

 Code

 Issues

 Pull requests 1

 Actions

 Projects

 Wiki

 Sec

 taskC-auth ▾

[CS3219_assignments](#) / [taskD](#) /

...

This branch is 4 commits ahead of main.

 #4



rtshkmr Add particulars to Task D README 

 2 minutes ago

 History

..

 resources

yesterday

 README.md

2 minutes ago

 docker-compose.yml

yesterday

 README.md



Task D: Pub-Sub Messaging via Kafka

Student Name: Ritesh Kumar

Matriculation Number: A0201829H

[GitHub Repository](#)

Background

0. Kafka is the message broker, groups messages into topics and handles topics.

Both pub and sub can be done

- here's a [thorough introduction to kafka](#)

1. Kafka ZooKeeper.

- here's a [primer on it](#)

Deliverables

The configuration for the servers is at [here](#). This indicates the configuration for a single zookeeper and 3 kafka nodes.

Simple Pub-Sub

1. Spin up the servers via `docker-compose up` (I prefer to open up non-detached instance docker so that the logs can be easily seen). Verify that the container is up by running `docker ps` to see this output:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
50d2350d44c0	confluentinc/cp-kafka:latest	/etc/confluent/docker/	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:29992->9092/tcp, 0.0.0.0:29993->9093/tcp
29092-29093/c0	confluentinc/cp-zookeeper:latest	/etc/confluent/docker/	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:29992->9092/tcp, 0.0.0.0:29993->9093/tcp
29092-29093/c1	confluentinc/cp-zookeeper:latest	/etc/confluent/docker/	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:29992->9092/tcp, 0.0.0.0:29993->9093/tcp
50d2350d44c1	confluentinc/cp-kafka:latest	/etc/confluent/docker/	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:49992->9092/tcp, 0.0.0.0:49993->9093/tcp
29092-49992/c1	confluentinc/cp-zookeeper:latest	/etc/confluent/docker/	3 minutes ago	Up 12 seconds	2888/tcp, 0.0.0.0:2181->2181/tcp, 0.0.0.0:4000->4000/tcp
29092-49992/c2	confluentinc/cp-zookeeper:latest	/etc/confluent/docker/	3 minutes ago	Up 12 seconds	2888/tcp, 0.0.0.0:2181->2181/tcp, 0.0.0.0:4000->4000/tcp
50d2350d44c2	src_nodejs_1	/bin/sh -c /usr/bin/wait-for-it	4 weeks ago	Up About an hour	0.0.0.0:4000->4000/tcp, 0.0.0.0:4000->4000/tcp

2. Now we enter one of the nodes in order to carry out actions within it:

- i. `docker exec -it taskd_kafka-1_1 bash` interactively spawns a bash shell within that node.

3. Create a dummy topic like so:

```
kafka-topics --create --zookeeper zoo1:2181 --replication-factor 2 --partitions 1  
# output: Created topic dummy-topic.
```

This creates a topic which can be produced to and consumed from.

4. Now, produce a message to the `dummy-topic`:

- i. Enter the producer console in order to publish messages to that topic:

```
kafka-console-producer --broker-list kafka-1:9092 --topic dummy-topic
```

- ii. On a separate shell, enter the consumer console in order to consume messages from that topic:

```
kafka-console-consumer --bootstrap-server kafka1:9092 --topic dummy-topic
```

- iii. type any message in the produce and see it appear in the consumer terminal:

The screenshot shows two terminal windows side-by-side. The left window is a Windows PowerShell session where a producer creates a topic named 'dummy-topic' with 3 partitions and replication factor 2. The right window is another Windows PowerShell session where a consumer reads from the 'dummy-topic' and prints the received messages to the console.

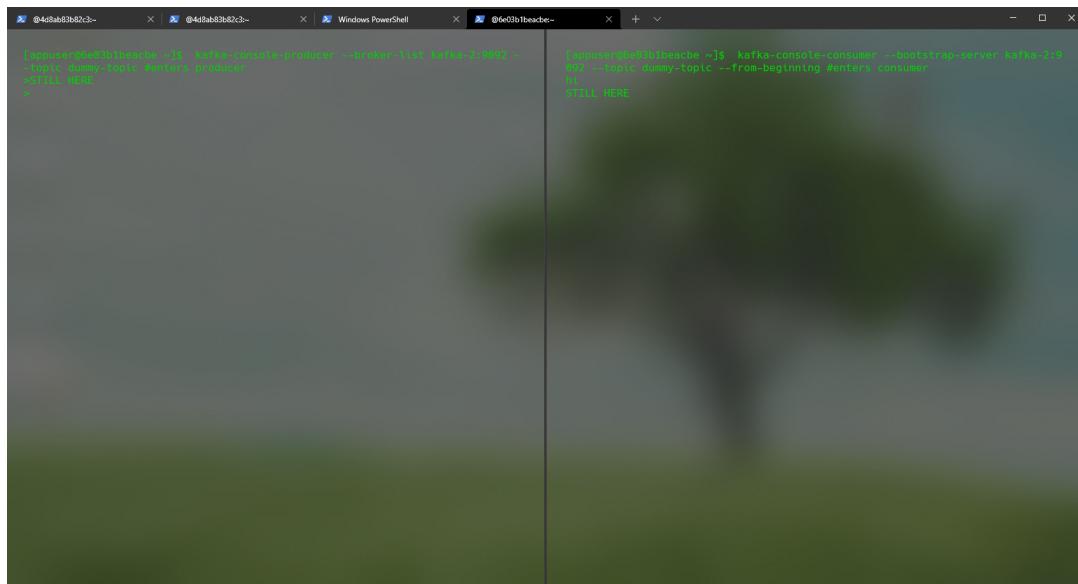
```
[root@blackmaiden:/mnt/c/Us] ~]$ kafka-topics --create --zookeeper localhost:2181 --replication-factor 3 --partitions 3 --topic dummy-topic
Topic: dummy-topic    TopicId: 8f01c6a914e1-Ky1D9fYv PartitionCount: 3    ReplicationFactor: 2    Configs:
  Topic: dummy-topic    Partition: 0    Leader: 3    Replicas: 3,1    Isr: 3,1
  Topic: dummy-topic    Partition: 1    Leader: 1    Replicas: 1,2    Isr: 1,2
  Topic: dummy-topic    Partition: 2    Leader: 2    Replicas: 2,3    Isr: 2,3
[roo
[root@blackmaiden:/mnt/c/Us] ~]$ kafka-console-consumer --bootstrap-server localhost:9092 --topic dummy-topic --from-beginning
Unregistered consumer
echo after me ya
>I see you brudder
>
```

Failover Demo

The screenshot shows a single terminal window with two commands. The first command runs `kafka-topics --zookeeper zoo1:2181 --describe --topic dummy-topic` and shows the current topic configuration with three partitions and a leader assigned to node 3. The second command runs the same command after killing the leader node (node 3) and shows that the leadership has been transferred to node 1.

```
[root@blackmaiden:83b82c3] ~]$ kafka-topics --zookeeper zoo1:2181 --describe --topic dummy-topic
Topic: dummy-topic    TopicId: 8f01c6a914e1-Ky1D9fYv PartitionCount: 3    ReplicationFactor: 2    Configs:
  Topic: dummy-topic    Partition: 0    Leader: 3    Replicas: 3,1    Isr: 3,1
  Topic: dummy-topic    Partition: 1    Leader: 1    Replicas: 1,2    Isr: 1,2
  Topic: dummy-topic    Partition: 2    Leader: 2    Replicas: 2,3    Isr: 2,3
[root@blackmaiden:83b82c3] ~]$ echo "after killing node 3"
after killing node 3
[root@blackmaiden:83b82c3] ~]$ kafka-topics --zookeeper zoo1:2181 --describe --topic dummy-topic
Topic: dummy-topic    TopicId: 8f01c6a914e1-Ky1D9fYv PartitionCount: 3    ReplicationFactor: 2    Configs:
  Topic: dummy-topic    Partition: 0    Leader: 1    Replicas: 3,1    Isr: 1
  Topic: dummy-topic    Partition: 1    Leader: 1    Replicas: 1,2    Isr: 1,2
  Topic: dummy-topic    Partition: 2    Leader: 2    Replicas: 2,3    Isr: 2
[root@blackmaiden:83b82c3] ~]$
```

1. Run `kafka-topics --zookeeper zoo1:2181 --describe --topic dummy-topic` to see the current description of the topic, which shows the master node
2. Kill the Leader node:
 - i. in the host shell, run `docker container kill taskd_kafka-3_1`
 - ii. here (node 3) will change the state like in the screenshot above, allocating other nodes.
3. The pub-sub functionality still works if we use another node:



The image shows a terminal window with two tabs open, each displaying a command-line interface for Apache Kafka's console tools.

The left tab contains the following command:

```
[kafka@centos7 ~]$ kafka-console-producer --bootstrap-server kafka01:9092 --topic many --property producer.interceptor.classes=io.confluent.monitoring.clients.interceptor.MonitoringInterceptor
```

The right tab contains the following command:

```
[kafka@centos7 ~]$ kafka-console-consumer --bootstrap-server kafka01:9092 --topic many --from-beginning --property consumer.interceptor.classes=io.confluent.monitoring.clients.interceptor.MonitoringInterceptor
```