Student Name: Tan Boon Ji

Matriculation Number: A0200362Y

Link to Github repository:
https://github.com/tanboonji/OTOT_Assignment/tree/main/OTOT_Task_D

Instructions on how to set up and run Kafka cluster:

1. On cmd, navigate your directory to the `OTOT_Task_D` folder.

```
D:\Project Folder\OTOT>cd OTOT_Task_D

D:\Project Folder\OTOT\OTOT_Task_D>
```

2. Start ZooKeeper and Kafka by using the command `docker-compose up` and wait for the images to build and run. (Ensure you have docker running on your computer)

```
D:\Project Folder\OTOT\OTOT_Task_A1>docker-compose up
```

3. Open another cmd, navigate the directory to the `OTOT_Task_D` folder again.

```
D:\Project Folder\OTOT>cd OTOT_Task_D

D:\Project Folder\OTOT\OTOT_Task_D>
```

4. Run the command `docker-compose ps` and ensure that all the services are up and running.

```
D:\Project Folder\OTOT\OTOT_Task_D>docker-compose ps
          Name                        Command               State   Ports
-------------------------------------------------------------------------
otot_task_d_kafka-1_1       /etc/confluent/docker/run   Up
otot_task_d_kafka-2_1       /etc/confluent/docker/run   Up
otot_task_d_kafka-3_1       /etc/confluent/docker/run   Up
otot_task_d_zookeeper-1_1   /etc/confluent/docker/run   Up
otot_task_d_zookeeper-2_1   /etc/confluent/docker/run   Up
otot_task_d_zookeeper-3_1   /etc/confluent/docker/run   Up
```

5. To verify that ZooKeeper is healthy, we will check the ZooKeeper logs by using the following command below.

   (Note that I am on Windows, and the following command uses `findstr` which is not available on Mac. For Mac users, replace `findstr` with `grep` instead.)

   `docker-compose logs zookeeper-1 | findstr "Created FOLLOWING"`

```
D:\Project Folder\OTOT\OTOT_Task_D>docker-compose logs zookeeper-1 | findstr "FOLLOWING Created"
```

   Ensure that you see messages like the following after running the command.

```
←[36mzookeeper-1_1  |←[0m [2021-11-11 14:10:24,620] INFO Created server with tickTime 2000 minSessionTimeout 4000 max
SessionTimeout 40000 clientPortListenBacklog -1 datadir /var/lib/zookeeper/log/version-2 snapdir /var/lib/zookeeper/d
ata/version-2 (org.apache.zookeeper.server.ZooKeeperServer)
```

```
←[36mzookeeper-1_1  |←[0m [2021-11-11 14:10:24,621] INFO FOLLOWING - LEADER ELECTION TOOK - 704 MS (org.apache.zookee
per.server.quorum.Learner)
```

6. There are three ZooKeeper instances and in Step 5, we only checked if zookeeper-1 was healthy. Repeat Step 5 to verify that zookeeper-2 and zookeper-3 are also healthy.

   `docker-compose logs zookeeper-2 | findstr “Created FOLLOWING”`

   `docker-compose logs zookeeper-3 | findstr “Created FOLLOWING”`

7. To verify that Kafka broker has successfully booted up, we will similarly be checking the Kafka logs by using the following command below.

   (Similarly, I am using `findstr` in the command. For Mac users, replace `findstr` with `grep`.)

   `docker-compose logs kafka-1 | findstr started`

   Ensure that you see the message like the following below after running the command.

```
←[36mkafka-1_1      |←[0m [2021-11-11 14:10:30,158] INFO [KafkaServer id=1] started (kafka.server.KafkaServer)
```

8. There are three Kafka brokers and in Step 7, we only checked if kafka-1 was successfully booted up. Repeat Step 7 to verify that kafka-2 and kafka-3 are also booted up.

   `docker-compose logs kafka-1 | findstr started`

   `docker-compose logs kafka-1 | findstr started`

9. To test that the brokers are working as expected, we first create a topic with the following command.

`docker run --net=host --rm confluentinc/cp-kafka:5.0.0 kafka-topics --create --topic bar --partitions 3 --replication-factor 3 --if-not-exists --zookeeper localhost:32181`

   You should see the following message after running the command.

```
Created topic "bar".
```

10. Now verify that the topic is created successfully by describing the topic with the following command.

`docker run --net=host --rm confluentinc/cp-kafka:5.0.0 kafka-topics --describe --topic bar --zookeeper localhost:32181`

    You should see this after running the command.

```
Topic:bar       PartitionCount:3        ReplicationFactor:3     Configs:
        Topic: bar      Partition: 0    Leader: 2       Replicas: 2,1,3 Isr: 2,1,3
        Topic: bar      Partition: 1    Leader: 3       Replicas: 3,2,1 Isr: 3,2,1
        Topic: bar      Partition: 2    Leader: 1       Replicas: 1,3,2 Isr: 1,3,2
```

    From the above, you can also see that there are nodes to take over the master node when the master node fails.

```
Topic:bar       PartitionCount:3        ReplicationFactor:3     Configs:
        Topic: bar      Partition: 0    Leader: 2       Replicas: 2,1,3 Isr: 2,1,3
        Topic: bar      Partition: 1    Leader: 3       Replicas: 3,2,1 Isr: 3,2,1
        Topic: bar      Partition: 2    Leader: 1       Replicas: 1,3,2 Isr: 1,3,2
```

The `Leader` is the master node, while the Replicas are the nodes that will take over the master node in the event of a failure.

11. Next, generate some data to the `bar` topic created in Step 9 by running the following command.

`docker run --net=host --rm confluentinc/cp-kafka:5.0.0 bash -c "seq 33 | kafka-console-producer --broker-list localhost:29092 --topic bar && echo 'Produced 33 messages.'"`

You should see the following message after the data has been successfully generated

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>Produced 33 messages.
```

12. The last step is to check that the consumer has received all the messages generated. We do so by reading the messages back using the Console Consumer using the following command.

`docker run --net=host --rm confluentinc/cp-kafka:5.0.0 kafka-console-consumer --bootstrap-server localhost:29092 --topic bar --from-beginning --max-messages 33`

You should see following messages.

```
2
5
8
11
14
17
20
23
26
29
32
3
6
9
12
15
18
21
24
27
30
33
1
4
7
10
13
16
19
22
25
28
31
Processed a total of 33 messages
```

You can see from the last line that a total of 33 messages have been processed. However, do note that the ordering of the messages is not sequential, but there will be total of 33 messages from 1 to 33.