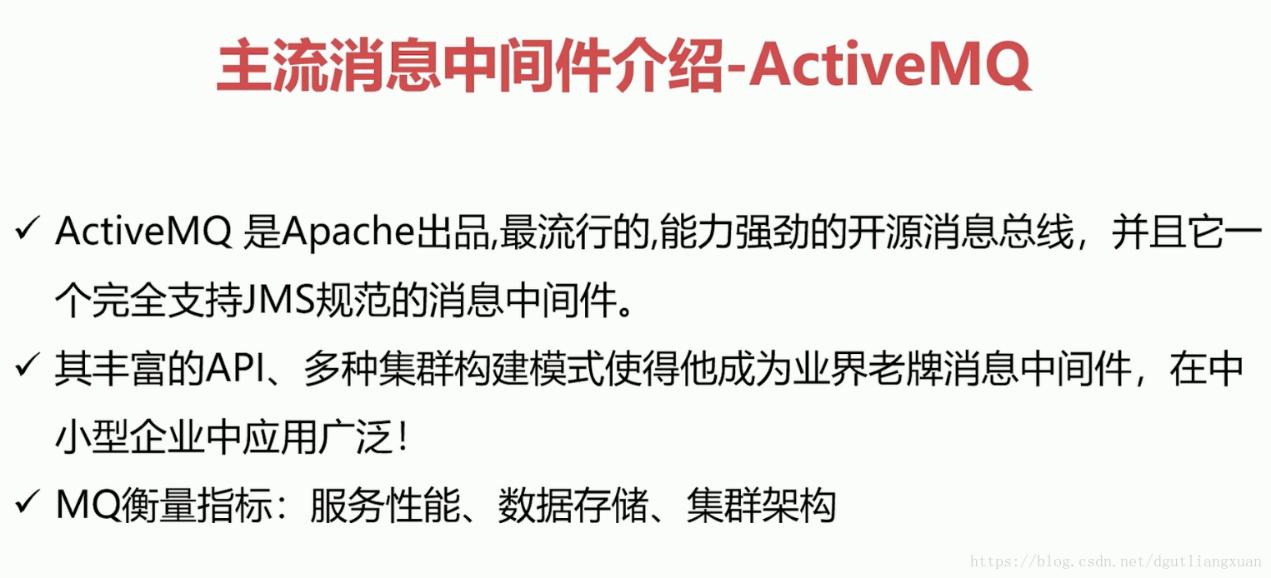
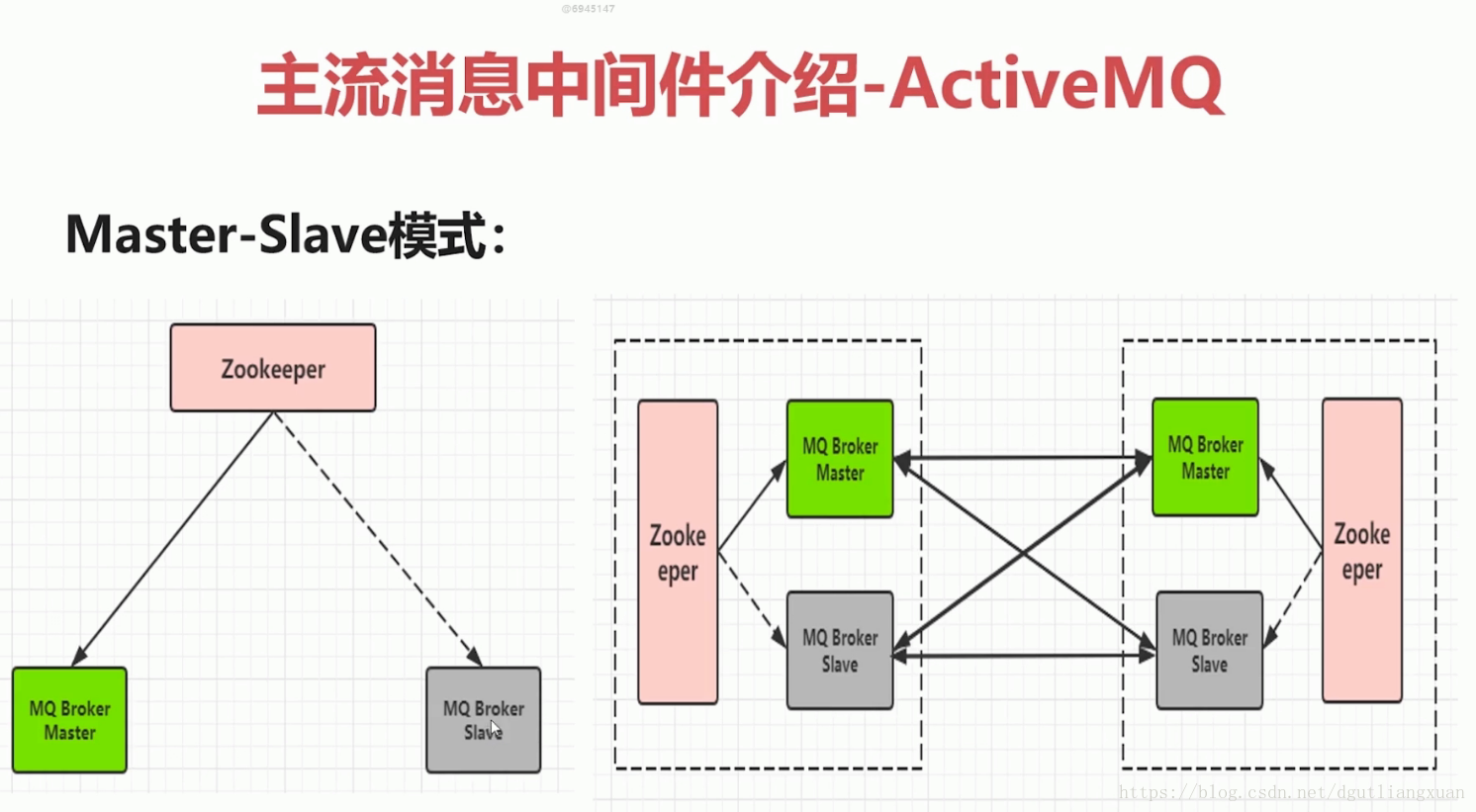
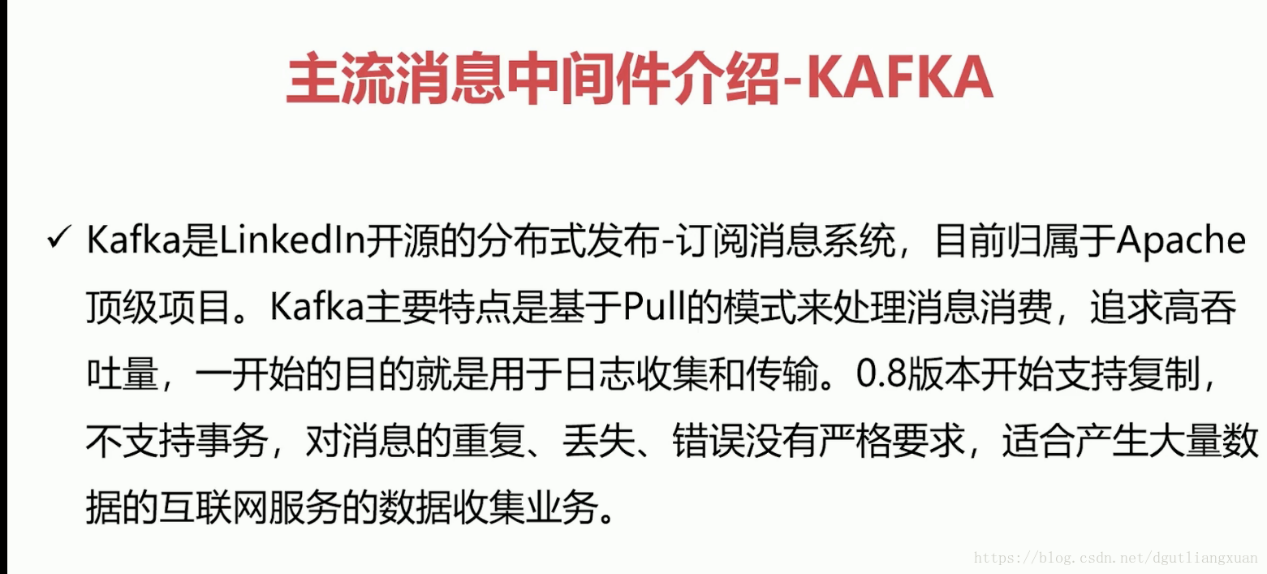
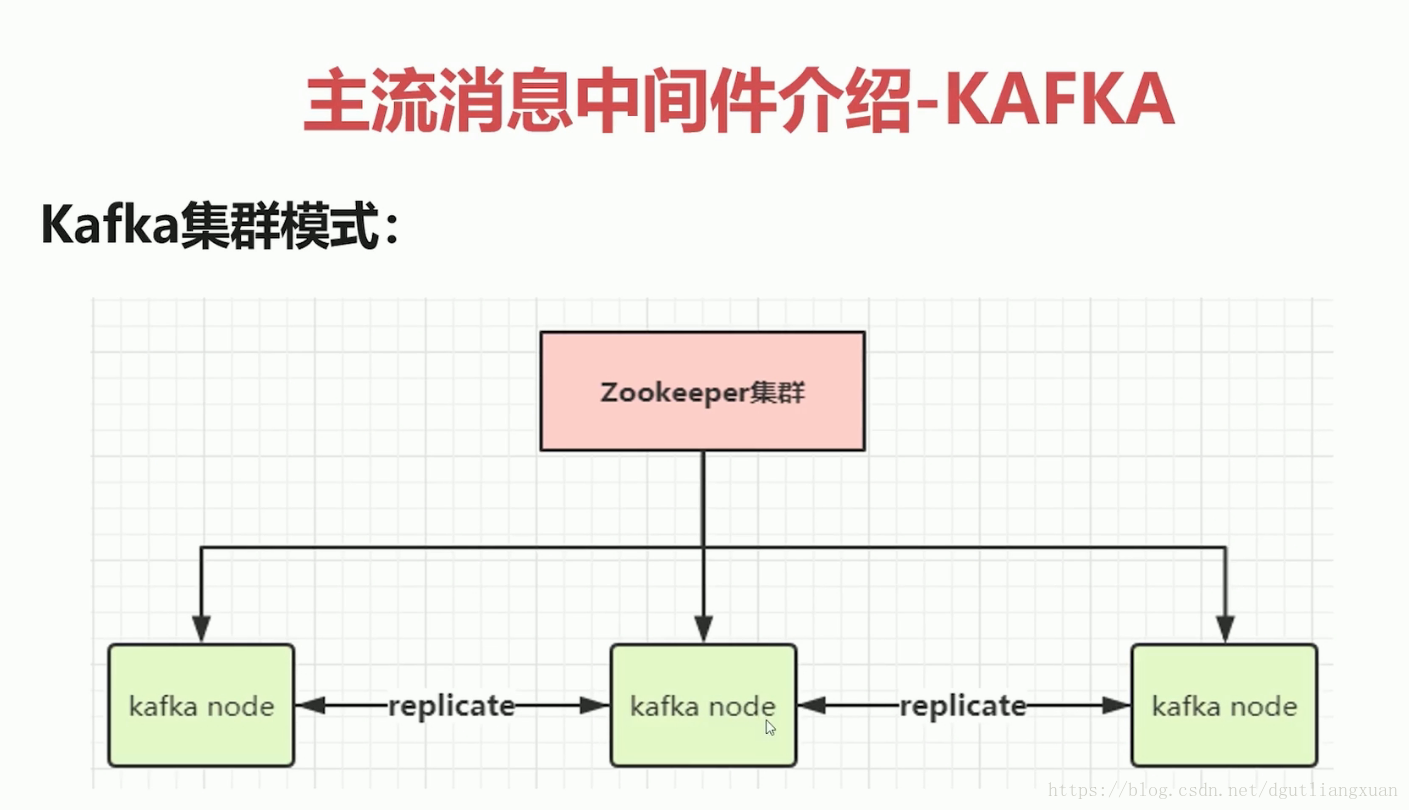
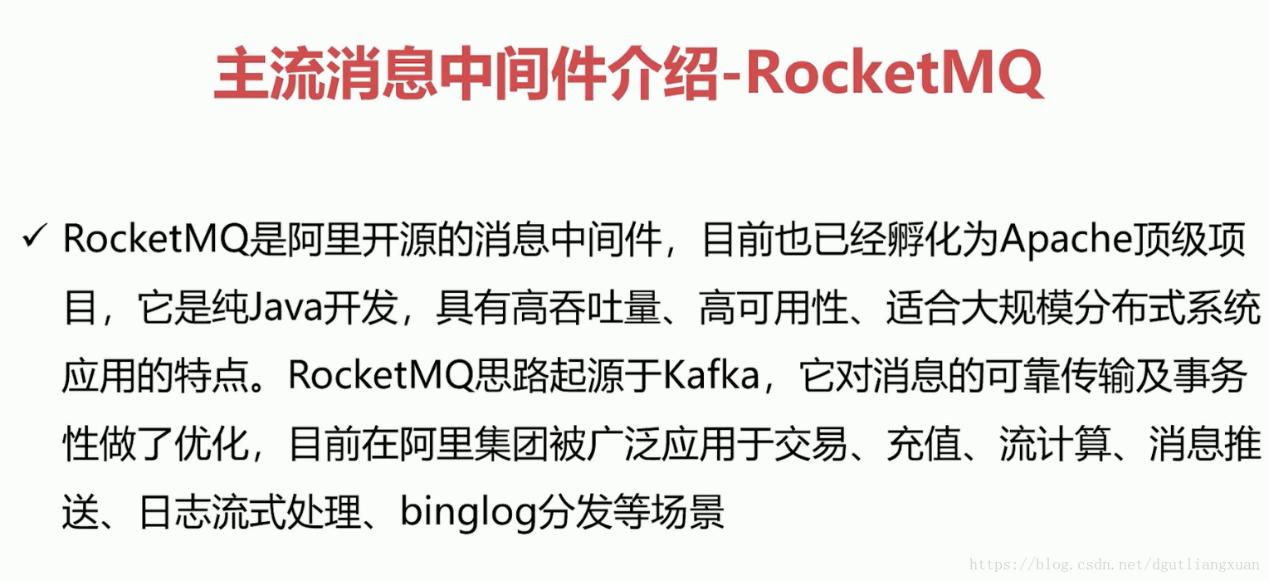
一．主流消息中间件

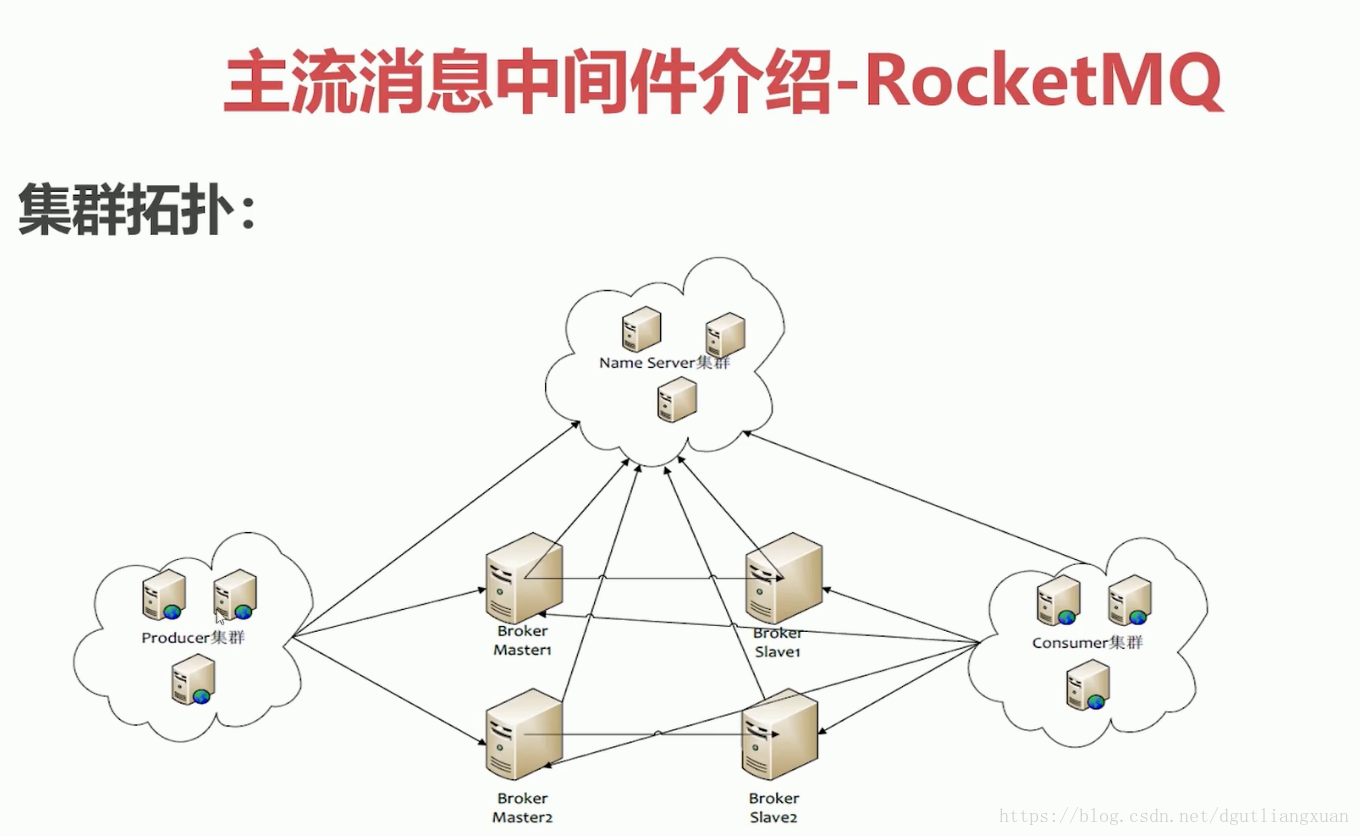


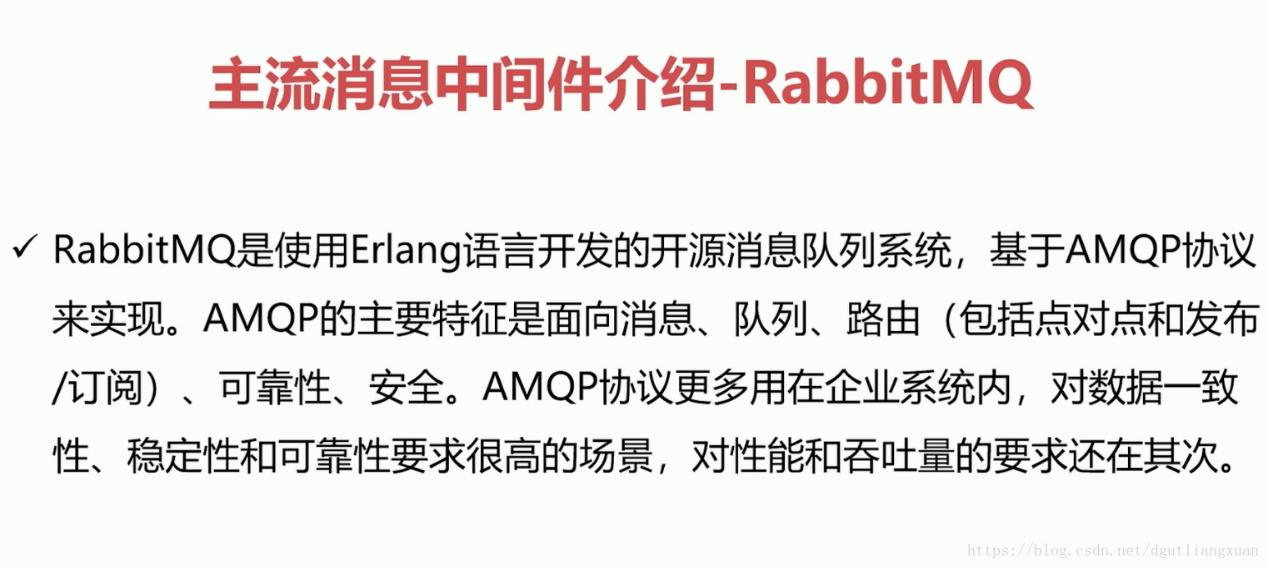


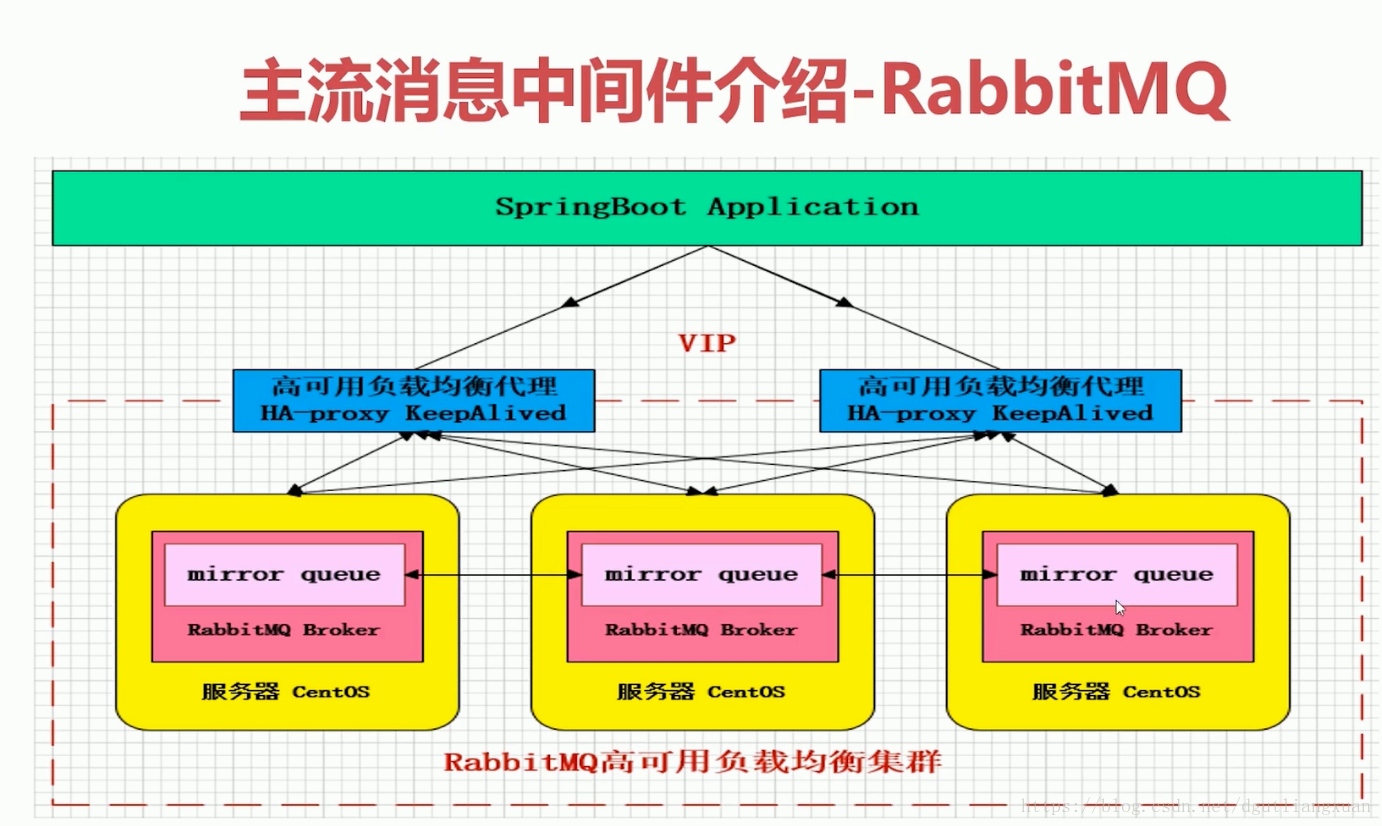




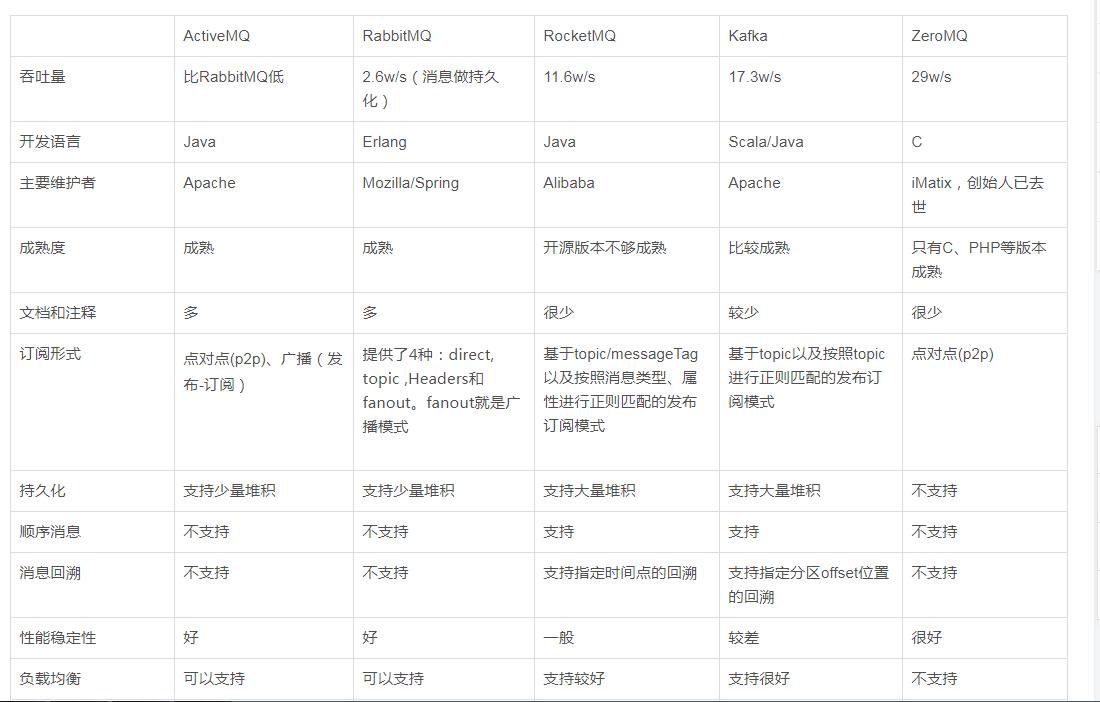








**基本对比信息：**



二、消息不丢失、不存在重复发送

根据阿里的测试结论，RocketMQ和Kafka在 消息不丢失 方面，做得差不多，服务器基本上不会丢失数据。结论如下：

1、在Broker进程被Kill的场景（在消息收发过程中，利用Kill -9 命令使Broker进程终止）， Kafka和RocketMQ都不丢消息，可靠性都比较高。

2、在宿主机掉电的场景，在“同步刷盘”策略下，Kafka与RocketMQ均能做到不丢消息，在“异步刷盘”策略下，Kafka和RMQ都无法保证掉电后不丢消息。

高并发

毋庸置疑，RabbitMQ 最高，原因是它的实现语言是天生具备高并发高可用的erlang 语言。

比较关注的比较， RabbitMQ 和 Kafka

RabbitMq 比Kafka 成熟，在可用性上，稳定性上，可靠性上，  [RabbitMq](https://www.sojson.com/tag_rabbitmq.html" \o "RabbitMq" \t "_blank)  胜于  [Kafka](https://www.sojson.com/tag_kafka.html)  （理论上）。

另外，Kafka 的定位主要在日志等方面， 因为Kafka 设计的初衷就是处理日志的，可以看做是一个日志（消息）系统一个重要组件，针对性很强，所以 如果业务方面还是建议选择 RabbitMq 。

还有就是，Kafka 的性能（吞吐量、TPS ）比RabbitMq 要高出来很多。

选型最后总结：

如果我们系统中已经有选择  [Kafka](https://www.sojson.com/tag_kafka.html)  ，或者   [RabbitMq](https://www.sojson.com/tag_rabbitmq.html" \o "RabbitMq" \t "_blank)  ，并且完全可以满足现在的业务，建议就不用重复去增加和造轮子。

可以在  [Kafka](https://www.sojson.com/tag_kafka.html)  和   [RabbitMq](https://www.sojson.com/tag_rabbitmq.html" \o "RabbitMq" \t "_blank)  中选择一个适合自己团队和业务的，这个才是最重要的。但是毋庸置疑现阶段，综合考虑没有第三选择。