实验五：Kafka 综合编程实践

“大数据工程”课程实验报告		
题目：Kafka 综合编程实践	学号姓名：郭加璐	日期：2024.5.3

实验环境：

虚拟机软件：VirtualBox 7.0.14

Linux 操作系统：Ubuntu Kylin 22.04.4，虚拟机名称 UbuntuRita

Java 版本：Oracle JDK 1.8

Java IDE：Eclipse

Kafka：3.7.0

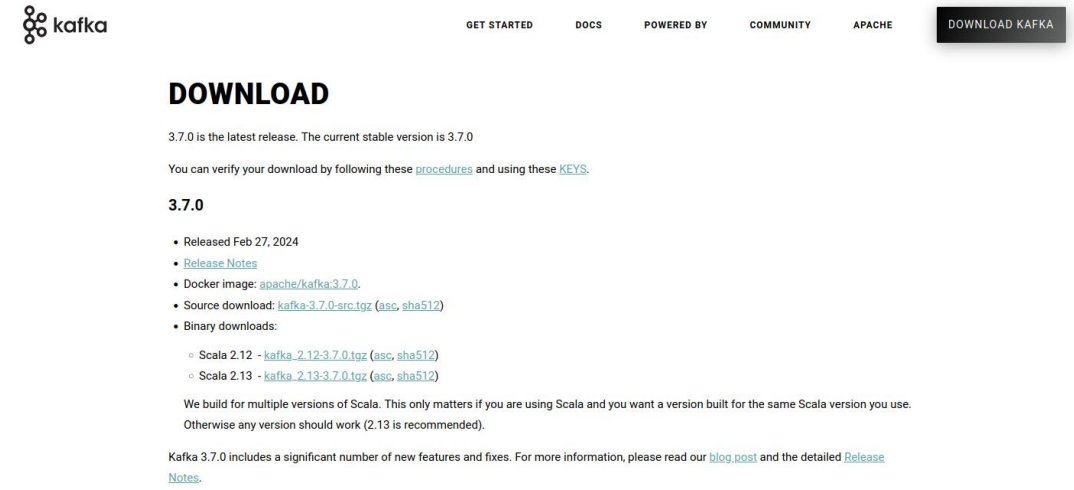
Redis：5:6.0.16-1ubuntu1

实验内容与完成情况：

一、安装部署 Kafka

1. 下载 Kafka

进入 Kafka 官网（https://kafka.apache.org/downloads），下载 kafka_2.13-3.7.0.tgz 至虚拟机“~/Downloads”目录下：



完成 Kafka 安装：

```
hadoop@UbuntuRita:~/Downloads$ cd ..
hadoop@UbuntuRita:~$ cd usr/local
bash: cd: usr/local: No such file or directory
hadoop@UbuntuRita:~$ cd ~/usr/local
bash: cd: /home/hadoop/usr/local: No such file or directory
hadoop@UbuntuRita:~$ cd ..
hadoop@UbuntuRita:/home$ cd ..
hadoop@UbuntuRita:/$ cd usr/local
hadoop@UbuntuRita:/usr/local$ sudo mv kafka_2.13-3.7.0 kafka
hadoop@UbuntuRita:/usr/local$ sudo chown -R hadoop ./kafka
hadoop@UbuntuRita:/usr/local$
```

2. 部署 Kafka 伪分布式集群

在 kafka 目录下建立./etc/目录，将 config 文件夹中的 zookeeper.properties 复制到该目录下：

```
hadoop@UbuntuRita:/usr/local$ cd kafka
hadoop@UbuntuRita:/usr/local/kafka$ sudo mkdir etc
hadoop@UbuntuRita:/usr/local/kafka$ cd config
hadoop@UbuntuRita:/usr/local/kafka/config$ ls
connect-console-sink.properties    consumer.properties
connect-console-source.properties  kraft
connect-distributed.properties     log4j.properties
connect-file-sink.properties       producer.properties
connect-file-source.properties     server.properties
connect-log4j.properties           tools-log4j.properties
connect-mirror-maker.properties    trogdor.conf
connect-standalone.properties      zookeeper.properties
hadoop@UbuntuRita:/usr/local/kafka/config$ sudo mv zookeeper.properties /usr/loc
al/kafka/etc
hadoop@UbuntuRita:/usr/local/kafka/config$ cd ..
hadoop@UbuntuRita:/usr/local/kafka$ cd etc
hadoop@UbuntuRita:/usr/local/kafka/etc$ ls
zookeeper.properties
hadoop@UbuntuRita:/usr/local/kafka/etc$
```

将 config 文件夹中的 server.properties 复制三份至 etc 文件目录中，分别命名未 server-0/1/2.properties：

```
hadoop@UbuntuRita:/usr/local/kafka/etc$ cd ..
hadoop@UbuntuRita:/usr/local/kafka$ cd config
hadoop@UbuntuRita:/usr/local/kafka/config$ sudo cp server.properties /usr/local/
kafka/etc/server-0.properties
hadoop@UbuntuRita:/usr/local/kafka/config$ sudo cp server.properties /usr/local/
kafka/etc/server-1.properties
hadoop@UbuntuRita:/usr/local/kafka/config$ sudo cp server.properties /usr/local/
kafka/etc/server-2.properties
hadoop@UbuntuRita:/usr/local/kafka/config$ cd ..
hadoop@UbuntuRita:/usr/local/kafka$ cd etc
hadoop@UbuntuRita:/usr/local/kafka/etc$ ls
server-0.properties  server-2.properties
server-1.properties  zookeeper.properties
hadoop@UbuntuRita:/usr/local/kafka/etc$
```

配置三个 server-X.properties 文件：

Broker.id = X

listeners = PLAINTEXT://:9092(9093/9094)

Log.dirs.=/tmp/kafka-logsX

```
hadoop@UbuntuRita:/usr/local/kafka/etc$ sudo vim server-0.properties
hadoop@UbuntuRita:/usr/local/kafka/etc$ sudo vim server-1.properties
hadoop@UbuntuRita:/usr/local/kafka/etc$ sudo vim server-2.properties
```

hadoop@UbuntuRita: /usr/local/kafka/etc

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################# Socket Server Settings #############################
##

# The address the socket server listens on. If not configured, the host name wil
l be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and
 port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
#advertised.listeners=PLAINTEXT://your.host.name:9092

# Maps listener names to security protocols, the default is for them to be the s
ame. See the config documentation for more details
-- INSERT --                                                    36,11         17%
```

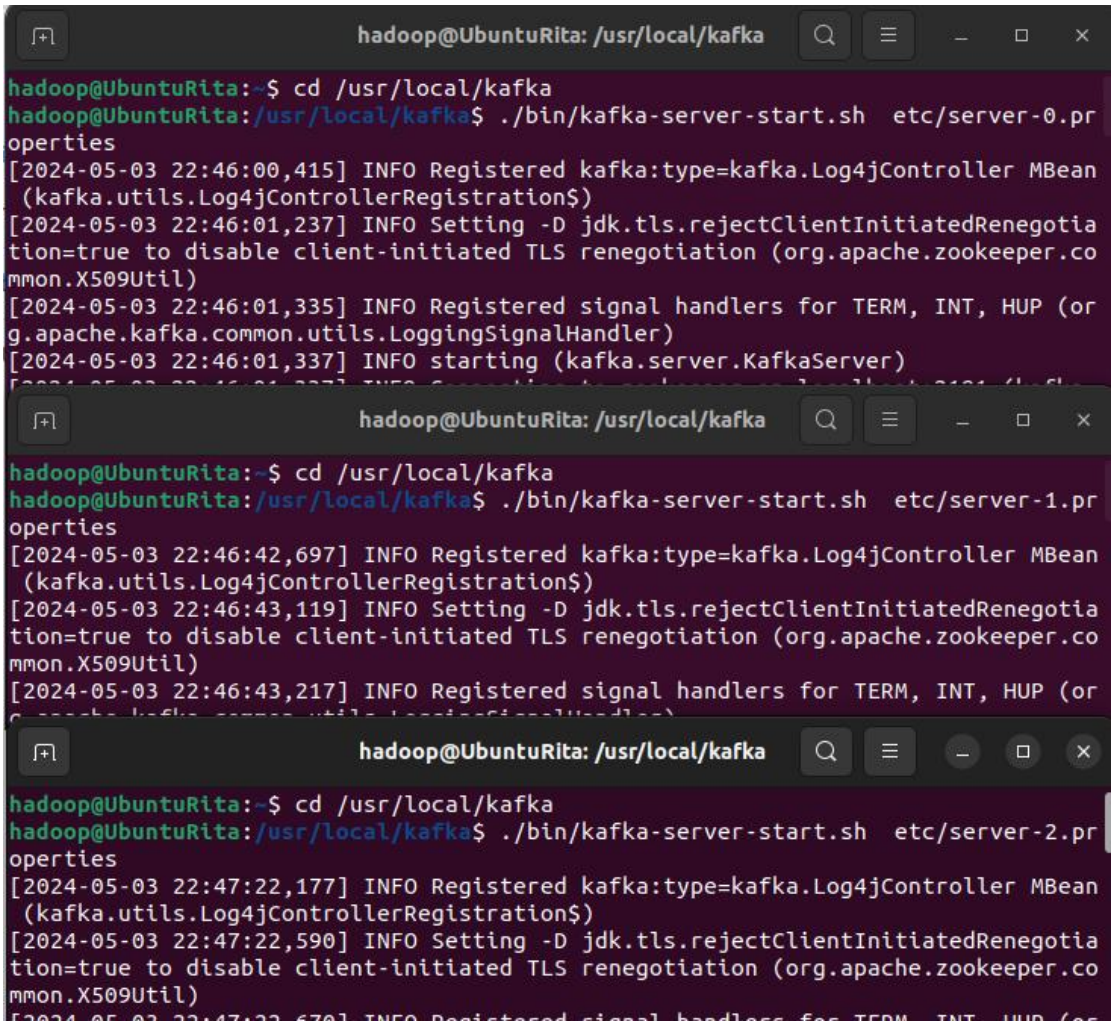
3. 启动 zookeeper 服务器和 kafka 集群：

首先启动 zookeeper：

```
hadoop@UbuntuRita:/usr/local/kafka/etc$ cd ..
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/zookeeper-server-start.sh  etc/zookeep
er.properties
[2024-05-03 22:44:41,219] INFO Reading configuration from: etc/zookeeper.propert
ies (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-05-03 22:44:41,223] WARN etc/zookeeper.properties is relative. Prepend ./
to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConf
ig)
[2024-05-03 22:44:41,269] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zoo
keeper.server.quorum.QuorumPeerConfig)
[2024-05-03 22:44:41,269] INFO secureClientPort is not set (org.apache.zookeeper
.server.quorum.QuorumPeerConfig)
[2024-05-03 22:44:41,269] INFO observerMasterPort is not set (org.apache.zookeep
er.server.quorum.QuorumPeerConfig)
[2024-05-03 22:44:41,269] INFO metricsProvider.className is org.apache.zookeeper
.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumP
```

启动 kafka 集群：

在三个端口分别启动 server-X。



4. 检查启动情况

启动成功：

二、问题 1

根据数据统计需求生成 JSON 文件。

1. 创建 topic

创建三个 topic（评论 comments、点赞 likes、分享 shares）以完成任务：

```
hadoop@UbuntuRita:~$ cd /usr/local/kafka
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/kafka-topics.sh --create --topic comme
nts --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092,loca
lhost:9093,localhost:9094
Created topic comments.
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/kafka-topics.sh --create --topic likes
 --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092,localho
st:9093,localhost:9094
Created topic likes.
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/kafka-topics.sh --create --topic share
s --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092,localh
ost:9093,localhost:9094
Created topic shares.
```

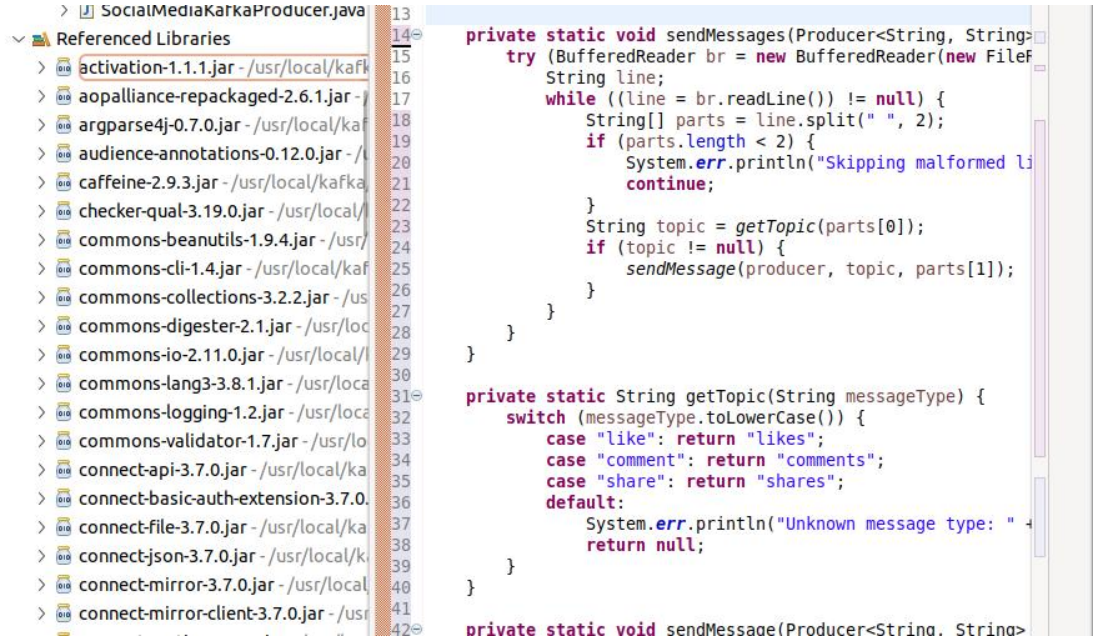
查看 topic 创建情况：

```
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/kafka-topics.sh --describe --topic comments --bootstrap-ser
ver localhost:9092,localhost:9093,localhost:9094
Topic: comments TopicId: 0Lit8eiuSY-2V1lQafN41g PartitionCount: 3       ReplicationFactor: 2    Confi
gs:
        Topic: comments Partition: 0    Leader: 0       Replicas: 0,2   Isr: 0,2
        Topic: comments Partition: 1    Leader: 2       Replicas: 2,1   Isr: 2,1
        Topic: comments Partition: 2    Leader: 1       Replicas: 1,0   Isr: 1,0
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/kafka-topics.sh --describe --topic likes --bootstrap-server
 localhost:9092,localhost:9093,localhost:9094
Topic: likes    TopicId: Ajc_uvMyTZaGD02rxtOQ3w PartitionCount: 3       ReplicationFactor: 2    Confi
gs:
        Topic: likes    Partition: 0    Leader: 2       Replicas: 2,0   Isr: 2,0
        Topic: likes    Partition: 1    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: likes    Partition: 2    Leader: 0       Replicas: 0,1   Isr: 0,1
hadoop@UbuntuRita:/usr/local/kafka$ ./bin/kafka-topics.sh --describe --topic shares --bootstrap-serve
r localhost:9092,localhost:9093,localhost:9094
Topic: shares   TopicId: b_M72rZaQIqwDxW-zAoKcg PartitionCount: 3       ReplicationFactor: 2    Confi
gs:
        Topic: shares   Partition: 0    Leader: 2       Replicas: 2,1   Isr: 2,1
        Topic: shares   Partition: 1    Leader: 1       Replicas: 1,0   Isr: 1,0
        Topic: shares   Partition: 2    Leader: 0       Replicas: 0,2   Isr: 0,2
hadoop@UbuntuRita:/usr/local/kafka$
```

2. 编写 Kafka 生产者程序

从 student_dataset.txt 文件中读取数据生产消息（用户社交媒体行为）放到 Kafka 消息队列中。

首先需要导入必需的 JAR 包：



编写 Kafka 生产者程序 SocialMediaProducer 读取数据生产消息（包括 likes、comments、shares），发送消息并放到 Kafka 消息队列中。具体代码实现如下：

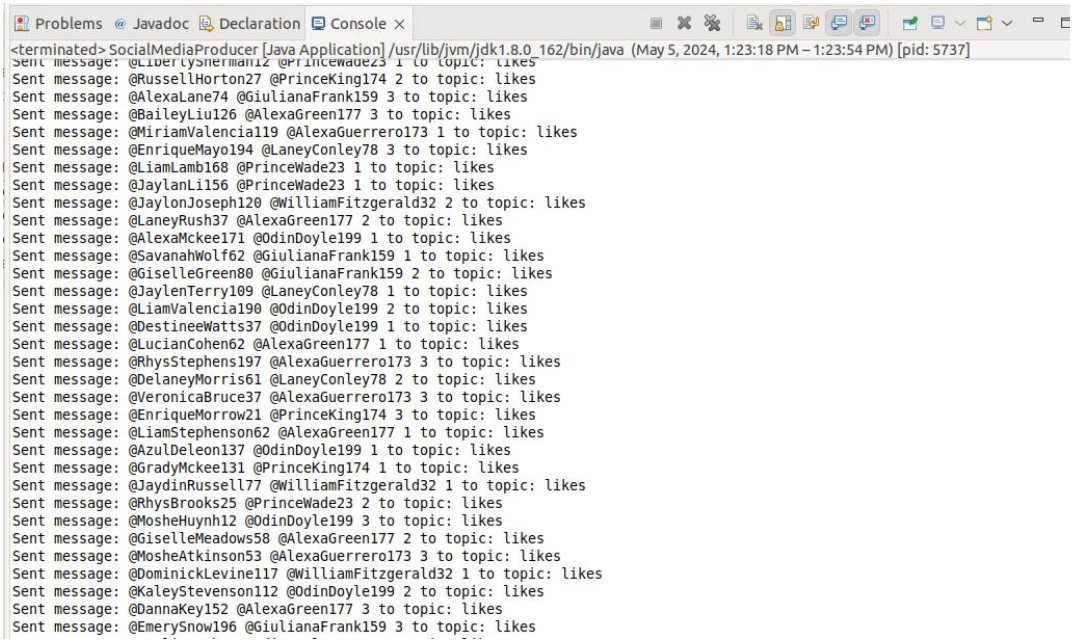
```
import org.apache.kafka.clients.producer.*;
import java.io.*;
import java.util.Properties;


public class SocialMediaProducer {
    private static Properties loadProducerProperties() {
        Properties props = new Properties();
        props.put("bootstrap.servers",
"localhost:9092,localhost:9093,localhost:9094");
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        return props;
    }
    private static void sendMessages(Producer<String, String> producer,
String inputFile) throws IOException {
        try (BufferedReader br = new BufferedReader(new
FileReader(inputFile))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(" ", 2);
                if (parts.length < 2) {
                    System.err.println("Skipping malformed line: " + line);
                    continue;
                }
                String topic = getTopic(parts[0]);
```

```
                    if (topic != null) {
                        sendMessage(producer, topic, parts[1]);
                    }
                }
            }
        }
        private static String getTopic(String messageType) {
            //three topics
            switch (messageType.toLowerCase()) {
                case "like": return "likes";
                case "comment": return "comments";
                case "share": return "shares";
                default:
                    System.err.println("Unknown message type: " + messageType);
                    return null;
            }
        }
        private static void sendMessage(Producer<String, String> producer,
String topic, String message) {
            producer.send(new ProducerRecord<>(topic, null, message),
(metadata, exception) -> {
                if (exception != null) {
                    System.err.println("Failed to send message: " + message +
" to topic: " + topic);
                    exception.printStackTrace();
                } else {
                    System.out.println("Sent message: " + message + " to topic:
" + topic);  //show message
                }
```

```
            });
        }
        public static void main(String[] args) {
            Properties props = loadProducerProperties();
            Producer<String, String> producer = new KafkaProducer<>(props);
            try {
                sendMessages(producer,
"/home/hadoop/Documents/dataset/student_dataset.txt");  //data set path
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                producer.close();
            }
        }
    }
```

运行消费者程序发送信息，并打印发送详情：

Likes：



Comments：

```
Sent message: @GermanMullins22 @OdinDoyle199 1 "gsXPYRsepXLJvwQiCCCcNVSZ " to topic: comments
Sent message: @ReeseLevine47 @PrinceWade23 2 "RgyEHyAxP LegAQlGEqkgBPOGwq KxPLNZwXXFUPhoSUzcULtAmcN ZjjKl " to topic: comments
Sent message: @LibertySherman12 @PrinceWade23 1 "jLEcnZIvqfHUDwlFlCnkPckF XKtMq fOduWihzMdZixOde " to topic: comments
Sent message: @RussellHorton27 @PrinceKing174 2 "ygnhoqzAHXkrRMSjbsOUFM " to topic: comments
Sent message: @AlexaLane74 @GiulianaFrank159 3 "lzRshYobxhG AzhdWRTGnZRmQ lcPdlgNNiJDfZRpJNDTfszoE " to topic: comments
Sent message: @BaileyLiu126 @AlexaGreen177 3 "MZKNEiCiLxVEyNhEO GWNtSWyozCBinMhqXYki spOfLb " to topic: comments
Sent message: @MiriamValencia119 @AlexaGuerrero173 1 "rTrUdWzghOYwkakguaIyPa AbzzAlrYvNj " to topic: comments
Sent message: @EnriqueMayo194 @LaneyConley78 3 "UeprHeGIx ZlVGbiQHaBTp gggZEBO zFyIyZ " to topic: comments
Sent message: @LiamLamb168 @PrinceWade23 1 "SVafqgCAjGnrjnaCeYtMh HDGCZMfLLRANUuWXWoQVpif rMtsCPmMdOVhycSNmWHVSej " to topic:
Sent message: @JaylanLi156 @PrinceWade23 1 "HNFanixIEEzvkJNhkmGds DjxRIMtFfgdSNkDcP haMqAexapCNwmyAbdQ " to topic: comments
Sent message: @JaylonJoseph120 @WilliamFitzgerald32 2 "xJvAllAeDDO ZokpPaBuKViBjysZhYToZyD PopyRn " to topic: comments
Sent message: @LaneyRush37 @AlexaGreen177 2 "rFDGdpBopaDYINDJZFx SkxjSzriExoWpzuXkEPe eDpHDNhlu mxkbQCU umeNMWoNuWSuEbKRxwdQHY
Sent message: @AlexaMckee171 @OdinDoyle199 1 "wlYkOe XNYffaJLwCVnISYJMZcHfbvlY wROReKrLgYoZk YbIGXFVuxkgcYSKyk xgdZAtixxRuqExQ
Sent message: @SavanahWolf62 @GiulianaFrank159 1 "mEMQNFKhRQbw " to topic: comments
Sent message: @GiselleGreen80 @GiulianaFrank159 2 "aweprJKmXqHxGLIngEuhzf EGzxZE IQPxRqMvitaKXZJ eTzjdywcuLbfdVu PRaGrOYTuPyXu
Sent message: @JaylenTerry109 @LaneyConley78 1 "MAIliaMNT hQGTSkXjUYwqvaCrwMv " to topic: comments
Sent message: @LiamValencia190 @OdinDoyle199 2 "IFapPaUsBJKrW " to topic: comments
Sent message: @DestineeWatts37 @OdinDoyle199 1 "xhAndksBghwhwbhJ " to topic: comments
Sent message: @LucianCohen62 @AlexaGreen177 1 "HLvFWfYDbyTVUceFNBwJi tStvtFYxVRnUv AEjsbmy GMVPvDYGtITeXu " to topic: comments
Sent message: @RhysStephens197 @AlexaGuerrero173 3 "YDglobyIAlaLIElmaFXlMkRzL BLynVa ukohppgsXgjNEciLsSC " to topic: comments
Sent message: @DelaneyMorris61 @LaneyConley78 2 "kcAdHYMz jvKoRxvZLUVTSODYbPYEdfun DkcKxSOVXEhsuPcEDZgns " to topic: comments
Sent message: @VeronicaBruce37 @AlexaGuerrero173 3 "PojDxxZiNaciZbl xQFkGYNMXHQaxhvVgkZlsrqR rKXNvtLkEAiZPayYp " to topic: com
Sent message: @EnriqueMorrow21 @PrinceKing174 3 "cLWdkXKKNruIsRxazbNJAsjcf VlBENK tRFzzgVfHBSafCNqx " to topic: comments
Sent message: @LiamStephenson62 @AlexaGreen177 1 "RsFRGVxPzEgtSmm GffgwRP " to topic: comments
Sent message: @AzulDeleon137 @OdinDoyle199 1 "TueCMJAQXaYgrMCRGA lmWqRIjRFtbUwJHxD TumsjCfqgYervZAVuTPkxoF " to topic: comment
Sent message: @GradyMckee131 @PrinceKing174 1 "JznKUuaOdg IjODfeDcSBFfYskixu nsIdxKImgNqZwwGZLX BlFwfCEwh " to topic: comments
Sent message: @JaydinRussell77 @WilliamFitzgerald32 1 "UQDsfBaVikrSqEFxeiOey mHNwhbEGkWPvOzumx cHJvnAOJTykuswcAFNnfcn KDEIxb "
Sent message: @RhysBrooks25 @PrinceWade23 2 "kHavTKpt " to topic: comments
Sent message: @MosheHuynh12 @OdinDoyle199 3 "rycPMQIKsNSCNoSIxmuIsCK eaYGYnFNuEPKvm RLTQJYvNdOTa pkaAoRHImNJB waHGQDWjXMaKbYgg
Sent message: @GiselleMeadows58 @AlexaGreen177 2 "KEncLDuBOGdTod " to topic: comments
Sent message: @MosheAtkinson53 @AlexaGuerrero173 3 "HrqYohx " to topic: comments
Sent message: @DominickLevine117 @WilliamFitzgerald32 1 "CUFANAXxusacSCnNiTL FDnzguF budqBesgNtUokxmwJYhL UaqbMan tubDzwhGU "
Sent message: @KaleyStevenson112 @OdinDoyle199 2 "hjlODpXqYRJL pjCZQBnFsUDAEUEKFbNXwsdI wOGBRsaFzFHy zVFDHAiHncZxsJWTKx wbrRiL
Sent message: @DannaKey152 @AlexaGreen177 3 "yuGWQnFvqsHCYG nOLizcxYlmaVyBnsscN " to topic: comments
```

Shares：

```
Sent message: @EnriqueMorrow21 @PrinceKing174 3 @AddysonKing27  to topic: shares
Sent message: @LiamStephenson62 @AlexaGreen177 1 @YamiletMullins66 @GermanWade128 @MilesRush147 @AddysonKing27 @DawsonLevine6
Sent message: @AzulDeleon137 @OdinDoyle199 1 @KarlieMeadows156 @YamiletMullins66 @KarlieMeadows156 @KarlieMeadows156 @ArelyOr
Sent message: @GradyMckee131 @PrinceKing174 1 @YamiletMullins66 @JustinAtkinson174 @DawsonLevine61 @AnahiKeller165 @MilesRush
Sent message: @JaydinRussell77 @WilliamFitzgerald32 1 @AnahiKeller165 @AddysonKing27 @AddysonKing27 @AnahiKeller165 @YamiletM
Sent message: @RhysBrooks25 @PrinceWade23 2 @AddysonKing27 @GermanWade128 @GermanWade128 @MilesRush147 @ArelyOrozco141 @Justi
Sent message: @MosheHuynh12 @OdinDoyle199 3 @AddysonKing27 @ArelyOrozco141 @GermanWade128 @JustinAtkinson174 @DawsonLevine61
Sent message: @GiselleMeadows58 @AlexaGreen177 2 @GermanWade128 @MilesRush147 @JustinAtkinson174 @AnahiKeller165 @GermanWade1
Sent message: @MosheAtkinson53 @AlexaGuerrero173 3 @KarlieMeadows156 @MilesRush147  to topic: shares
Sent message: @DominickLevine117 @WilliamFitzgerald32 1 @ArelyOrozco141 @JustinAtkinson174 @ArelyOrozco141 @AnahiKeller165 @A
Sent message: @KaleyStevenson112 @OdinDoyle199 2 @AddysonKing27 @ArelyOrozco141 @YamiletMullins66 @MilesRush147 @MilesRush147
Sent message: @DannaKey152 @AlexaGreen177 3 @JustinAtkinson174 @KarlieMeadows156 @ArelyOrozco141 @AddysonKing27 @AnahiDennis1
Sent message: @EmerySnow196 @GiulianaFrank159 3 @JustinAtkinson174 @AnahiDennis140 @AnahiKeller165 @JustinAtkinson174 @Justin
Sent message: @KarlieRush61 @OdinDoyle199 1 @KarlieMeadows156 @JustinAtkinson174 @MilesRush147 @DawsonLevine61 @ArelyOrozco14
Sent message: @RebekahGonzalez79 @LaneyConley78 1 @AnahiKeller165 @MilesRush147 @AddysonKing27 @AddysonKing27 @AddysonKing27
Sent message: @JaydinWise185 @LaneyConley78 2 @JustinAtkinson174 @AnahiKeller165 @JustinAtkinson174  to topic: shares
Sent message: @DaisyEllis78 @LaneyConley78 3 @ArelyOrozco141 @ArelyOrozco141  to topic: shares
Sent message: @JaxMoyer45 @PrinceWade23 1 @AnahiKeller165 @JustinAtkinson174 @ArelyOrozco141 @AnahiDennis140 @AddysonKing27 @
Sent message: @MauricioGonzalez107 @OdinDoyle199 1 @AddysonKing27 @DawsonLevine61 @JustinAtkinson174 @KarlieMeadows156 @Anahi
Sent message: @SavanahWolf62 @AlexaGuerrero173 2 @YamiletMullins66 @ArelyOrozco141 @AnahiKeller165 @KarlieMeadows156 @AnahiKe
Sent message: @MarisolKey54 @AnnikaLiu115 3 @MilesRush147 @GermanWade128 @DawsonLevine61 @DawsonLevine61 @YamiletMullins66 @Y
Sent message: @PrinceMeadows1 @AlexaGuerrero173 3 @GermanWade128 @ArelyOrozco141 @AnahiKeller165 @JustinAtkinson174 @KarlieMe
Sent message: @ArelyKey182 @WilliamFitzgerald32 2 @AnahiDennis140 @KarlieMeadows156 @MilesRush147 @DawsonLevine61 @DawsonLevi
Sent message: @ShylaLin96 @OdinDoyle199 3 @DawsonLevine61 @DawsonLevine61 @YamiletMullins66 @GermanWade128 @GermanWade128 @Mi
Sent message: @LiamRush105 @LaneyConley78 1 @AnahiKeller165 @JustinAtkinson174 @KarlieMeadows156 @KarlieMeadows156 @DawsonLev
Sent message: @YamiletNguyen36 @ThaliaMckee62 1 @YamiletMullins66 @DawsonLevine61 @YamiletMullins66 @AnahiKeller165 @KarlieMe
```

3. 编写 Kafka 消费者程序

从 Kafka 消息队列中消费消息，并进行统计计算 likes、comments、popularity，最后把统计结果输出存储到 JSON 文件中。

编写 Kafka 消费者程序 SocialMediaConsumer 消费信息并保存为 JSON 文件。具体代码实现如下：

```
import org.apache.kafka.clients.consumer.*;

import org.apache.kafka.common.serialization.StringDeserializer;



import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.databind.node.ObjectNode;
```

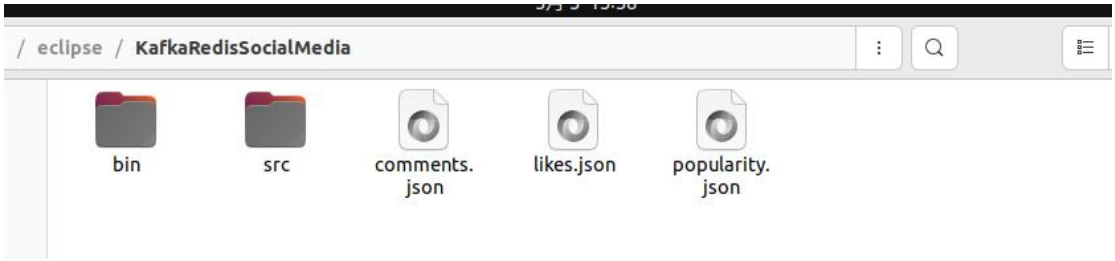
```
    import java.io.FileWriter;
    import java.util.*;
    import java.util.concurrent.TimeUnit;
    public class SocialMediaConsumer {
        private static final ObjectMapper mapper = new ObjectMapper();
        private static Properties loadConsumerProperties() {
            Properties props = new Properties();
            props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092,localhost:9093,localhost:9094");
            props.put(ConsumerConfig.GROUP_ID_CONFIG, "social-media-group");
            props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
            props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
            props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
            return props;
        }
        private static void processMessages(KafkaConsumer<String, String>
consumer) {
            Map<String, Set<String>> userComments = new HashMap<>();
            Map<String, Map<String, Integer>> userLikes = new HashMap<>();
            Map<String, Integer> userPopularity = new HashMap<>();
            final int giveUp = 10000; // 10 seconds timeout
            int noRecordsCount = 0;
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(100);
                if (records.count() == 0) {
                    noRecordsCount += 100;
                    if (noRecordsCount > giveUp) break;
```

```
                } else {
                    noRecordsCount = 0;
                    records.forEach(record -> processRecord(record,
userComments, userLikes, userPopularity));
                }
                if (!records.isEmpty()) {
                    try {
                        writeToJsonFiles(userComments, userLikes,
userPopularity);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }
        private static void processRecord(ConsumerRecord<String, String>
record, Map<String, Set<String>> userComments, Map<String, Map<String,
Integer>> userLikes, Map<String, Integer> userPopularity) {
            String[] parts = record.value().split(" ");
            String topic = record.topic();
            String userWhoPosted = parts[1];
            String postId = parts[2];
            switch (topic) {
                case "likes":
                    userLikes.computeIfAbsent(userWhoPosted, k -> new
HashMap<>())
                             .merge(postId, 1, Integer::sum);
                    userPopularity.merge(userWhoPosted, 1, Integer::sum);
                    break;
```

```
                case "comments":
                    userComments.computeIfAbsent(userWhoPosted, k -> new
HashSet<>()).add(parts[3]);
                    userPopularity.merge(userWhoPosted, 5, Integer::sum);
                    break;
                case "shares":
                    int shareCount = parts.length - 3;
                    userPopularity.merge(userWhoPosted, 20 * shareCount,
Integer::sum);  //compute popularity
                    break;
            }
        }

        //output Json file
        private static void writeToJsonFiles(Map<String, Set<String>>
userComments, Map<String, Map<String, Integer>> userLikes, Map<String,
Integer> userPopularity) throws Exception {
            try (FileWriter commentsWriter = new FileWriter("comments.json");
                 FileWriter likesWriter = new FileWriter("likes.json");
                 FileWriter popularityWriter = new
FileWriter("popularity.json")) {
                mapper.writeValue(commentsWriter, userComments);
                mapper.writeValue(likesWriter, userLikes);
                ObjectNode popularityJson = mapper.createObjectNode();
                userPopularity.forEach((user, popularity) -> {
                    popularityJson.put(user, popularity / 1000.0);
                });
                mapper.writeValue(popularityWriter, popularityJson);
            }
```

```
        }

        public static void main(String[] args) {

            Properties props = loadConsumerProperties();

            try (KafkaConsumer<String, String> consumer = new
KafkaConsumer<>(props)) {

                consumer.subscribe(Arrays.asList("likes", "comments",
"shares"));

                processMessages(consumer);

            } catch (Exception e) {

                e.printStackTrace();

            }

        }

    }
```

得到三个 JSON 文件：



Likes：

Comments：

Popularity：

三、问题 2

改写问题 1 中的 Kafka 消费者程序，将 Kafka 消费者程序中原来将统计结果输出存储到 JSON 文件的操作，转变成将统计结果输出存储到 Redis 数据库中。

1. Redis 数据库部署

首先，在虚拟机配置 Redis 数据库：

```
hadoop@UbuntuRita:~$ sudo apt-get install redis-server
[sudo] password for hadoop:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjemalloc2 liblua5.1-0 liblzf1 lua-bitop lua-cjson redis-tools
Suggested packages:
  ruby-redis
The following NEW packages will be installed:
  libjemalloc2 liblua5.1-0 liblzf1 lua-bitop lua-cjson redis-server
  redis-tools
0 upgraded, 7 newly installed, 0 to remove and 70 not upgraded.
Need to get 1,273 kB of archives.
After this operation, 5,725 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/universe amd64 libjemallo
c2 amd64 5.2.1-4ubuntu1 [240 kB]
```

查看 Redis 是否成功启动：

```
hadoop@UbuntuRita:~$ service redis-server status
● redis-server.service - Advanced key-value store
     Loaded: loaded (/lib/systemd/system/redis-server.service; enabled; vendor >
     Active: active (running) since Sun 2024-05-05 13:43:18 CST; 48s ago
       Docs: http://redis.io/documentation,
             man:redis-server(1)
   Main PID: 6393 (redis-server)
     Status: "Ready to accept connections"
      Tasks: 5 (limit: 9430)
     Memory: 2.6M
        CPU: 184ms
     CGroup: /system.slice/redis-server.service
             └─6393 "/usr/bin/redis-server 127.0.0.1:6379" "" "" "" "" "" "" "">
lines 1-12/12 (END)
```

修改 Kafka 消费者程序 SocialMediaRedis：

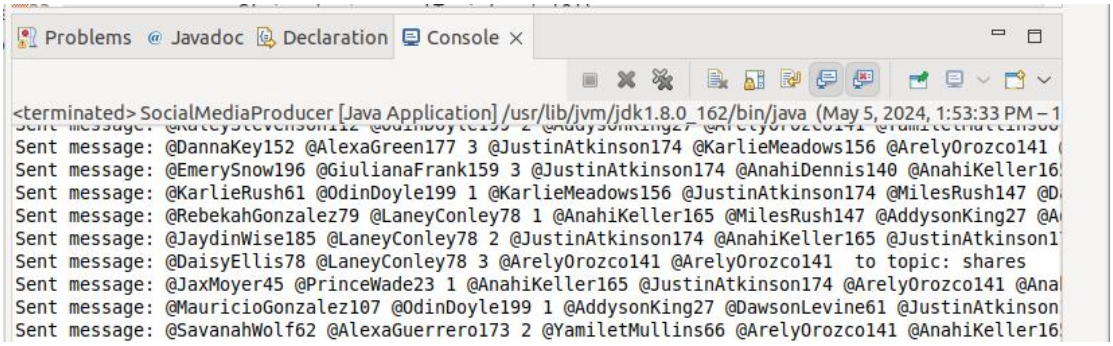
```
import redis.clients.jedis.Jedis;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.common.serialization.StringDeserializer;

import java.util.*;
public class SocialMediaRedis {
    private static final String REDIS_HOST = "localhost";  //Redis host
    private static final int REDIS_PORT = 6379;  //Redis port
    private static Properties createConsumerProperties() {
```

```
            Properties props = new Properties();
            props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092,localhost:9093,localhost:9094");
            props.put(ConsumerConfig.GROUP_ID_CONFIG, "social-media-group");
            props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
            props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
            props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
            return props;
        }
        private static void consumeMessages(KafkaConsumer<String, String>
consumer, Jedis jedis) {
            final int giveUp = 10000;
            int noRecordsCount = 0;
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(100);
                if (records.count() == 0) {
                    noRecordsCount += 100;
                    if (noRecordsCount > giveUp) break;
                } else {
                    noRecordsCount = 0;
                    System.out.println("Received " + records.count() + "
records");
                    records.forEach(record -> processRecord(record, jedis));
                }
            }
        }
        private static void processRecord(ConsumerRecord<String, String>
record, Jedis jedis) {
            String[] parts = record.value().split(" ");
            String topic = record.topic();
            String userWhoPosted = parts[1];
            String postId = parts[2];
            //three topics
            switch (topic) {
                case "likes":
                    jedis.hincrBy("likes:" + userWhoPosted, postId, 1);
                    break;
                case "comments":
                    String comment = parts[3];
                    jedis.rpush("comments:" + userWhoPosted, comment);
                    break;
                case "shares":
```

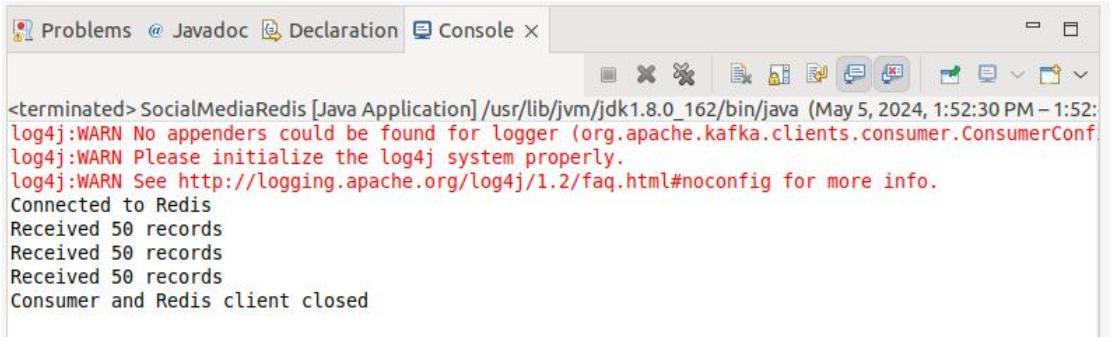
```
                    int shareCount = parts.length - 3;
                    jedis.incrBy("popularity:" + userWhoPosted, 20 *
shareCount);
                    break;
                }
            }
            public static void main(String[] args) {
                try (Jedis jedis = new Jedis(REDIS_HOST, REDIS_PORT);
                     KafkaConsumer<String, String> consumer = new
KafkaConsumer<>(createConsumerProperties())) {
                    System.out.println("Connected to Redis");
                    consumer.subscribe(Arrays.asList("likes", "comments",
"shares"));
                    consumeMessages(consumer, jedis);
                } catch (Exception e) {
                    System.err.println("Error in processing: " + e.getMessage());
                    e.printStackTrace();
                } finally {
                    System.out.println("Consumer and Redis client closed");
                }
            }
        }
```

运行结果如下：

Producer：



RedisConsumer：



查看 Redis 数据库中结果：

```
hadoop@UbuntuRita:~$ redis-cli
127.0.0.1:6379> KEYS *
 1) "popularity:@LaneyConley78"
 2) "popularity:@OdinDoyle199"
 3) "comments:@AlexaGreen177"
 4) "likes:@OdinDoyle199"
 5) "comments:@AlexaGuerrero173"
 6) "comments:@PrinceWade23"
 7) "comments:@ThaliaMckee62"
 8) "likes:@AlexaGreen177"
 9) "comments:@AnnikaLiu115"
10) "comments:@GiulianaFrank159"
11) "popularity:@WilliamFitzgerald32"
12) "popularity:@AlexaGreen177"
13) "popularity:@GiulianaFrank159"
14) "comments:@PrinceKing174"
15) "comments:@OdinDoyle199"
16) "popularity:@ThaliaMckee62"
17) "popularity:@PrinceKing174"
18) "likes:@ThaliaMckee62"
19) "likes:@AnnikaLiu115"
20) "likes:@PrinceWade23"
21) "likes:@PrinceKing174"
22) "likes:@WilliamFitzgerald32"
23) "likes:@AlexaGuerrero173"
24) "comments:@WilliamFitzgerald32"
25) "comments:@LaneyConley78"
26) "likes:@LaneyConley78"
27) "likes:@GiulianaFrank159"
28) "popularity:@PrinceWade23"
29) "popularity:@AnnikaLiu115"
30) "popularity:@AlexaGuerrero173"
127.0.0.1:6379>
```

出现的问题：

1. 安装 Kafka 后导入 libs 中的 jar 包后，仍缺少相关 jar 包。

解决方案（列出遇到的问题和解决办法，列出没有解决的问题）：

1. 逐个搜索导入所需 jar 包。

注：报告篇幅可根据实际题目情况进行调整。