

CS3219 OTOT Task D

Name: Teo Yick Fong Alex

Student Number: A022144R

Task D-1

1. Create a docker compose project containing 6 docker containers, 3 for kafka brokers and 3 for zookeeper nodes
2. Create the containers by using the docker-compose.yml file in this repo
 - `.docker compose up -d`
3. Verify that the containers are running
 - `.docker ps`

```
PS C:\Users\alex> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fe9967266e7e	confluentinc/cp-kafka:latest	"/etc/confluent/dock..."	29 minutes ago	Up 29 minutes		kafka-kafka-1-1
3e79511cca79	confluentinc/cp-kafka:latest	"/etc/confluent/dock..."	29 minutes ago	Up 29 minutes		kafka-kafka-2-1
2b4a681fb630	confluentinc/cp-kafka:latest	"/etc/confluent/dock..."	29 minutes ago	Up 29 minutes		kafka-kafka-3-1
a3ab5cac8795	confluentinc/cp-zookeeper:latest	"/etc/confluent/dock..."	29 minutes ago	Up 29 minutes		kafka-zookeeper-3-1
6da769db1f0f	confluentinc/cp-zookeeper:latest	"/etc/confluent/dock..."	29 minutes ago	Up 29 minutes		kafka-zookeeper-1-1
40a8105bc61d	confluentinc/cp-zookeeper:latest	"/etc/confluent/dock..."	29 minutes ago	Up 29 minutes		kafka-zookeeper-2-1

1. Create a new topic

1. `docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --create --topic topic1 --partitions 3 --replication-factor 3 --if-not-exists --bootstrap-server localhost:39092`

```
PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --create --topic topic1 --partitions 3 --replication-factor 3 --if-not-exists --bootstrap-server localhost:39092
Created topic topic1.
PS C:\Users\alex> |
```

2. Check that the topic has been created successfully

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

```
PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --list --bootstrap-server localhost:39092
topic1
PS C:\Users\alex> |
```

3. Start another shell and begin listening for incoming messages on `topic1` by using `kafka-console-consumer`

1. `docker run --net=host --rm confluentinc/cp-kafka:latest kafka-console-consumer --topic topic1 --bootstrap-server localhost:39092`
2. Now, any messages published to `topic1` will be displayed on this shell.

```
PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-console-consumer --topic topic1 --bootstrap-server localhost:39092
|
```

4. Send messages to the message broker using `kafka-console-producer`

1. For this, we will send 10 messages from **1** to **10** by starting **kafka-console-producer** and echoing the values into the shell
2. `docker run --net=host --rm confluentinc/cp-kafka:latest bash -c "seq 10 | kafka-console-producer --bootstrap-server localhost:39092 --topic topic1"`

The image shows two terminal windows. The left window shows the command to run a Kafka console producer: `PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest bash -c "seq 10 | kafka-console-producer --bootstrap-server localhost:39092 --topic topic1"`. The right window shows the command to run a Kafka console consumer: `PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-console-consumer --topic topic1 --bootstrap-server localhost:39092`. The consumer window shows the output of the sequence: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

5. The messages **1** to **10** should be displayed on the shell that is consuming the message and it will be waiting for further messages published to **topic1**.

Task D-2

1. Describe current cluster

1. `docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic topic1 --bootstrap-server localhost:39092`
2. Currently, broker 3 is the leader of partition 2, with all brokers being in-sync replicas.

The image shows a terminal window with the command `PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic topic1 --bootstrap-server localhost:39092`. The output shows the topic details for 'topic1' with 3 partitions. Partition 0 has leader 2 and replicas 2,3,1. Partition 1 has leader 3 and replicas 3,1,2. Partition 2 has leader 1 and replicas 1,2,3.

2. Kill broker 3

3. Describe cluster again

1. `docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic topic1 --bootstrap-server localhost:29092`
2. Broker 3 is killed and the other 2 nodes take over as the leader for all 3 partitions.
3. In-sync replicas are left with broker 1 and 2 as 3 is dropped from the cluster.

The image shows a terminal window with the command `PS C:\Users\alex> docker run --net=host --rm confluentinc/cp-kafka:latest kafka-topics --describe --topic topic1 --bootstrap-server localhost:29092`. The output shows the topic details for 'topic1' with 3 partitions. Partition 0 has leader 2 and replicas 2,1. Partition 1 has leader 1 and replicas 3,1,2. Partition 2 has leader 1 and replicas 1,2. This indicates that broker 3 has been removed from the cluster and its replicas have been reassigned to the remaining brokers.