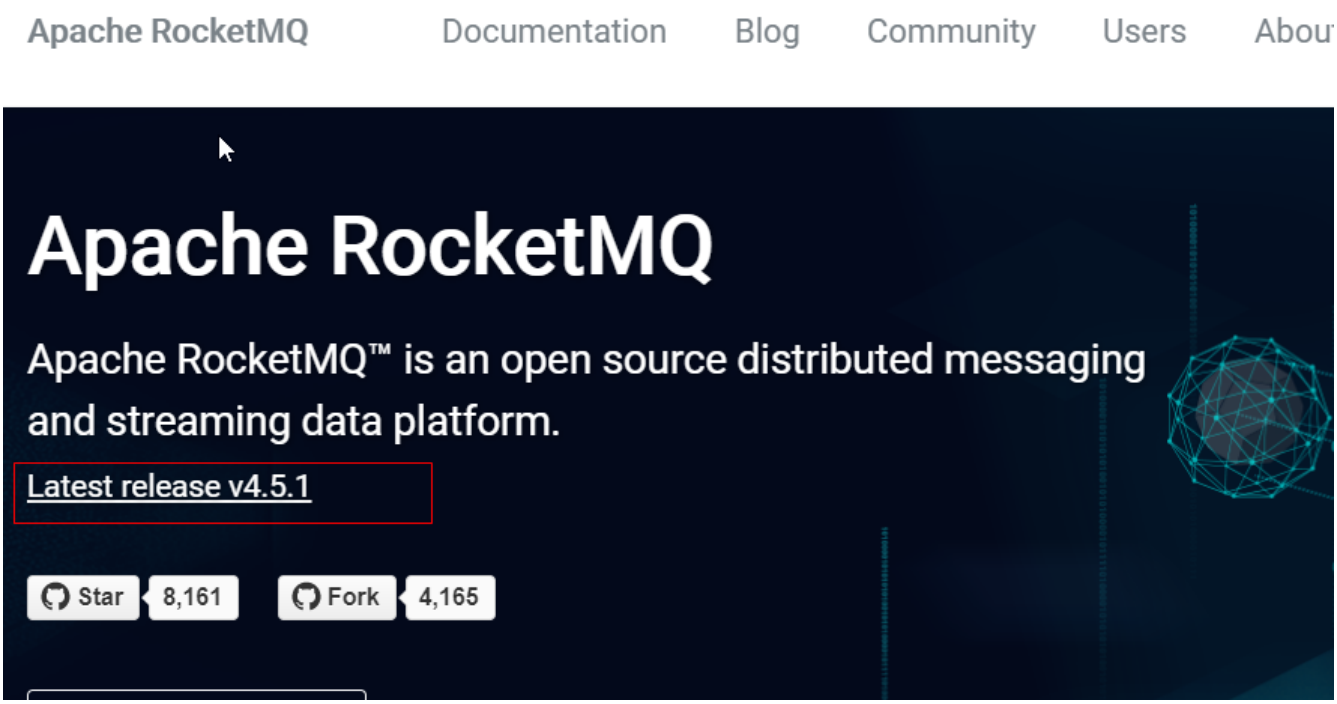
RocketMQ安装

一、RocketMQ安装

1、RocketMQ下载



可以手动下载，也可以直接在Linux服务器下载

undefined

2、安装

2.1、安装命令

```
#解压rocketmq
unzip rocketmq-all-4.5.1-bin-release.zip
#启动nameserver
bin/mqnamesrv
```

2.2、错误（一）

```
[root@464839504d37 bin]# ./mqnamesrv
Unrecognized VM option 'MetaspaceSize=128m'
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
[root@464839504d37 bin]# cd /opt/
```

错误原因：JDK版本不匹配，JDK必须是1.8以上

解决方案：重新安装软件，配置JDK，安装1.8

```
[root@464839504d37 opt]# java -version
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)
```

重新启动：

```
./mqnamesrv
```

2.3、错误（二）

```
[root@464839504d37 bin]# ./mqnamesrv
Java HotSpot(TM) 64-Bit Server VM warning: Using the DefNew young collector with the CMS collector is deprecated and will likely be removed in
ure release
Java HotSpot(TM) 64-Bit Server VM warning: UseCMSCompactAtFullCollection is deprecated and will likely be removed in a future release.
Java HotSpot(TM) 64-Bit Server VM warning: INFO: os::commit_memory(0x00000006c0000000, 2147483648, 0) failed; error='Cannot allocate memory' (
12)
#
# There is insufficient memory for the Java Runtime Environment to continue.
# Native memory allocation (mmap) failed to map 2147483648 bytes for committing reserved memory.
# An error report file with more information is saved as:
```

错误原因：是因为内存不够，导致启动失败，原因：RocketMQ的配置默认是生产环境的配置，设置的jvm的内存 大小值比较大，对于学习而言没有必要设置这么大，测试环境的内存往往都不是很大，所以需要调整默认值

解决方案：

解决办法，找到runserver.sh和runbroker.sh，编辑

JAVA_OPT="${JAVA_OPT} -server -Xms256m -Xmx256m -Xmn125m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m"

修改runserver文件：

```
vi runserver.sh
```

```
#===========================================================================================
# JVM Configuration                                                    太大，对于测试环境，不需要怎么大
#===========================================================================================
JAVA_OPT="${JAVA_OPT} -server -Xms4g -Xmx4g -Xmn2g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m"
JAVA_OPT="${JAVA_OPT} -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccup
 -XX:SoftRefLRUPolicyMSPerMB=0 -XX:+CMSClassUnloadingEnabled -XX:SurvivorRatio=8  -XX:-UseParNewGC"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/rmq_srv_gc.log -XX:+PrintGCDetails"
JAVA_OPT="${JAVA_OPT} -XX:-OmitStackTraceInFastThrow"
JAVA_OPT="${JAVA_OPT}  -XX:-UseLargePages"
JAVA_OPT="${JAVA_OPT} -Djava.ext.dirs=${JAVA_HOME}/jre/lib/ext:${BASE_DIR}/lib"
```

修改为如下所示：

```
#===========================================================================================
# JVM Configuration
#===========================================================================================
JAVA_OPT="${JAVA_OPT} -server -Xms256m -Xmx256m -Xmn128m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320
JAVA_OPT="${JAVA_OPT} -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccupar
 -XX:SoftRefLRUPolicyMSPerMB=0 -XX:+CMSClassUnloadingEnabled -XX:SurvivorRatio=8  -XX:-UseParNewGC"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/rmq_srv_gc.log -XX:+PrintGCDetails"
JAVA_OPT="${JAVA_OPT} -XX:-OmitStackTraceInFastThrow"
JAVA_OPT="${JAVA_OPT}  -XX:-UseLargePages"
JAVA_OPT="${JAVA_OPT} -Djava.ext.dirs=${JAVA_HOME}/jre/lib/ext:${BASE_DIR}/lib"
```

修改runbroker文件：

```
vi runbroker.sh
```

```
#=================================================================================
JAVA_OPT="${JAVA_OPT} -server -Xms8g -Xmx8g -Xmn4g"
JAVA_OPT="${JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupan
B=0"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/mq_gc_%p.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -X
rintAdaptiveSizePolicy"
JAVA_OPT="${JAVA_OPT} -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m"
```

太大，修改小一点

修改为：

```
# JVM Configuration
#=================================================================================
JAVA_OPT="${JAVA_OPT} -server -Xms256m -Xmx256m -Xmn128m"
JAVA_OPT="${JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeap
B=0"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/mq_gc_%p.log -XX:+PrintGCDetails -XX:+PrintGCDateSt
rintAdaptiveSizePolicy"
```

重启rocketMQ:

```
./mqnamesrv
#后台启动
#如果不存在nohup,使用命令安装:yum install coreutils
#如果 which nohup在usr/bin/nohup目录下存在，只是没有配置环境变量，要进行配置
#nohub必须配置vi ~/.bash_profile文件，在环境变量PATH中加入/usr/bin
#source ~/.bash_profile
#nohup --v #查询nohup的版本
#后台启动mqnamesrv
nohup ./mqnamesrv > /dev/null 2>&1 &
#后台启动broker
nohup ./mqbroker -n 172.17.0.2:9876 > /dev/null 2>&1 &
```

发现运行成功：

```
[root@464839504d37 bin]# ./mqnamesrv
Java HotSpot(TM) 64-Bit Server VM warning: Using the DefNew young collector with the CMS collector
ure release
Java HotSpot(TM) 64-Bit Server VM warning: UseCMSCompactAtFullCollection is deprecated and will lik
The Name Server boot success. serializeType=JSON
```

2.4、启动broker

```
#前台启动
./mqbroker -n 172.17.0.2:9876 autoCreateTopicEnable=true
#后台启动：端口必须指定9876
nohup ./mqbroker -n 172.17.0.2:9876 autoCreateTopicEnable=true > /dev/null 2>&1 &
```

启动成功：

```
[root@464839504d37 bin]# ./mqbroker -n 172.17.0.2:9999
The broker[464839504d37, 172.17.0.2:10911] boot success. serializeType=JSON and name server is 172.17.0.2:9999
```

注意：端口不能任意指定，无法测试通过

2.5、消息测试

配置nameserver地址：

```
#编辑profile文件
vi /etc/profile
#刷新
source /etc/profile
#设置nameserver服务器
export NAMESRV_ADDR=172.17.0.2:9876
#测试消息发送命令
sh tools.sh org.apache.rocketmq.example.quickstart.Producer
#测试消息接收命令
sh tools.sh org.apache.rocketmq.example.quickstart.Consumer
```

2.6、发送消息

```
#测试消息发送命令
sh tools.sh org.apache.rocketmq.example.quickstart.Producer
#测试结果
SendResult [sendStatus=SEND_OK, msgId=AC110001473C7D4991AD336AEA5703E0,
offsetMsgId=AC11000100002A9F00000000000E8580, messageQueue=MessageQueue
[topic=TopicTest, brokerName=itcast, queueId=3], queueOffset=1323]
SendResult [sendStatus=SEND_OK, msgId=AC110001473C7D4991AD336AEA5903E1,
offsetMsgId=AC11000100002A9F00000000000E8634, messageQueue=MessageQueue
[topic=TopicTest, brokerName=itcast, queueId=0], queueOffset=1323]
SendResult [sendStatus=SEND_OK, msgId=AC110001473C7D4991AD336AEA5F03E2,
offsetMsgId=AC11000100002A9F00000000000E86E8, messageQueue=MessageQueue
[topic=TopicTest, brokerName=itcast, queueId=1], queueOffset=1323]
SendResult [sendStatus=SEND_OK, msgId=AC110001473C7D4991AD336AEA6103E3,
offsetMsgId=AC11000100002A9F00000000000E879C, messageQueue=MessageQueue
[topic=TopicTest, brokerName=itcast, queueId=2], queueOffset=1323]
#可以正常发送消息
```

消息发送正常：

```
SendResult [sendStatus=SEND_OK, msgId=AC1100020B4E0D7163618907022502E1, offsetMsgId=AC11000200002A9F0000000000205C6, messageQueue=MessageQueue [t
ic=TopicTest, brokerName=464839504d37, queueId=2], queueOffset=184]
SendResult [sendStatus=SEND_OK, msgId=AC1100020B4E0D7163618907022A02E2, offsetMsgId=AC11000200002A9F000000000002067A, messageQueue=MessageQueue [t
ic=TopicTest, brokerName=464839504d37, queueId=3], queueOffset=184]
SendResult [sendStatus=SEND_OK, msgId=AC1100020B4E0D7163618907022F02E3, offsetMsgId=AC11000200002A9F000000000002072E, messageQueue=MessageQueue [t
ic=TopicTest, brokerName=464839504d37, queueId=0], queueOffset=184]
SendResult [sendStatus=SEND_OK, msgId=AC1100020B4E0D7163618907023502E4, offsetMsgId=AC11000200002A9F00000000000207E2, messageQueue=MessageQueue [t
ic=TopicTest, brokerName=464839504d37, queueId=1], queueOffset=185]
SendResult [sendStatus=SEND_OK, msgId=AC1100020B4E0D7163618907023702E5, offsetMsgId=AC11000200002A9F0000000000020896, messageQueue=MessageQueue [t
ic=TopicTest, brokerName=464839504d37, queueId=2], queueOffset=185]
SendResult [sendStatus=SEND_OK, msgId=AC1100020B4E0D7163618907023902E6, offsetMsgId=AC11000200002A9F000000000002094A, messageQueue=MessageQueue [t
```

注意事项：

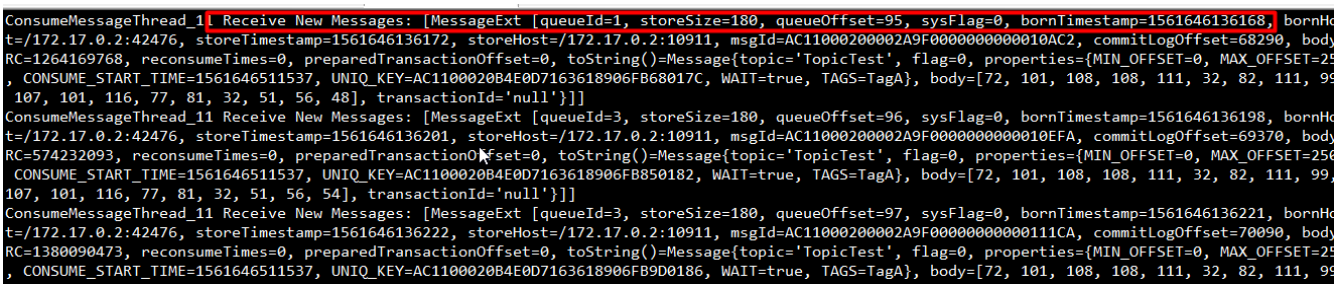
runserver.sh,runbroker.sh文件内存大小必须设置合适：

```
#大小必须设置合适
runserver.sh
JAVA_OPT="${JAVA_OPT} -server -Xms256m -Xmx256m -Xmn512m -XX:MetaspaceSize=128m -
XX:MaxMetaspaceSize=320m"
#大小设置合适
runbroker.sh
JAVA_OPT="${JAVA_OPT} -server -Xms256m -Xmx256m -Xmn128m"
#如果报错
#Native memory allocation (mmap) failed to map 805306368 bytes for committing
#使用命令jps查询发现除了很多进程，全部干掉重新启动
jps
kill -9 pid(全部的pid)
```

2.7、接收消息

```
#测试消息接收命令
sh tools.sh org.apache.rocketmq.example.quickstart.Consumer
#测试结果
ConsumeMessageThread_7 Receive New Messages: [MessageExt [queueId=2, storeSize=180,
queueOffset=1322, sysFlag=0, bornTimestamp=1544456244818,
bornHost=/172.16.55.185:33702, storeTimestamp=1544456244819,
storeHost=/172.17.0.1:10911, msgId=AC11000100002A9F00000000000E84CC,
commitLogOffset=951500, bodyCRC=684865321, reconsumeTimes=0,
preparedTransactionOffset=0, toString()=Message{topic='TopicTest', flag=0,
properties={MIN_OFFSET=0, MAX_OFFSET=1325, CONSUME_START_TIME=1544456445397,
UNIQ_KEY=AC110001473C7D4991AD336AEA5203DF, WAIT=true, TAGS=TagA}, body=[72, 101, 108,
108, 111, 32, 82, 111, 99, 107, 101, 116, 77, 81, 32, 57, 57, 49],
transactionId='null'}]]
ConsumeMessageThread_6 Receive New Messages: [MessageExt [queueId=2, storeSize=180,
queueOffset=1323, sysFlag=0, bornTimestamp=1544456244833,
bornHost=/172.16.55.185:33702, storeTimestamp=1544456244835,
storeHost=/172.17.0.1:10911, msgId=AC11000100002A9F00000000000E879C,
commitLogOffset=952220, bodyCRC=801108784, reconsumeTimes=0,
preparedTransactionOffset=0, toString()=Message{topic='TopicTest', flag=0,
properties={MIN_OFFSET=0, MAX_OFFSET=1325, CONSUME_START_TIME=1544456445397,
UNIQ_KEY=AC110001473C7D4991AD336AEA61103E3, WAIT=true, TAGS=TagA}, body=[72, 101, 108,
108, 111, 32, 82, 111, 99, 107, 101, 116, 77, 81, 32, 57, 57, 53],
transactionId='null'}]]
#从结果中，可以看出，接收消息正常
```

发现接受消息成功：



3、docker安装

3.1、镜像安装

```
#拉取镜像
docker pull foxiswho/rocketmq:server-4.3.2
docker pull foxiswho/rocketmq:broker-4.3.2
#创建nameserver容器
docker create -p 9876:9876 --name rmqserver \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt" \
-v /kkb/rmq/rmqserver/logs:/opt/logs \
-v /kkb/rmq/rmqserver/store:/opt/store \
foxiswho/rocketmq:server-4.3.2
#创建broker容器
docker create -p 10911:10911 -p 10909:10909 --name rmqbroker \
-e "JAVA_OPTS=-Duser.home=/opt" \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-v /kkb/rmq/rmqbroker/conf/broker.conf:/etc/rocketmq/broker.conf \
-v /kkb/rmq/rmqbroker/logs:/opt/logs \
-v /kkb/rmq/rmqbroker/store:/opt/store \
```

```
foxiswho/rocketmq:broker-4.3.2

#第二种更简单的创建方式（上面那种创建方式，不是很好使）
#创建broker-server
docker run -di -p 9877:9876 --name=rmqserver02 \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt" \
foxiswho/rocketmq:server-4.5.1


#创建broker
docker run -di -p 10911:10911 -p 10909:10909 --name=rmqbroker -e "JAVA_OPTS=-Duser.home=/opt"
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" foxiswho/rocketmq:broker-4.5.1

#配置broker容器的配置文件
docker exec -it rmqbroker /bin/bash
cd /etc/rocketmq/
vi broker.conf

#配置内容
brokerIP1=172.17.0.3 #内网IP
namesrvAddr=192.168.66.66:9876
brokerName=kkb-a

#启动容器
docker start rmqserver rmqbroker
#停止删除容器
docker stop rmqbroker rmqserver
docker rm rmqbroker rmqserver
```

经测试，可以正常发送、接收消息。

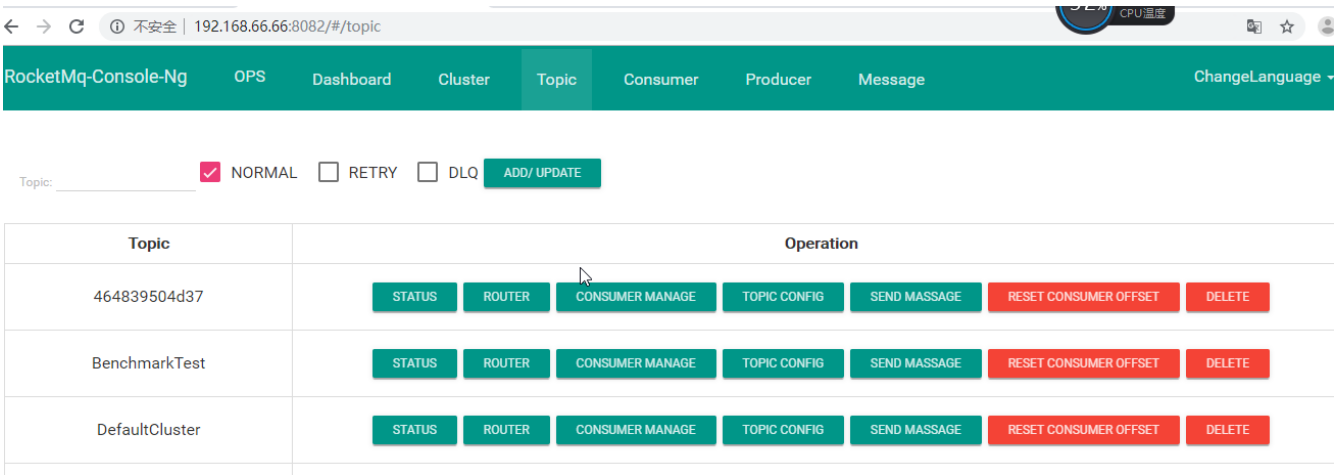
3.2、管理工具

RocketMQ提供了UI管理工具，名为rocketmq-console、项目地址：https://github.com/apache/rocketmq-externals/tree/master/rocketmq-console。该工具支持docker以及非docker安装，这里我们选择使用docker安装

安装命令：

```
 #拉取镜像
docker pull styletang/rocketmq-console-ng:1.0.0
#创建并启动容器
docker run -e "JAVA_OPTS=-Drocketmq.namesrv.addr=172.17.0.3:9876 -
Dcom.rocketmq.sendMessageWithVIPChannel=false" -p 8082:8080 -t styletang/rocketmq-console-
ng:1.0.0
```

启动效果：



3.3、可能错误

测试结果会发现，发送消息会报错、原因是什么呢?

答案就是由于IP地址，和端口可能造成访问不到的原因。

仔细观察broker启动的信息： 会发现，broker的ip地址是172.17.0.1，那么在开发机上是不可能访问到的。 所以，需要指定broker的ip地址。 错误信息如下所示：

```
Exception in thread "main"
org.apache.rocketmq.remoting.exception.RemotingTooMuchRequestException:    sendDefaultImpl
call timeout    at
org.apache.rocketmq.client.impl.producer.DefaultMQProducerImpl.sendDefaultImpl(Defaul
tMQProducerImpl.java:612)    at
org.apache.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducer
Impl.java:1253)    at
org.apache.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducer
Impl.java:1203)    at
org.apache.rocketmq.client.producer.DefaultMQProducer.send(DefaultMQProducer.java:214    )
at cn.itcast.rocketmq.SyncProducer.main(SyncProducer.java:26)
```

启动信息:

```
The broker[jackhu, 172.17.0.1:10911] boot success. serializeType=JSON and name server    is
172.16.185.55:9876
```

解决问题:

创建broker配置文件

```
vi /kkb/rmq/rmqbroker/conf/broker.conf
brokerIP1=172.17.0.2
namesrvAddr=172.17.0.2:9876
brokerName=jackhu
```

启动broker，通过 -c 指定配置文件

```
bin/mqbroker -c /kkb/rmq/rmqbroker/conf/broker.conf
The broker[jackhu, 172.17.0.2:10911] boot success. serializeType=JSON and name
server is 172.17.0.2:9876  #这样就可以进行访问了
```

二、集群构建

1、集群模式

在RocketMQ中，集群的部署模式是比较多的，有以下几种：

```
public class ConsumerDemo {
  public static void main(String[] args) throws Exception {
    DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("test-group");
    consumer.setNamesrvAddr("172.16.55.185:9876");
    // 订阅topic，接收此Topic下的所有消息
    consumer.subscribe("my-test-topic", "*");
    consumer.registerMessageListener(new MessageListenerConcurrently() {
      @Override
      public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt> msgs,
ConsumeConcurrentlyContext context) {
        for (MessageExt msg : msgs) {
          try {
            System.out.println(new String(msg.getBody(), "UTF-8"));
         } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
         }
       }
        System.out.println("收到消息->" + msgs);
        if(msgs.get(0).getReconsumeTimes() >= 3){
          // 重试3次后，不再进行重试
          return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
       }
        return ConsumeConcurrentlyStatus.RECONSUME_LATER;
     }
   });
    consumer.start();
 }
}
```

单个Master 这种方式风险较大，一旦Broker重启或者宕机时，会导致整个服务不可用，不建议线上环境使用。 多Master模式 一个集群无Slave，全是Master，例如2个Master或者3个Master 单台机器宕机期间，这台机器上未被消费的消息在机器恢复之前不可订阅，消息实时性会受到影响。 多Master多Slave模式，异步复制 每个Master配置一个Slave，有多对Master-Slave，HA采用异步复制方式，主备有短暂消息延迟，毫秒 级。 优点：即使磁盘损坏，消息丢失的非常少，且消息实时性不会受影响，因为Master宕机后，消费者仍然 可以从Slave消费，此过程对应用透明，不需要人工干预。性能同多Master模式几乎一样。 缺点：Master宕机，磁盘损坏情况，会丢失少量消息。 多Master多Slave模式，同步双写 每个Master配置一个Slave，有多对Master-Slave，HA采用同步双写方式，主备都写成功，向应用返回 成功。 优点：数据与服务都无单点，Master宕机情况下，消息无延迟，服务可用性与数据可用性都非常高。 缺点：性能比异步复制模式略低，大约低10%左右。

2、搭建2m2s集群

下面通过docker搭建2master+2slave的集群。

```
#创建2个master
#nameserver1
docker create -p 9876:9876 --name rmqserver01 \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt" \
foxiswho/rocketmq:server-4.5.1
#nameserver2
docker create -p 9877:9876 --name rmqserver02 \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
-e "JAVA_OPTS=-Duser.home=/opt" \
foxiswho/rocketmq:server-4.5.1



#创建第1个master broker
#master broker01
docker create --net host --name rmqbroker01 \
-e "JAVA_OPTS=-Duser.home=/opt" \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
foxiswho/rocketmq:broker-4.5.1
#配置
namesrvAddr=172.16.55.185:9876;172.16.55.185:9877
brokerClusterName=testCluster
brokerName=broker01
brokerId=0
deleteWhen=04
fileReservedTime=48
brokerRole=SYNC_MASTER
flushDiskType=ASYNC_FLUSH
brokerIP1=172.16.55.185
brokerIp2=172.16.55.185
listenPort=10911

#创建第2个master broker
#master broker02
docker create --net host --name rmqbroker02 \
-e "JAVA_OPTS=-Duser.home=/opt" \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
foxiswho/rocketmq:broker-4.5.1
#master broker02
namesrvAddr=172.16.55.185:9876;172.16.55.185:9877
brokerClusterName=testCluster
brokerName=broker02
brokerId=0
deleteWhen=04
fileReservedTime=48
brokerRole=SYNC_MASTER
flushDiskType=ASYNC_FLUSH
brokerIP1=172.16.55.185
brokerIp2=172.16.55.185
listenPort=10811

#创建第1个slave broker
#slave broker01
docker create --net host --name rmqbroker03 \
-e "JAVA_OPTS=-Duser.home=/opt" \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
foxiswho/rocketmq:broker-4.5.1
#slave broker01
namesrvAddr=172.16.55.185:9876;172.16.55.185:9877
brokerClusterName=testCluster
```

```
brokerName=broker01
brokerId=1
deleteWhen=04
fileReservedTime=48
brokerRole=SLAVE
flushDiskType=ASYNC_FLUSH
brokerIP1=172.16.55.185
brokerIp2=172.16.55.185
listenPort=10711

#创建第2个slave broker
#slave broker01
docker create --net host --name rmqbroker04 \
-e "JAVA_OPTS=-Duser.home=/opt" \
-e "JAVA_OPT_EXT=-server -Xms128m -Xmx128m -Xmn128m" \
foxiswho/rocketmq:broker-4.5.1
#slave broker02
namesrvAddr=172.16.55.185:9876;172.16.55.185:9877
brokerClusterName=testCluster
brokerName=broker02
brokerId=1
deleteWhen=04
fileReservedTime=48
brokerRole=SLAVE
flushDiskType=ASYNC_FLUSH
brokerIP1=172.16.55.185
brokerIp2=172.16.55.185
listenPort=10611



#启动容器
docker start rmqserver01 rmqserver02
docker start rmqbroker01 rmqbroker02 rmqbroker03 rmqbroker04



#
```