1. 简介
   1. 常用企业消息中间件：RocketMQ RabbitMQ ActiveMQ Kafka ZeroMQ
   2. RocketMQ常用于数据量很大的情形
2. 什么是企业消息中间件
   1. 想象就是一个消息邮局
3. 应用场合
   1. 异步
   2. 解耦
   3. 削峰
4. 下载
   1. github
   2. 找咨询老师
5. 安装
   1. 解压到/usr/local
   2. cd /usr/local/alibaba-rocketmq
   3. cd bin
   4. chmod +x \*
6. 启动nameserver
   1. ./mqnamesrv
   2. nohup ./mqnamesrv > namesrv.out 2 > &1 &
   3. 如果发现内存不足错误，修改runserver.sh
7. 启动broker
   1. only master模式启动一台broker
   2. nohup sh mqbroker -n "192.168.56.200:9876" -c ../conf/2m-noslave/broker-a.properties > broker.out &
   3. 如果有内存不足错误日志产生，修改runbroker.sh
8. 使用./mqadmin clusterList -n 192.168.56.200:9876来观察namesrv和master的情况
9. Producer.java：

**import** com.alibaba.rocketmq.client.producer.DefaultMQProducer;

**import** com.alibaba.rocketmq.client.producer.SendResult;

**import** com.alibaba.rocketmq.common.message.Message;

**public** **class** Producer {

**public** **static** **void** main(String[] args) **throws** Exception {

DefaultMQProducer producer = **new** DefaultMQProducer("group");

producer.setNamesrvAddr("192.168.56.200:9876");

producer.start();

Message msg = **new** Message("orders", "order1".getBytes());

SendResult result = producer.send(msg);

System.***out***.println(result);

System.***out***.println(msg + " send out");

producer.shutdown();

Thread.sleep(500);

}

}

1. PushConsumer.java

**import** java.util.List;

**import** com.alibaba.rocketmq.client.consumer.DefaultMQPushConsumer;

**import** com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext;

**import** com.alibaba.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;

**import** com.alibaba.rocketmq.client.consumer.listener.MessageListenerConcurrently;

**import** com.alibaba.rocketmq.common.consumer.ConsumeFromWhere;

**import** com.alibaba.rocketmq.common.message.Message;

**import** com.alibaba.rocketmq.common.message.MessageExt;

**public** **class** PushConsumer {

**public** **static** **void** main(String[] args){

DefaultMQPushConsumer consumer =

**new** DefaultMQPushConsumer("consumers");

consumer.setNamesrvAddr("192.168.56.200:9876");

**try** {

//订阅PushTopic下Tag为的消息

consumer.subscribe("orders", **null**);

//程序第一次启动从消息队列头取数据

consumer.setConsumeFromWhere(

ConsumeFromWhere.***CONSUME\_FROM\_FIRST\_OFFSET***);

consumer.registerMessageListener(

**new** MessageListenerConcurrently() {

**public** ConsumeConcurrentlyStatus consumeMessage(

List<MessageExt> list,

ConsumeConcurrentlyContext Context) {

Message msg = list.get(0);

System.***out***.println(**new** String(msg.getBody()));

**return** ConsumeConcurrentlyStatus.***CONSUME\_SUCCESS***;

}

}

);

consumer.start();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

1. 更复杂的架构