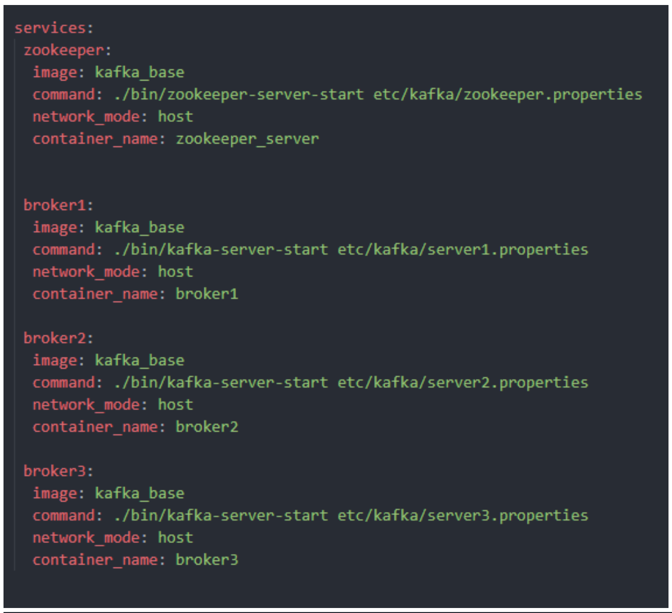
OTOT_Task_D

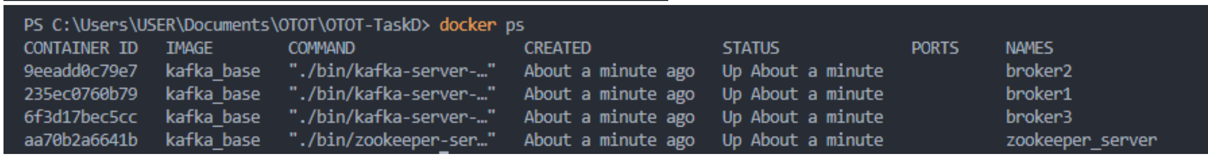
Goh Ee Liang, A0202170B

GitHub link: https://github.com/Elgoh/PubSubApp

Steps to begin:

1. Ensure that docker engine and docker-compose is installed.
    a. https://docs.docker.com/get-docker/
2. Clone the repo from the GitHub link above
3. Open a new terminal and run `docker build -t kafka_base .`
    a. The build is based on the Dockerfile in the repo
    b. Open-jdk-8 & configuration files for kafka broker are installed and downloaded
    c. The image will be the be created
4. Using the same terminal, run `docker-compose up`
    a. This launches the cluster with 4 containers as seen in the docker-compose.yml file
    b. 

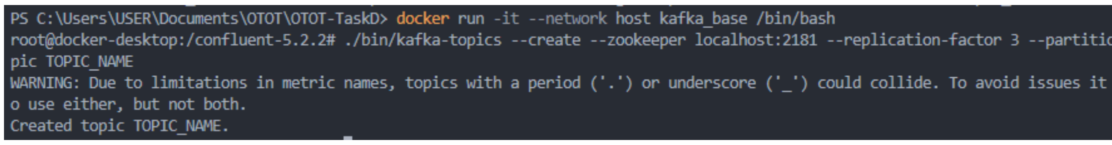
```
services:
 zookeeper:
  image: kafka_base
  command: ./bin/zookeeper-server-start etc/kafka/zookeeper.properties
  network_mode: host
  container_name: zookeeper_server


 broker1:
  image: kafka_base
  command: ./bin/kafka-server-start etc/kafka/server1.properties
  network_mode: host
  container_name: broker1

 broker2:
  image: kafka_base
  command: ./bin/kafka-server-start etc/kafka/server2.properties
  network_mode: host
  container_name: broker2

 broker3:
  image: kafka_base
  command: ./bin/kafka-server-start etc/kafka/server3.properties
  network_mode: host
  container_name: broker3
```

    c. 

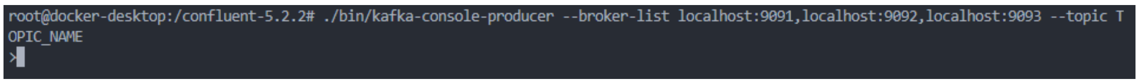
```
PS C:\Users\USER\Documents\OTOT\OTOT-TaskD> docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED              STATUS              PORTS     NAMES
9eeadd0c79e7   kafka_base   "./bin/kafka-server-…"   About a minute ago   Up About a minute             broker2
235ec0760b79   kafka_base   "./bin/kafka-server-…"   About a minute ago   Up About a minute             broker1
6f3d17bec5cc   kafka_base   "./bin/kafka-server-…"   About a minute ago   Up About a minute             broker3
aa70b2a6641b   kafka_base   "./bin/zookeeper-ser…"   About a minute ago   Up About a minute             zookeeper_server
```

Steps to test:

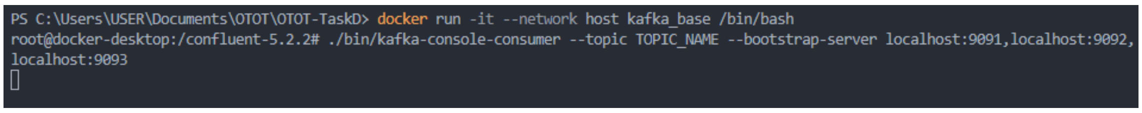
1. Creation of topic
    a. First run a client with the command:
        i. Open a new terminal
        ii. `docker run -it --network host kafka_base /bin/bash`
    b. Next create a topic with the command:
        i. `./bin/kafka-topics --create --zookeeper localhost:2181 --replication-factor 3 --partitions 3 --topic TOPIC_NAME`
        ii. You can change TOPIC_NAME to whatever you like
        iii. NOTE: message published on `TOPIC_NAME` can only be consumed by consumer that subscribes to `TOPIC_NAME`

iv. 

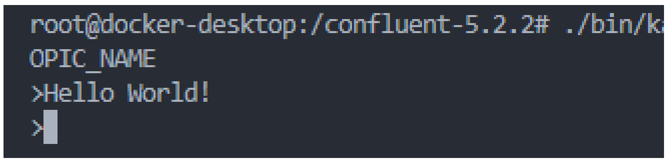
```
PS C:\Users\USER\Documents\OTOT\OTOT-TaskD> docker run -it --network host kafka_base /bin/bash
root@docker-desktop:/confluent-5.2.2# ./bin/kafka-topics --create --zookeeper localhost:2181 --replication-factor 3 --partitic
pic TOPIC_NAME
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it
o use either, but not both.
Created topic TOPIC_NAME.
```

2. Creation of Publisher/Producer
   a. Using the above terminal,
   b. Run `./bin/kafka-console-producer --broker-list localhost:9091,localhost:9092,localhost:9093 --topic TOPIC_NAME`
   c. 

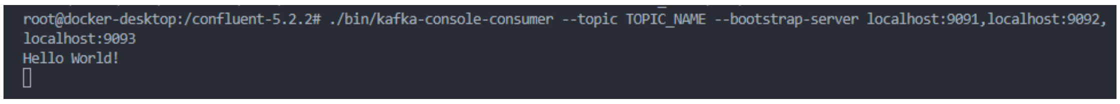
```
root@docker-desktop:/confluent-5.2.2# ./bin/kafka-console-producer --broker-list localhost:9091,localhost:9092,localhost:9093 --topic T
OPIC_NAME
>
```

3. Creation of Subscriber/Consumer
   a. Open a new terminal
   b. Run `docker run -it --network host kafka_base /bin/bash`
   c. Run `./bin/kafka-console-consumer --topic TOPIC_NAME --bootstrap-server localhost:9091,localhost:9092,localhost:9093`
   d. 

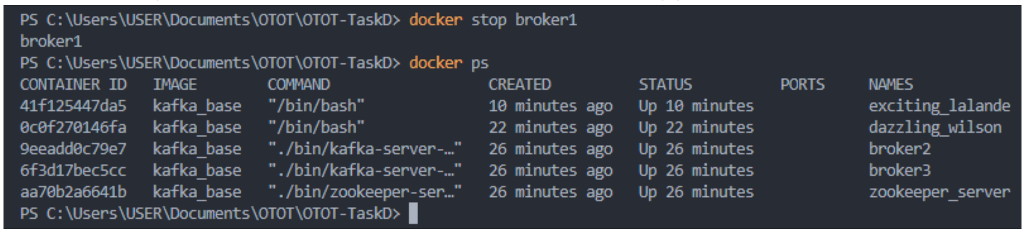
```
PS C:\Users\USER\Documents\OTOT\OTOT-TaskD> docker run -it --network host kafka_base /bin/bash
root@docker-desktop:/confluent-5.2.2# ./bin/kafka-console-consumer --topic TOPIC_NAME --bootstrap-server localhost:9091,localhost:9092,
localhost:9093
```

4. Test that message can be passed from Publisher to Subscriber
   a. Type a message in the Publisher's terminal as below
   b. 

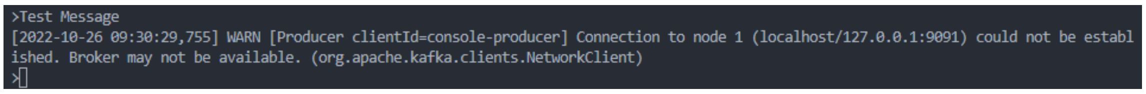
```
root@docker-desktop:/confluent-5.2.2# ./bin/k
OPIC_NAME
>Hello World!
>
```

   c. Check Subscriber's terminal and it should contain the message as it is subscribed to the same topic as the Publisher
   d. 

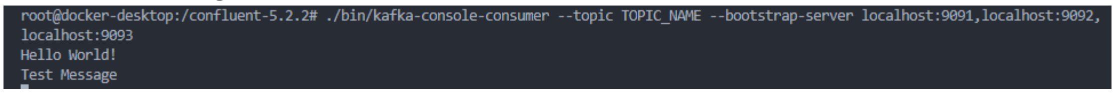
```
root@docker-desktop:/confluent-5.2.2# ./bin/kafka-console-consumer --topic TOPIC_NAME --bootstrap-server localhost:9091,localhost:9092,
localhost:9093
Hello World!
```

5. Test that if a Kafka broker is down, messages will still be able to pass through other brokers
   a. Open a new terminal
   b. Run `docker stop broker1`
   c. Run `docker ps` and check that broker 1 has indeed stopped
   d. 

```
PS C:\Users\USER\Documents\OTOT\OTOT-TaskD> docker stop broker1
broker1
PS C:\Users\USER\Documents\OTOT\OTOT-TaskD> docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS          PORTS     NAMES
41f125447da5   kafka_base   "/bin/bash"              10 minutes ago   Up 10 minutes             exciting_lalande
0c0f270146fa   kafka_base   "/bin/bash"              22 minutes ago   Up 22 minutes             dazzling_wilson
9eeadd0c79e7   kafka_base   "./bin/kafka-server-…"   26 minutes ago   Up 26 minutes             broker2
6f3d17bec5cc   kafka_base   "./bin/kafka-server-…"   26 minutes ago   Up 26 minutes             broker3
aa70b2a6641b   kafka_base   "./bin/zookeeper-ser…"   26 minutes ago   Up 26 minutes             zookeeper_server
PS C:\Users\USER\Documents\OTOT\OTOT-TaskD>
```

   e. Send a message from the publisher
   f. 

```
>Test Message
[2022-10-26 09:30:29,755] WARN [Producer clientId=console-producer] Connection to node 1 (localhost/127.0.0.1:9091) could not be establ
ished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
>
```

   g. Check that message is received from subscriber
   h. 

```
root@docker-desktop:/confluent-5.2.2# ./bin/kafka-console-consumer --topic TOPIC_NAME --bootstrap-server localhost:9091,localhost:9092,
localhost:9093
Hello World!
Test Message
```

   i. This can be done on any broker and the message will still pass through, if time permits, do try it.