Kafka VS RabbitMQ

二者均是async方式消息传递，双端不必同时在线

Message Queue Pattern (each Message can only be consumed by one consumer)
Pub/Sub Pattern

RabbitMQ: 开源的分布式消息代理（Message Broker）

```
desc: 处理复杂的路由场景，支持多种消息协议（AMQP，MQTT，and STOMP），支持分布式部署及HA,社区庞大，
    可以通过Pivot进行商业支持
arch: producers,exchanges,queues,consumers

使用模式: producer发送message到exchange，然后exchange路由到queue或者路由到其他exchange，
      然后consumer连接到queue进行消费。

queue: 是一个顺序的数据结构， producer生成消息到queue的tail端（尾部），consumer从queue的head部分进行消费 ，即先进先出

message-exchange: producer可以发送消息到四种exchange:
    Direct Exchange: 根据消息携带的routing key进行路由，routing key是字符串，
                     中间可以是英文句号分隔。

        Fanout-Exchange: 路由消息到所有的可用的queue，是一种广播机制，此时，若消息携带routing key，则会被忽略

      Topic Exchange: 根据routing key的匹配情况可能路由到一个或者多个queue

        Header Exchange: 根据message的消息头进行路由，其中消息头中可以携带比routing key更多的属性信息
```

Kafka:
开源的分布式事件流平台，linkedIn最初用于跟踪website的activity，当下主要用于building real-time data pipelines and streaming apps

选用的依据:

```
Scalability: 分布式架构利于水平扩展 ，
performance: 秒级百万级的数据处理能力，
flexibility: 面向各种系统的系统对接，有广阔的API支持
available: 负载均衡与replication
```

arch: producers,consumers，clusters,brokers,topics,paritions

消息处理模式： producer 发送records到cluster，cluster存储这些record，然后发送到consumers。
cluster中的每个节点都是一个broker，每个节点存储数据直到被consumer消费。

topics:

```
    不再是queue，kafka中使用topic,一个topic是一个单个record构成的数据流，
    就如filesystem中的一个文件，每个topic会被分割成partitions，partition是有一系列
    不可变的record构成，每个record有一个序列ID,称为offset.producer append records to topic partition,consumer subscribe to
```

RabbitMQ 与 Kafka的消息模式:

```
rabbit使用Exchage来路由消息到queue，而kafka使用的是多pub/sub模式（一个producer可以发送
message到一个具体的topic，然后，一个或者多个consumer—group consume message，
 consumer 与 consumer-group可以订到多个topic或event）

 kafaka的消息保留：只要决定将消息保留在分区中，消费者就可以重复读取以前存储的消息，
 这使得event sourcing and log aggregation称为可能。
```

安全：

与 Kafka 一样，RabbitMQ 支持 TLS 加密、SASL 认证和 RBAC。除了 **CLI** 工具之外，
RabbitMQ 还提供了一个基于浏览器的 API，用于管理和监控用户和队列。
还可以使用开源或商业工具来监控来自节点、集群、队列等的指标.

Apache Kafka 用例

一些最好的 Kafka 用例利用了该平台的高吞吐量和流处理能力。

高吞吐量活动跟踪：Kafka 可用于各种大容量、高吞吐量的活动跟踪应用程序。例如，您可以使用 Kafka
来跟踪网站活动（其原始用例）、从物联网传感器获取数据、在医院环境中监控患者或密切关注运输情况。

流处理：Kafka 使您能够实现基于事件流的应用程序逻辑。您可以保持对事件类型的运行计数或
计算持续几分钟的事件过程中的平均值。例如，如果您有一个包含自动温度计的 IoT 应用程序，
可以随着时间的推移跟踪平均温度，并在读数偏离目标温度范围时触发警报。

事件溯源：Kafka 可用于支持事件溯源，其中应用程序状态的更改存储为事件序列。
因此，例如，您可以将 Kafka 与银行应用程序一起使用。如果帐户余额以某种方式损坏，
可以根据存储的交易历史重新计算余额。

日志聚合：类似于事件溯源，您可以使用 Kafka 收集日志文件并将它们存储在一个集中的地方。
这些存储的日志文件然后可以为您的应用程序提供单一的真实来源.

总结：

Kafka 和 RabbitMQ 都允许您推送和拉取消息，并在消费者忙碌或不可用时缓冲消息。
两者还提供了一种一次获取多个消息的方法。对于 RabbitMQ，这被称为“pre-fetching”，
而对于 Kafka，它被称为以“batch size”处理消息.

RabbitMQ 是一个通用的消息代理，支持包括 MQTT、AMQP 和 STOMP 在内的协议。
它可以处理高吞吐量的用例，例如在线支付处理。它可以处理后台作业或充当微服务之间的消息代理。

Kafka 是为高入口数据重放和流而开发的消息总线。 Kafka 是一个持久的消息代理，
它使应用程序能够处理、持久化和重新处理流数据。 Kafka 有一种简单的路由方法，
它使用路由键将消息发送到主题.

二者是如何处理消息的（How Do They Handle Messaging）：

Tool	Apache Kafka	RabbitMQ
Message Ordering	Provides message ordering because of its partitions. Messages are sent to topics by message key.	Not supported.
Message Lifetime	Kafka is a log, which means that messages are always there. You can manage this by specifying a message retention policy.	RabbitMQ is a queue, so messages are done away with once consumed, and acknowledgment is provided.
Delivery Guarantees	Retains order only inside a partition. In a partition, Kafka guarantees that the whole batch of messages either fails or passes.	Doesn't guarantee atomicity, even in relation to transactions involving a single queue.
Message Priorities	N/A	In RabbitMQ, you can specify message priorities, and consumed message with high priority at the onset.

性能：

Apache Kafka:

Kafka offers much higher performance than message brokers like RabbitMQ. It uses sequential disk I/O to boost performance, making it a suitable option for implementing queues. It can achieve high throughput (millions of messages per second) with limited resources, a necessity for big data use cases.

RabbitMQ:

RabbitMQ can also process a million messages per second but requires more resources (around 30 nodes). You can use RabbitMQ for many of the same use cases as Kafka, but you'll need to combine it with other tools like Apache Cassandra.

用例：

Apache Kafka Use Cases

Apache Kafka provides the broker itself and has been designed towards stream processing scenarios. Recently, it has added Kafka Streams, a client library for building applications and microservices. Apache Kafka supports use cases such as metrics, activity tracking, log aggregation, stream processing, commit logs, and event sourcing.

The following messaging scenarios are especially suited for Kafka:

- Streams with complex routing, throughput of 100K/sec events or more, with “at least once” partitioned ordering.
- Applications requiring a stream history, delivered in “at least once” partitioned ordering. Clients can see a “replay” of the event stream.
- Event sourcing, modeling changes to a system as a sequence of events.
- Stream processing data in multi-stage pipelines. The pipelines generate graphs of real-time data flows.

RabbitMQ Use Cases

RabbitMQ can be used when web servers need to quickly respond to requests. This eliminates the need to perform resource-intensive activities while the user waits for a result. RabbitMQ is also used to convey a message to various recipients for consumption or to share loads between workers under high load (20K+ messages/second).

Scenarios that RabbitMQ can be used for:

- Applications that need to support legacy protocols, such as STOMP, MQTT, AMQP, 0-9-1.
- Granular control over consistency/set of guarantees on a per-message basis
- Complex routing to consumers
- Applications that need a variety of publish/subscribe, point-to-point request/reply messaging capabilities.