Task D

Wednesday, 9 November 2022   11:51 pm

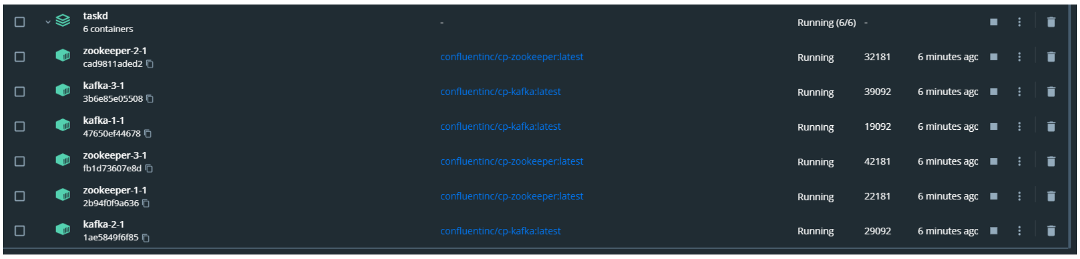
Name: Nikki Than Win
Matriculation Number: A0217715E
Link:

Task D1

1. Clone my github repo and you will find docker-compose.yml file in the root directory of my repo.
2. Build the docker container that will contain the kafka and zookeeper nodes using the following command in the root directory of the repo.

   docker compose up -d

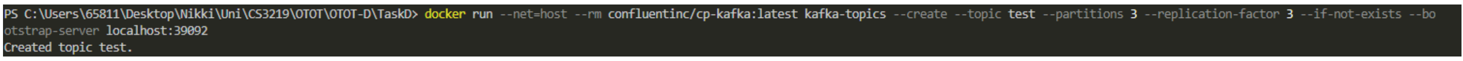
3. After successfully building, you should see the following container in your docker.



□	taskd 6 containers	-	Running (6/6)	-		
□	zookeeper-2-1 cad9811aded2	confluentinc/cp-zookeeper:latest	Running	32181	6 minutes agc	
□	kafka-3-1 3b6e85e05508	confluentinc/cp-kafka:latest	Running	39092	6 minutes agc	
□	kafka-1-1 47650ef44678	confluentinc/cp-kafka:latest	Running	19092	6 minutes agc	
□	zookeeper-3-1 fb1d73607e8d	confluentinc/cp-zookeeper:latest	Running	42181	6 minutes agc	
□	zookeeper-1-1 2b94f0f9a636	confluentinc/cp-zookeeper:latest	Running	22181	6 minutes agc	
□	kafka-2-1 1ae5849f6f85	confluentinc/cp-kafka:latest	Running	29092	6 minutes agc	

4. Next create a topic using the following command. In this case, I am naming my topic "test"

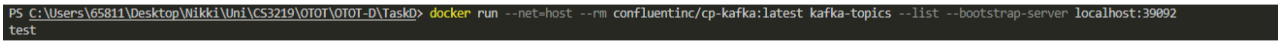
   docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --create --topic test --partitions 3 --replication-factor 3 --if-not-exists --bootstrap-server localhost:39092



```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --create --topic test --partitions 3 --replication-factor 3 --if-not-exists --bo
otstrap-server localhost:39092
Created topic test.
```

5. Check that your topic has been created by running the following command, you should see the topic that you have just created:

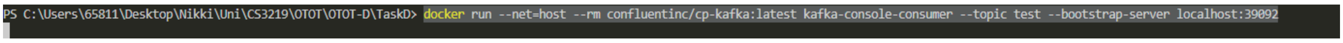
   docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --list --bootstrap-server localhost:39092



```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --list --bootstrap-server localhost:39092
test
```

6. Next run the following command for your kafka node consumer to start listening for messages posted to this topic.

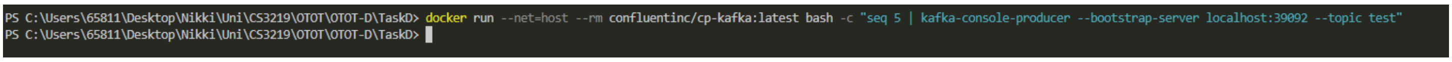
   docker run --net=host --rm confluentinc/cp-kafka:latest kafka-console-consumer --topic test --bootstrap-server localhost:39092



```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-console-consumer --topic test --bootstrap-server localhost:39092
```

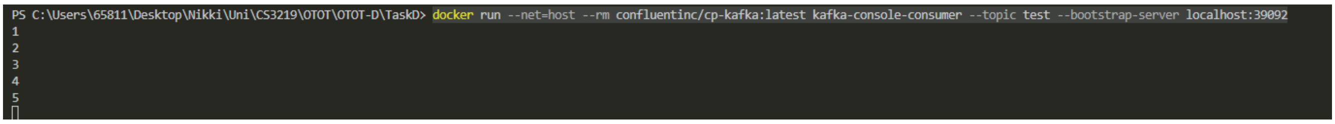
7. Open another terminal, now we will set up our producer to send 1 - 5 sequentially to our consumer. Run the following command:

   docker run --net=host --rm confluentinc/cp-kafka:latest bash -c "seq 5 | kafka-console-producer --bootstrap-server localhost:39092 --topic test"



```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest bash -c "seq 5 | kafka-console-producer --bootstrap-server localhost:39092 --topic test"
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD>
```

   If you have been following the steps thus far, in your first terminal you will see 1 - 5 recieved sequentially:



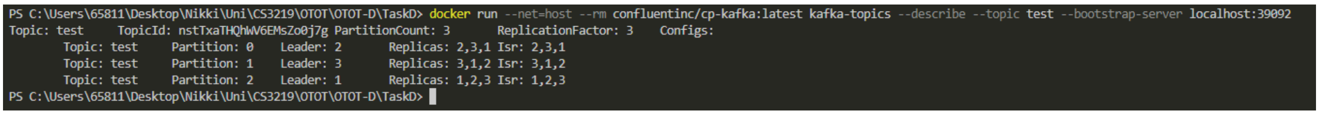
```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-console-consumer --topic test --bootstrap-server localhost:39092
1
2
3
4
5
```

Task D2

Next we will be showing the failure of the master node being taken over by another node to become the new master node.

1. Run the following command to describe your current cluster.

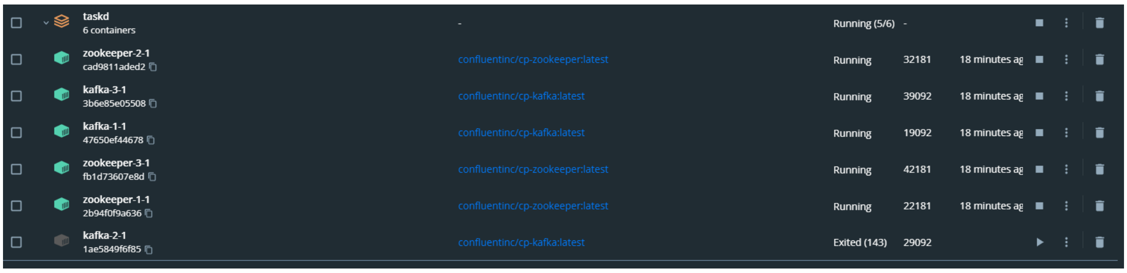
   docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic test --bootstrap-server localhost:39092



```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic test --bootstrap-server localhost:39092
Topic: test     TopicId: nstTxaTHQhWV6EMsZo0j7g PartitionCount: 3       ReplicationFactor: 3    Configs:
        Topic: test     Partition: 0    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
        Topic: test     Partition: 1    Leader: 3       Replicas: 3,1,2 Isr: 3,1,2
        Topic: test     Partition: 2    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD>
```

   Here you will see the different partitions in your kafka cluster as well as the leaders for each of the different partitions.

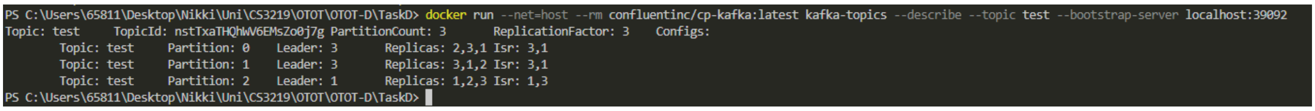
2. We observe that for partition 0, broker 2 is the leader. Let us kill this broker within the docker application.



□	taskd 6 containers	-	Running (5/6)	-		
□	zookeeper-2-1 cad9811aded2	confluentinc/cp-zookeeper:latest	Running	32181	18 minutes ag	
□	kafka-3-1 3b6e85e05508	confluentinc/cp-kafka:latest	Running	39092	18 minutes ag	
□	kafka-1-1 47650ef44678	confluentinc/cp-kafka:latest	Running	19092	18 minutes ag	
□	zookeeper-3-1 fb1d73607e8d	confluentinc/cp-zookeeper:latest	Running	42181	18 minutes ag	
□	zookeeper-1-1 2b94f0f9a636	confluentinc/cp-zookeeper:latest	Running	22181	18 minutes ag	
□	kafka-2-1 1ae5849f6f85	confluentinc/cp-kafka:latest	Exited (143)	29092		▶

3. We now describe our cluster again using the same command:

```
docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic test --
bootstrap-server localhost:39092
```



```
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic test --bootstrap-server localhost:39092
Topic: test     TopicId: nstTxaTHQhWV6EMsZo0j7g PartitionCount: 3       ReplicationFactor: 3    Configs:
        Topic: test     Partition: 0    Leader: 3       Replicas: 2,3,1 Isr: 3,1
        Topic: test     Partition: 1    Leader: 3       Replicas: 3,1,2 Isr: 3,1
        Topic: test     Partition: 2    Leader: 1       Replicas: 1,2,3 Isr: 1,3
PS C:\Users\65811\Desktop\Nikki\Uni\CS3219\OTOT\OTOT-D\TaskD>
```

We notice that since broker 2 is killed, for the same partition 0, broker 3 takes over as the leader(master node). Thus showing successful management of the failure of the master node being taken over by another.