1. Kafka背景：

Kafka 是一个消息系统，原本开发自 LinkedIn，用作 LinkedIn 的活动流（Activity Stream）和运营数据处理管道（Pipeline）的基础

1. kafka简介：

Kafka 是一种分布式的，基于发布 / 订阅的消息系统。主要设计目标如下：

* 1. 以时间复杂度为 O(1) 的方式提供消息持久化能力，即使对 TB 级以上数据也能保证常数时间复杂b.度的访问性能。
  2. 高吞吐率。即使在非常廉价的商用机器上也能做到单机支持每秒 100K 条以上消息的传输。
  3. 支持 Kafka Server 间的消息分区，及分布式消费，同时保证每个 Partition 内的消息顺序传输。

同时支持离线数据处理和实时数据处理。

* 1. Scale out：支持在线水平扩展。

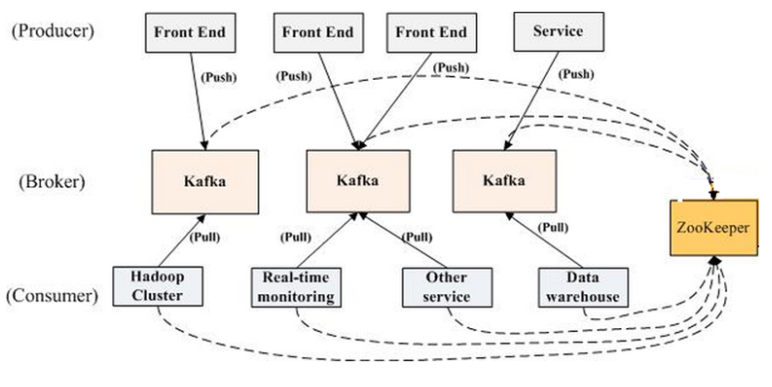
1. 专用术语：
   1. Broker：Kafka 集群包含一个或多个服务器，这种服务器被称为 broker。
   2. Topic：每条发布到 Kafka 集群的消息都有一个类别，这个类别被称为 Topic。（物理上不同 Topic 的消息分开存储，逻辑上一个 Topic 的消息虽然保存于一个或多个 broker 上，但用户只需指定消息的 Topic 即可生产或消费数据而不必关心数据存于何处）。
   3. Partition：Partition 是物理上的概念，每个 Topic 包含一个或多个 Partition。
   4. Producer：负责发布消息到 Kafka broker。
   5. Consumer：消息消费者，向 Kafka broker 读取消息的客户端。
   6. Consumer Group：每个 Consumer 属于一个特定的 Consumer Group（可为每个 Consumer 指定 group name，若不指定 group name 则属于默认的 group）。
2. 交互流程：

Kafka 是一个基于分布式的消息发布-订阅系统，它被设计成快速、可扩展的、持久的。与其他消息发布-订阅系统类似，Kafka 在主题当中保存消息的信息。生产者向主题写入数据，消费者从主题读取数据。由于 Kafka 的特性是支持分布式，同时也是基于分布式的，所以主题也是可以在多个节点上被分区和覆盖的。

信息是一个字节数组，程序员可以在这些字节数组中存储任何对象，支持的数据格式包括 String、JSON、Avro。Kafka 通过给每一个消息绑定一个键值的方式来保证生产者可以把所有的消息发送到指定位置。属于某一个消费者群组的消费者订阅了一个主题，通过该订阅消费者可以跨节点地接收所有与该主题相关的消息，每一个消息只会发送给群组中的一个消费者，所有拥有相同键值的消息都会被确保发给这一个消费者。

Kafka 设计中将每一个主题分区当作一个具有顺序排列的日志。同处于一个分区中的消息都被设置了一个唯一的偏移量。Kafka 只会保持跟踪未读消息，一旦消息被置为已读状态，Kafka 就不会再去管理它了。Kafka 的生产者负责在消息队列中对生产出来的消息保证一定时间的占有，消费者负责追踪每一个主题 (可以理解为一个日志通道) 的消息并及时获取它们。基于这样的设计，Kafka 可以在消息队列中保存大量的开销很小的数据，并且支持大量的消费者订阅。

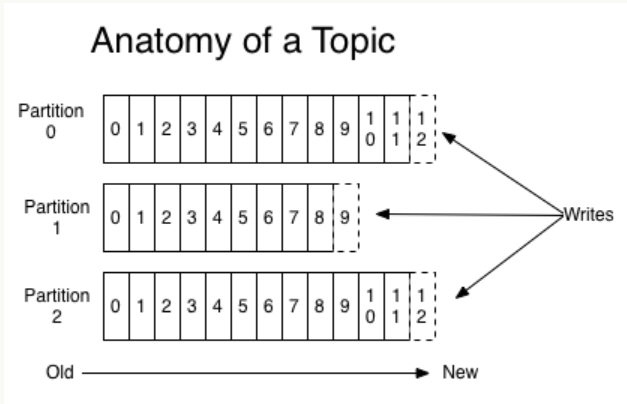
1. 拓扑结构：



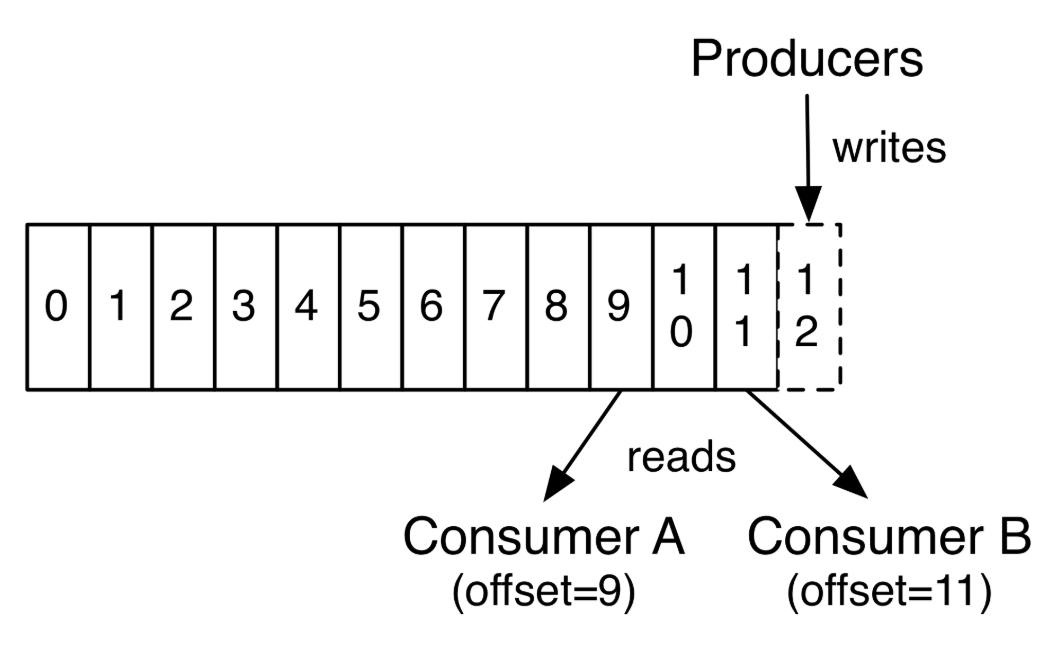
一个典型的 Kafka 集群中包含若干 Producer（可以是 web 前端产生的 Page View，或者是服务器日志，系统 CPU、Memory 等），若干 broker（Kafka 支持水平扩展，一般 broker 数量越多，集群吞吐率越高），若干 Consumer Group，以及一个Zookeeper集群。Kafka 通过 Zookeeper 管理集群配置，选举 leader，以及在 Consumer Group 发生变化时进行 rebalance。Producer 使用 push 模式将消息发布到 broker，Consumer 使用 pull 模式从 broker 订阅并消费消息。

1. Topic & Partition & log：

主题（topic）是发布的数据流的类别或名称。在逻辑上可以被认为是一个 queue，每条消费都必须指定它的 Topic，可以简单理解为必须指明把这条消息放进哪个 queue 里。为了使得 Kafka 的吞吐率可以线性提高，物理上把 Topic 分成一个或多个 Partition，每个 Partition 在物理上对应一个文件夹，该文件夹下存储这个 Partition 的所有消息和索引文件。若创建 topic1 和 topic2 两个 topic，且分别有 13 个和 19 个分区，则整个集群上会相应会生成共 32 个文件夹。topic在Kafka中，总是支持多订阅者的; 也就是说，主题可以有零个，一个或多个消费者订阅写到相应主题的数据. 对应每一个主题，Kafka集群会维护像一个如下这样的分区的日志:



每个分区都是一个有序的，不可变的，并且不断被附加的记录序列，也就是一个结构化提交日志（commit log）.为了保证唯一标性识分区中的每个数据记录，分区中的记录每个都会被分配一个一个叫做偏移（offset）顺序的ID号. 通过一个可配置的保留期，Kafka集群会保留所有被发布的数据，不管它们是不是已经被消费者处理. 例如，如果保留期设置为两天，则在发布记录后的两天内，数据都可以被消费，之后它将被丢弃以释放空间。 Kafka的性能是不为因为数据量大小而受影响的，因此长时间存储数据并不成问题。



事实上，在每个消费者上保留的唯一元数据是消费者在日志中的偏移位置。这个偏移由消费者控制：通常消费者会在读取记录时线性地提高其偏移值（offset++），但实际上，由于偏移位置由消费者控制，它可以以任何顺序来处理数据记录。 例如，消费者可以重置为较旧的偏移量以重新处理来自过去的数据，或者跳过之前的记录，并从“现在”开始消费。 这种特征的组合意味着Kafka消费者非常轻量级 ——随意的开启和关闭并不会对其他的消费者有大的影响。例如，您可以使用我们的命令行工具tail来查看任何主题的内容，而无需更改任何现有消费者所消耗的内容。

日志中的分区有几个目的。 首先，它保证日志的扩展性，主题的大小不受单个服务器大小的限制。每个单独的分区大小必须小于托管它的服务器磁盘大小，但主题可能有很多分区，因此它可以处理任意数量的海量数据。第二，它可以作为并行处理的单位。

1. 数据的分配:

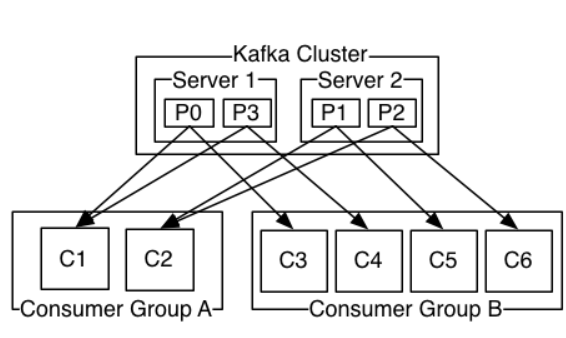
在Kafka集群中，不同分区日志的分布在相应的不同的服务器节点上，每个服务器节点处理自己分区对应的数据和请求。每个分区都会被复制备份到几个（可配置）服务器节点，以实现容错容灾。 分布在不同节点的同一个分区都会有一个服务器节点作为领导者（”leader”）和0个或者多个跟随者（”followers”）. 分区的领导者会处理所有的读和写请求，而跟随者只会被动的复制领导者.如果leader挂了, 一个follower会自动变成leader。每个服务器都会作为其一些分区的领导者，但同时也可能作为其他分分区的跟随者，Kafka以此来实现在集群内的负载平衡

1. 生产者：

生产者将数据发布到他们选择的主题。 生产者负责选择要吧数据分配给主题中哪个分区。这可以通过循环方式（round-robin）简单地平衡负载，或者可以根据某些语义分区（例如基于数据中的某些关键字）来完成

1. 消费者：

消费者们使用消费群组名称来标注自己，几个消费者共享一个组群名，每一个发布到主题的数据会被传递到每个消费者群组中的一个消费者实例。 消费者实例可以在不同的进程中或不同的机器上。 如果所有的消费者实例具有相同的消费者组，则记录将在所有的消费者实例上有效地负载平衡,每个数据只发到了一个消费者 如果所有的消费者实例都有不同的消费者群体，那么每个记录将被广播给所有的消费者进程，每个数据都发到了所有的消费者。



1. 保证：

在高可用的Kafka集群中，我们有如下的保证:

(1)生产者发送到特定主题分区的消息将按照发送的顺序进行追加。 也就是说，如果记录M1由与记录M2相同的制造者发送，并且首先发送M1，则M1将具有比M2更低的偏移并且在日志中较早出现。

(2)消费者实例观察到数据的顺序，与它们存储在日志中的顺序一致。

(3)对于具有复制因子N的主题，我们将容忍最多N-1个服务器故障，而不会丢失提交到日志的任何记录。

1. kafka作为消息系统：

Kafka的数据流概念与传统的企业消息系统相比如何？ 消息系统传统上有两种模式: 队列和发布-订阅. 在队列中，消费者池可以从服务器读取，每条记录都转到其中一个; 在发布订阅中，记录将广播给所有消费者。 这两个模型中的每一个都有优点和缺点。 排队的优点是它允许您在多个消费者实例上分配数据处理，从而可以扩展您的处理。 不幸的是，队列支持多用户，一旦一个进程读取数据就没有了。 发布订阅允许您将数据广播到多个进程，但无法缩放和扩容，因为每个消息都发送给每个订阅用户。 卡夫卡消费群体概念概括了这两个概念。 与队列一样，消费者组允许您通过一系列进程（消费者组的成员）来划分处理。 与发布订阅一样，Kafka允许您将消息广播到多个消费者组。 Kafka模型的优点是，每个主题都具有这两个属性，它可以进行缩放处理，也是多用户的，没有必要选择一个而放弃另一个。 卡夫卡也比传统的消息系统有更强大的消息次序保证。 传统队列在服务器上保存顺序的记录，如果多个消费者从队列中消费，则服务器按照存储顺序输出记录。 然而，虽然服务器按顺序输出记录，但是记录被异步传递给消费者，所以它们可能会在不同的消费者处按不确定的顺序到达。 这意味着在并行消耗的情况下，记录的排序丢失。 消息传递系统通常通过使“唯一消费者”的概念只能让一个进程从队列中消费，但这当然意味着处理中没有并行性。 卡夫卡做得更好。通过分区，在一个主题之内的并行处理，Kafka能够在消费者流程池中，即提供排序保证，也负载平衡。这是通过将主题中的分区分配给消费者组中的消费者来实现的，以便每一个分区由组中的一个消费者使用。 通过这样做，我们确保消费者是该分区的唯一读者，并按顺序消耗数据。 由于有许多分区，这仍然平衡了许多消费者实例的负载。 但是请注意，消费者组中的消费者实例个数不能超过分区的个数。

1. kafka作为存储系统：

任何允许发布消息，解耦使用消息的消息队列，都在本质上充当传输中途消息的存储系统。 卡夫卡的不同之处在于它是一个很好的存储系统。 写入Kafka的数据写入磁盘并进行复制以进行容错。 Kafka允许生产者等待写入完成的确认，这样在数据完全复制之前，写入是未完成的，并且即使写入服务器失败，也保证持久写入。 Kafka的磁盘结构使用可以很好的扩容，无论您在服务器上是否有50KB或50TB的持久数据，Kafka都能保持稳定的性能。 由于对存储花费了很多精力，并允许客户端控制其读取位置，您可以将Kafka视为，专用于高性能，低延迟的日志存储复制和传播的专用分布式文件系统。

1. kafka用于流数据处理:

仅读取，写入和存储数据流是不够的，Kafka的目的是实现流的实时处理。 在Kafka中，流处理器的定义是：任何从输入主题接收数据流，对此输入执行一些处理，并生成持续的数据流道输出主题的组件。 例如，零售应用程序可能会收到销售和出货的输入流，并输出根据该数据计算的重新排序和价格调整的输出流。 当然我们也可以直接用producer and consumer APIs在做简单的出列. 然而对于更复杂的转换，Kafka提供了一个完全集成的Streams API。这允许我们构建应用程序进行更复杂的运算，或者聚合，或将流连接在一起。 该设施有助于解决这种类型的应用程序面临的困难问题：处理无序数据，重新处理输入作为代码更改，执行有状态计算等。 Stream API基于Kafka提供的核心原语构建：它使用生产者和消费者API进行输入，使用Kafka进行有状态存储，并在流处理器实例之间使用相同的组机制来实现容错。

参考：1.简介：<https://www.ibm.com/developerworks/cn/opensource/os-cn-kafka/index.html>

2.官方文档：<http://ifeve.com/kafka-introduction/>