Name: Lim Yong Shen, Kevin
Matriculation Number: A0183562B
GitHub Link: https://github.com/lyskevin/docker-apache-kafka-zookeeper.git

Intructions To Run The Kafka Cluster

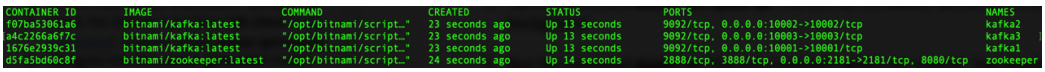
1. Ensure that Docker, Docker-Compose, and Kafkacat are installed on your computer (you should be able to execute the commands docker, docker-compose, and kafkacat, respectively, from the terminal).

2. Clone the GitHub repository (**git clone** https://github.com/lyskevin/docker-apache-kafka-zookeeper.git). A folder called **docker-apache-kafka-zookeeper** should appear.

3. Navigate into the **docker-apache-kafka-zookeeper** folder.

4. Run the command **docker-compose up -d** to run containers in the background. This will run the docker containers in the background. You may also run **docker-compose up** to view all the logs but you will need to open a separate terminal window to run commands and also send CTRL+C before running **docker-compose down** at the end.

5. Run the command **docker ps**. This lists containers which are running. You should see something similar to the following image:

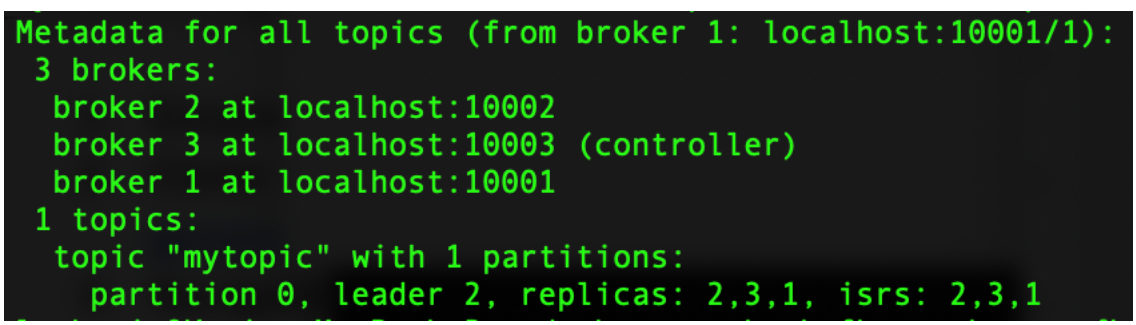


   There are 4 containers running: 3 for the Apache-Kafka nodes and 1 for Zookeeper.

Pub-Sub Service

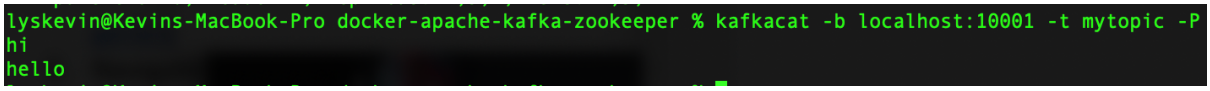
1. Run the command **docker exec -it kafka1 kafka-topics.sh --bootstrap-server kafka1:9092 --create --topic mytopic --partitions 1 --replication-factor 3**. This creates a topic called “mytopic” in the Kafka cluster. It should output a line saying “Created topic mytopic”. Feel free to replace **mytopic** in the command with another topic of your choice.

2. Examine the current state of the cluster by running the command **kafkacat -L -b localhost:10001**. You should see something similar to the following image:



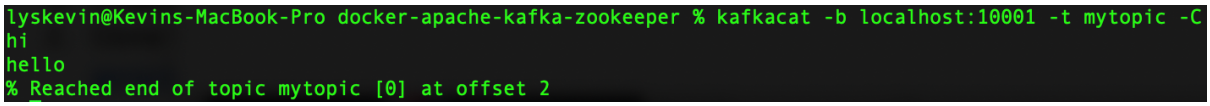
3. To send messages to the topic, enter the command **kafkacat -b localhost:10001 -t mytopic -P**. Replace **mytopic** with your own in the command if you changed it when creating the topic.

4. At this point, the terminal window will expect some input from the user. Each message is ended by the newline character (i.e. pressing the enter key). You may press CTRL+D (sends the EOF command) when you are done entering the messages.



I entered the messages “hi” and “hello” in the above image.

5. In order to read the messages, enter the command **kafkacat -b localhost:10001 -t mytopic -C**. You will see something similar to the following image:

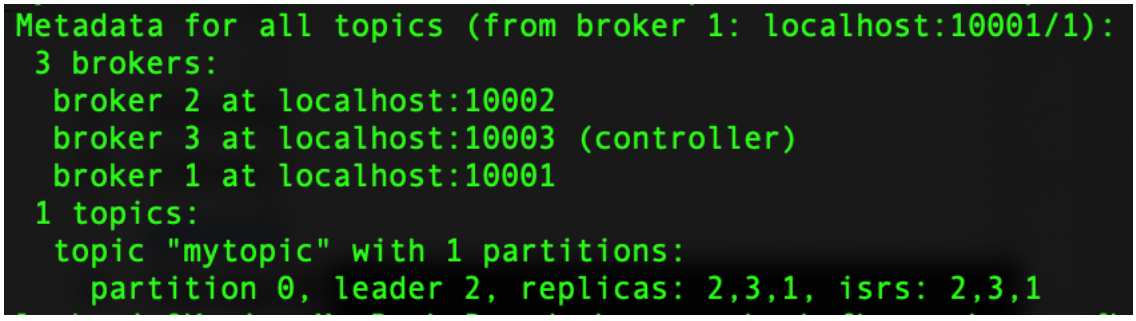


6. And we are done! You may open two terminal windows (one for the publisher and one for the subscriber; i.e. one for each of steps 4 and 5) if you would like to see real-time message sending and receiving. The windows should have their respective commands running in order to see the real-time messages.

Successful Delegation During Node Failure

1. After you have tried the first part of the instructions, you should have already created a topic. Otherwise, feel free to create one now.

2. Let’s examine the state of the cluster before one of the nodes fails. Run the command **kafkacat -L -b localhost:10001**. You should see something similar to the following image:

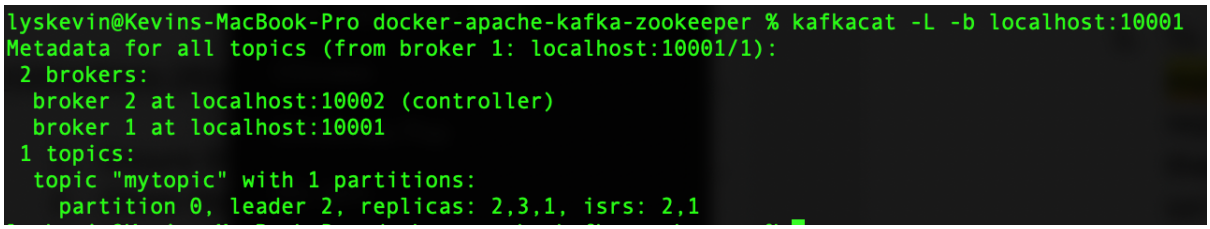


Right now, broker (node) 3 is the controller (this may be different for you).

3. Stop the controller node by running the command **docker kill kafka<controller-ID>**. In my case, it was **docker kill kafka3**. You should replace the 3 with the number of the broker with (controller) next to it.

4. Wait for > 18 seconds (let's say 30 seconds), which is the maximum amount of time before Zookeeper detects that the controller broker has timed out. It will then attempt to reassign the controller role to another existing broker.

5. To verify that this has occurred, run the command **kafkacat -b localhost:10001 -t mytopic -C** once more. Note that if you killed kafka1 in step 3, then you should replace **localhost:10001** with either **localhost10002** or **localhost:10003**, which are the addresses that map to kafka2 and kafka3, respectively. You should see something similar to the following image:



   In my case, the controller role was reassigned to kafka2.

6. Let's also verify that messages still get received even if a node goes down before they are received. At this point, you should have 2 nodes remaining. In my case, I had kafka1 and kafka2. Take note of which one is the leader of the topic. You can see from the above image that my leader was kafka2 (it says leader 2 at the least line). This is different from being the controller node, but a node could be both a topic leader and a controller, as seen above.

7. Run the command **kafkacat -b localhost:1000<leader-ID> -t mytopic -P** and send some messages. Press CTRL+D after you are done. In my case, I sent them to **localhost:10002** since kafka2 was the leader.

8. Now, before we receive the messages, run **docker kill kafka<leader-ID>**. In my case, it was kafka2. Again, wait for at least 20 seconds for zookeeper to reassign topic leader and controller roles.

9. Open a separate terminal window and run **kafkacat -b localhost:1000<non-leader-ID> -t mytopic -C**, with **<non-leader-ID>** being the number of the non-leader node. In my case, this was kafka1, so I used **localhost:10001**. You may receive messages that you sent in the past, but you should also see your most recent message(s) (the ones which were sent just before killing the topic leader). This demonstrates that messages are still received even if the node that we initially sent them to is down. This is because of the replication factor of 3 that we included when creating the topic; every message that is sent to the topic are copied to 2 additional brokers if possible. Again, press CTRL+C to terminate the process once you have finished sending your messages. You may close this terminal window.

10. Let's also verify that the topic leader has indeed changed to the only remaining node (assuming that you ran steps 1 to 9 and have killed two kafka nodes so far). In the original window, run the command **kafkacat -L -b localhost:1000<remaining-node-**

ID>. In my case, it was **localhost:10001**. You should see that this remaining node has become the topic leader.

11. Run **docker-compose down** to stop the Docker containers from running.