

 Code

 Issues

 Pull requests

 Actions

 Projects

 Wiki

 Security

 taskD-kafka ▾

...

CS3219_assignments / taskD / README.md



rtshkmr Complete Task D ✓

 History

1 contributor

57 lines (42 sloc) | 2.4 KB

...

Task D: Pub-Sub Messaging via Kafka

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:

```
ntech@blackmaiden:/mnt/c/Us
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
346241861f0c	confluentinc/cp-kafka:latest	/etc/confluent/docker/run	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:9091->9092/tcp, ...
33900212005772	cp-kafka_2_1	/etc/confluent/docker/run	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:9091->9092/tcp, ...
334411665072	confluentinc/cp-kafka:latest	/etc/confluent/docker/run	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:9091->9092/tcp, ...
3299021200572	cp-kafka_2_1	/etc/confluent/docker/run	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:9091->9092/tcp, ...
3209021200572	confluentinc/cp-kafka:latest	/etc/confluent/docker/run	3 minutes ago	Up 9 seconds	9092/tcp, 0.0.0.0:9091->9092/tcp, ...
4090021200572	cp-kafka_2_1	/etc/confluent/docker/run	3 minutes ago	Up 12 seconds	2888/tcp, 0.0.0.0:2181->2181/tcp, ...
4110021200572	cp-kafka_2_1	/etc/confluent/docker/run	3 minutes ago	Up 12 seconds	2888/tcp, 0.0.0.0:2181->2181/tcp, ...
4149021200572	cp-kafka_2_1	/etc/confluent/docker/run	4 weeks ago	Up About an hour	0.0.0.0:4000->4000/tcp, 0.0.0.0:4000->4000/tcp

```
PS C:\Users\ntech\Us
```

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-to
```

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

The screenshot shows three terminal windows side-by-side:

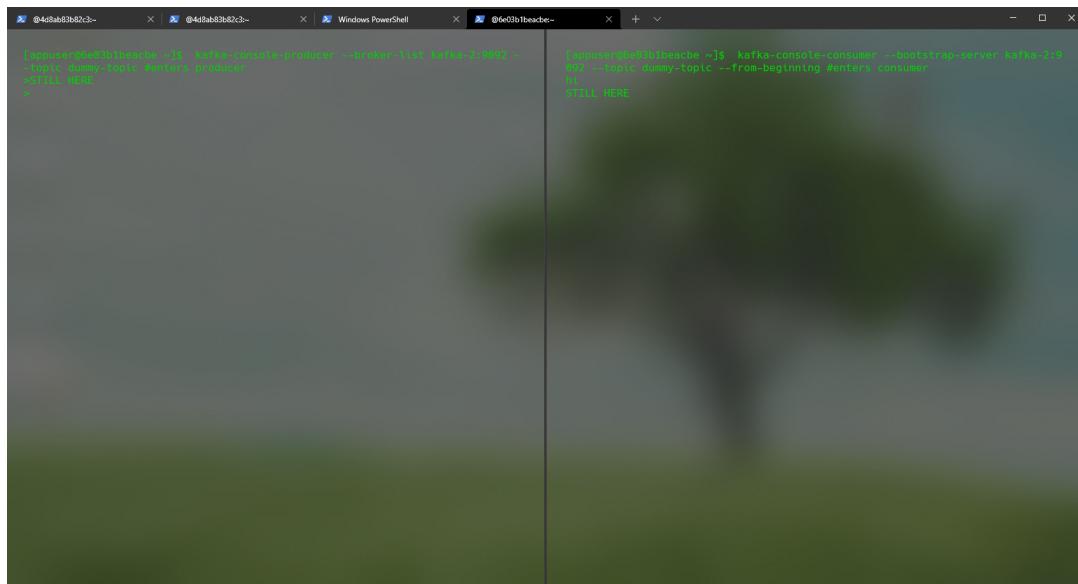
- Terminal 1 (Left):** A user creates a topic named "dummy-topic" with 3 partitions and replication factor 1 using the command: `kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic dummy-topic`. The output shows the topic was created successfully.
- Terminal 2 (Middle):** A user creates a producer and sends a message to the "dummy-topic" on broker 1 (localhost:9092) using the command: `kafka-console-producer --broker-list localhost:9092 --topic dummy-topic`. The message "echo after me ya" is sent.
- Terminal 3 (Right):** A user creates a consumer and consumes from the "dummy-topic" on broker 1 (localhost:9092) using the command: `kafka-console-consumer --bootstrap-server kafka-1:9092 --topic dummy-topic --from-beginning`. The response "I see you brudder" is received.

Failover Demo

The screenshot shows two terminal windows:

- Terminal 1 (Left):** A user runs `kafka-topics --zookeeper zoo1:2181 --describe --topic dummy-topic` to check the current topic configuration. The output shows the master node (node 3) as the leader for all partitions.
- Terminal 2 (Right):** A user runs `echo "after killing node 3"` to simulate a node failure.
- Terminal 3 (Left):** After killing the leader node, the user runs `kafka-topics --zookeeper zoo1:2181 --describe --topic dummy-topic` again. The output shows the master node (node 3) is no longer listed as a leader, and node 1 has become the new leader for all partitions.

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
```