**4.X文档 RocketMQ**

**消息构成：** Topic, 必填项。在RocketMQ-JMS中，createQueue是不支持的，此外messageSelector也不支持。body ，properties ，transactionId 。

Tag：Topic 是一级分类，而 Tag 可以理解为是二级分类。使用 Tag 可以实现对 Topic 中的消息进行过滤。Tag 可以用来区分同一个 Topic 下相互关联的消息，例如全集和子集的关系、流程先后的关系。

Keys：在业务层面的设置唯一标识码 keys 字段，方便将来定位消息丢失问题**A screen shot of a computer program

AI-generated content may be incorrect.**

Producer：消息的发送者，需要注意的是，生产环境中不同消息类型需要使用不同的主题。必须填写生产组的名称。 同步发送：消息发送方发出一条消息后，会在收到服务端同步响应之后才发下一条消息。

异步发送：送方发出一条消息后，不等服务端返回响应，接着发送下一条消息。使用到SendCallback接口，在send时引用。

单向模式发送：只发送请求不等待应答

顺序消息：对于一个指定的Topic，消息严格按照先进先出（FIFO）的原则进行消息发布和消费。

延迟消息： 延迟一定时间后才投递到Consumer进行消费

批量消息： 将一些消息聚成一批以后进行发送

事务消息： 对本地事务执行结果二次确认，Commit则投递，否则不投递。超时后，服务端会发起消息回查。

Consumer：每个消费者必须指定一个消费组。ConsumerGroup：两种模式：集群模式,消息只消费一次。 同一个 ConsumerGroup 中的 Consumer 实例是负载均衡消费 ； 广播模式：每条消息被消费组的每个消费者处理。从消费位点来说，针对于一个消费组，在集群模式下，消费位点是由客户端提给交服务端保存的，在广播模式下，消费位点是由客户端自己保存的

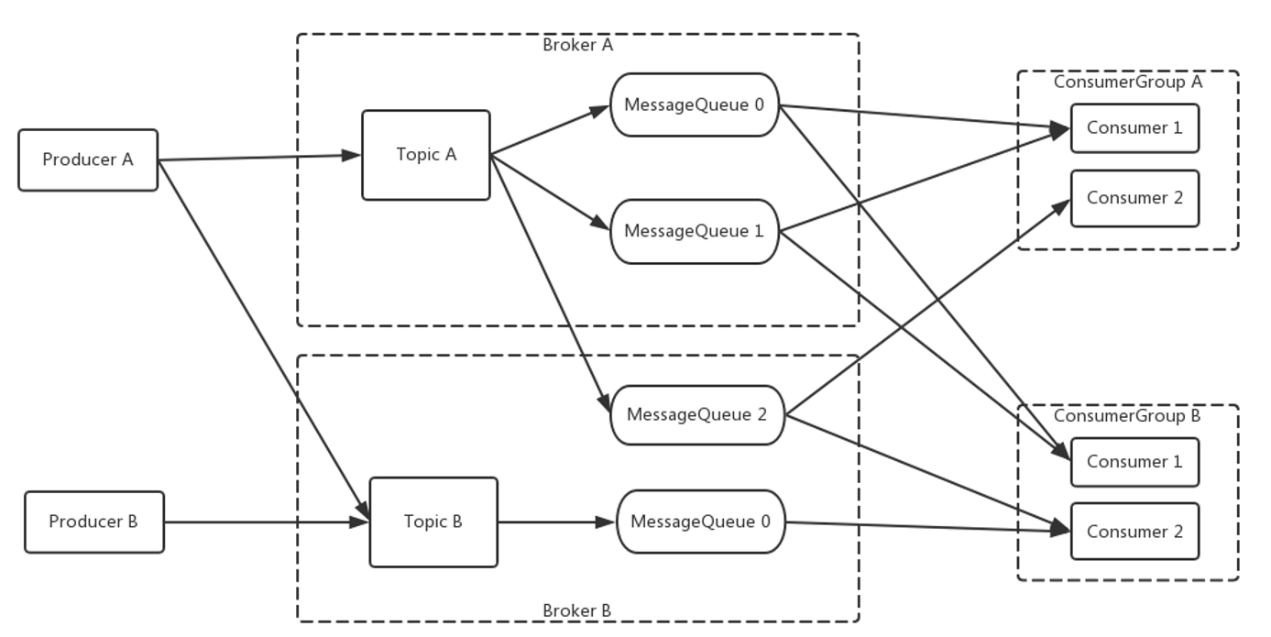
消费模式：Push是服务端主动推送消息给客户端；及时性较好，大量消息到客户端时，会导致客户端消息堆积甚至崩溃。

消息重试：Consumer消费某条消息失败，则RocketMQ会在重试间隔时间后，将消息重新投递给Consumer消费，若达到最大重试次数后消息还没有成功被消费，则消息将被投递至死信队列

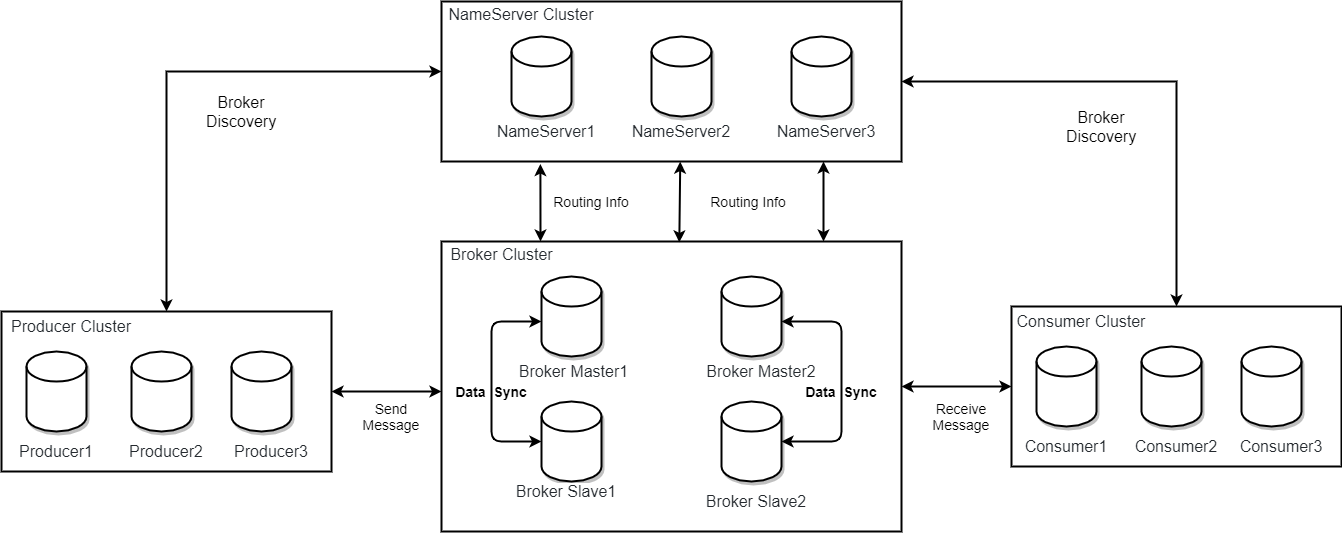
Pull是客户端需要主动到服务端取数据。优点是客户端可以依据自己的消费能力进行消费，拉取的频率需要控制，频繁容易造成服务端和客户端的压力，拉取间隔长又容易造成消费不及时。

Apache RocketMQ既提供了Push模式也提供了Pull模式

Topic : 每条消息只能属于一个Topic，是RocketMQ进行消息订阅的基本单位。

为了支持高并发和水平扩展，需要对 Topic 进行分区，在 RocketMQ 中这被称为队列。一个 Topic 可能有多个队列，并且可能分布在不同的 Broker 上

RocketMQ-JMS连接时，必须设置ProducerID和ConsumerID



NameServer：一个简单的 Topic 路由注册中心，支持 Topic、Broker 的动态注册与发现。无状态，各实例间相互不进行信息通讯，Broker是向每一台NameServer注册自己的路由信息。也可用于客户端查询的队列信息。通过NameServer就可以知道整个Broker集群的路由信息，从而进行消息的投递和消费。

Broker：主要负责消息的存储、投递和查询以及服务高可用保证。Broker 与 NameServer 集群中的所有节点建立长连接，保持心跳。在 Master-Slave 架构中，Broker 分为 Master 与 Slave，一个Master可以对应多个Slave，但是一个Slave只能对应一个Master。

Producer 与 NameServer 集群中的其中一个节点建立长连接，定期从 NameServer 获取Topic路由信息，并向提供 Topic 服务的 Master 建立长连接，且定时向 Master 发送心跳。Producer 完全无状态

Consumer 与 NameServer 集群中的其中一个节点建立长连接，定期从 NameServer 获取 Topic 路由信息，并向提供 Topic 服务的 Master、Slave 建立长连接，且定时向 Master、Slave发送心跳。Consumer 既可以从 Master 订阅消息，也可以从Slave订阅消息。

启动流程：

NameServer

Broker

创建 Topic，需要指定该 Topic 要存储在哪些 Broker 上，也可以在发送消息时自动创建。

发送

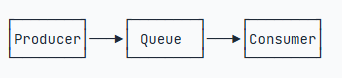
消费

ActiveMQ:

完全基于JMS 规范实现的一个消息中间件产品。

JMS的两个相互独立的消息模型：

Queue，端到端： 每条消息只被一个消费者接收



Topic，Pub/Sub: 广播式消费

A diagram of a diagram

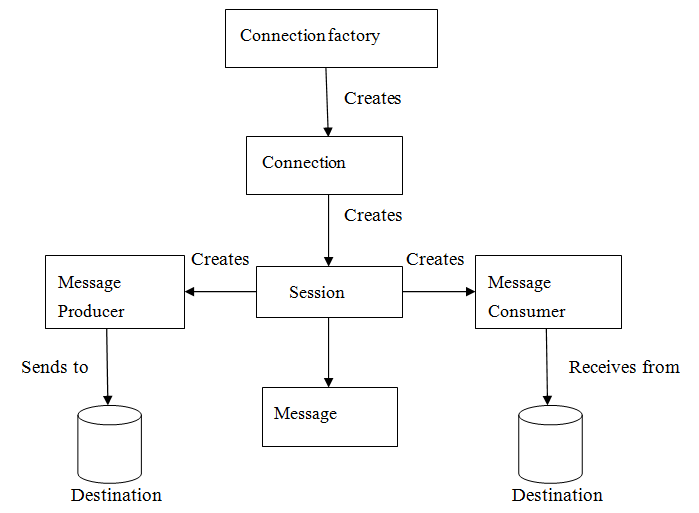
AI-generated content may be incorrect.

消息接收：

同步接收：Consumer调用 receive()方法。在receive()中，消息未到达或在到达指定时间之前，方法会阻塞，直到消息可用。

异步接收： 使用到MessageListener。JMS服务提供者（ActiveMQ ）会通过调用监听器的onMessage()递送消息。

JMS应用程序：



JMS管理对象（Administered objects）：是预先配置的JMS对象，由ActiveMQ为使用JMS的客户端创建，主要有两个被管理的对象，ConnectionFactory，Destination

ConnectionFactory：创建了ActiveMQ和客户端之间的连接

Destination：有两个实现，Topic和Queue，对应两种消息模型。

Connection ：要注意close，否则占用资源或者内存泄露。

Session：用于生产和消费消息，可以创建出消息生产者和消息消费者

MessageProducer，MessageConsumer ：可以为目的地、队列或话题创建生产者和消费者。

JMS消息结构：

消息头：JMS消息头预定义了若干字段用于客户端与ActiveMQ之间识别和发送消息

消息属性：可以给消息设置自定义属性，这些属性主要是提供给应用程序的

消息体：在消息体中，JMS API定义了五种类型的消息格式，Text，Object，Bytes，Stream，Map。

Rabbit