flush()
```

- KafkaProducer class provides partitionFor method, which helps in getting the partition metadata for a given topic. This can be used for custom partitioning. The signature of this method is as follows −

```
public Map metrics()
```

It returns the map of internal metrics maintained by the producer.

- public void close() − KafkaProducer class provides close method blocks until all previously sent requests are completed.

Producer API

The central part of the Producer API is Producer class. Producer class provides an option to connect Kafka broker in its constructor by the following methods.

The Producer Class

The producer class provides send method to send messages to either single or multiple topics using the following signatures.

```
public void send(KeyedMessaget<k,v> message)

- sends the data to a single topic,par-titioned by key using either sync or async producer.

public void send(List<KeyedMessage<k,v>>messages)

- sends data to multiple topics.

Properties prop = new Properties();

prop.put(producer.type,"async")

ProducerConfig config = new ProducerConfig(prop);
```

public void close()

Producer class provides close method to close the producer pool connections to all Kafka bro-kers.

ProducerRecord API

ProducerRecord is a key/value pair that is sent to Kafka cluster.ProducerRecord class constructor for creating a record with partition, key and value pairs using the following signature.

```
public ProducerRecord (string topic, int partition, k key, v value)
```

- Topic − user defined topic name that will appended to record.
- Partition − partition count
- Key − The key that will be included in the record.
- Value − Record contents

public ProducerRecord (string topic, k key, v value)

ProducerRecord class constructor is used to create a record with key, value pairs and without partition.

- Topic − Create a topic to assign record.
- Key − key for the record.
- Value − record contents.

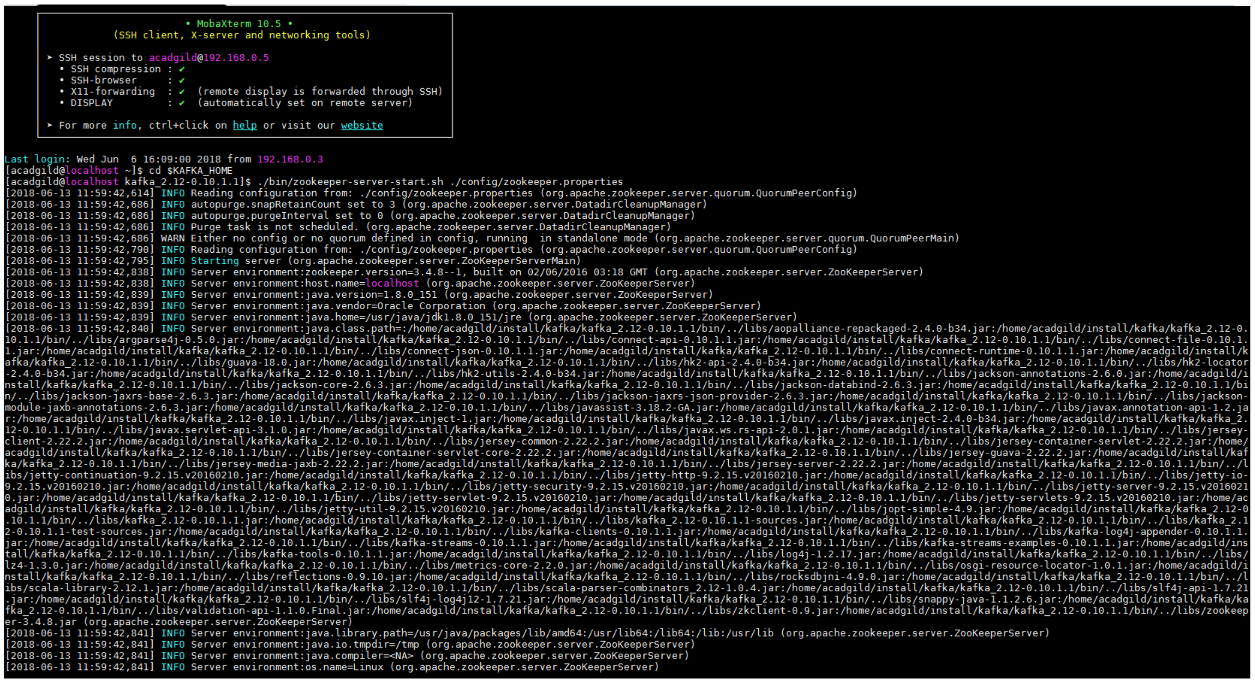
public ProducerRecord (string topic, v value)

ProducerRecord class creates a record without partition and key.

- Topic − create a topic.
- Value − record contents.

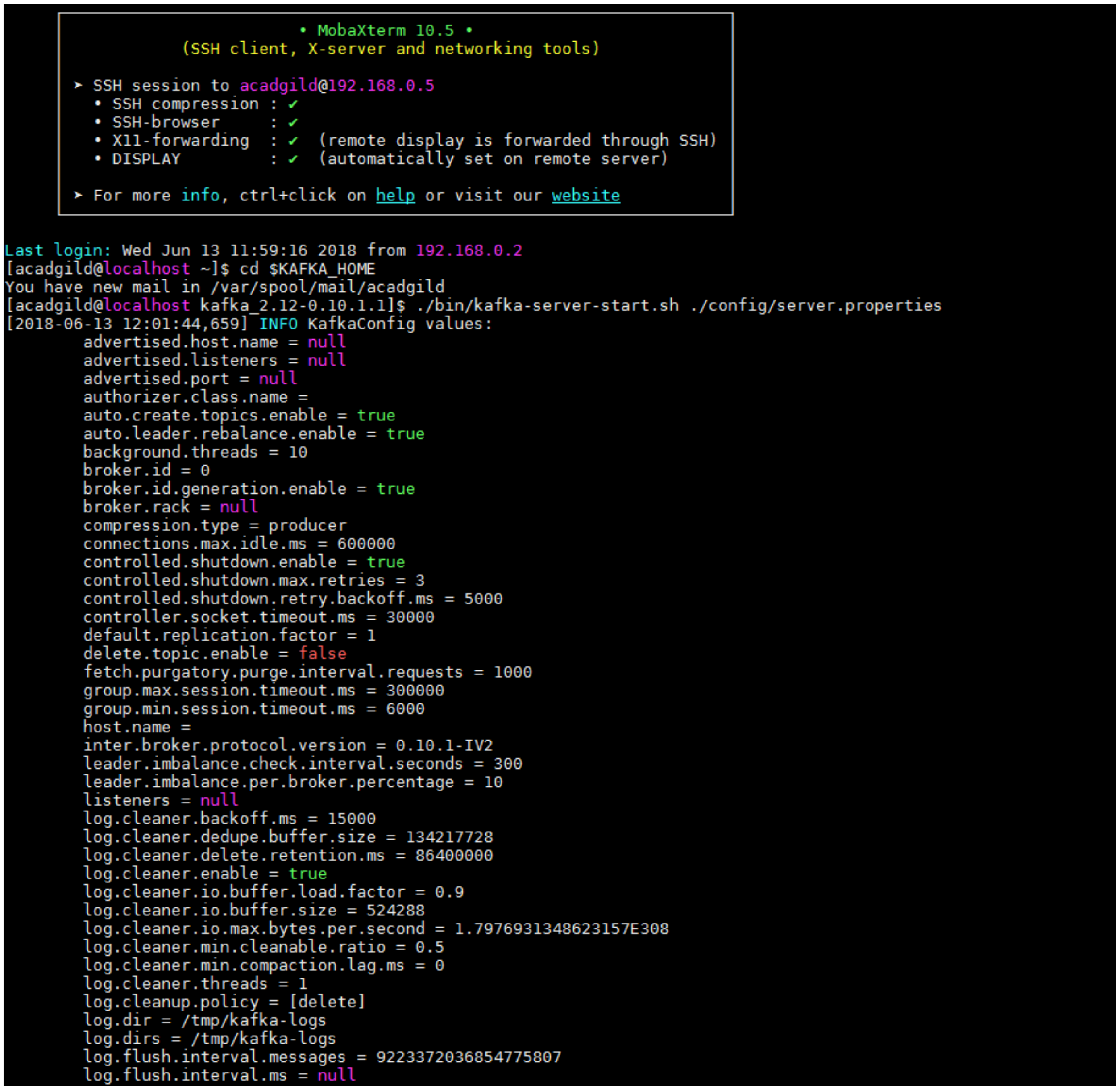
Pre-requisite

1. Start the ZooKeeper with below command
   Commands
   cd $KAFKA_HOME ($KAFKA_HOME = /home/acadgild/install/kafka/kafka_2.12-0.10.1.1/)
   ./bin/zookeeper-server-start.sh ./config/zookeeper.properties



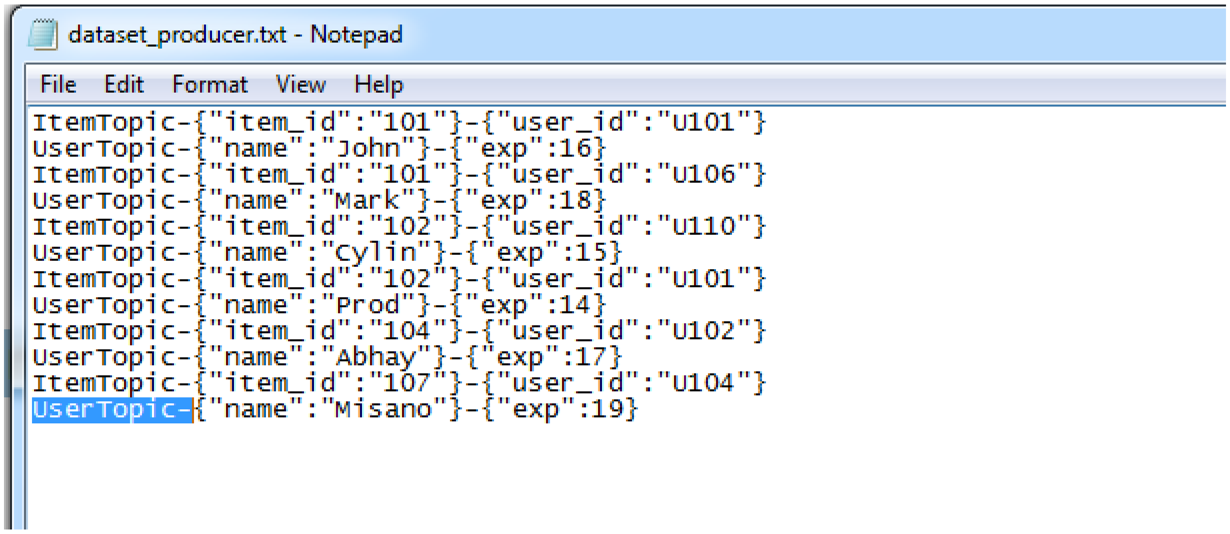
```
                    • MobaXterm 10.5 •
          (SSH client, X-server and networking tools)

  ➤ SSH session to acadgild@192.168.0.5
    • SSH compression : ✔
    • SSH-browser     : ✔
    • X11-forwarding  : ✔  (remote display is forwarded through SSH)
    • DISPLAY         : ✔  (automatically set on remote server)

  ➤ For more info, ctrl+click on help or visit our website

Last login: Wed Jun  6 16:09:00 2018 from 192.168.0.3
[acadgild@localhost ~]$ cd $KAFKA_HOME
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/zookeeper-server-start.sh ./config/zookeeper.properties
[2018-06-13 11:59:42,614] INFO Reading configuration from: ./config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2018-06-13 11:59:42,686] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2018-06-13 11:59:42,686] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2018-06-13 11:59:42,686] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2018-06-13 11:59:42,686] WARN Either no config or no quorum defined in config, running  in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2018-06-13 11:59:42,790] INFO Reading configuration from: ./config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2018-06-13 11:59:42,795] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2018-06-13 11:59:42,838] INFO Server environment:zookeeper.version=3.4.8--1, built on 02/06/2016 03:18 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,838] INFO Server environment:host.name=localhost (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,839] INFO Server environment:java.version=1.8.0_151 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,839] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,839] INFO Server environment:java.home=/usr/java/jdk1.8.0_151/jre (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,840] INFO Server environment:java.class.path=:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/aopalliance-repackaged-2.4.0-b34.jar:/home/acadgild/install/kafka/kafka_2.12-0.
10.1.1/bin/../libs/argparse4j-0.5.0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/connect-api-0.10.1.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/connect-file-0.10.1.
1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/connect-json-0.10.1.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/connect-runtime-0.10.1.1.jar:/home/acadgild/install/k
afka/kafka_2.12-0.10.1.1/bin/../libs/guava-18.0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/hk2-api-2.4.0-b34.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/hk2-locator
-2.4.0-b34.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/hk2-utils-2.4.0-b34.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jackson-annotations-2.6.0.jar:/home/acadgild/i
nstall/kafka/kafka_2.12-0.10.1.1/bin/../libs/jackson-core-2.6.3.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jackson-databind-2.6.3.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bi
n/../libs/jackson-jaxrs-base-2.6.3.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jackson-jaxrs-json-provider-2.6.3.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jackson-
module-jaxb-annotations-2.6.3.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/javassist-3.18.2-GA.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/javax.annotation-api-1.2.ja
r:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/javax.inject-1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/javax.inject-2.4.0-b34.jar:/home/acadgild/install/kafka/kafka_2.
12-0.10.1.1/bin/../libs/javax.servlet-api-3.1.0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/javax.ws.rs-api-2.0.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jersey-
client-2.22.2.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jersey-common-2.22.2.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jersey-container-servlet-2.22.2.jar:/home/
acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jersey-container-servlet-core-2.22.2.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jersey-guava-2.22.2.jar:/home/acadgild/install/kaf
ka/kafka_2.12-0.10.1.1/bin/../libs/jersey-media-jaxb-2.22.2.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jersey-server-2.22.2.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../l
ibs/jetty-continuation-9.2.15.v20160210.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-http-9.2.15.v20160210.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-io-
9.2.15.v20160210.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-security-9.2.15.v20160210.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-server-9.2.15.v2016021
0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-servlet-9.2.15.v20160210.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-servlets-9.2.15.v20160210.jar:/home/ac
adgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jetty-util-9.2.15.v20160210.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/jopt-simple-4.9.jar:/home/acadgild/install/kafka/kafka_2.12-0
.10.1.1/bin/../libs/kafka_2.12-0.10.1.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka_2.12-0.10.1.1-sources.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka_2.1
2-0.10.1.1-test-sources.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka-clients-0.10.1.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka-log4j-appender-0.10.1.1.
jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka-streams-0.10.1.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka-streams-examples-0.10.1.1.jar:/home/acadgild/ins
tall/kafka/kafka_2.12-0.10.1.1/bin/../libs/kafka-tools-0.10.1.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/log4j-1.2.17.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/
lz4-1.3.0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/metrics-core-2.2.0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/osgi-resource-locator-1.0.1.jar:/home/acadgild/i
nstall/kafka/kafka_2.12-0.10.1.1/bin/../libs/reflections-0.9.10.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/rocksdbjni-4.9.0.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../l
ibs/scala-library-2.12.1.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/scala-parser-combinators_2.12-1.0.4.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/slf4j-api-1.7.21
.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/slf4j-log4j12-1.7.21.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/snappy-java-1.1.2.6.jar:/home/acadgild/install/kafka/ka
fka_2.12-0.10.1.1/bin/../libs/validation-api-1.1.0.Final.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/zkclient-0.9.jar:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/bin/../libs/zookeep
er-3.4.8.jar (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,841] INFO Server environment:java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,841] INFO Server environment:java.io.tmpdir=/tmp (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,841] INFO Server environment:java.compiler=<NA> (org.apache.zookeeper.server.ZooKeeperServer)
[2018-06-13 11:59:42,841] INFO Server environment:os.name=Linux (org.apache.zookeeper.server.ZooKeeperServer)
```

2. Start the Kafka Broker
   Commands
   cd $KAFKA_HOME
   ./bin/kafka-server-start.sh ./config/server.properties



```
                      • MobaXterm 10.5 •
            (SSH client, X-server and networking tools)

  ➤ SSH session to acadgild@192.168.0.5
    • SSH compression : ✔
    • SSH-browser     : ✔
    • X11-forwarding  : ✔  (remote display is forwarded through SSH)
    • DISPLAY         : ✔  (automatically set on remote server)

  ➤ For more info, ctrl+click on help or visit our website

Last login: Wed Jun 13 11:59:16 2018 from 192.168.0.2
[acadgild@localhost ~]$ cd $KAFKA_HOME
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-server-start.sh ./config/server.properties
[2018-06-13 12:01:44,659] INFO KafkaConfig values:
        advertised.host.name = null
        advertised.listeners = null
        advertised.port = null
        authorizer.class.name =
        auto.create.topics.enable = true
        auto.leader.rebalance.enable = true
        background.threads = 10
        broker.id = 0
        broker.id.generation.enable = true
        broker.rack = null
        compression.type = producer
        connections.max.idle.ms = 600000
        controlled.shutdown.enable = true
        controlled.shutdown.max.retries = 3
        controlled.shutdown.retry.backoff.ms = 5000
        controller.socket.timeout.ms = 30000
        default.replication.factor = 1
        delete.topic.enable = false
        fetch.purgatory.purge.interval.requests = 1000
        group.max.session.timeout.ms = 300000
        group.min.session.timeout.ms = 6000
        host.name =
        inter.broker.protocol.version = 0.10.1-IV2
        leader.imbalance.check.interval.seconds = 300
        leader.imbalance.per.broker.percentage = 10
        listeners = null
        log.cleaner.backoff.ms = 15000
        log.cleaner.dedupe.buffer.size = 134217728
        log.cleaner.delete.retention.ms = 86400000
        log.cleaner.enable = true
        log.cleaner.io.buffer.load.factor = 0.9
        log.cleaner.io.buffer.size = 524288
        log.cleaner.io.max.bytes.per.second = 1.7976931348623157E308
        log.cleaner.min.cleanable.ratio = 0.5
        log.cleaner.min.compaction.lag.ms = 0
        log.cleaner.threads = 1
        log.cleanup.policy = [delete]
        log.dir = /tmp/kafka-logs
        log.dirs = /tmp/kafka-logs
        log.flush.interval.messages = 9223372036854775807
        log.flush.interval.ms = null
```

Dataset Used for this assignment is as below


dataset_producer.txt - Notepad

File Edit Format View Help

```
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}
```

Topics Creation from Command Line

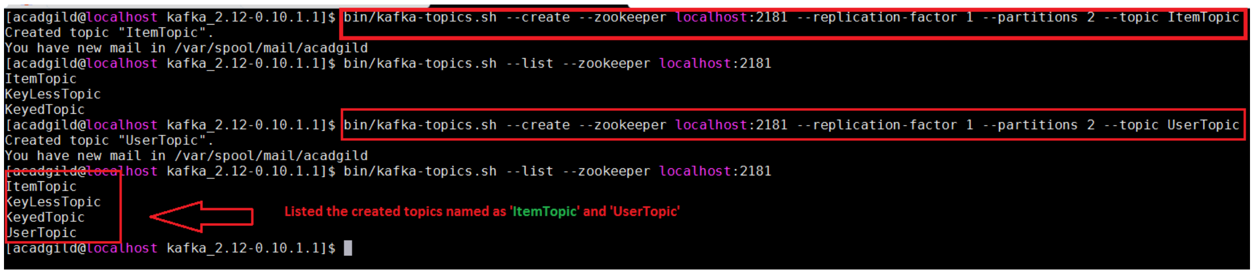
It has two topics. These topics needs to be created from command line as below

- Created topic named as ***ItemTopic***

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

- Created topic named as ***UserTopic***

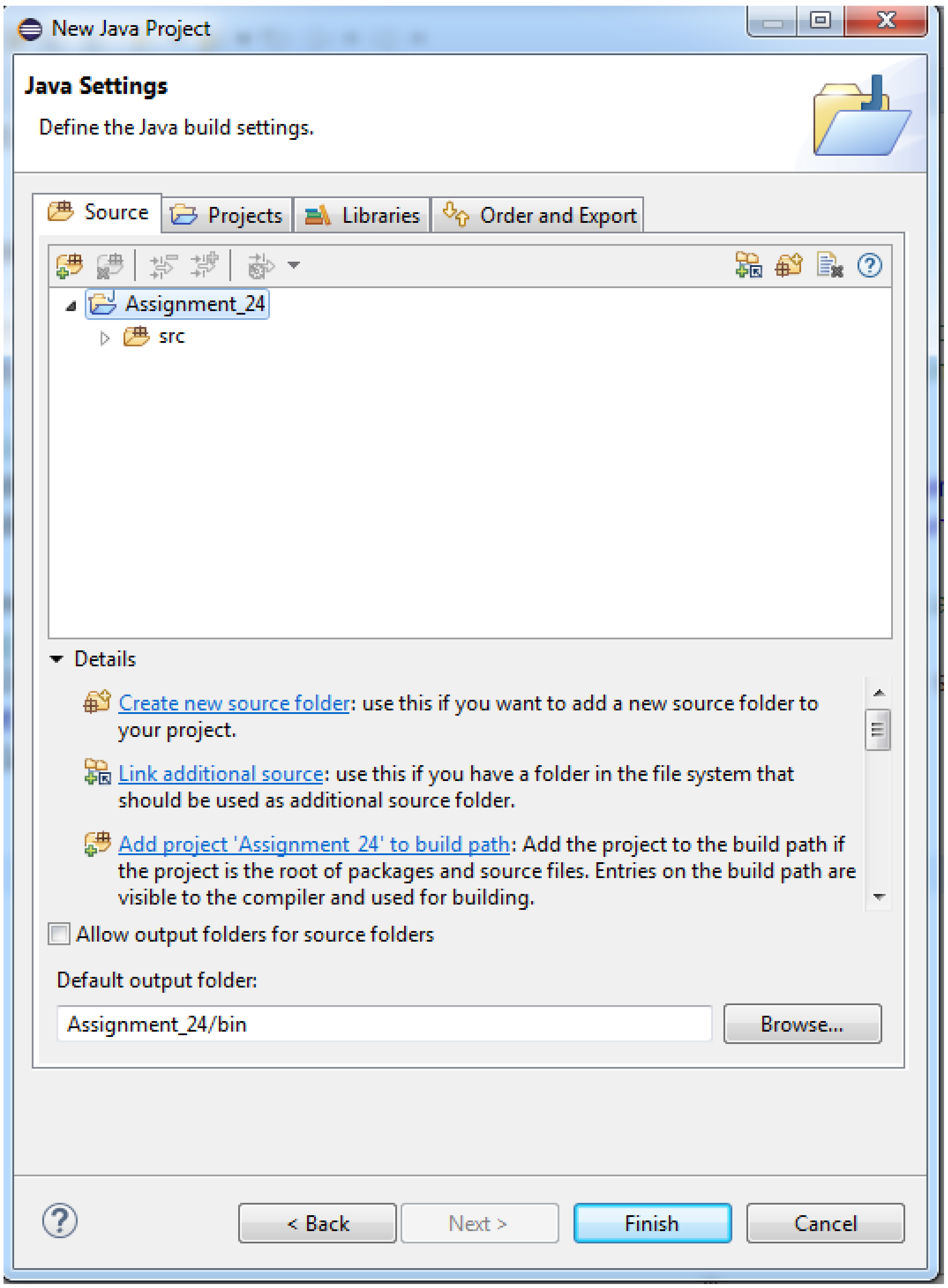
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 2 --topic ***UserTopic***



```
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 2 --topic ItemTopic
Created topic "ItemTopic".
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --list --zookeeper localhost:2181
ItemTopic
KeyLessTopic
KeyedTopic
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 2 --topic UserTopic
Created topic "UserTopic".
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --list --zookeeper localhost:2181
ItemTopic
KeyLessTopic
KeyedTopic
UserTopic
[acadgild@localhost kafka_2.12-0.10.1.1]$
```

Listed the created topics named as 'ItemTopic' and 'UserTopic'

Created project in Java


New Java Project

Java Settings

Define the Java build settings.

Assignment_24
src

Details

Create new source folder: use this if you want to add a new source folder to your project.

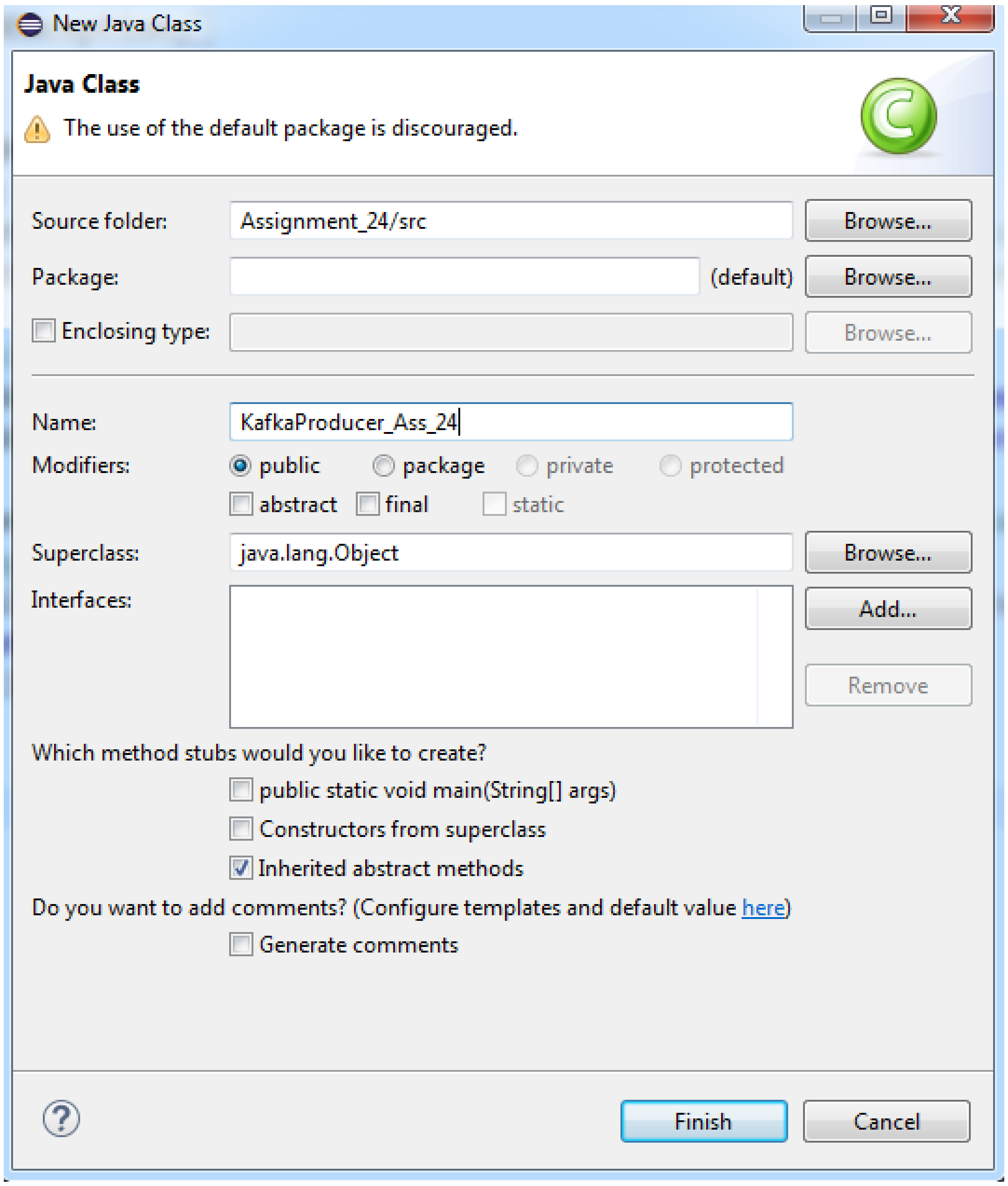
Link additional source: use this if you have a folder in the file system that should be used as additional source folder.

Add project 'Assignment_24' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

[ ] Allow output folders for source folders

Default output folder: Assignment_24/bin

Add a class to have kafka producer API


New Java Class

Java Class

The use of the default package is discouraged.

Source folder: Assignment_24/src

Package: (default)

[ ] Enclosing type:

Name: KafkaProducer_Ass_24

Modifiers: [● public] ○ package ○ private ○ protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[ ] public static void main(String[] args)

[ ] Constructors from superclass

[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

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.

Solution Approach

After creating the Java class imported required kafka jars

```
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;
```

As per requirement need to check if two arguments are passed

```
//We must pass file name and delimiter as input while
execution
        if (args.length != 2) {
            System.out.println("Please provide command line
arguments as file name as delimiter");
            System.exit(-1);
        }
```

Set all the properties for kafka producer

```
//Configuring the properties for Kafka Producers
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("acks", "all");
```

- 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

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

- 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

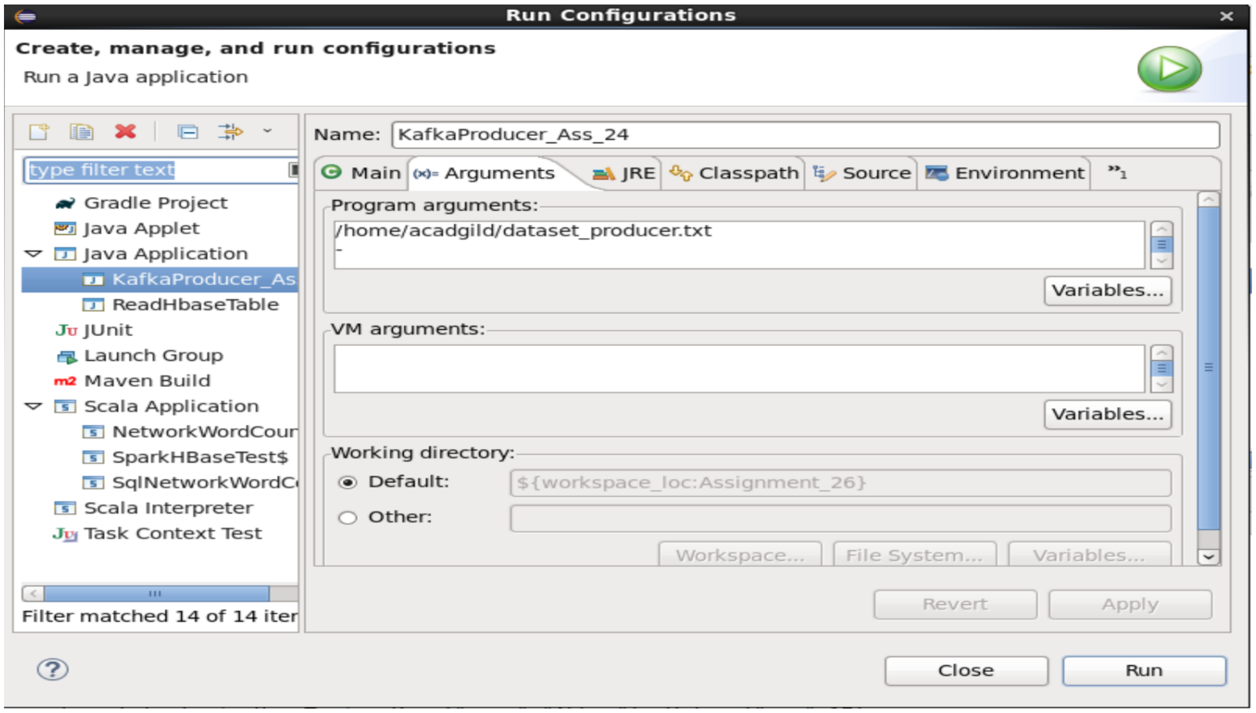
```
//Read the file (file name is passed through command line 
argument with delimiter)
        String fileName = args[0];
        String delimiter = args[1];
```

- 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.
- 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 inally, close the producer.

```
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();
            }
```

- Run this program in eclipse, by giving the arguments in "Run Configurations" as shown below:


Run Configurations

Create, manage, and run configurations

Run a Java application

Name: KafkaProducer_Ass_24

Program arguments:

```
/home/acadgild/dataset_producer.txt
-
```

VM arguments:

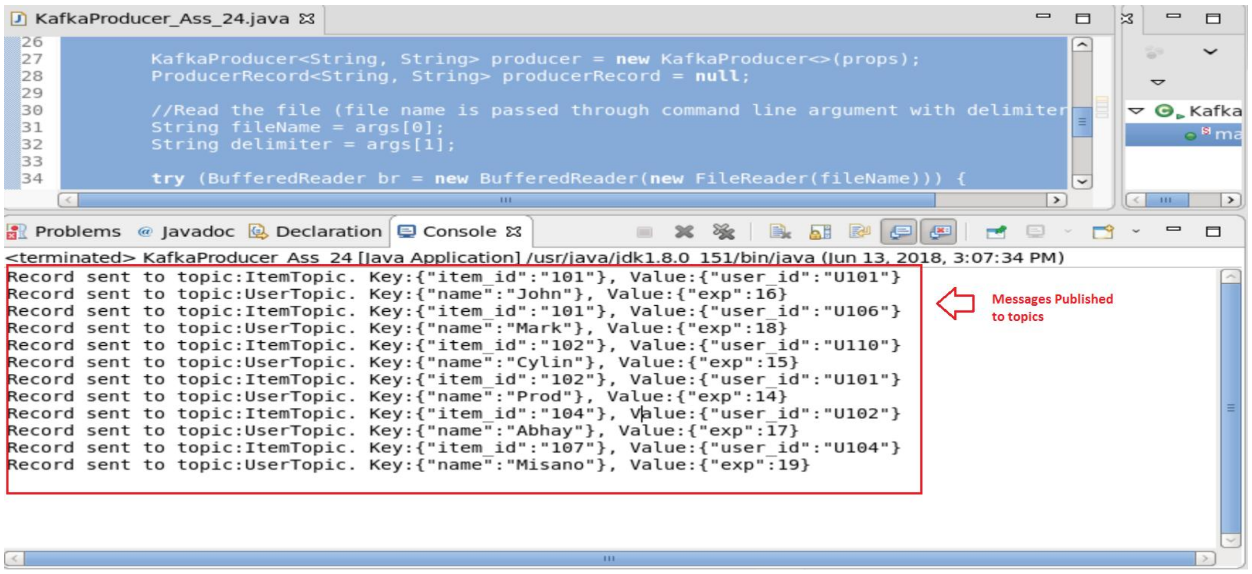
Working directory:
Default: ${workspace_loc:Assignment_26}

Complete Code

```
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;

public class KafkaProducer_Ass_24 {
    public static void main(String[] args) throws
IOException {
        //We must pass file name and delimiter as input
while execution
        if (args.length != 2) {
            System.out.println("Please provide command line
arguments as file name as delimiter");
            System.exit(-1);
        }


        //Configuring the properties for Kafka Producers
        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;

        //Read the file (file name is passed through command
line argument with delimiter)
        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();
      }


}
```

Output

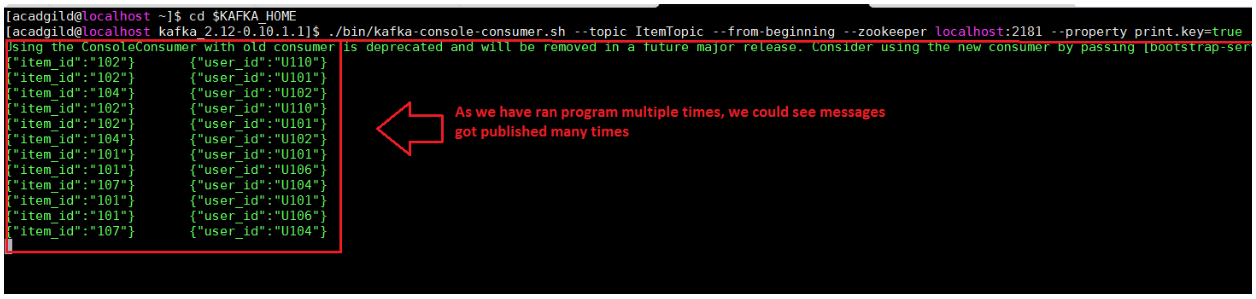


```
KafkaProducer<String, String> producer = new KafkaProducer<>(props);
ProducerRecord<String, String> producerRecord = null;

//Read the file (file name is passed through command line argument with delimiter
String fileName = args[0];
String delimiter = args[1];

try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
```

```
<terminated> KafkaProducer_Ass_24 [Java Application] /usr/java/jdk1.8.0_151/bin/java (Jun 13, 2018, 3:07:34 PM)
Record sent to topic:ItemTopic. Key:{"item_id":"101"}, Value:{"user_id":"U101"}
Record sent to topic:UserTopic. Key:{"name":"John"}, Value:{"exp":16}
Record sent to topic:ItemTopic. Key:{"item_id":"101"}, Value:{"user_id":"U106"}
Record sent to topic:UserTopic. Key:{"name":"Mark"}, Value:{"exp":18}
Record sent to topic:ItemTopic. Key:{"item_id":"102"}, Value:{"user_id":"U110"}
Record sent to topic:UserTopic. Key:{"name":"Cylin"}, Value:{"exp":15}
Record sent to topic:ItemTopic. Key:{"item_id":"102"}, Value:{"user_id":"U101"}
Record sent to topic:UserTopic. Key:{"name":"Prod"}, Value:{"exp":14}
Record sent to topic:ItemTopic. Key:{"item_id":"104"}, Value:{"user_id":"U102"}
Record sent to topic:UserTopic. Key:{"name":"Abhay"}, Value:{"exp":17}
Record sent to topic:ItemTopic. Key:{"item_id":"107"}, Value:{"user_id":"U104"}
Record sent to topic:UserTopic. Key:{"name":"Misano"}, Value:{"exp":19}
```

Reading the topics from kafka Console Consumers

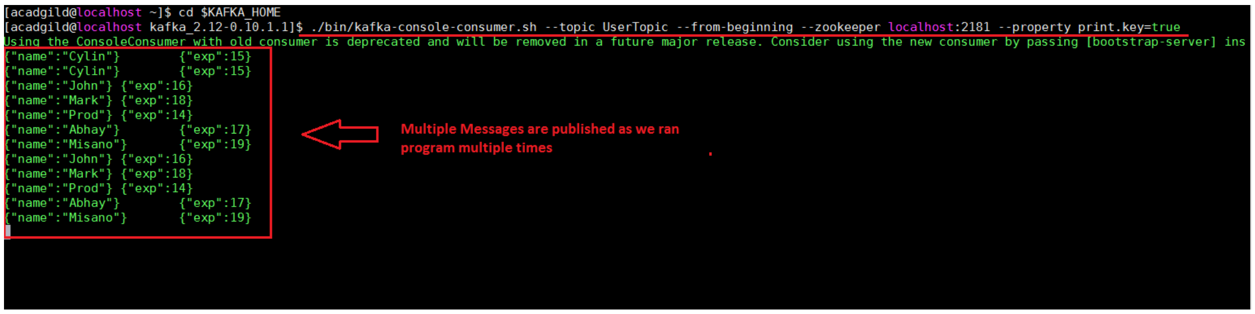
./bin/kafka-console-consumer.sh --topic ItemTopic --from-beginning --zookeeper localhost:2181 --property print.key=true



```
[acadgild@localhost ~]$ cd $KAFKA_HOME
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic ItemTopic --from-beginning --zookeeper localhost:2181 --property print.key=true
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-ser
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"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"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
```

As we have ran program multiple times, we could see messages got published many times

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



```
[acadgild@localhost ~]$ cd $KAFKA_HOME
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic UserTopic --from-beginning --zookeeper localhost:2181 --property print.key=true
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] ins
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
```

Multiple Messages are published as we ran program multiple times

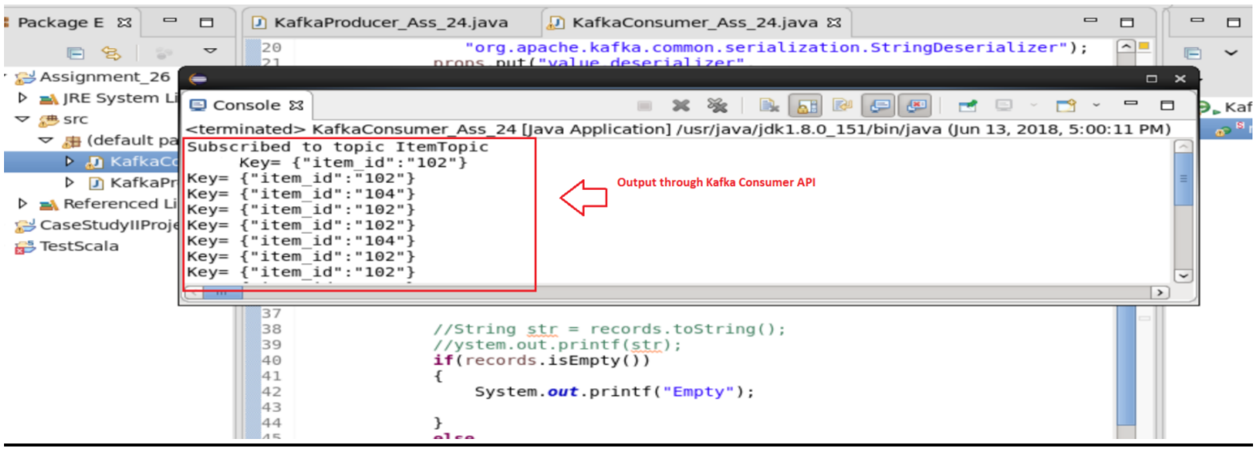
Reading Topics through Kafka consumers

Complete Code

```
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 KafkaConsumer_Ass_24 {
     public static void main(String[] args) throws Exception {
          //we have to read two topics which should be passed as
arguments
          if(args.length !=2){
              System.out.println("Enter 2 topic names");
              return;
           }
          String topicName1 = args[0].toString();
          String topicName2 = args[1].toString();
          Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("group.id","ItemTopic");
```

```
          props.put("session.timeut.ms", "300000");
          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(topicName1));
             //print the topic name
               System.out.println("Subscribed to topic " + topicName1);
               consumer.poll(0);
               consumer.seekToBeginning(consumer.assignment());
               while(true)
               {

               ConsumerRecords<String, String> records =
consumer.poll(1);

               //String str = records.toString();
               //ystem.out.printf(str);
               if(records.isEmpty())
               {
                  System.out.printf("Empty");

               }
               else
               {
                  for (ConsumerRecord<String, String> consumerRecord :
records) {
                  System.out.printf("Key= %s \n", consumerRecord.key(),
consumerRecord.value());
                  }
               }
            }



      }
}
```

Item Topic



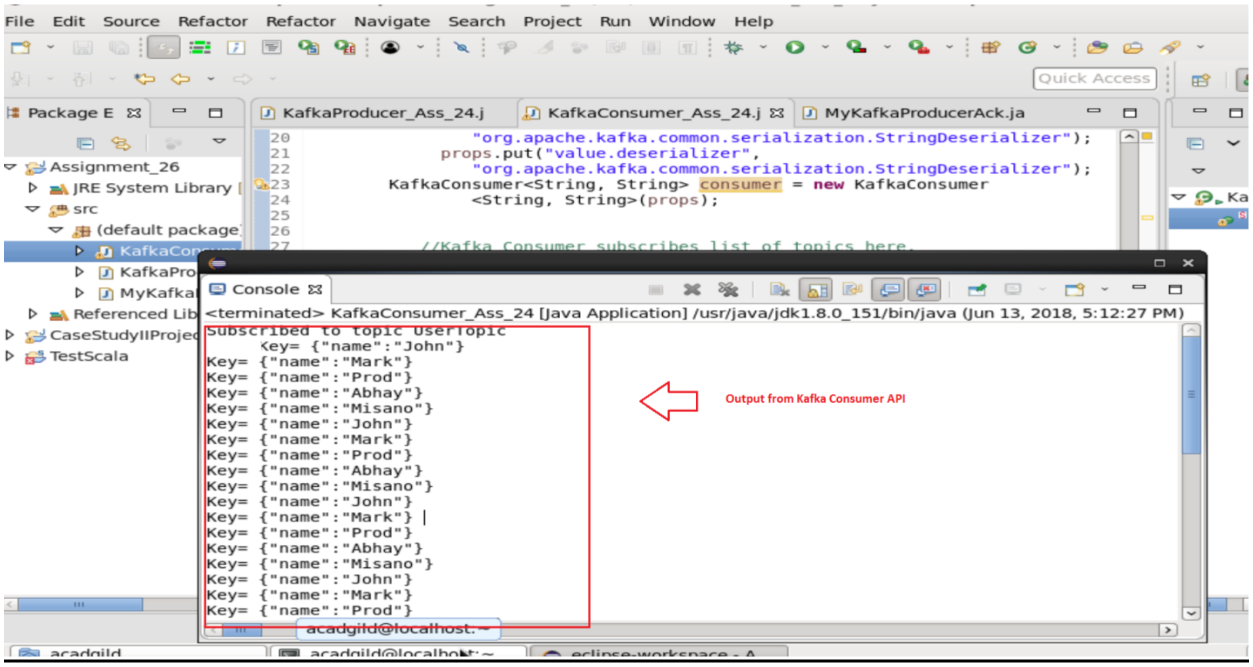
```
"org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer"
```

```
<terminated> KafkaConsumer_Ass_24 [Java Application] /usr/java/jdk1.8.0_151/bin/java (Jun 13, 2018, 5:00:11 PM)
Subscribed to topic ItemTopic
          Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
```

Output through Kafka Consumer API

```
        //String str = records.toString();
        //ystem.out.printf(str);
        if(records.isEmpty())
        {
            System.out.printf("Empty");

        }
        else
```

User Topic



```
        "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.
```

```
<terminated> KafkaConsumer_Ass_24 [Java Application] /usr/java/jdk1.8.0_151/bin/java (Jun 13, 2018, 5:12:27 PM)
Subscribed to topic UserTopic
          Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
Key= {"name":"Abhay"}
Key= {"name":"Misano"}
Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
Key= {"name":"Abhay"}
Key= {"name":"Misano"}
Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
Key= {"name":"Abhay"}
Key= {"name":"Misano"}
Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
```

Output from Kafka Consumer API

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.

Solution Approach –

The entire code will remain as above only we need to add two more properties for creating the kafka producer as below

We configure the properties for KafkaProducer:

- 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.

```
props.put("acks", "all");
props.put("retries", 3);
```

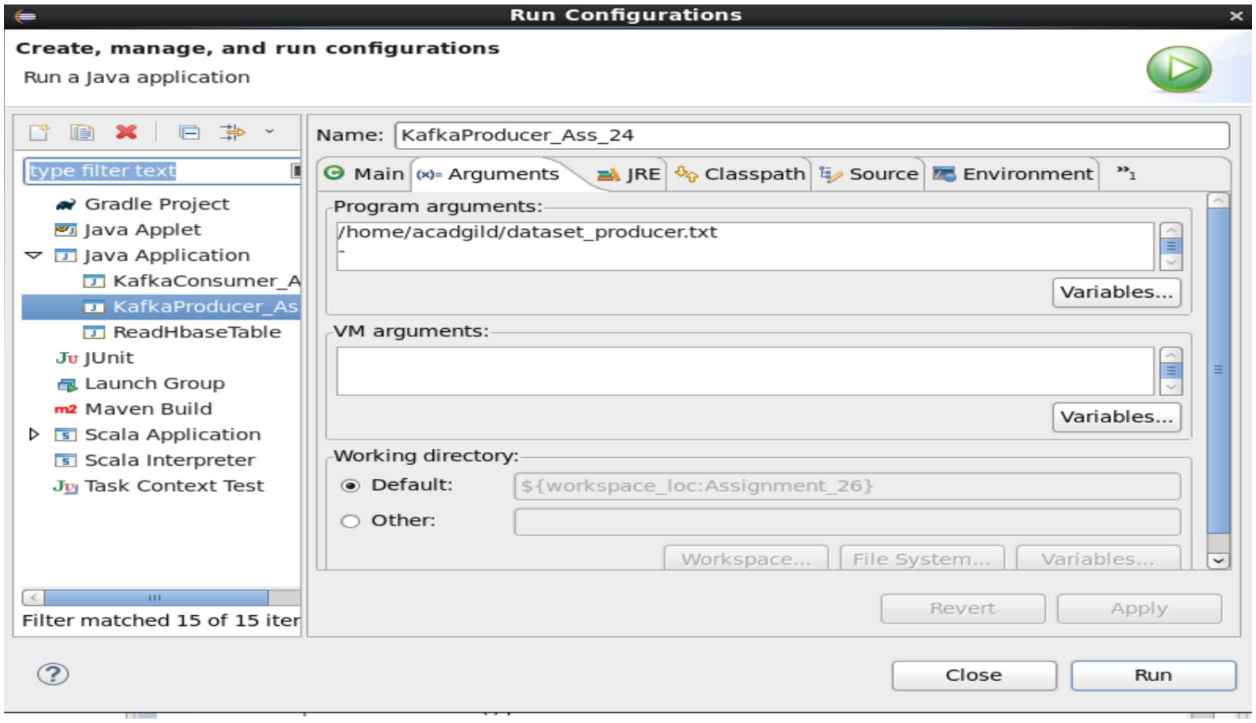
Complete Code

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import java.util.concurrent.ExecutionException;

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

public class MyKafkaProducerAck {
	public static void main(String[] args) throws IOException {
		//We must pass file name and delimiter as input while
execution
```

```
        if (args.length != 2) {
            System.out.println("Please provide command line arguments
as file name as delimiter");
            System.exit(-1);
        }


        //Configuring the properties for Kafka Producers
        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");


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

        //Read the file (file name is passed through command line
argument with delimiter)
        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).get();
                System.out.printf("Record sent to topic & Acknowledged
as well:%s. Key:%s, Value:%s\n", topic, key, value);
            }
        } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
           } catch (ExecutionException e) {
                // TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                }
            producer.close();
        }


}
```

Code Execution



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: KafkaProducer_Ass_24

Program arguments:

```
/home/acadgild/dataset_producer.txt
-
```

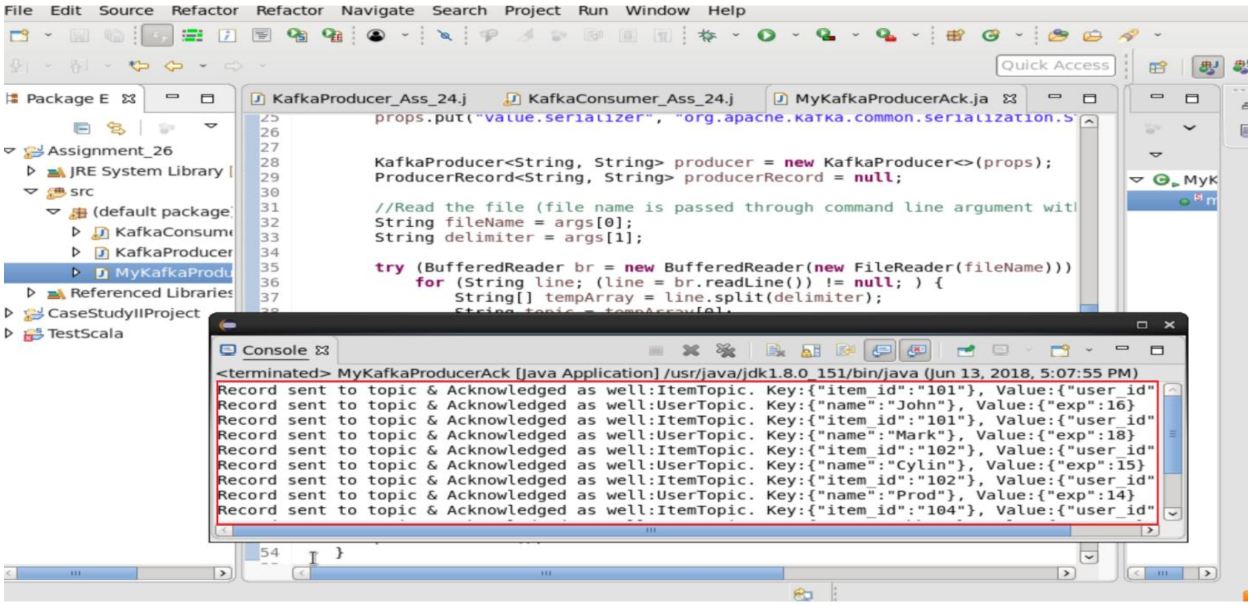
VM arguments:

Working directory:

Default: ${workspace_loc:Assignment_26}

Other:

Output

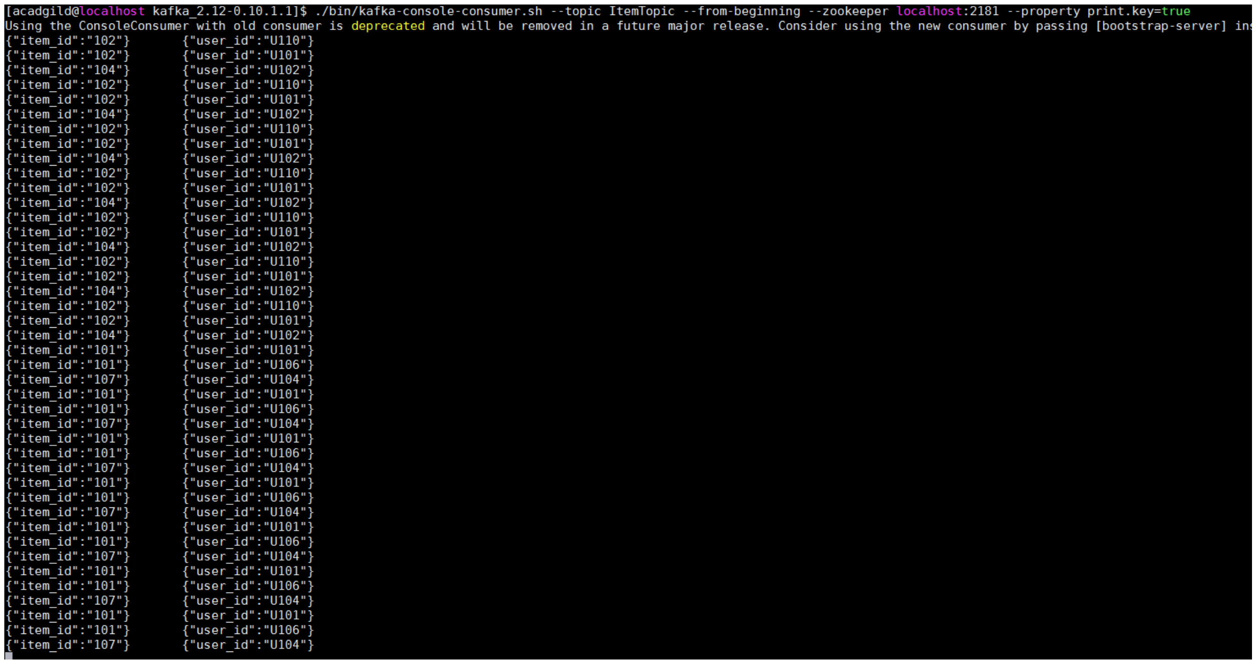


```
props.put("value.serializer", "org.apache.kafka.common.serialization.S

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

//Read the file (file name is passed through command line argument wit
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);
```

```
<terminated> MyKafkaProducerAck [Java Application] /usr/java/jdk1.8.0_151/bin/java (Jun 13, 2018, 5:07:55 PM)
Record sent to topic & Acknowledged as well:ItemTopic. Key:{"item_id":"101"}, Value:{"user_id"
Record sent to topic & Acknowledged as well:UserTopic. Key:{"name":"John"}, Value:{"exp":16}
Record sent to topic & Acknowledged as well:ItemTopic. Key:{"item_id":"101"}, Value:{"user_id"
Record sent to topic & Acknowledged as well:UserTopic. Key:{"name":"Mark"}, Value:{"exp":18}
Record sent to topic & Acknowledged as well:ItemTopic. Key:{"item_id":"102"}, Value:{"user_id"
Record sent to topic & Acknowledged as well:UserTopic. Key:{"name":"Cylin"}, Value:{"exp":15}
Record sent to topic & Acknowledged as well:ItemTopic. Key:{"item_id":"102"}, Value:{"user_id"
Record sent to topic & Acknowledged as well:UserTopic. Key:{"name":"Prod"}, Value:{"exp":14}
Record sent to topic & Acknowledged as well:ItemTopic. Key:{"item_id":"104"}, Value:{"user_id"
```

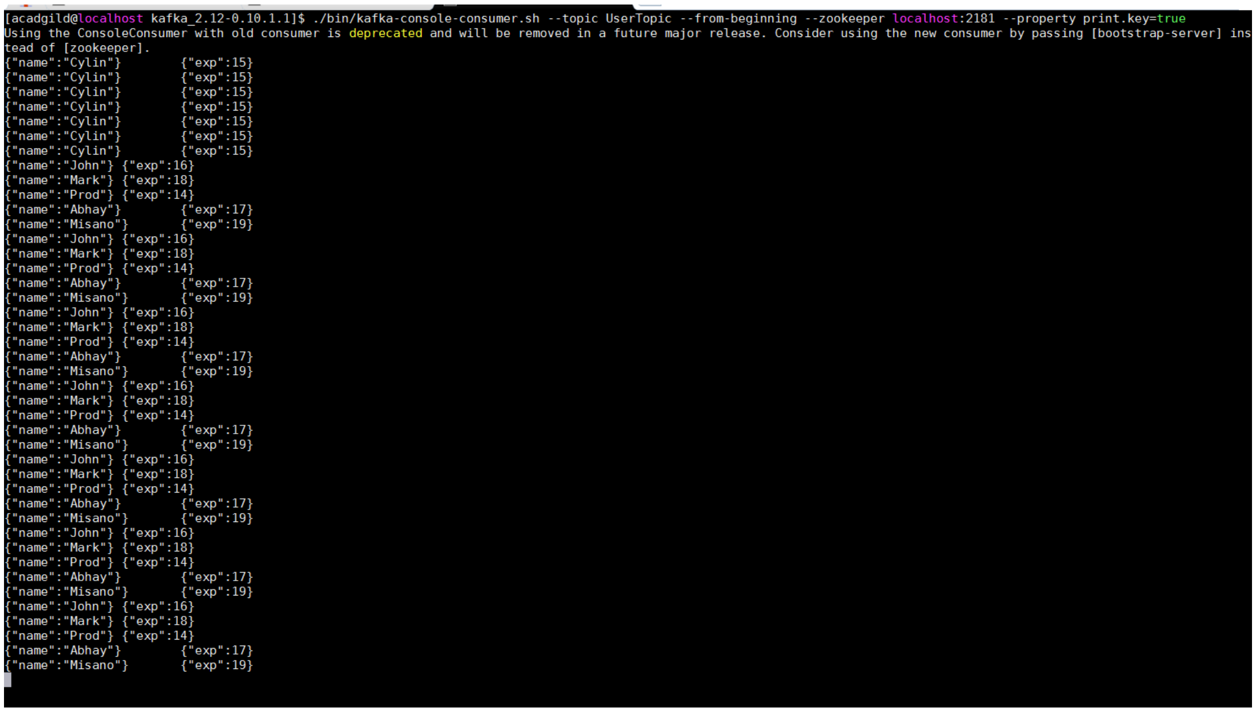
Reading the topics from kafka Console Consumers

./bin/kafka-console-consumer.sh --topic ItemTopic --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
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] ins
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"item_id":"102"}       {"user_id":"U110"}
{"item_id":"102"}       {"user_id":"U101"}
{"item_id":"104"}       {"user_id":"U102"}
{"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"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
{"item_id":"101"}       {"user_id":"U101"}
{"item_id":"101"}       {"user_id":"U106"}
{"item_id":"107"}       {"user_id":"U104"}
```

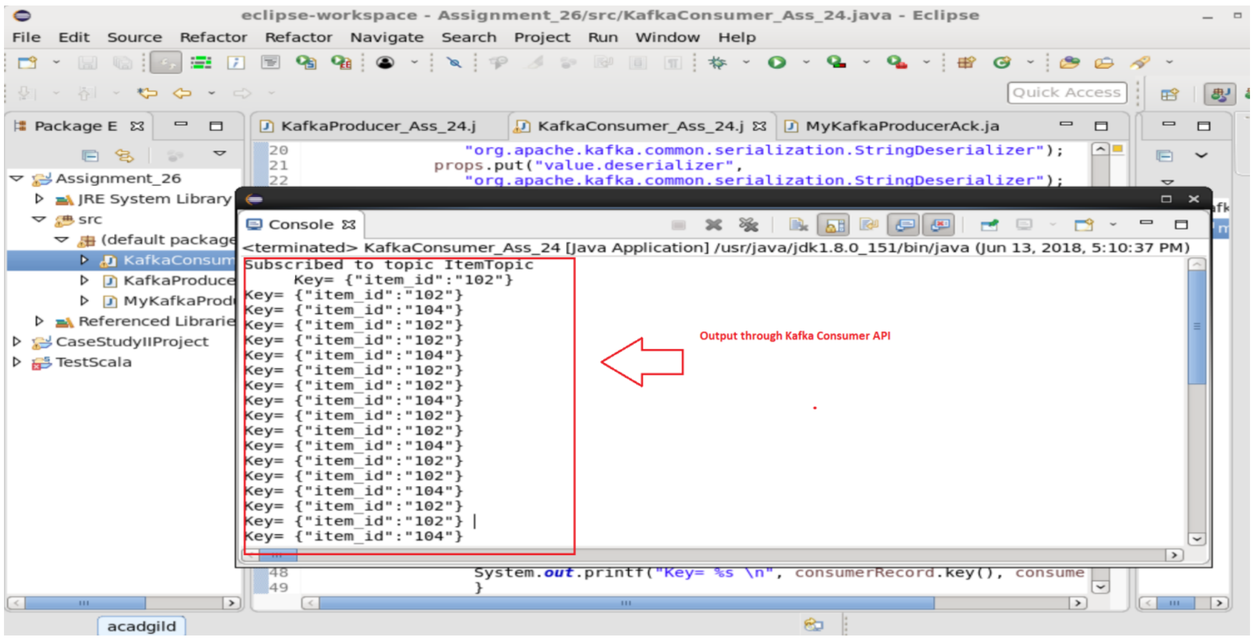
./bin/kafka-console-consumer.sh --topic UserTopic --from-beginning --zookeeper localhost:2181 --property print.key=true



```
[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
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] ins
tead of [zookeeper].
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"Cylin"}        {"exp":15}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
{"name":"John"} {"exp":16}
{"name":"Mark"} {"exp":18}
{"name":"Prod"} {"exp":14}
{"name":"Abhay"}        {"exp":17}
{"name":"Misano"}       {"exp":19}
```

Reading Topics through Kafka consumers

Item Topic

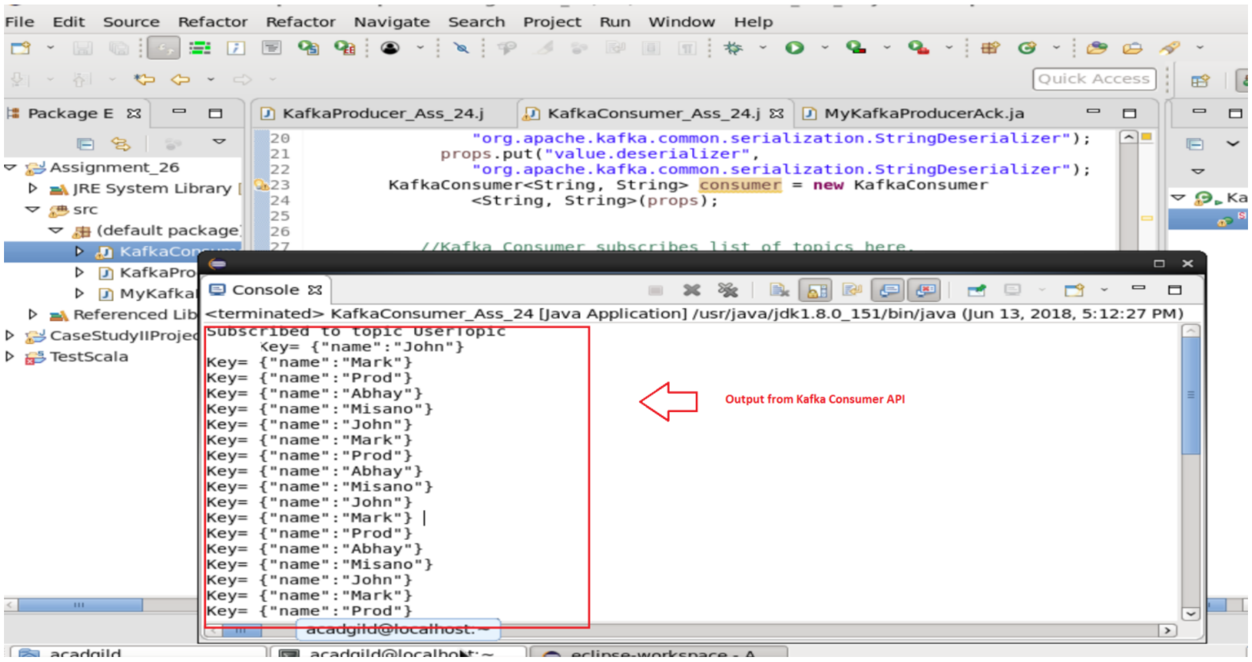


```
                "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer",
                "org.apache.kafka.common.serialization.StringDeserializer");
```

```
<terminated> KafkaConsumer_Ass_24 [Java Application] /usr/java/jdk1.8.0_151/bin/java (Jun 13, 2018, 5:10:37 PM)
Subscribed to topic ItemTopic
     Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
Key= {"item_id":"102"}
Key= {"item_id":"102"}
Key= {"item_id":"104"}
```

```
                System.out.printf("Key= %s \n", consumerRecord.key(), consume
                }
```

UserTopic



```
                "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.
```

```
<terminated> KafkaConsumer_Ass_24 [Java Application] /usr/java/jdk1.8.0_151/bin/java (Jun 13, 2018, 5:12:27 PM)
Subscribed to topic UserTopic
     Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
Key= {"name":"Abhay"}
Key= {"name":"Misano"}
Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
Key= {"name":"Abhay"}
Key= {"name":"Misano"}
Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
Key= {"name":"Abhay"}
Key= {"name":"Misano"}
Key= {"name":"John"}
Key= {"name":"Mark"}
Key= {"name":"Prod"}
```