暨南大学本科实验报告专用纸

课程名称 云计算实验 成绩评定

实验项目名称 分布式消息处理中间件 Kafaka 的部署与使用

指导教师 梁倬骞、魏林锋

实验项目编号 0806030806 实验项目类型 验证型 实验地点 N116

学生姓名 陈宇 学号 2020101642

学院 信息科学技术 系 计算机 专业 计算机科学与技术

实验时间 2022 年 11 月 2 日 上午～11 月 2 日 中午 温度 ℃湿度

6.1 实验目的

1）理解 Kafka 工作原理。

2）通过实验掌握分布式消息中间件 Kafka 的安装过程。

3）熟悉分布式消息中间件 Kafka 的使用。

6.2 实验内容

1）完成分布式消息中间件 Kafka 的安装。

2）完成在多台机器上启用多个 broker 的操作。

3）完成在一台机器上启用多个 broker 的操作。

4）实现 Kafka 的生成和消费。

6.3 实验环境

已经配置完成的 Hadoop 伪分布式或完全分布式环境。环境配置如下：

Hadoop01：192.168.24.91 Hadoop02：192.168.24.92 Hadoop03：192.168.24.93

管理员用户：root / admin@1

Hadoop 用户：hadoop / hadoop

6.4 实验步骤

6.4.1 在多台机器上启用多个 broker

1、使用 xftp 工具将 kafka_2.12-2.2.0.tgz 文件上传到服务器上，解压 kafka_2.12-2.2.0.tgz 文件，并重命名 kafka 文件夹。命令如下：

```
[root@master ~]# tar -zvxf kafka_2.12-2.2.0.tgz -C /usr/
[root@master ~]# mv /usr/kafka_2.12-2.2.0/ /usr/kafka
```

2、添加环境变量，并使其生效。命令如下：

```
[root@master ~]# vi /etc/profile
export KAFKA_HOME=/usr/kafka

export PATH=$KAFKA_HOME/bin:$PATH
[root@master ~]# source /etc/profile
```

3、进入/usr/kafka/config 目录。命令如下：

```
[root@master ~]# cd /usr/kafka/config
```

4、修改 server.pronperties。命令如下：

```
[root@master config] # vim server.pronperties
#broker 的全局唯一编号，不能重复
broker.id=0
#用来监听链接的端口，producer 或 consumer 将在此端口建立连接
listeners=PLAINTEXT://master:9092
#处理网络请求的线程数量
num.network.threads=3
#用来处理磁盘 IO 的线程数量
num.io.threads=8
#发送套接字的缓冲区大小
socket.send.buffer.bytes=102400
```

```
#接受套接字的缓冲区大小
socket.receive.buffer.bytes=102400
#请求套接字的缓冲区大小
socket.request.max.bytes=104857600
#kafka 消息存放的路径
log.dirs= /usr/kafka/logs
#topic 在当前 broker 上的分片个数
num.partitions=2
#用来恢复和清理 data 下数据的线程数量
num.recovery.threads.per.data.dir=1
#segment 文件保留的最长时间，超时将被删除
log.retention.hours=168

#滚动生成新的 segment 文件的最大时间
log.roll.hours=168
#日志文件中每个 segment 的大小，默认为 1G
log.segment.bytes=1073741824
#周期性检查文件大小的时间
log.retention.check.interval.ms=300000
#日志清理是否打开
log.cleaner.enable=true
#broker 需要使用 zookeeper 保存 meta 数据
zookeeper.connect=master:2181,slave1:2181,slave2:2181
#zookeeper 链接超时时间
zookeeper.connection.timeout.ms=6000
#partion buffer 中，消息的条数达到阈值，将触发 flush 到磁盘
log.flush.interval.messages=10000
#消息 buffer 的时间，达到阈值，将触发 flush 到磁盘
log.flush.interval.ms=3000
#删除 topic 需要 server.properties 中设置 delete.topic.enable=true 否则只是标记删除
delete.topic.enable=true
#此处的 host.name 为本机 IP(重要),如果不改,则客户端会抛出:Producerconnection to localhost:9092 unsuccessful 错误!
host.name=master
```

5、修改 producer.properties。命令如下：

```
[root@master config] # vim producer.properties
#指定 kafka 节点列表，用于获取 metadata，不必全部指定
metadata.broker.list=master:9092,slave1:9092,slave2:9092
# 指定分区处理类。默认 kafka.producer.DefaultPartitioner，表通过 key 哈希到对应分区
#partitioner.class=kafka.producer.DefaultPartitioner
# 是否压缩，默认 0 表示不压缩，1 表示用 gzip 压缩，2 表示用 snappy 压缩。压缩后消息中
```

会有头来指明消息压缩类型，故在消费者端消息解压是透明的无需指定。
compression.codec=none
# 指定序列化处理类
serializer.class=kafka.serializer.DefaultEncoder
# 如果要压缩消息，这里指定哪些 topic 要压缩消息，默认 empty，表示不压缩。
#compressed.topics=
# 设置发送数据是否需要服务端的反馈,有三个值 0,1,-1
# 0: producer 不会等待 broker 发送 ack
# 1: 当 leader 接收到消息之后发送 ack
# -1: 当所有的 follower 都同步消息成功后发送 ack.
request.required.acks=0
#在向 producer 发送 ack 之前,broker 允许等待的最大时间 ，如果超时,broker 将会向 producer 发送一个 error ACK.意味着上一次消息因为某种原因未能成功(比如 follower 未能同步成功)
request.timeout.ms=10000
# 同步还是异步发送消息，默认“sync”表同步，"async"表异步。异步可以提高发送吞吐量,也意味着消息将会在本地 buffer 中,并适时批量发送，但是也可能导致丢失未发送过去的消息
producer.type=sync
# 在 async 模式下,当 message 被缓存的时间超过此值后,将会批量发送给 broker,默认为 5000ms
# 此值和 batch.num.messages 协同工作.
queue.buffering.max.ms = 5000
# 在 async 模式下,producer 端允许 buffer 的最大消息量
# 无论如何,producer 都无法尽快的将消息发送给 broker,从而导致消息在 producer 端大量沉积
# 此时,如果消息的条数达到阀值,将会导致 producer 端阻塞或者消息被抛弃，默认为 10000
queue.buffering.max.messages=20000
# 如果是异步，指定每次批量发送数据量，默认为 200
batch.num.messages=500
# 当消息在 producer 端沉积的条数达到"queue.buffering.max.meesages"后
# 阻塞一定时间后,队列仍然没有 enqueue(producer 仍然没有发送出任何消息)
# 此时 producer 可以继续阻塞或者将消息抛弃,此 timeout 值用于控制"阻塞"的时间
# -1: 无阻塞超时限制,消息不会被抛弃
# 0:立即清空队列,消息被抛弃
queue.enqueue.timeout.ms=-1
# 当 producer 接收到 error ACK,或者没有接收到 ACK 时,允许消息重发的次数
# 因为 broker 并没有完整的机制来避免消息重复,所以当网络异常时(比如 ACK 丢失)
# 有可能导致 broker 接收到重复的消息,默认值为 3.
message.send.max.retries=3
# producer 刷新 topicmetada 的时间间隔,producer 需要知道 partitionleader 的位置,以及当前 topic 的情况
# 因此 producer 需要一个机制来获取最新的 metadata,当 producer 遇到特定错误时,将会立即刷新
#(比如 topic 失效,partition 丢失,leader 失效等),此外也可以通过此参数来配置额外的刷新机

制，默认值 600000

6、修改 consumer.properties。命令如下：

```
[root@master config]# vim consumer.properties
# zookeeper 连接服务器地址
zookeeper.connect=master:2181,slave1:2181, slave2:2181
# zookeeper 的 session 过期时间，默认 5000ms，用于检测消费者是否挂掉
zookeeper.session.timeout.ms=5000
#当消费者挂掉，其他消费者要等该指定时间才能检查到并且触发重新负载均衡
zookeeper.connection.timeout.ms=10000
# 指定多久消费者更新 offset 到 zookeeper 中。注意 offset 更新时基于 time 而不是每次获得的消息。一旦在更新 zookeeper 发生异常并重启，将可能拿到已拿到过的消息
zookeeper.sync.time.ms=2000
#指定消费组
group.id=xxx
# 当 consumer 消费一定量的消息之后,将会自动向 zookeeper 提交 offset 信息
# 注意 offset 信息并不是每消费一次消息就向 zk 提交一次,而是现在本地保存(内存),并定期提交,默认为 true
auto.commit.enable=true
# 自动更新时间。默认 60 * 1000
auto.commit.interval.ms=1000
# 当前 consumer 的标识,可以设定,也可以有系统生成,主要用来跟踪消息消费情况,便于观察
conusmer.id=xxx
# 消费者客户端编号，用于区分不同客户端，默认客户端程序自动产生
client.id=xxxx
# 最大取多少块缓存到消费者(默认 10)
queued.max.message.chunks=50
# 当有新的 consumer 加入到 group 时,将会 reblance,此后将会有 partitions 的消费端迁移到新的 consumer 上,如果一个 consumer 获得了某个 partition 的消费权限,那么它将会向 zk 注册 "Partition Owner registry"节点信息,但是有可能此时旧的 consumer 尚没有释放此节点，此值用于控制,注册节点的重试次数.
rebalance.max.retries=5
# 获取消息的最大尺寸,broker 不会像 consumer 输出大于此值的消息 chunk  每次 feth 将得到多条消息,此值为总大小,提升此值,将会消耗更多的 consumer 端内存
fetch.min.bytes=6553600
# 当消息的尺寸不足时,server 阻塞的时间,如果超时,消息将立即发送给 consumer
fetch.wait.max.ms=5000
socket.receive.buffer.bytes=655360
# 如果 zookeeper 没有 offset 值或 offset 值超出范围。那么就给个初始的 offset。有 smallest、largest、anything 可选，分别表示给当前最小的 offset、当前最大的 offset、抛异常。默认 largest
auto.offset.reset=smallest
# 指定序列化处理类
```

```
derializer.class=kafka.serializer.DefaultDecoder
```

7、创建/usr/kafka/logs/文件夹。命令如下：

```
[root@master conf]# mkdir /usr/kafka/logs
```

8、修改 flume 文件夹权限。命令如下：

```
[root@master conf]# chown -R hadoop:hadoop /usr/kafka
```

9、将 Kafka 复制到其他两台主机上。在其他两台机上修改 server.pronperties，更改 borker.id 和主机名。命令如下：

```
[root@master conf]# scp -r /usr/kafka/ slave1:/usr/
[root@master conf]# scp -r /usr/kafka/ slave2:/usr/
[root@slave1 conf]#vi server.pronperties
#broker 的全局唯一编号，不能重复
broker.id=1
#用来监听链接的端口，producer 或 consumer 将在此端口建立连接
listeners=PLAINTEXT://slave1:9092
host.name=slave1

[root@slave2 conf]#vi server.pronperties
broker.id=2
#用来监听链接的端口，producer 或 consumer 将在此端口建立连接
listeners=PLAINTEXT://slave2:9092
host.name= slave2
```

10、切换到 hadoop 用户。命令如下：

```
[root@master conf]# su hadoop
```

11、启动 Hadoop 集群。命令如下：

```
[hadoop @master ~]$ start-all.sh
```

12、启动 Zookeeper 集群。命令如下：

```
[hadoop @master ~]$ zkServer.sh start
[hadoop @ slave1~]$ zkServer.sh start
[hadoop @ slave2 ~]$ zkServer.sh start
```

13、启动 Kafka。命令如下：

```
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server.properties &
[hadoop @slave1 ~]$ kafka-server-start.sh /usr/kafka/config/server.properties &
[hadoop @slave2 ~]$ kafka-server-start.sh /usr/kafka/config/server.properties &
```

```
[hadoop@master ~]$ kafka-server-start.sh /usr/kafka/config/server.properties &
[1] 6448
[hadoop@master ~]$ [2022-11-20 00:16:02,301] INFO Registered kafka:type=kafka.Log4
jController MBean (kafka.utils.Log4jControllerRegistration$)
[2022-11-20 00:16:02,756] INFO starting (kafka.server.KafkaServer)
[2022-11-20 00:16:02,757] INFO Connecting to zookeeper on master:2181,slave1:2181,
slave2:2181 (kafka.server.KafkaServer)
[2022-11-20 00:16:02,777] INFO [ZooKeeperClient] Initializing a new session to mas
ter:2181,slave1:2181,slave2:2181. (kafka.zookeeper.ZooKeeperClient)
[2022-11-20 00:16:02,782] INFO Client environment:zookeeper.version=3.4.13-2d71af4
dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apache.zooke
eper.ZooKeeper)
```

```
[hadoop@slave1 ~]$ kafka-server-start.sh /usr/kafka/config/server.properties &
[1] 5539
[hadoop@slave1 ~]$ [2022-11-20 00:32:23,064] INFO Registered kafka:type=kafka.Log4
jController MBean (kafka.utils.Log4jControllerRegistration$)
[2022-11-20 00:32:23,548] INFO starting (kafka.server.KafkaServer)
[2022-11-20 00:32:23,549] INFO Connecting to zookeeper on master:2181,slave1:2181,
slave2:2181 (kafka.server.KafkaServer)
[2022-11-20 00:32:23,571] INFO [ZooKeeperClient] Initializing a new session to mas
ter:2181,slave1:2181,slave2:2181. (kafka.zookeeper.ZooKeeperClient)
[2022-11-20 00:32:23,577] INFO Client environment:zookeeper.version=3.4.13-2d71af4
dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apache.zooke
eper.ZooKeeper)
```

```
[hadoop@slave2 ~]$ kafka-server-start.sh /usr/kafka/config/server.properties &
[1] 5352
[hadoop@slave2 ~]$ [2022-11-20 00:32:46,917] INFO Registered kafka:type=kafka.Log4
jController MBean (kafka.utils.Log4jControllerRegistration$)
[2022-11-20 00:32:47,374] INFO starting (kafka.server.KafkaServer)
[2022-11-20 00:32:47,375] INFO Connecting to zookeeper on master:2181,slave1:2181,
slave2:2181 (kafka.server.KafkaServer)
[2022-11-20 00:32:47,396] INFO [ZooKeeperClient] Initializing a new session to mas
ter:2181,slave1:2181,slave2:2181. (kafka.zookeeper.ZooKeeperClient)
[2022-11-20 00:32:47,401] INFO Client environment:zookeeper.version=3.4.13-2d71af4
dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apache.zooke
eper.ZooKeeper)
```

14、查看 Kafka 是否启动成功。命令如下：

```
[hadoop @master ~]$ jps
```

```
[hadoop@master ~]$ jps
4224 SecondaryNameNode
4496 ResourceManager
6448 Kafka
6401 QuorumPeerMain
7026 Jps
3860 NameNode
4004 DataNode
4743 NodeManager
```

[hadoop @slave1 ~]$ jps

```
[hadoop@slave1 ~]$ jps
3331 DataNode
4755 QuorumPeerMain
5539 Kafka
3465 NodeManager
5918 Jps
```

[hadoop @ slave12~]$ jps

```
[hadoop@slave2 ~]$ jps
3459 NodeManager
4568 QuorumPeerMain
5352 Kafka
3325 DataNode
5711 Jps
```

15、关闭 kafka。命令如下：

```
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server.properties
```

6.4.2 在一台机器上启用多个 broker

1、将 server.pronperties 复制三份，定义为 server1.properties、server2.properties、server3.properties。命令如下：

```
[hadoop @master config]$ cp server.pronperties server1.pronperties
[hadoop @master config]$ cp server.pronperties server2.pronperties
[hadoop @master config]$ cp server.pronperties server3.pronperties
```

```
[root@master config]# ls
connect-console-sink.properties     producer.properties
connect-console-source.properties   server1.properties
connect-distributed.properties      server2.properties
connect-file-sink.properties        server3.properties
connect-file-source.properties      server.properties
```

2、分别修改 server2.properties、server3.properties。命令如下：

```
[root@master config] # vim server2.pronperties
```

```
broker.id=1
listeners=PLAINTEXT://master:9093
log.dirs= /usr/kafka/logs2


[root@master config] # vim server3.pronperties
broker.id=2
listeners=PLAINTEXT://master:9094
log.dirs= /usr/kafka/logs3
```

3、创建 log2 和 log3 两个文件夹。命令如下：

```
[root@master config] # mkdir /usr/kafka/logs2
[root@master config] # mkdir /usr/kafka/logs3
```

4、启动 kafka。命令如下：

```
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server1.properties &
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server2.properties &
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server3.properties &
```

```
[hadoop@master ~]$ kafka-server-start.sh /usr/kafka/config/server1.properties &
[1] 18822
[hadoop@master ~]$ [2022-11-20 03:28:59,845] INFO Registered kafka:type=kafka.Log4
jController MBean (kafka.utils.Log4jControllerRegistration$)
```

```
[hadoop@master ~]$ kafka-server-start.sh /usr/kafka/config/server2.properties &
[2] 19193
[hadoop@master ~]$ [2022-11-20 03:29:51,342] INFO Registered kafka:type=kafka.Log4
jController MBean (kafka.utils.Log4jControllerRegistration$)
```

```
[hadoop@master ~]$ kafka-server-start.sh /usr/kafka/config/server3.properties &
[3] 19590
[hadoop@master ~]$ [2022-11-20 03:30:40,244] INFO Registered kafka:type=kafka.Log4
jController MBean (kafka.utils.Log4jControllerRegistration$)
```

5、查看 Kafka 是否启动成功。命令如下：

```
[hadoop @master ~]$ jps
```

```
[hadoop@master ~]$ jps
4224 SecondaryNameNode
4496 ResourceManager
6401 QuorumPeerMain
3860 NameNode
4004 DataNode
18822 Kafka
19590 Kafka
4743 NodeManager
19193 Kafka
19950 Jps
```

注意：在开启 kafka 之前，必须要打开 Hadoop 集群和各台机的 Zookeeper 服务。

15、关闭 kafka。命令如下：

```
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server1.properties
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server2.properties
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server3.properties
```

6.4.3 分布式消息中间件 Kafka 的使用

1、启动 kafka。命令如下：

```
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server1.properties &
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server2.properties &
[hadoop @master ~]$ kafka-server-start.sh /usr/kafka/config/server3.properties &
```

2、新建一个 topic。通过 - zookeeper 指定 zookeeper 的地址和端口， - partitions 指定 partition 的数量， - replication-factor 指定数据副本的数量。也就是说，如果有 100 条数据，会被切分成 10 份，每一份有三个副本，存放在不同的 partition 里。命令如下：

```
[hadoop @master ~]$ kafka-topics.sh --zookeeper master:2181 --create --topic demo --partitions
10 --replication-factor 3
[root@master ~]# /usr/local/kafka/bin/kafka-topics.sh --zookeeper localhost:21
81 --create --topic demo --partitions 10 --replication-factor 3
Created topic demo.
```

```
[hadoop@master ~]$ kafka-topics.sh --zookeeper master:2181 --create --topic demo --partitions 10 --replication-factor 3
Created topic demo.
```

3、查看 topic 列表，命令如下：

[hadoop @master ~]$ kafka-topics.sh --zookeeper master:2181 --list

```
[root@master ~]# /usr/local/kafka/bin/kafka-topics.sh --zookeeper localhost:2181 --list
demo
```

```
[hadoop@master ~]$ kafka-topics.sh --zookeeper master:2181 --list
demo
```

4、查看指定 topic 明细，命令如下：

[hadoop @master ~]$ kafka-topics.sh --describe --zookeeper master:2181 --topic demo

```
[root@master ~]# /usr/local/kafka/bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic demo
Topic:demo      PartitionCount:10       ReplicationFactor:3     Configs:
        Topic: demo     Partition: 0    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
        Topic: demo     Partition: 1    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
        Topic: demo     Partition: 2    Leader: 2       Replicas: 2,0,1 Isr: 2,0,1
        Topic: demo     Partition: 3    Leader: 0       Replicas: 0,2,1 Isr: 0,2,1
        Topic: demo     Partition: 4    Leader: 1       Replicas: 1,0,2 Isr: 1,0,2
        Topic: demo     Partition: 5    Leader: 2       Replicas: 2,1,0 Isr: 2,1,0
        Topic: demo     Partition: 6    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
        Topic: demo     Partition: 7    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
        Topic: demo     Partition: 8    Leader: 2       Replicas: 2,0,1 Isr: 2,0,1
        Topic: demo     Partition: 9    Leader: 0       Replicas: 0,2,1 Isr: 0,2,1
```

从第一排可以看到 topic 的名称，partition 数量，副本数量。使用 3 份副本，就是保证数据的可用性，即使有两台 broker 服务器挂了，也能保证 kafka 的正常运行。从第二排开始，表格包含了五列，显示 partition 的情况，分别表示：topic 名称、partition 编号，此 partions 的 leader broker 编号，副本存放的 broker 编号，同步 broker 编号。因为我们开启了三个 broker 服务，对应的 broker.id 分别为 0、1、2，而每个 partion 有三个副本，所以就有把所有的 broker 都使用了，只不过每个 partition 的 leader 不同。

```
[hadoop@master ~]$ kafka-topics.sh --describe --zookeeper master:2181 --topic demo
Topic: demo     PartitionCount:10       ReplicationFactor:3     Configs:
        Topic: demo     Partition: 0    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
        Topic: demo     Partition: 1    Leader: 2       Replicas: 2,0,1 Isr: 2,0,1
        Topic: demo     Partition: 2    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
        Topic: demo     Partition: 3    Leader: 1       Replicas: 1,0,2 Isr: 1,0,2
        Topic: demo     Partition: 4    Leader: 2       Replicas: 2,1,0 Isr: 2,1,0
        Topic: demo     Partition: 5    Leader: 0       Replicas: 0,2,1 Isr: 0,2,1
        Topic: demo     Partition: 6    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
        Topic: demo     Partition: 7    Leader: 2       Replicas: 2,0,1 Isr: 2,0,1
        Topic: demo     Partition: 8    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
        Topic: demo     Partition: 9    Leader: 1       Replicas: 1,0,2 Isr: 1,0,2
```

5、修改 Topic。增加 partion 数量，从 10 个 partition 增加到 20 个，命令如下：

[hadoop @master ~]$ kafka-topics.sh --zookeeper master:2181 --alter --topic demo --partitions 20

```
[root@master ~]# /usr/local/kafka/bin/kafka-topics.sh --zookeeper localhost:21
81 --alter --topic demo --partitions 20
WARNING: If partitions are increased for a topic that has a key, the partition
 logic or ordering of the messages will be affected
Adding partitions succeeded!
```

```
[hadoop@master ~]$ kafka-topics.sh --zookeeper master:2181 --alter --topic demo --
partitions 20
WARNING: If partitions are increased for a topic that has a key, the partition log
ic or ordering of the messages will be affected
Adding partitions succeeded!
```

6、开启生产者和消费者，命令如下：

[hadoop @master ~]$ kafka-console-producer.sh --broker-list master:9092 --topic demo

[hadoop @master ~]$ kafka-console-consumer.sh --bootstrap-server master:9092 --topic demo

注意这两条命令要在两个不同的 shell 终端运行，两个 shell 终端均指向 master 这台机。如果加上 from-beginning 指定从第一条数据开始消费。

7、往生产者中写入数据，消费者消费数据，命令如下：

生产者终端输入:

```
[hadoop@master config]$ kafka-console-producer.sh --broker-list master:9092 --topic demo
>[2022-10-24 08:04:22,849] INFO [GroupCoordinator 1]: Preparing to rebalance group console-consumer-54761 in state Prepa
ringRebalance with old generation 0 (__consumer_offsets-24) (reason: Adding new member consumer-1-ebf8d81a-497d-4a89-9ac
2-9c65e44f9b19) (kafka.coordinator.group.GroupCoordinator)
[2022-10-24 08:04:25,859] INFO [GroupCoordinator 1]: Stabilized group console-consumer-54761 generation 1 (__consumer_of
fsets-24) (kafka.coordinator.group.GroupCoordinator)
[2022-10-24 08:04:25,868] INFO [GroupCoordinator 1]: Assignment received from leader for group console-consumer-54761 fo
r generation 1 (kafka.coordinator.group.GroupCoordinator)
abcd
>abcd
>
```

消费者终端输出:

```
[hadoop@master ~]$ kafka-console-consumer.sh --bootstrap-server master:9092 --topic demo
abcd
abcd
```

生产者：

```
abcd
>[2022-11-20 03:59:01,410] INFO [GroupMetadataManager brokerId=0] Removed 0 expire
d offsets in 0 milliseconds. (kafka.coordinator.group.GroupMetadataManager)
0123
```

消费者：

```
[hadoop@master ~]$ kafka-console-consumer.sh --bootstrap-server master:9092 --to
pic demo
abcd
0123
```

8、删除 topic。在删除之前，需要对三个 server*.properties 文件进行修改，然后再完成删除 topic 的操作。

```
[hadoop @master ~]$ cd /usr/kafka/config
[hadoop @master ~]$ vi server1.properties
delete.topic.enable=true
[hadoop @master ~]$ vi server2.properties
delete.topic.enable=true
[hadoop @master ~]$ vi server3.properties
delete.topic.enable=true
[hadoop @master ~]$ kafka-topics.sh --zookeeper master:2181 --delete --topic demo
[root@master ~]# /usr/local/kafka/bin/kafka-topics.sh --zookeeper localhost:21
81 --delete --topic demo
Topic demo is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

可以看到上面的 topic 只是被标记删除。如果该 topic 还在有数据交换，那么查看 topic list 的时候，会显示该 topic 为标记删除。直到没有客户端使用该 topic，才会真正的被删除。

```
[hadoop@master ~]$ kafka-topics.sh --zookeeper master:2181 --delete --topic demo
Topic demo is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

删除结果：

```
[hadoop@master ~]$ kafka-topics.sh --zookeeper master:2181 --list
__consumer_offsets
```

9、关闭 kafka。命令如下：

```
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server1.properties
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server2.properties
```

```
[hadoop @master ~]$ kafka-server-stop.sh /usr/kafka/config/server3.properties
```