全部开发者教程

第14周-消息队列之Kafka从入门到小牛

1 初识Kafka

2 Kafka集群安装部署

3 Kafka使用初体验

4 Kafka核心扩展内容

5 Kafka核心之存储和容错机制

6 Kafka生产消费者实战

7 Kafka技巧篇

8 Kafka小试牛刀实战篇

9 Kafka核心复盘

第15周-极速上手内存数据库Redis

徐老师 • 更新于 2020-09-25

上一节 2 Kafka集群安装… 4 Kafka核心扩展… 下一节

Kafka中Topic的操作

kafka集群安装好了以后我们就想向kafka中添加一些数据
想要添加数据首先需要创建topic
那接下来看一下针对topic的一些操作

- 新增Topic：指定2个分区，2个副本，注意：副本数不能大于集群中Broker的数量

因为每个partition的副本必须保存在不同的broker，否则没有意义，如果partition的副本都保存在同一个broker，那么这个broker挂了，则partition数据依然会丢失
在这里我使用的是3个节点的kafka集群，所以副本数我就暂时设置为2，最大可以设置为3
如果你们用的是单机kafka的话，这里的副本数就只能设置为1了，这个需要注意一下

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --create --zookeeper l
Created topic hello.
```

-查询Topic：查询Kafka中的所有Topic列表以及查看指定Topic的详细信息

查询kafka中所有的topic列表

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --list --zookeeper loc
hello
```

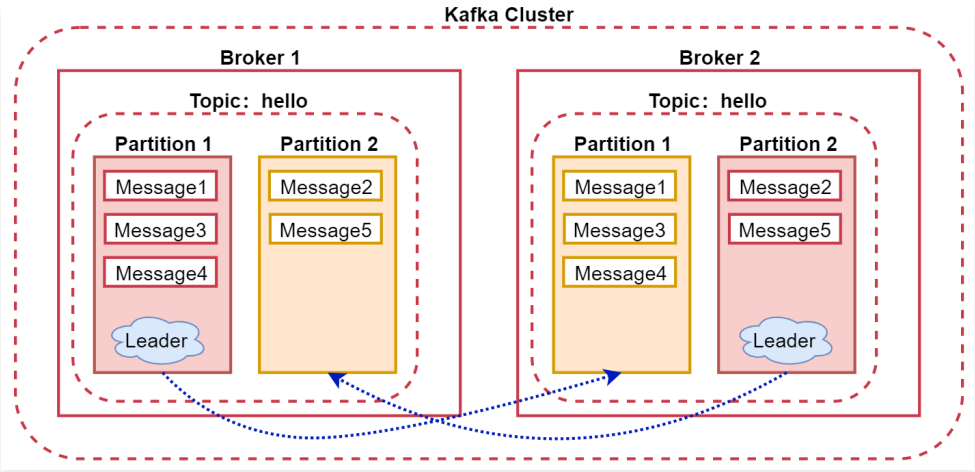
查看指定topic的详细信息

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --describe --zookeeper
Topic: hello    PartitionCount: 2       ReplicationFactor: 2    Configs:
        Topic: hello    Partition: 0    Leader: 2       Replicas: 2,0      Is
        Topic: hello    Partition: 1    Leader: 0       Replicas: 0,1      Is
```

第一个行显示指定topic所有partitions的一个总结
PartitionCount：表示这个Topic一共有多少个partition
ReplicationFactor：表示这个topic中partition的副本因子是几
Config：这个表示创建Topic时动态指定的配置信息，在这我们没有额外指定配置信息

下面每一行给出的是一个partition的信息，如果只有一个partition，则只显示一行。
Topic：显示当前的topic名称
Partition：显示当前topic的partition编号
Leader：Leader partition所在的节点编号，这个编号其实就是broker.id的值，
来看这个图：



这个图里面的hello这个topic有两个partition，其中partition1的leader所在的节点是broker1，partition2的leader所在的节点是broker2

Replicas：当前partition所有副本所在的节点编号【包含Leader所在的节点】，如果设置多个副本的话，这里会显示多个，不管该节点是否是Leader以及是否存活。
Isr：当前partition处于同步状态的所有节点，这里显示的所有节点都是存活状态的，并且跟Leader同步的(包含Leader所在的节点)

所以说Replicas和Isr的区别就是
如果某个partition的副本所在的节点宕机了，在Replicas中还是会显示那个节点，但是在Isr中就不会显示了，Isr中显示的都是处于正常状态的节点。

- 修改Topic：修改Topic的partition数量，只能增加

为什么partition只能增加?
因为数据是存储在partition中的，如果可以减少partition的话，那么partition中的数据就丢了

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --alter --zookeeper lo
WARNING: If partitions are increased for a topic that has a key, the partitio
Adding partitions succeeded!
```

修改之后再来查看一下topic的详细信息

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --describe --zookeeper
Topic: hello    PartitionCount: 5       ReplicationFactor: 2    Configs:
        Topic: hello    Partition: 0    Leader: 2       Replicas: 2,0       I
        Topic: hello    Partition: 1    Leader: 0       Replicas: 0,1       -
        Topic: hello    Partition: 2    Leader: 1       Replicas: 1,2
        Topic: hello    Partition: 3    Leader: 2       Replicas: 2,1
        Topic: hello    Partition: 4    Leader: 0       Replicas: 0,2
```

- 删除Topic：删除Kafka中的指定Topic

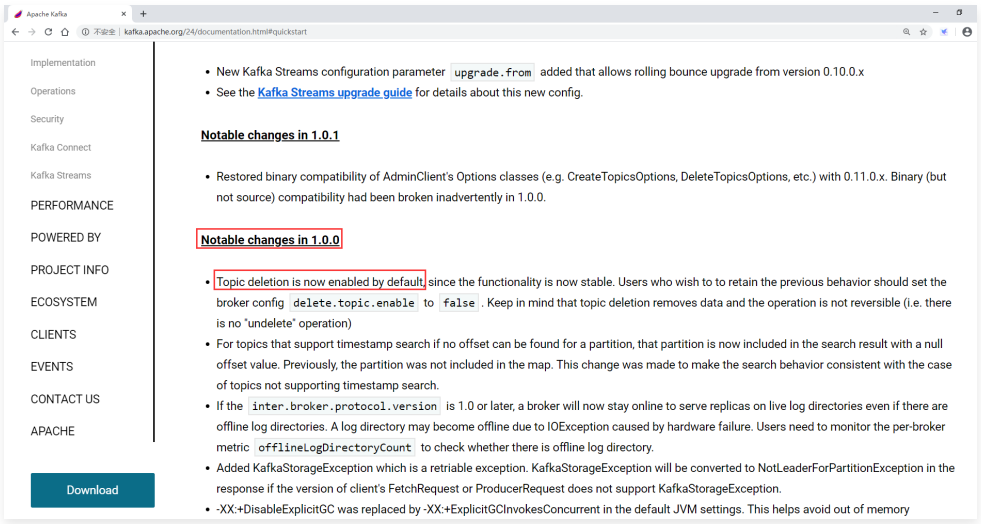
代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --delete --zookeeper l
Topic hello is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

> 注意：Kafka从1.0.0开始默认开启了删除操作，之前的版本只会把Topic标记为删除状态，需要设置delete.topic.enable为true才可以真正删除

如果不想开启删除功能，可以设置 `delete.topic.enable` 为 `false` ，这样删除topic的时候只会把它标记为删除状态，此时这个topic依然可以正常使用。

`delete.topic.enable` 可以配置在 `server.properties` 文件中



Kafka中的生产者和消费者

前面我们学习了Kafka中的topic的创建方式，下面我们可以向topic中生产数据以及消费数据了
生产数据需要用到生产者
消费数据需要用到消费者

kafka默认提供了基于控制台的生产者和消费者，方便测试使用
生产者： `bin/kafka-console-producer.sh`
消费者： `bin/kafka-console-consumer.sh`

先来看一下如何向里面生产数据
直接使用kafka提供的基于控制台的生产者
先创建一个topic【5个分区，2个副本】：

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --create --zookeeper l
Created topic hello.
```

向这个topic中生产数据
`broker-list`： kafka的服务地址[多个用逗号隔开]、
`topic`： topic名称

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-console-producer.sh --broker-lis
>hehe
```

下面来创建一个消费者消费topic中的数据
`bootstrap-server`： kafka的服务地址
`topic` :具体的topic

代码块

```
ootstrap-
```

意见反馈 收藏教程 标记书签

发现消费不到刚才生产的数据，为什么呢?
因为kafka的消费者默认是消费最新生产的数据，如果想消费之前生产的数据需要添加一个参数–from-beginning，表示从头消费的意思

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-console-consumer.sh --bootstrap-
hehe
```

案例：QQ群聊天

通过kafka可以模拟QQ群聊天的功能，我们来看一下
首先在kafka中创建一个新的topic，可以认为是我们在QQ里面创建了一个群，群号是88888888

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-topics.sh --create --zookeeper l
Created topic 88888888.
```

然后我把你们都拉到这个群里面，这样我在群里面发消息你们就都能收到了
在bigdata02和bigdata03上开启消费者，可以认为是把这两个人拉到群里面了

代码块

```
[root@bigdata02 kafka_2.12-2.4.1]# bin/kafka-console-consumer.sh --bootstrap-
```

代码块

```
[root@bigdata03 kafka_2.12-2.4.1]# bin/kafka-console-consumer.sh --bootstrap-
```

然后我在bigdata01上开启生产者发消息，这样bigdata02和bigdata03都是可以收到的。
这样就可以认为在群里的人都能收到我发的消息，类似于发广播。
这个其实主要利用了kafka中的多消费者的特性，每个消费者都可以消费到相同的数据

代码块

```
[root@bigdata01 kafka_2.12-2.4.1]# bin/kafka-console-producer.sh --broker-lis
>hello everyone
```

2 Kafka集群安装部署 上一节 下一节 4 Kafka核心扩展内容

我要提出意见反馈

意见反馈 收藏教程 标记书签