Docker部署zookeeper和kafka

1. 拉取镜像

- 拉取已经配置好的镜像

```
docker pull 879314144/mykafka:v2
```

2. 启动镜像和容器

- 启动镜像

  ```
  docker run -itd -p 8088:8088 -p 9870:9870 -p 9000:9000 -p 9090:9090 --privileged=true 879314144/mykafka:v2 /sbin/init
  ```

　　将可能要用到的端口映射到宿主机上，8088: yarn web默认端口，9870: hdfs web默认端口，9000: 打算用来配置kafka web UI，9090: 打算用来配置zookeeper web UI

- 启动容器
  - 每次启动镜像会给我们一个容器，可通过以下命令查看所有容器

    ```
    docker ps -a
    ```

    ```
    CONTAINER ID   IMAGE                  COMMAND         CREATED        STATUS          PORTS
                                                                                                         NAMES
    03c8e31f43c4   879314144/mykafka:v1   "/sbin/init"    2 hours ago    Up 21 minutes   0.0.0.0:2181-2183->2181-2183/tcp, 0.0.0.0:
    8088->8088/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:9090->9090/tcp, 0.0.0.0:9092-9094->9092-9094/tcp, 0.0.0.0:9870->9870/tcp   compassionate_tu
    ```

    我们可以看容器是基于哪个镜像的，这里我们要启动的容器ID为03c8e31f43c4，用以下命令启动

    ```
    docker start 03c8e31f43c4 //也可以直接用ID前几位符号，只要能和其他镜像区分开来就行，比如这里可以用docker start 03c
    ```

    然后可以用下面命令进入容器，进入容器后可以用命令行进行交互

    ```
    docker exec -it 03c /bin/bash
    ```

3. 启动zookeeper和kafka伪集群

- 为什么叫伪集群?
  - 三个zookeeper节点和三个kafka节点都在同一个容器内（或者说在同一台机器上）
  - 三个zookeeper节点的client端口分别是2181, 2182, 2183，三个kafka的生产消费端口分别是9092, 9093, 9094
- 因为容器里的zookeeper和kafka都已经配置好了，为了方便，我这里直接写了一个脚本来启动所有节点，并进入交互式模式

```
#!/bin/zsh
docker start 03c //别忘记改为自己的容器的ID
docker exec -it 03c /myStart.sh
docker exec -it 03c /usr/zk1/apache-zookeeper-3.8.0-bin/bin/zkServer.sh start
docker exec -it 03c /usr/zk2/apache-zookeeper-3.8.0-bin/bin/zkServer.sh start
docker exec -it 03c /usr/zk3/apache-zookeeper-3.8.0-bin/bin/zkServer.sh start
docker exec 03c /usr/kafka_2.13-3.2.1/bin/kafka-server-start.sh
/usr/kafka_2.13-3.2.1/config/server-1.properties &
docker exec 03c /usr/kafka_2.13-3.2.1/bin/kafka-server-start.sh
/usr/kafka_2.13-3.2.1/config/server-2.properties &
docker exec 03c /usr/kafka_2.13-3.2.1/bin/kafka-server-start.sh
/usr/kafka_2.13-3.2.1/config/server-3.properties &
docker exec -it 03c /bin/bash
```

- 如果是Mac OS，则用chmod +x把文件变为可执行文件；如果是Windows，请自己百度，搜了一下可能是把后缀变为bat；如果是linux，后缀改为sh。
- 这里myStart.sh文件是容器里的一个脚本，脚本内容如下

```
#!/bin/sh
echo "127.0.0.1 zk1 zk2 zk3" >> /etc/hosts
```

主要是使主机ip有对应的名字，因为在配置的时候用了主机名，没用ip地址。不知道为啥写入/etc/hosts里的内容每次启动都要写一遍，并没有持久化保存。

- 等所有节点启动完成后，用jps命令查看正在运行的java进程

```
66 QuorumPeerMain
306 Kafka
229 QuorumPeerMain
2822 ConsoleProducer
167 QuorumPeerMain
3993 Jps
3819 jar
333 Kafka
367 Kafka
```

如果有三个QuorumPeerMain和三个Kafka则说明启动成功

4. Flink生产者

- 这里我先放命令行模式，到时候再看看java代码怎么写

```
cd /usr/kafka_2.13-3.2.1/bin/
./kafka-topics.sh --create \
                  --bootstrap-server localhost:9092 \   //这行的localhost可以改为
zk1(zk2, zk3也可以)或者ip地址127.0.0.1
                  --replication-factor 1 \
                  --partitions 1  \
                  --topic flink-stream-in-topic //topic名字
```

启动生产者

```
# cd /usr/kafka_2.13-3.2.1/bin/
./kafka-console-producer.sh --broker-list localhost:9092 --topic flink-stream-in-topic //进入console producer模式，即在console里进行输入。可以启动多个生产者往多个kafka节点写入东西，需要打开多个终端，进入该容器，然后只要修改上面命令中的端口号就行。
```

5. 消费者

- 消费者直接写了java代码

```
public class KafkaConsumer {
    public static void main(String[] args) {
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        Properties properties = new Properties();
        properties.setProperty("bootstrap.servers",
"localhost:9092,localhost:9093,localhost:9094");
        DataStream<String> stream = env
                .addSource(new FlinkKafkaConsumer010<>("flink-stream-in-topic",
new SimpleStringSchema(), properties));
        stream.print();
        try {
            env.execute("Flink Streaming");
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

这里监听了三个kafka节点端口，也就是可以从三个kafka节点进行消费

- 整个jar包我放在github上，需要copy到容器中执行
  - 从本地copy到容器内的命令

    ```
    docker cp 本地文件路径 容器ID:容器内路径
    ```

  - 容器内执行jar包

    ```
    java -jar xxx.jar
    ```

  - 可以看一下效果
    - 消费者，即运行的jar包

```
        ssl.trustmanager.algorithm = PKIX
        ssl.truststore.location = null
        ssl.truststore.password = null
        ssl.truststore.type = JKS
        value.deserializer = class org.apache.kafka.common.serialization.ByteArr
ayDeserializer

06:25:20,065 INFO  org.apache.kafka.common.utils.AppInfoParser
 - Kafka version : 0.10.2.1
06:25:20,065 INFO  org.apache.kafka.common.utils.AppInfoParser
 - Kafka commitId : e89bffd6b2eff799
06:25:20,098 INFO  org.apache.kafka.clients.consumer.internals.AbstractCoordinat
or  - Discovered coordinator localhost:9094 (id: 2147483645 rack: null) for grou
p .
2> a
2> b
[2022-08-06 06:29:05,346] INFO [BrokerToControllerChannelManager broker=0 name=f
orwarding] Node 2 disconnected. (org.apache.kafka.clients.NetworkClient)
                                                                        [2022-08
-06 06:34:51,398] INFO [BrokerToControllerChannelManager broker=1 name=forwardin
g] Node 2 disconnected. (org.apache.kafka.clients.NetworkClient)
                                                              2> c
2> d
```

消费了字符串a，字符串b，字符串c，字符串d

- 生产者，这里我开了两个

```
Last login: Sat Aug  6 14:06:32 on ttys000
 ~/ docker cp /Users/apple/IdeaProjects/flink/target/flink-1.0-SNAPSHOT.jar 03c
:/
 ~/ docker exec -it 03c /bin/bash                                  (3s)[14:23:30]
[root@03c8e31f43c4 /]# cd /usr/kafka_2.13-3.2.1/bin
[root@03c8e31f43c4 bin]# ./kafka-console-producer.sh --broker-list localhost:909
3 --topic flink-stream-in-topic
>b
>c
>
```

```
[root@03c8e31f43c4 bin]# ./kafka-console-producer.sh --broker-list localhost:909
2 --topic flink-stream-in-topic
>a
>d
>
```

一个是9092端口的kafka节点，另一个是9093端口的节点

输入了以后，上面的消费者都已经进行了消费