

CS3219 Task D: Pub-Sub Messaging

Name: Koh Vinleon

Matric Number: A0202155W

GitHub Link: <https://github.com/glatiuden/CS3219-OTOT-TaskD>

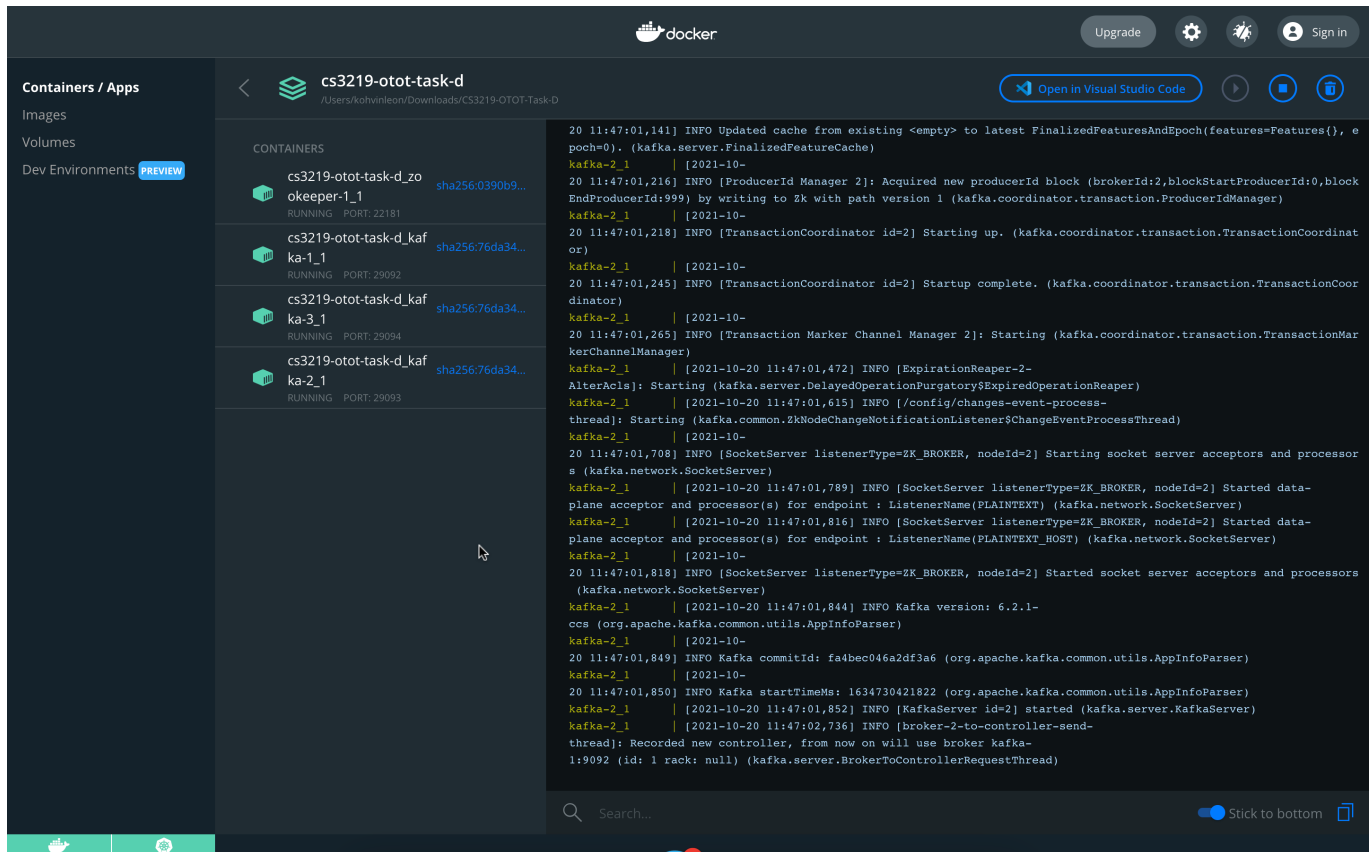
Set Up Instructions

All the configuration has been set in `docker-compose.yml` in the same directory.

To run the containers, execute the following command:

```
docker-compose up -d
```

The four containers (1 Zookeeper & 3 Kafka nodes) should start running.



In addition, we can verify whether are the servers listening to the port by executing:

```
nc -z localhost 22181
nc -z localhost 29092
nc -z localhost 29093
nc -z localhost 29094
```

```
CS3219-OTOT-Task-D — fish /Users/kohvinleon/Downloads/CS3219-OTOT-Task-D — -fish — 107x28
[kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D> nc -z localhost 22181
Connection to localhost port 22181 [tcp/*] succeeded!
[kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D> nc -z localhost 29092
Connection to localhost port 29092 [tcp/*] succeeded!
[kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D> nc -z localhost 29093
Connection to localhost port 29093 [tcp/*] succeeded!
[kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D> nc -z localhost 29094
Connection to localhost port 29094 [tcp/*] succeeded!
[kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D> ]
```

Pub/Sub

To start off, we need to create a topic within our Kafka's container. Before we can run any commands in our Kafka container, we need to execute the following command in command prompt/terminal:

```
docker exec -it cs3219-otot-task-d_kafka-1_1 bash
```

Upon executing, your command prompt/terminal should be running bash within the Kafka's container.

Topic

We need a topic before we can demonstrate the Pub/Sub messaging capability. We will be creating a topic called **ps-topic** for this demonstration with this command:

```
kafka-topics --create --topic ps-topic --partitions 3 --replication-factor 3 --zookeeper zookeeper-1:2181
```

Your command prompt/terminal should output:

```
Created topic ps-topic.
```

Producer

Now, we will be entering the producer console.

If you have exited the bash terminal within the Kafka container, please refer to [Pub/Sub](#) to execute the command again.

To enter the producer console, we can execute

```
kafka-console-producer --topic ps-topic --bootstrap-server kafka-1:9092
```

Consumer

Now, we will be entering the consumer console. For demonstration purpose, please open a separate command prompt/terminal for the following instructions this while maintaining the producer console as connected.

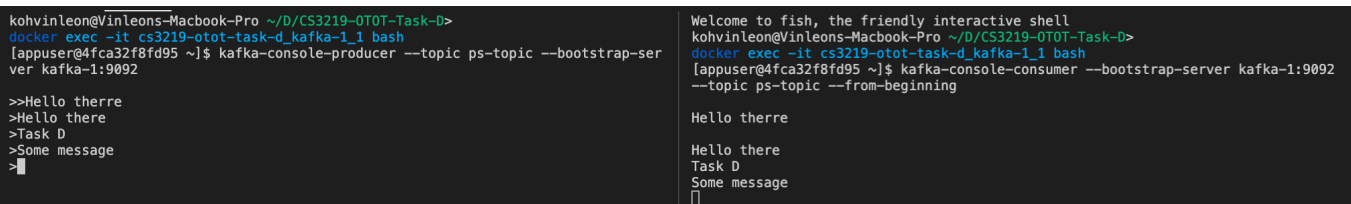
Please refer to [Pub/Sub](#) to connect to Kafka's container bash.

To enter the producer console, we can execute

```
kafka-console-consumer --bootstrap-server kafka-1:9092 --topic ps-topic --from-beginning
```

Demonstration

Please ensure you have both [Producer](#) and [Consumer](#) console running. For the demonstration, you can enter any message in the producer console and it should reflect on the consumer's console instantaneously.



The screenshot shows two terminal windows side-by-side. The left window is the Kafka Producer console, and the right window is the Kafka Consumer console. Both are running in a Docker container named 'cs3219-otot-task-d_kafka-1.1'. The Producer console shows messages being sent: 'Hello therre', 'Hello there', 'Task D', and 'Some message'. The Consumer console shows these same messages being received and displayed.

```
kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D>
docker exec -it cs3219-otot-task-d_kafka-1.1 bash
[appuser@4fca32f8fd95 ~]$ kafka-console-producer --topic ps-topic --bootstrap-ser
ver kafka-1:9092

>>Hello therre
>Hello there
>Task D
>Some message
>

Welcome to fish, the friendly interactive shell
kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D>
docker exec -it cs3219-otot-task-d_kafka-1.1 bash
[appuser@4fca32f8fd95 ~]$ kafka-console-consumer --bootstrap-server kafka-1:9092
--topic ps-topic --from-beginning

Hello therre

Hello there
Task D
Some message
█
```

Successful management of the failure of the master node in the cluster

If you are continuing from the previous section, you may stop either one of the producer or consumer console by pressing **Ctrl + C**, otherwise open a new terminal and follow the instructions [here](#).

Commands to be executed are in the Kafka's container bash terminal otherwise stated.

First, we can view the master (or leader) nodes of each partition to see the before and after. To do this, we can run:

```
kafka-topics --zookeeper zookeeper-1:2181 --describe --topic ps-topic
```

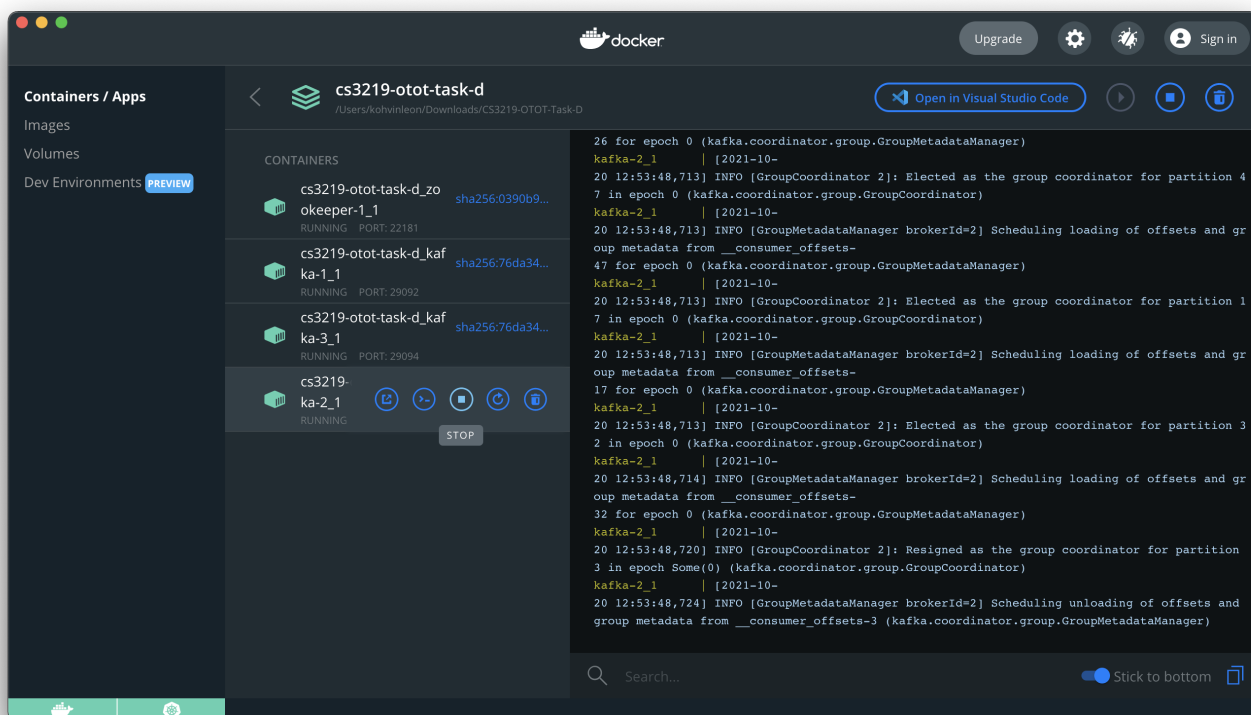
```
kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D> docker exec -it cs3219-otot-task-d_kafka-1_1 bash
[appuser@4fca32f8fd95 ~]$ kafka-topics --zookeeper zookeeper-1:2181 --describe --topic ps-topic

Topic: ps-topic TopicId: IPJBz9_ZRsSl2m8PMU3XNQ PartitionCount: 3      ReplicationFactor: 3      Configs:
  Topic: ps-topic Partition: 0    Leader: 1      Replicas: 1,2,3 Isr: 1,2,3
  Topic: ps-topic Partition: 1    Leader: 2      Replicas: 2,3,1 Isr: 2,3,1
  Topic: ps-topic Partition: 2    Leader: 3      Replicas: 3,1,2 Isr: 3,1,2
[appuser@4fca32f8fd95 ~]$
[appuser@4fca32f8fd95 ~]$
```

As we can see here, node 1 is the master (or leader) for partition 0, node 2 for partition 1 and node 3 for partition 2.

To demonstrate the "failure" event, we will be killing one of the Kafka's node. You may kill any nodes of your choice. For this demonstration, we will be killing node 2, named **cs3219-otot-task-d_kafka-2_1**.

You may kill the node via the Docker Desktop GUI by pressing stop button:



...or open another terminal/command prompt (on your desktop, not in Kafka's container), and execute:

```
docker container kill cs3219-otot-task-d_kafka-2_1
```

Once the node has stop, we can run `kafka-topics --zookeeper zookeeper-1:2181 --describe --topic ps-topic` once again to see which node takeover as master for partition 1.

```
kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D>
docker exec -it cs3219-otot-task-d_kafka-1_1 bash
[appuser@4fca32f8fd95 ~]$ kafka-topics --zookeeper zookeeper-1:2181 --describe -
--topic ps-topic
Topic: ps-topic TopicId: IPJBz9_ZRs5l2m8PMU3XNQ PartitionCount: 3 Replicati
onFactor: 3 Configs:
  Topic: ps-topic Partition: 0 Leader: 1 Replicas: 1,2,3 Isr: 1,3
  Topic: ps-topic Partition: 1 Leader: 3 Replicas: 2,3,1 Isr: 3,1
  Topic: ps-topic Partition: 2 Leader: 3 Replicas: 3,1,2 Isr: 3,1
[appuser@4fca32f8fd95 ~]$
```

```
kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D>
docker container kill cs3219-otot-task-d_kafka-2_1
cs3219-otot-task-d_kafka-2_1
kohvinleon@Vinleons-Macbook-Pro ~/D/CS3219-OTOT-Task-D>
```

We can see that node 3 has become the leader for partition 1, successfully taking over as the master node when node 2 was killed.

References

- <https://www.baeldung.com/ops/kafka-docker-setup/>
- <https://kafka-tutorials.confluent.io/kafka-console-consumer-producer-basics/kafka.html>
- <https://medium.com/big-data-engineering/hello-kafka-world-the-complete-guide-to-kafka-with-docker-and-python-f788e2588cfc>