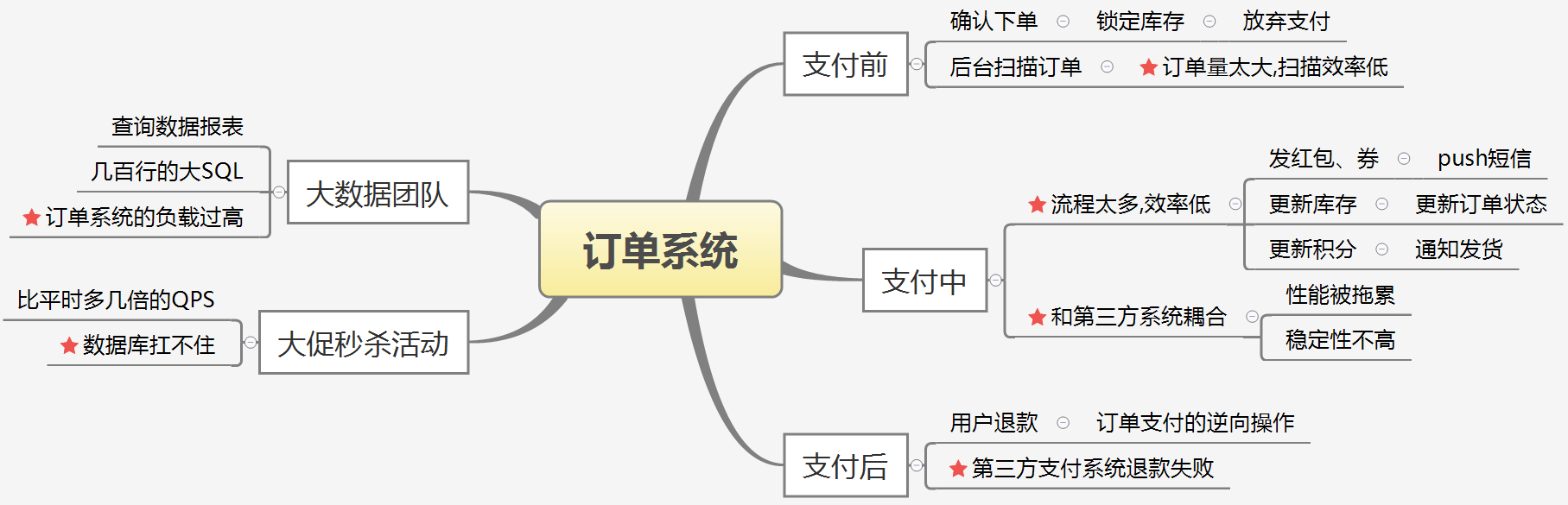
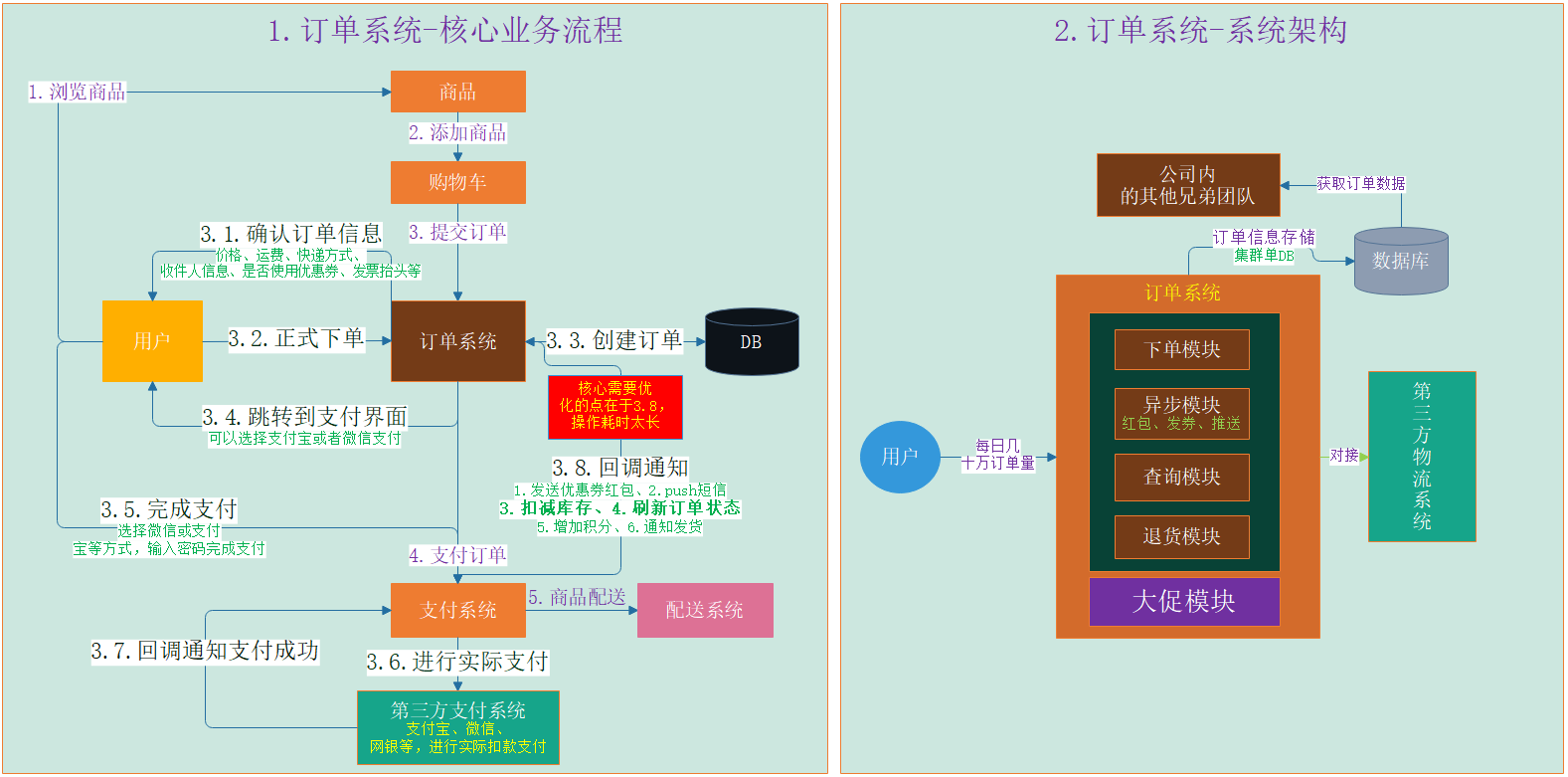
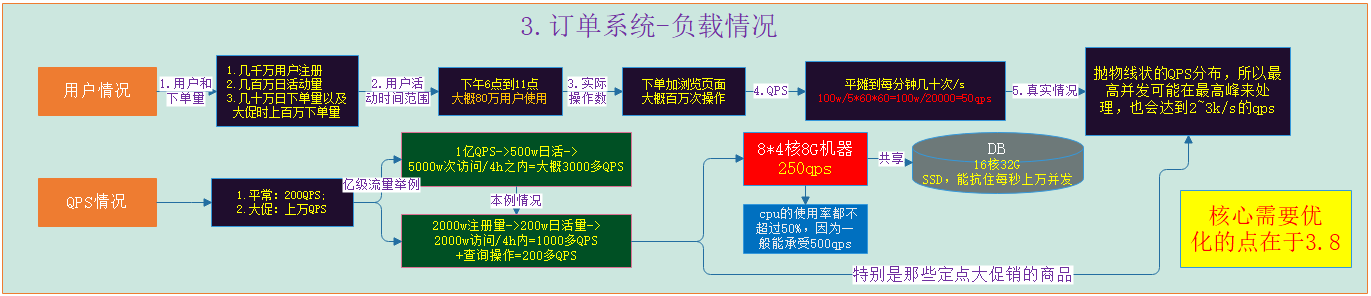
# 一个真实订单系统面临的技术挑战

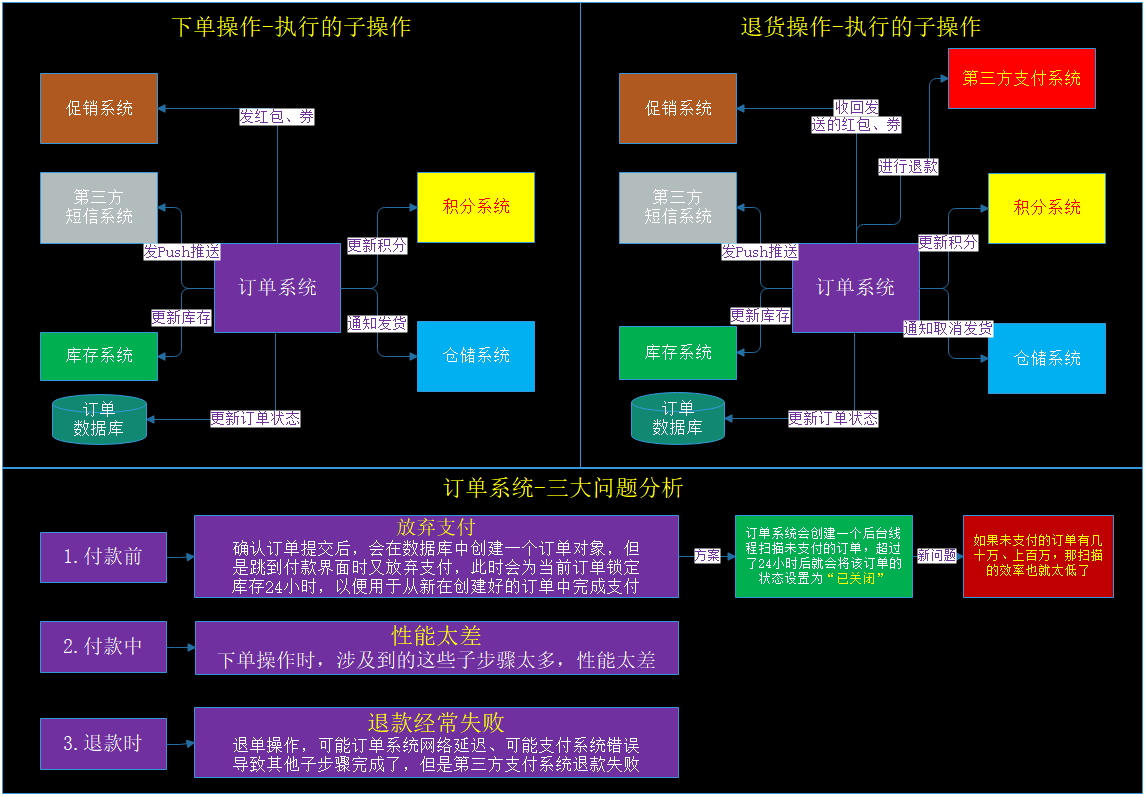


## 一个真实的电商系统业务流程、系统架构和负载情况

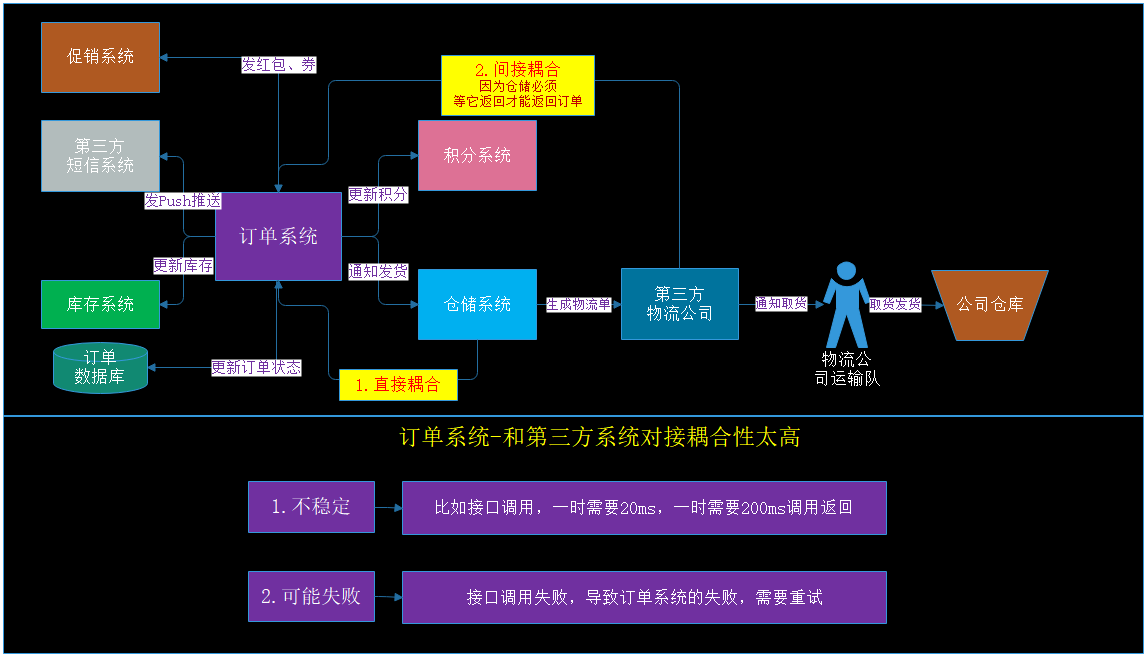




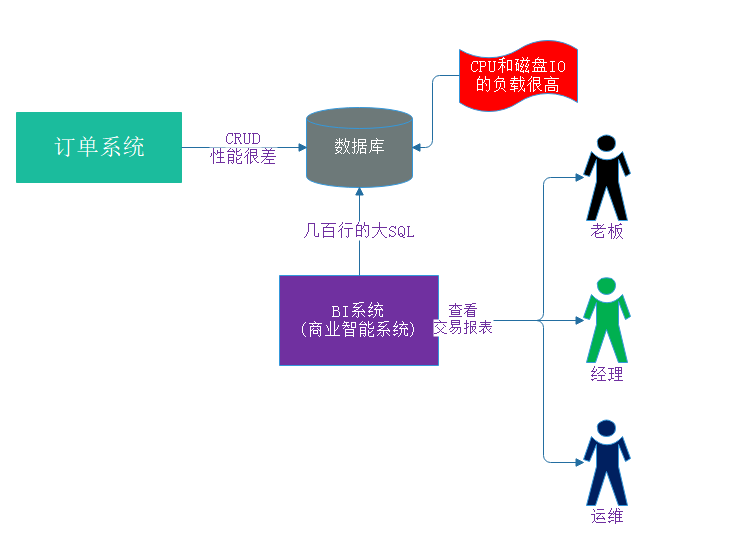
## 系统面临的问题-订单支付前中后三大问题



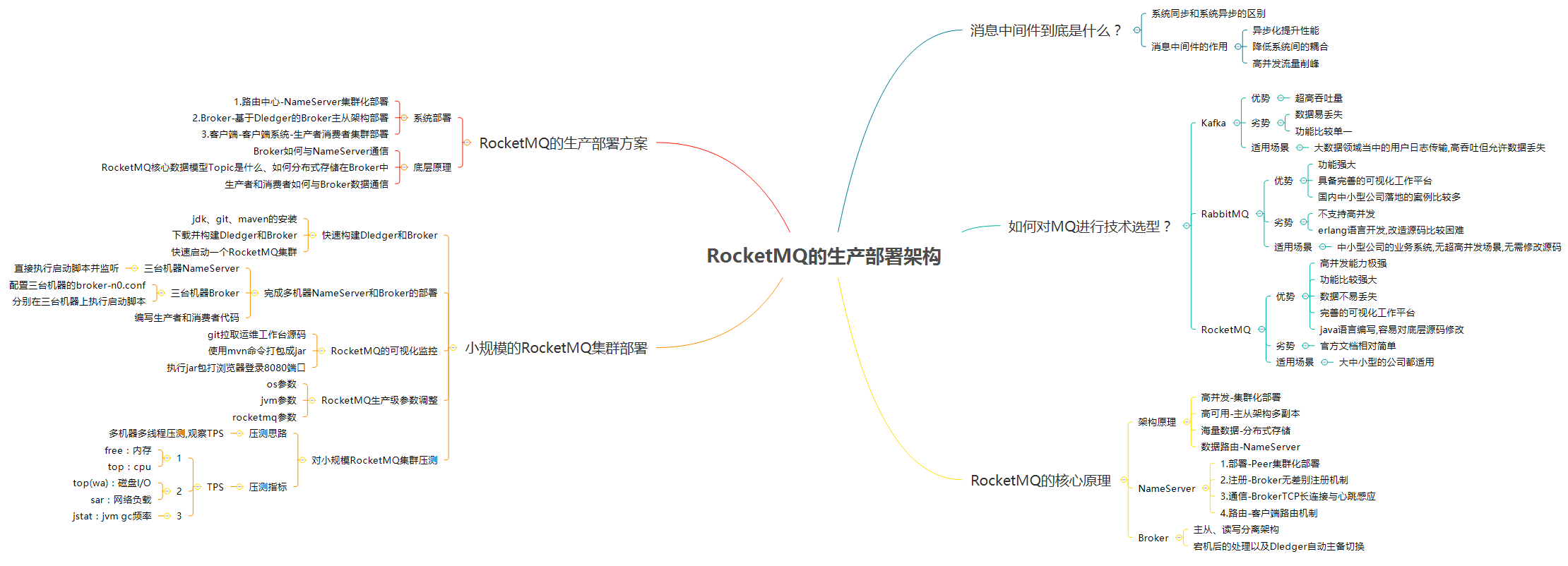
## 系统面临的问题-和第三方系统对接耦合性太高



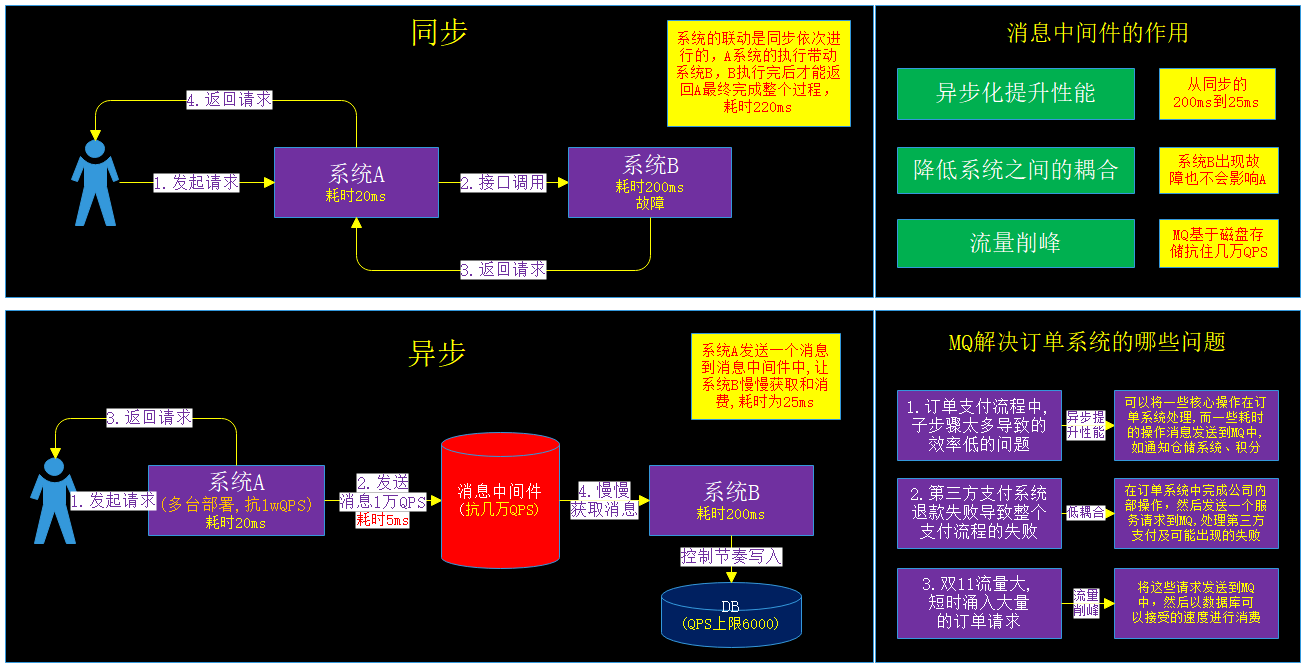
## 系统面临的问题-大数据团队要订单系统数据带来的问题



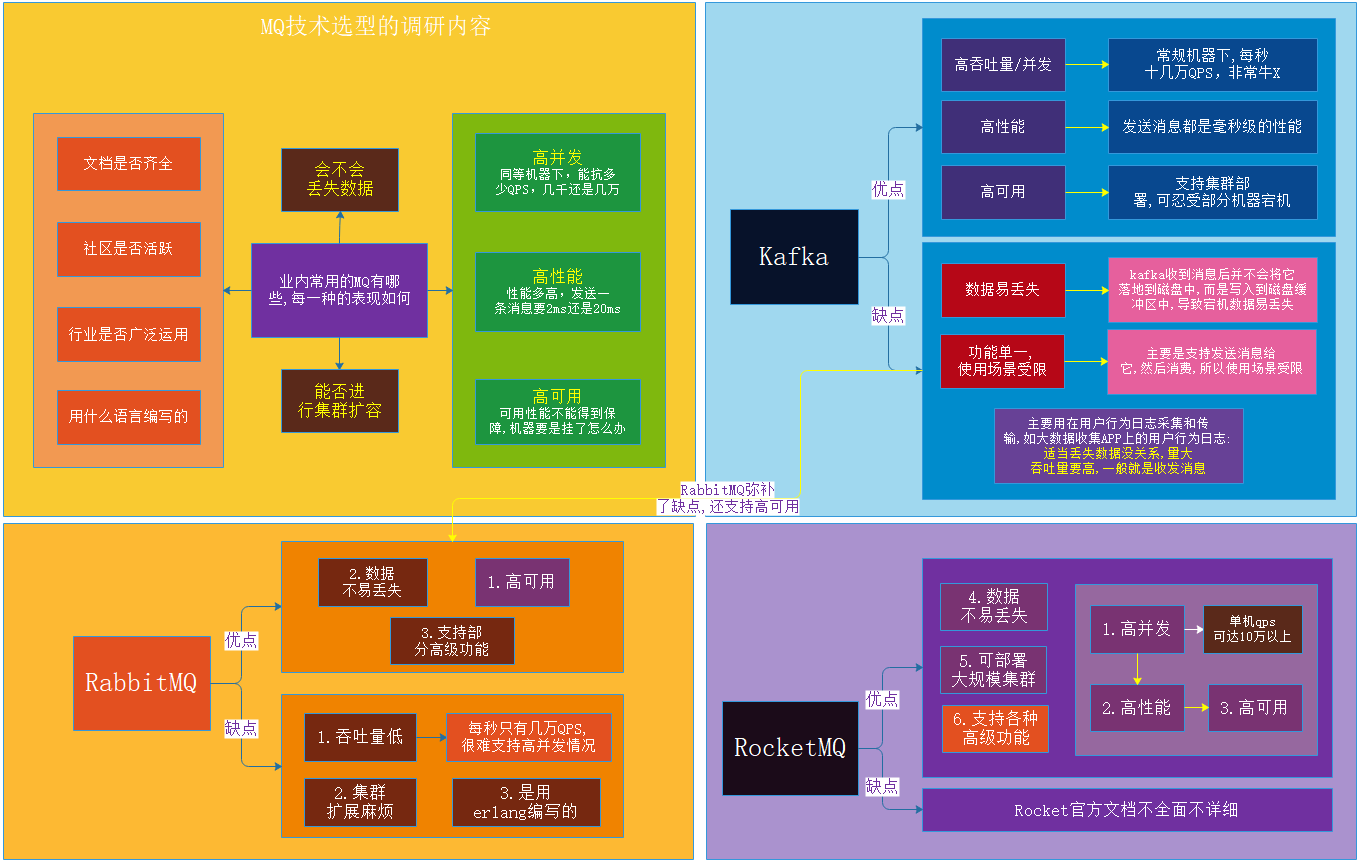
# 基于真实负载设计RocketMQ的生产部署架构



## 消息中间件到底是什么

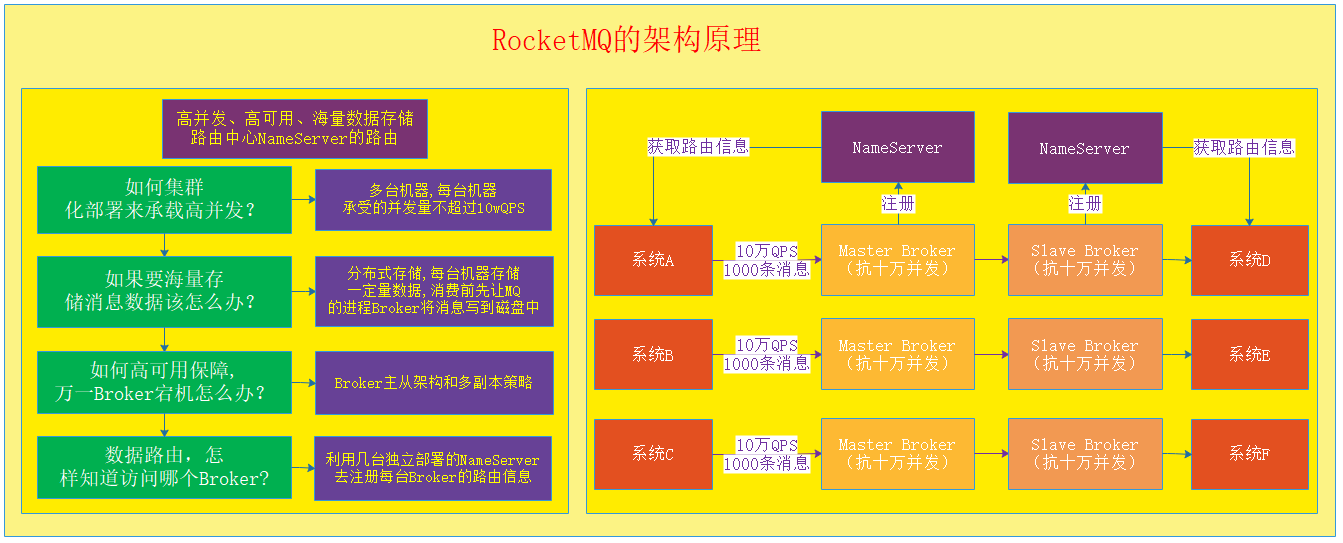


## MQ进行技术选型调研

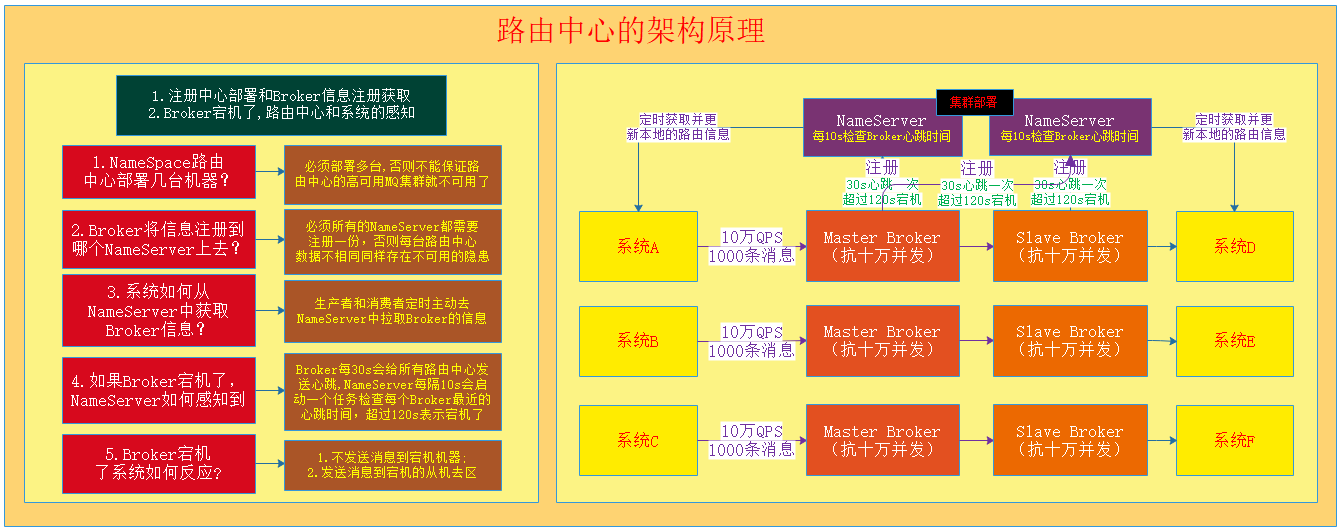


## RocketMQ的核心原理

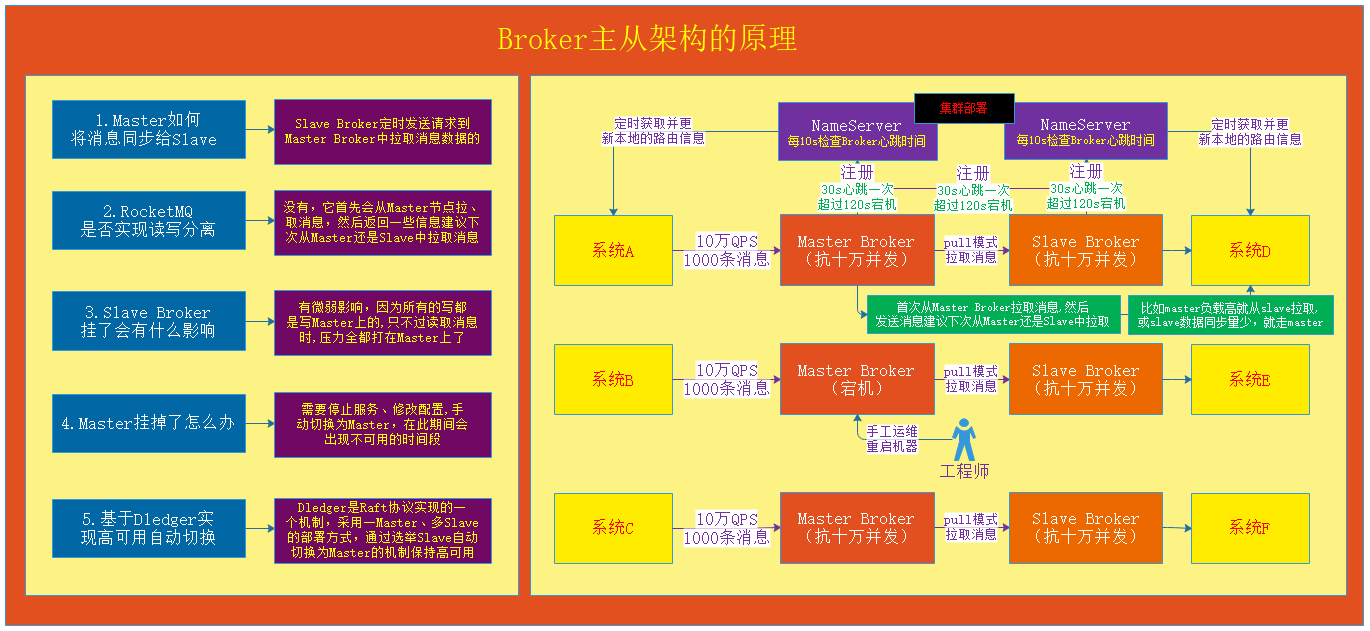
### RocketMQ架构原理



### NameServer架构原理



### Broker主从架构原理



## 高可用的MQ-生产部署架构

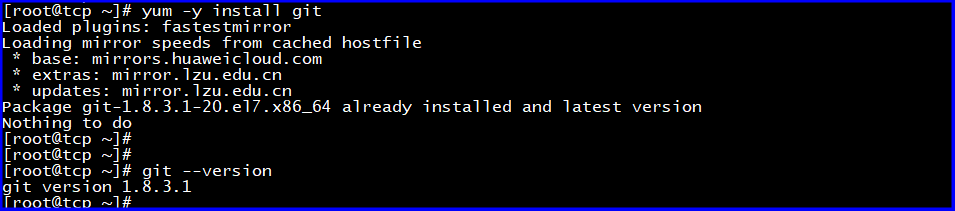


## 部署一个小规模的RocketMQ集群



### 快速构建Dledger和RocketMQ

（1）git的安装



（2）maven的安装

//1.下载apache-maven-3.2.5-bin.tar.gz并解压到/usr/local

**cd /usr/local**

**tar –zxvf apache-maven-3.2.5-bin.tar.gz**

//2.配置环境变量-进入/etc/profile文件并添加如下配置

**export MAVEN\_HOME=/usr/local/apache-maven-3.2.5**

**export PATH=$PATH:$MAVEN\_HOME/bin**

//3.让配置文件生效

**source /etc/profile**

//4.查看maven版本信息

**mvn –v**

（3）构建Dledger

//1.进入目录/usr/local下,利用git下载dledger

**cd /usr/local**

**git clone**[**https://github.com/openmessaging/openmessaging-storage-dledger.git**](https://github.com/openmessaging/openmessaging-storage-dledger.git)

//2.进入dledger目录中构建它

**cd openmessaging-storage-dledger**

**mvn clean install –DskipTests**

（4）构建RocketMQ

**//1.进入目录/usr/local下，利用git下载rocketmq**

**cd /usr/local**

**git clone** [**https://github.com/apache/rocketmq.git**](https://github.com/apache/rocketmq.git)**（30min）**

**//2.进入rocketmq目录构建它**

**cd rocketmq**

**git checkout -b store\_with\_dledger origin/store\_with\_dledger**

**mvn -Prelease-all -DskipTests clean install -U (30min)**

**//3.修改rocketmq配置文件中的jdk路径**

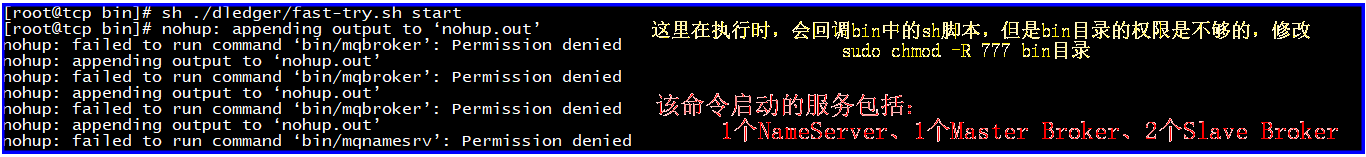
#1.如下图分别修改**/usr/local/rocketmq/distribution/target/apache-rocketmq/bin** 目录下的：runserver.sh、runbroker.sh、tools.sh

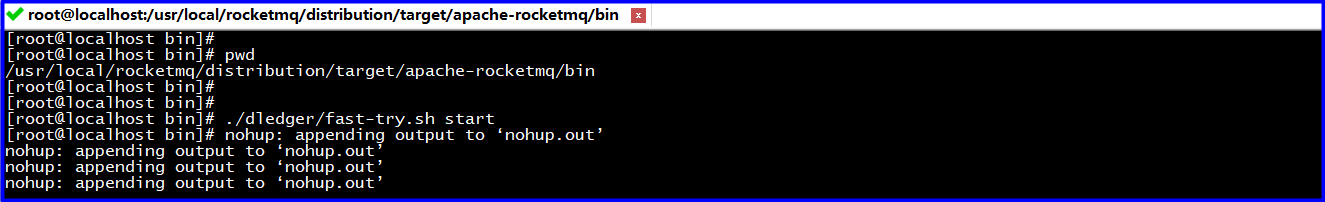


#2.快速启动rocketmq

**cd /usr/local/rocketmq/distribution/target/apache-rocketmq**

**sh bin/dledger/fast-try.sh start**

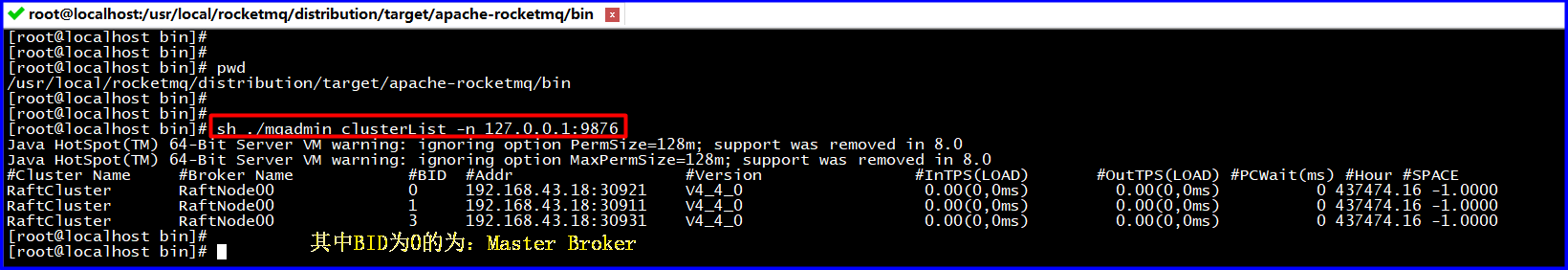




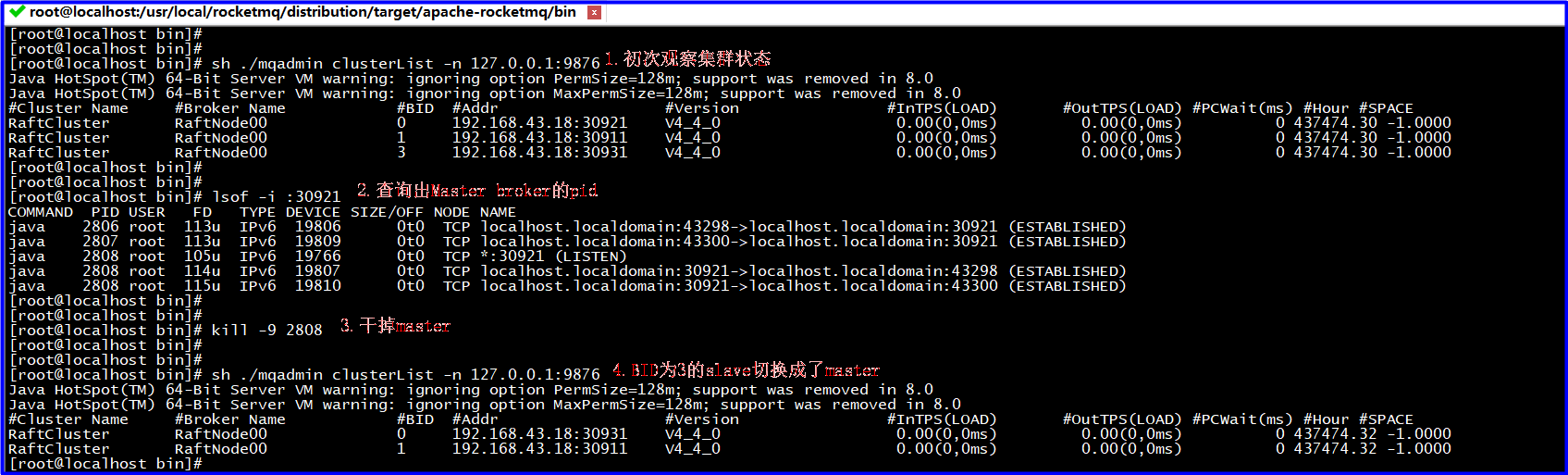
#3.检查rocketmq集群状态

//其中9876就是NameServer监听的端口

**sh ./mqadmin clusterList –n 127.0.0.1:9876**



#4.kill掉master broker，观察主备切换



### 完成NameServer和Broker的部署

（1）完成三台NameServer的部署

准备三台机器，如前面两步：完成jdk安装、git安装、maven安装、Dledger和RocketMQ的构建

//注意，此处也需要将runserver.sh、runbroker.sh和tools.sh中的jdk参数和jvm参数配置下

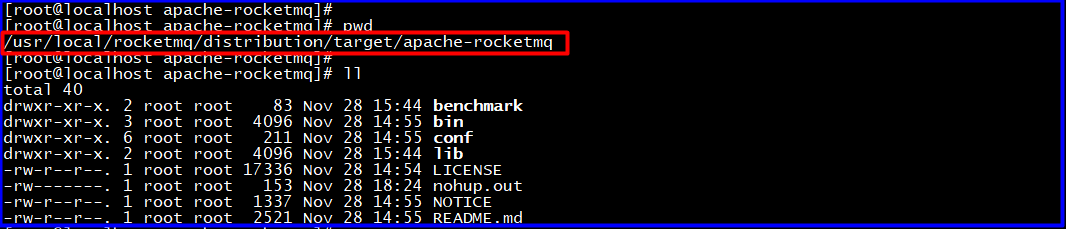
**cd /usr/local/rocketmq/distribution/target/apache-rocketmq/bin/**

**nohup sh mqnamesrv &**

**lsof -i:9876//查看nameserver端口9876服务**

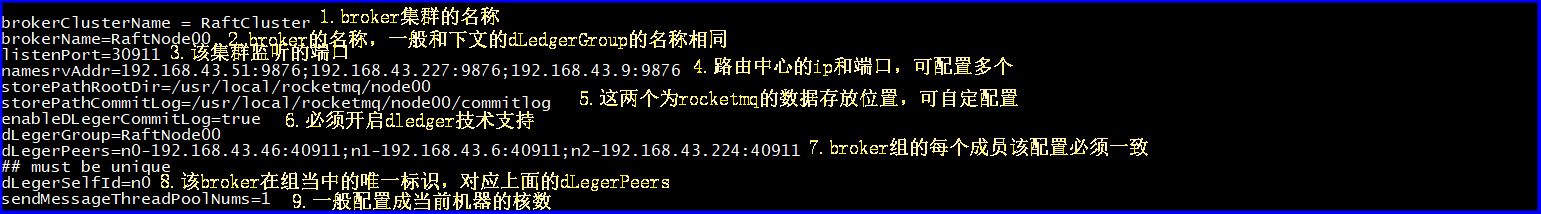
（2）完成一组Broker的部署

//1.进入该目录下cd /usr/local/rocketmq/distribution/target/apache-rocketmq,修改配置文件



//2.分配在三台机器上配置conf,三台机器的配置中,分别都只修改./conf/dledger/broker-n0.conf

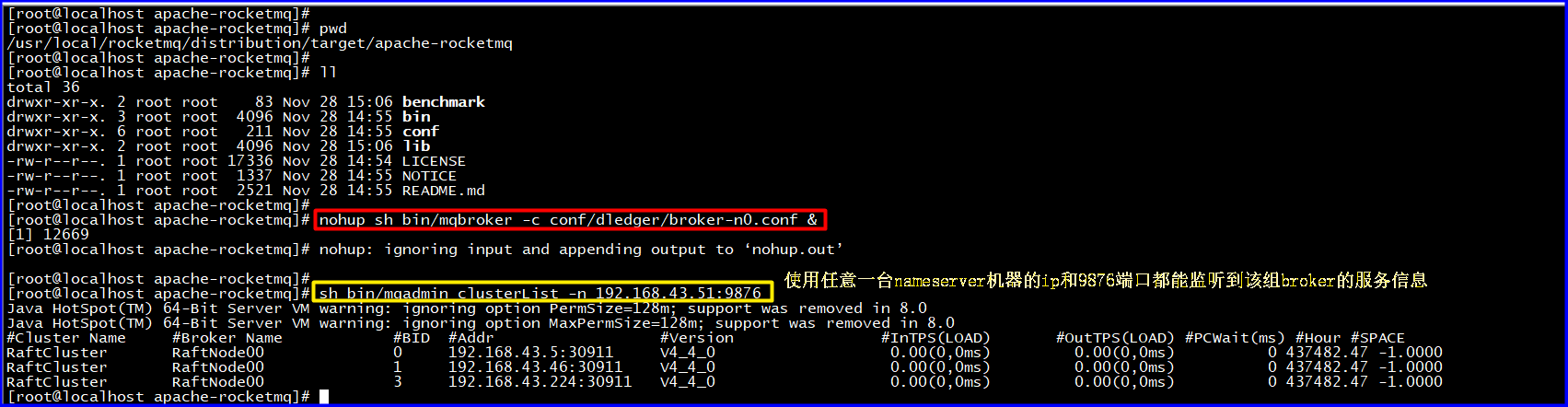
**三台机器中的该配置文件，唯一不同在于dLegerSelfId，标识broker组的唯一标识**



//3.分别在三台机器上启动broker

**nohup sh bin/mqbroker -c conf/dledger/broker-n0.conf &**

**sh bin/mqadmin clusterList -n 192.168.43.51:9876**



//4.nameserver和broker服务的关闭

**sh ./bin/mqshutdown namesrv**

**sh ./bin/mqshutdown broker**

（3）编写生产者和消费者消费者代码

****

### 对RocketMQ可视化的监控

//1.在目录/usr/local/下拉取运维工作台的源码

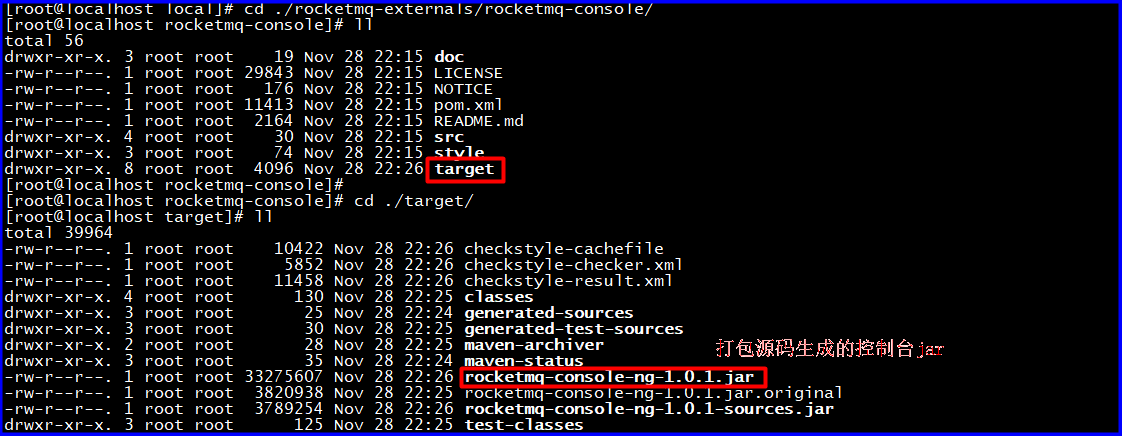
**cd /usr/local**

**git clone** [**https://github.com/apache/rocketmq-externals.git**](https://github.com/apache/rocketmq-externals.git)

//2.将rocketmq-console打包成一个jar

**cd rocketmq-externals/rocketmq-console/**

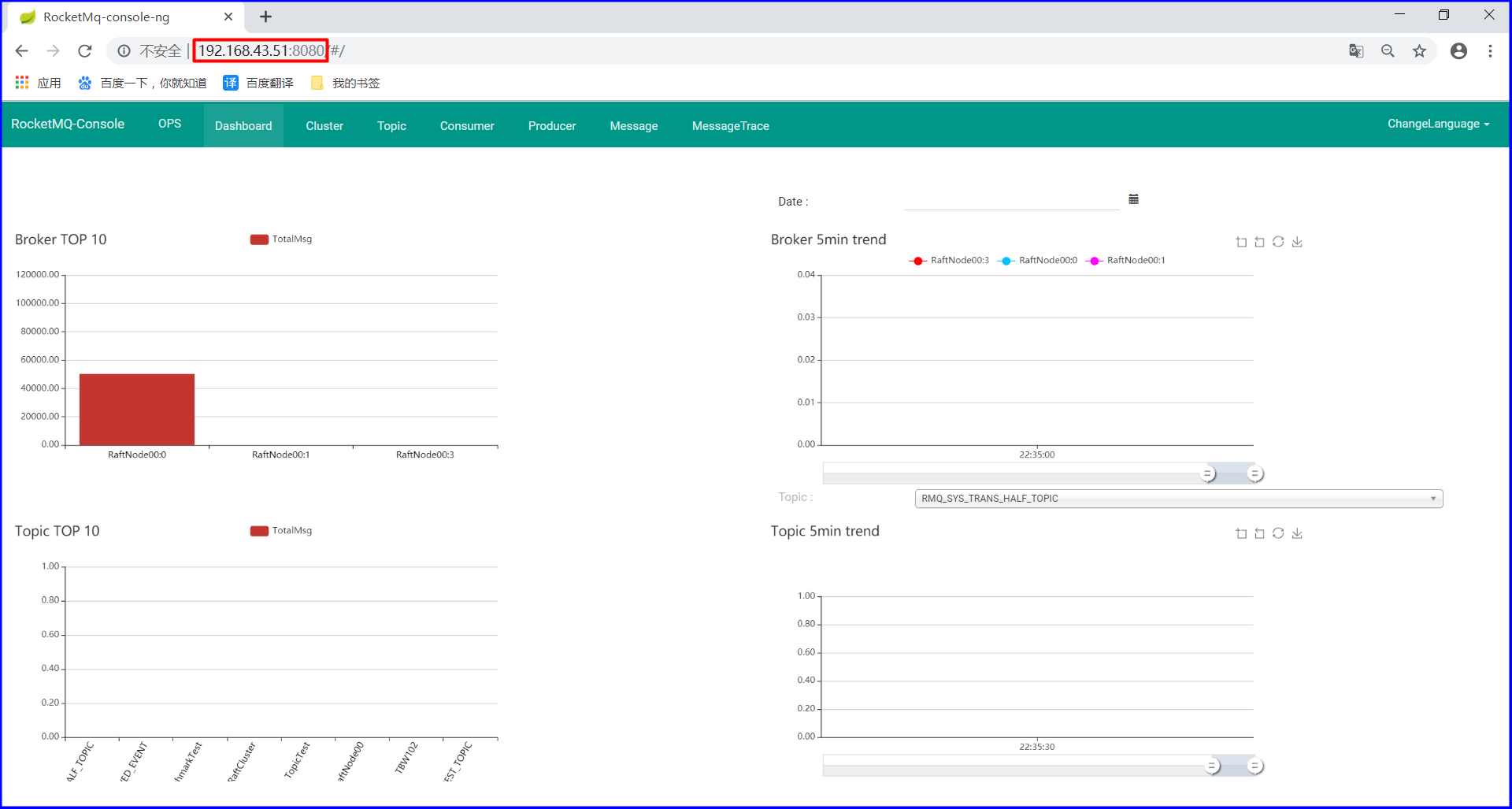
**mvn package –DskipTests**



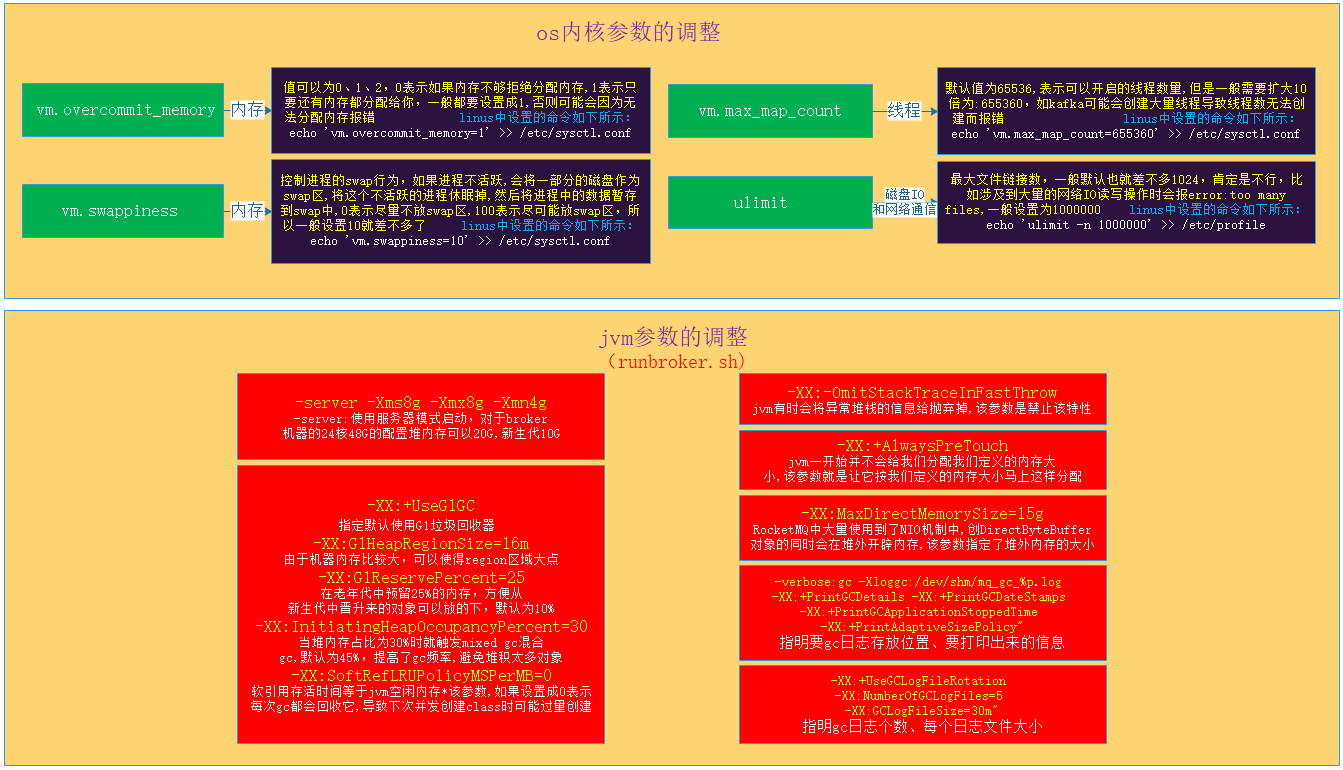
//3.启动工作台

**java -jar rocketmq-console-ng-1.0.1.jar --server.port=8080 --rocketmq.config.namesrvAddr=192.168.43.51:9876;192.168.43.227:9876;192.168.43.9:9876**

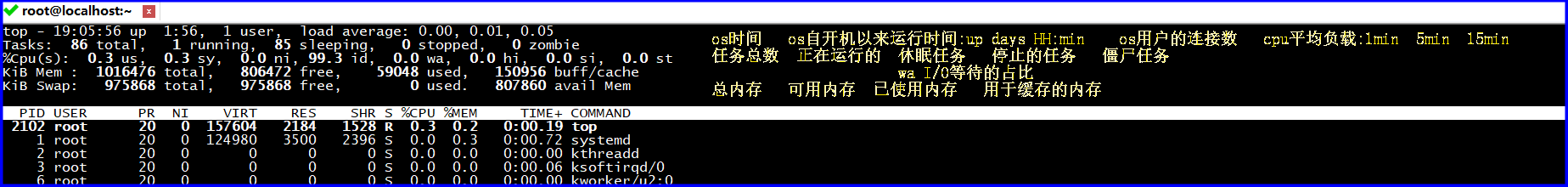
**//然后通过浏览器访问该虚拟机的8080端口即可看到工作台**

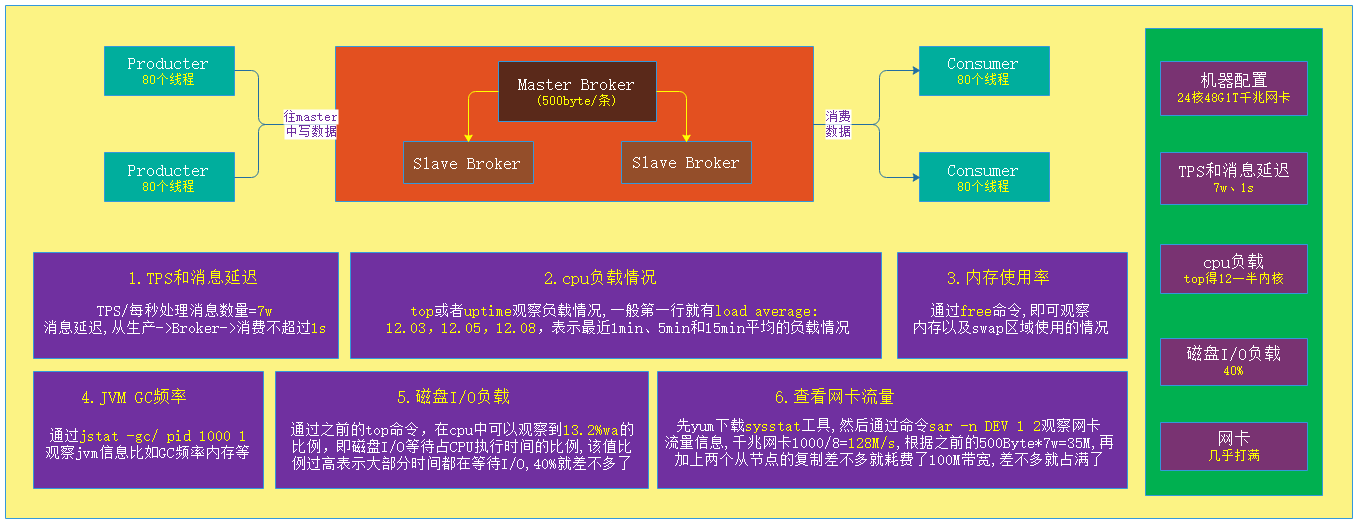


### 对os内核参数和jvm参数进行调整



### 对小规模的RocketMQ集群进行压测

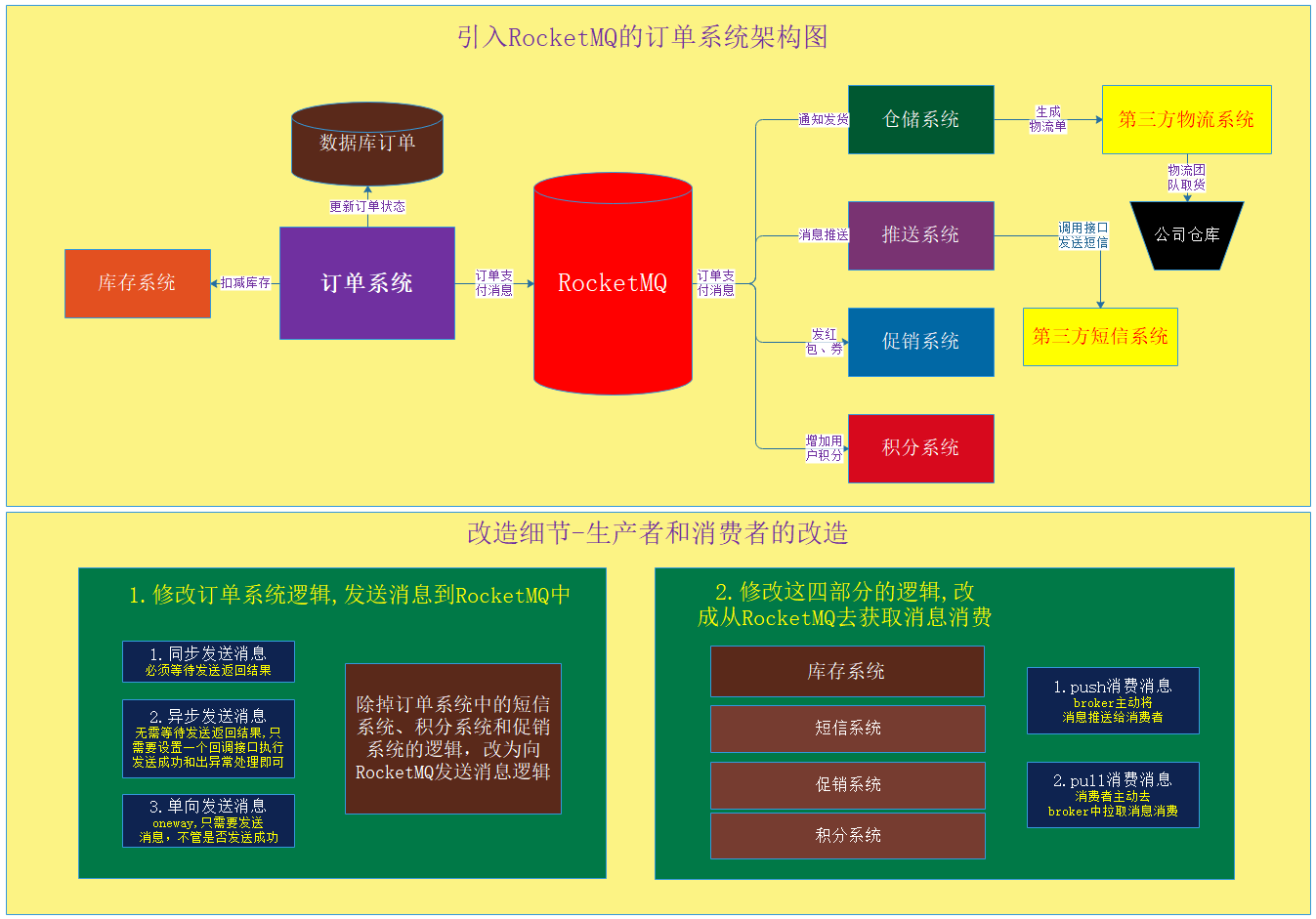




# 基于RocketMQ完成订单系统的架构升级改造

## 异步化提升性能和解耦：基于MQ实现订单系统核心流程异步化改造

### 引入RocketMQ的订单系统架构

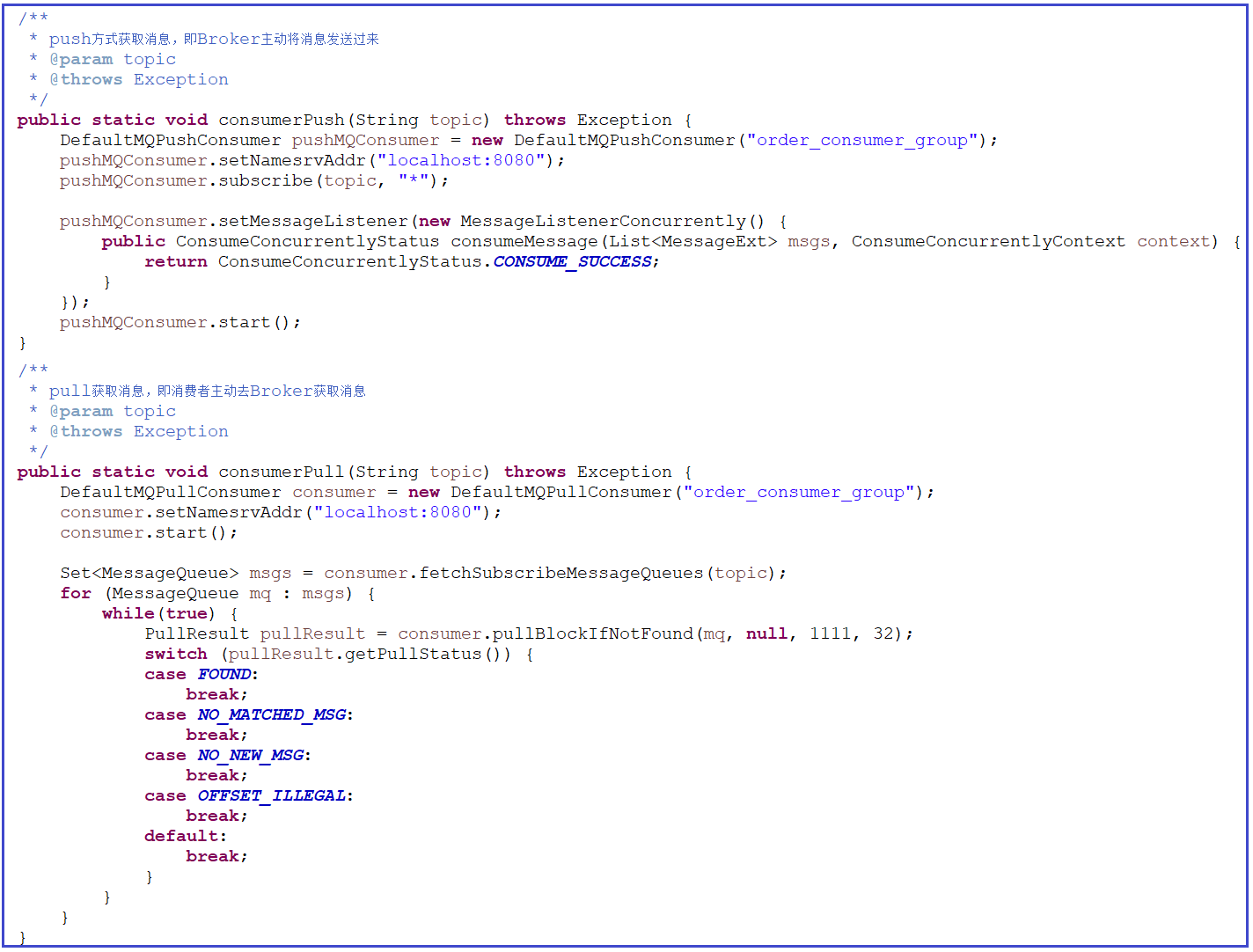
****

### 引入RocketMQ后的生产者和消费者代码

****

### RocketMQ的三种生产消息和两种消费消息方式



****

## 大数据团队：基于MQ实现将订单数据同步给大数据团队数据库

