DSC650-Week-07-Mukherjee

Author: Chitramoy Mukherjee

Date: 01/27/2024

Exercise 1: Topic Creation and Verification in Kafka (On One Terminal Only)

Exercise 1: Create a Kafka topic named ‘my-topic’.

/opt/kafka_2.13-2.8.1/bin/kafka-topics.sh --create --topic my-topic --bootstrap-server localhost:9092

Exercise 2: List the topics to verify that ‘my-topic’ has been successfully created.

/opt/kafka_2.13-2.8.1/bin/kafka-topics.sh --list --bootstrap-server localhost:9092

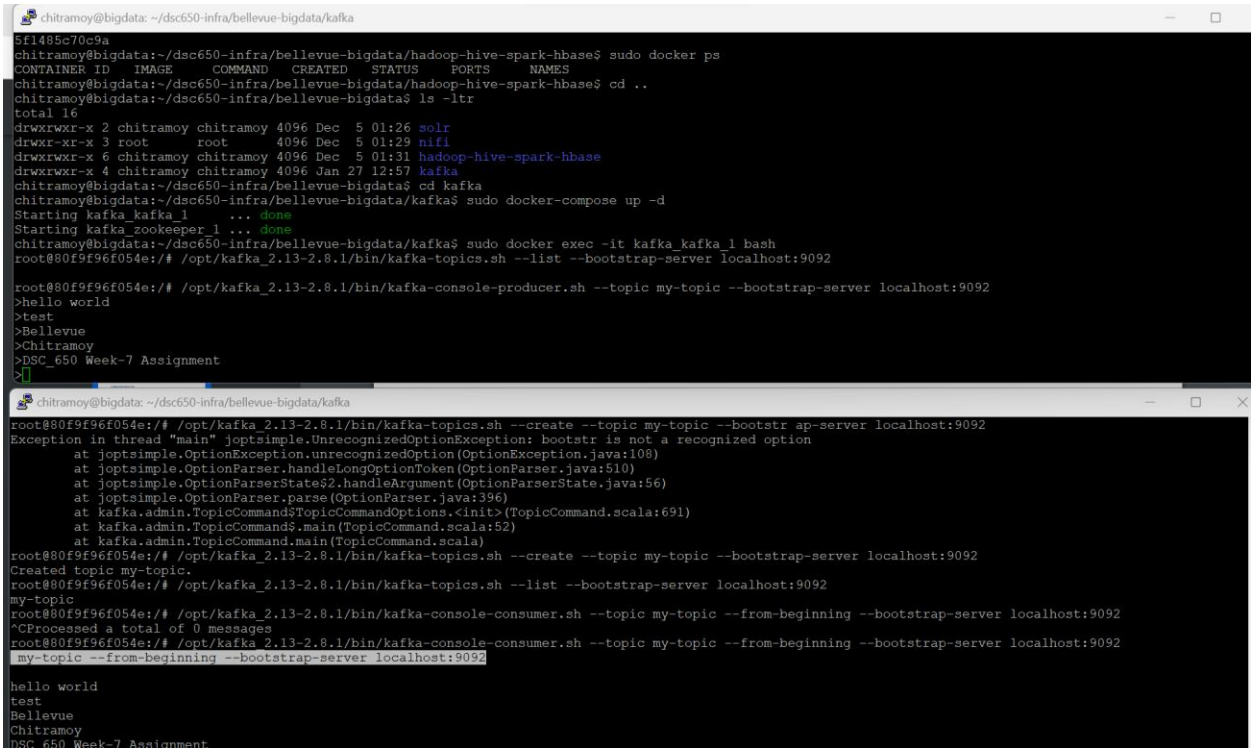
```
root@80f9f96f054e:/# /opt/kafka_2.13-2.8.1/bin/kafka-topics.sh --create --topic my-topic --bootstrap-server localhost:9092
Created topic my-topic.
root@80f9f96f054e:/# /opt/kafka_2.13-2.8.1/bin/kafka-topics.sh --list --bootstrap-server localhost:9092
my-topic
root@80f9f96f054e:/#
```

Exercise 3: In the first terminal, start a Kafka consumer.

/opt/kafka_2.13-2.8.1/bin/kafka-console-consumer.sh --topic my-topic --from-beginning --bootstrap-server localhost:9092

Exercise 4: In the second terminal, start a Kafka producer:

/opt/kafka_2.13-2.8.1/bin/kafka-console-producer.sh --topic my-topic --bootstrap-server localhost:9092

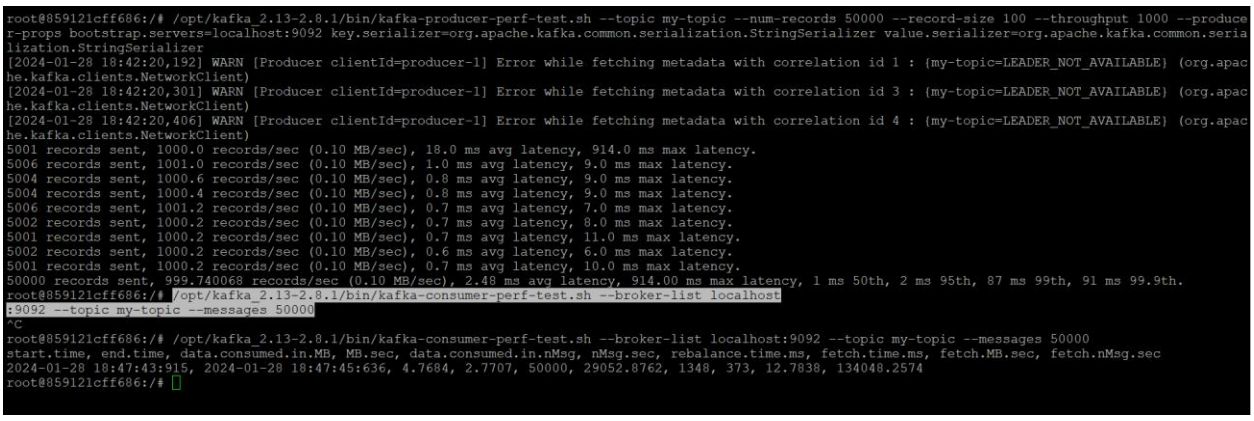


Exercise 5: Run a performance test on the producer using the Kafka producer performance test script with provided arguments.

/opt/kafka_2.13-2.8.1/bin/kafka-producer-perf-test.sh --topic my-topic --num-records 50000 --record-size 100 --throughput 1000 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer

Exercise 6: Following the producer test, run a consumer performance test on ‘my-topic’:

/opt/kafka_2.13-2.8.1/bin/kafka-consumer-perf-test.sh --broker-list localhost:9092 --topic my-topic --messages 50000



Result Description:

The performance test started on January 28, 2024, at 18:47:43:915 and ended at 18:47:45:636.During the test, 4.7684 megabytes of data were consumed at a rate of 2.7707 megabytes per second. A total of 50,000 messages were consumed at a rate of 29,052.8762 messages per second. The rebalancing process took 1,348 milliseconds. The time taken for message fetching was 373 milliseconds.The rate of fetching data was 12.7838 megabytes per second. The rate of fetching messages was 134,048.2574 messages per second.
This output provides insights into the performance characteristics of the Kafka system during the specific test scenario, including data consumption rates, rebalancing times, and message fetching metrics.

Exercise 7: Create a topic, this time partitioned and replicated across all three Kafka instances:

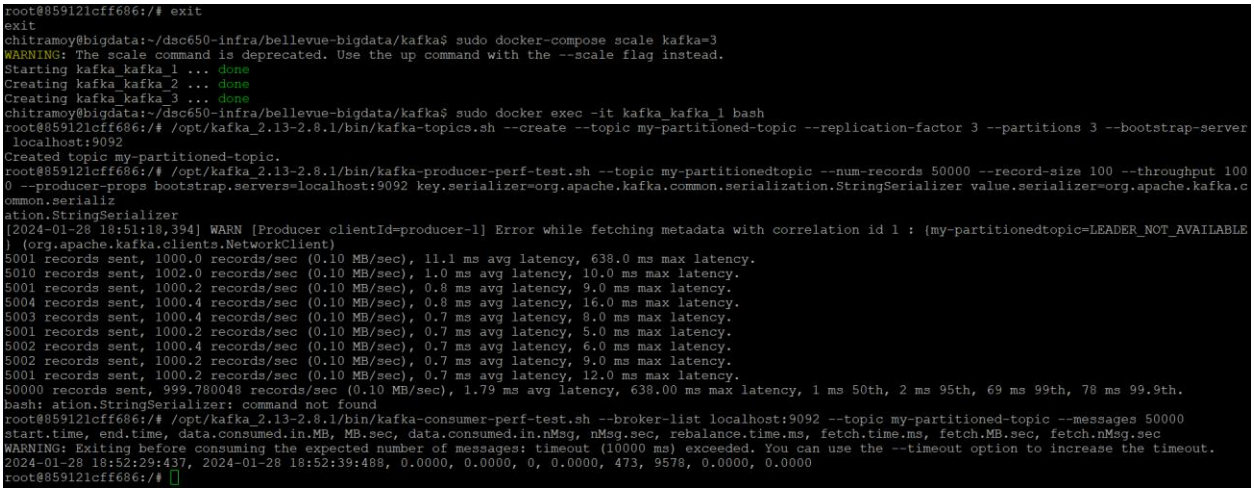
/opt/kafka_2.13-2.8.1/bin/kafka-topics.sh --create --topic my-partitioned-topic --replication-factor 3 --partitions 3 --bootstrap-server localhost:9092

Exercise 8: Conduct the producer and consumer performance tests on the new topic, observing differences:

/opt/kafka_2.13-2.8.1/bin/kafka-producer-perf-test.sh --topic my-partitioned-topic --num-records 50000 --record-size 100 --throughput 1000 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer

Followed by:

/opt/kafka_2.13-2.8.1/bin/kafka-consumer-perf-test.sh --broker-list localhost:9092 --topic my-partitioned-topic --messages 50000



Result Description:

These exercises involve creating a partitioned and replicated Kafka topic and then performing producer and consumer performance tests on that topic. The partitioning and replication configurations are crucial for scalability, fault tolerance, and ensuring high availability in a distributed Kafka environment. It is vital to comprehend the proper way to partition data, as this knowledge is essential for enhancing the performance and reliability of your Kafka configuration. The performance tests help assess the throughput and efficiency of the Kafka producers and consumers under specified conditions. For larger data sets, partitioning proves to be significantly more optimal, particularly in terms of parallel computing and fault tolerance.