咕泡学院 VIP 课：分布式消息通信 Kafka(一)

课程目标

1. Kafka 产生的背景
2. kafka 的架构
3. Kafka 安装部署及集群部署
4. Kafka 的基本操作
5. Kafka 的应用

Kafka 的简介

什么是 Kafka

Kafka 是一款分布式消息发布和订阅系统，具有高性能、高吞吐量的特点而被广泛应用与大数据传输场景。它是由 LinkedIn 公司开发，使用 Scala 语言编写，之后成为 Apache 基金会的一个顶级项目。kafka 提供了类似 JMS 的特性，但是在设计和实现上是完全不同的，而且他也不是 JMS 规范的实现。

kafka 产生的背景

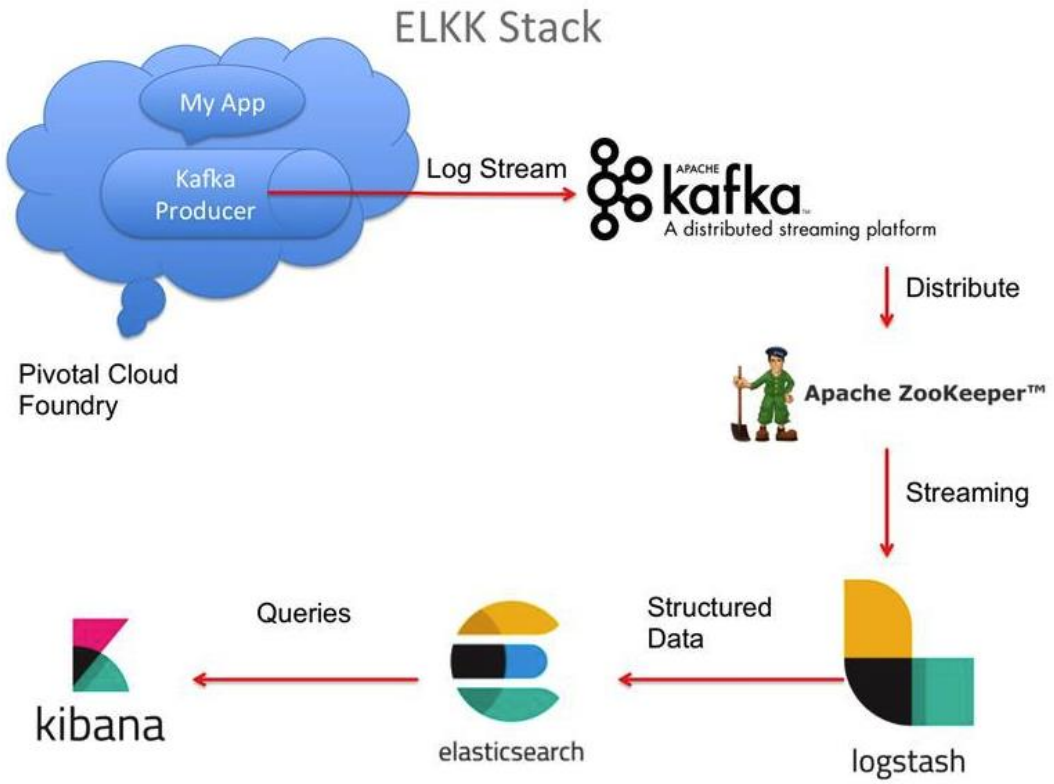
kafka 作为一个消息系统，早起设计的目的是用作 LinkedIn 的活动流（Activity Stream）和运营数据处理管道（Pipeline）。活动流数据是所有的网站对用户的使用情况做分析的时候要用到的最常规的部分,活动数据包括页面的访问量（PV）、被查看内容方面的信息以及搜索内容。这种数据通常的处理方式是先把各种活动以日志的形式写入某种文件，然后周期性的对这些文件进行统计分析。运营数据指的是服务器的性能数据（CPU、IO 使用率、请求时间、服务日志等）。

Kafka 的应用场景

由于 kafka 具有更好的吞吐量、内置分区、冗余及容错性的优点(kafka 每秒可以处理几十万消息)，让 kafka 成为了一个很好的大规模消息处理应用的解决方案。所以在企业级应用长，主要会应用于如下几个方面

Ø 行为跟踪：kafka 可以用于跟踪用户浏览页面、搜索及其他行为。通过发布-订阅模式实时记录到对应的 topic 中，通过后端大数据平台接入处理分析，并做更进一步的实时处理和监控

Ø 日志收集：日志收集方面，有很多比较优秀的产品，比如 Apache Flume，很多公司使用 kafka 代理日志聚合。日志聚合表示从服务器上收集日志文件，然后放到一个集中的平台（文件服务器）进行处理。在实际应用开发中，我们应用程序的 log 都会输出到本地的磁盘上，排查问题的话通过 linux 命令来搞定，如果应用程序组成了负载均衡集群，并且集群的机器有几十台以上，那么想通过日志快速定位到问题，就是很麻烦的事情了。所以一般都会做一个日志统一收集平台管理 log 日志用来快速查询重要应用的问题。所以很多公司的套路都是把应用日志几种到 kafka 上，然后分别导入到 es 和 hdfs 上，用来做实时检索分析和离线统计数据备份等。而另一方面，kafka 本身又提供了很好的 api 来集成日志并且做日志收集



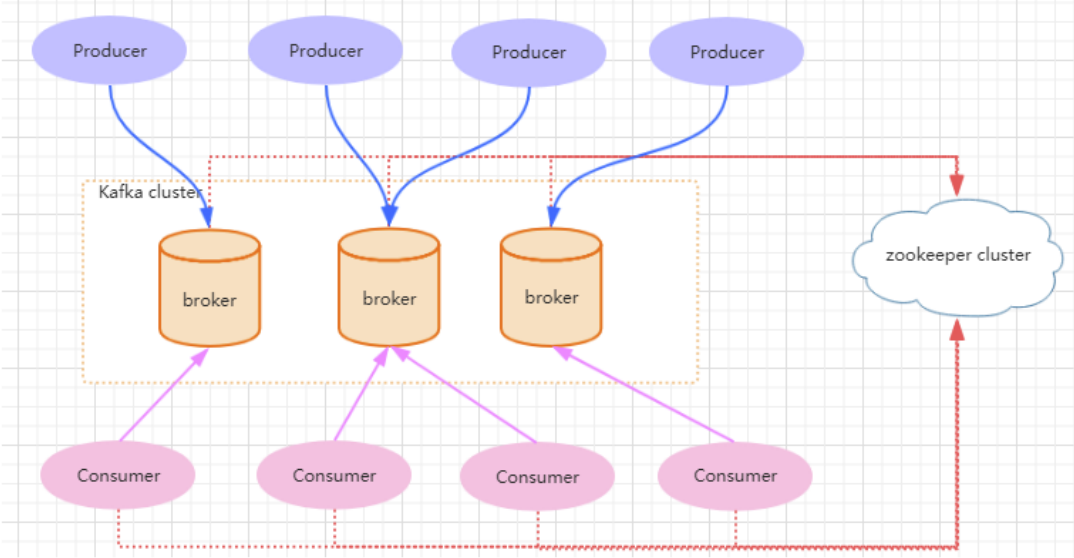
Kafka 本身的架构

一个典型的 kafka 集群包含若干 Producer（可以是应用节点产生的消息，也可以是通过 Flume 收集日志产生的事件），若干个 Broker（kafka 支持水平扩展）、若干个 Consumer Group，以及一个 zookeeper 集群。kafka 通过 zookeeper 管理集群配置及服务协同。

Producer 使用 push 模式将消息发布到 broker，consumer 通过监听使用 pull 模式从 broker 订阅并消费消息。

多个 broker 协同工作，producer 和 consumer 部署在各个业务逻辑中。三者通过 zookeeper 管理协调请求和转发。这样就组成了一个高性能的分布式消息发布和订阅系统。

图上有一个细节是和其他 mq 中间件不同的点，producer 发送消息到 broker 的过程是 push，而 consumer 从 broker 消费消息的过程是 pull，主动去拉数据。而不是 broker 把数据主动发送给 consumer



kafka 的安装部署

下载安装包

https://www.apache.org/dyn/closer.cgi?path=/kafka/1.1.0/kafka_2.11-1.1.0.tgz

安装过程

1. tar -zxvf 解压安装包

kafka 目录介绍

1. /bin 操作 kafka 的可执行脚本

2. /config 配置文件

3. /libs 依赖库目录

/logs 日志数据目录

启动/停止 kafka

1. 需要先启动 zookeeper，如果没有搭建 zookeeper 环境，可以直接运行 kafka 内嵌的 zookeeper

启动命令： bin/zookeeper-server-start.sh config/zookeeper.properties &

2. 进入 kafka 目录，运行 bin/kafka-server-start.sh {-daemon 后台启动} config/server.properties &

3. 进入 kafka 目录，运行 bin/kafka-server-stop.sh config/server.properties

Kafka 的基本操作

创建 topic

./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

Replication-factor 表示该 topic 需要在不同的 broker 中保存几份，这里设置成 1，表示在两个 broker 中保存两份

Partitions 分区数

查看 topic

./kafka-topics.sh --list --zookeeper localhost:2181

查看 topic 属性

./kafka-topics.sh --describe --zookeeper localhost:2181 --topic test

消费消息

./kafka-console-consumer.sh –bootstrap-server localhost:9092 --topic test --from-beginning

发送消息

./kafka-console-producer.sh --broker-list localhost:9092 --topic test

安装集群环境

修改 server.properties 配置

1. 修改 server.properties. broker.id=0 / 1

2. 修改 server.properties 修改成本机 IP

advertised.listeners=PLAINTEXT://192.168.11.153:9092

当 Kafka broker 启动时，它会在 ZK 上注册自己的 IP 和端口号，客户端就通过这个 IP 和端口号来连接

Kafka JAVA API 的使用

课程代码->详见 gitlab->mic-vip

配置信息分析

发送端的可选配置信息分析

acks

acks 配置表示 producer 发送消息到 broker 上以后的确认值。有三个可选项

Ø 0：表示 producer 不需要等待 broker 的消息确认。这个选项时延最小但同时风险最大（因为当 server 宕机时，数据将会丢失）。

Ø 1：表示 producer 只需要获得 kafka 集群中的 leader 节点确认即可，这个选择时延较小同时确保了 leader 节点确认接收成功。

Ø all(-1)：需要 ISR 中所有的 Replica 给予接收确认，速度最慢，安全性最高，但是由于 ISR 可能会缩小到仅包含一个 Replica，所以设置参数为 all 并不能一定避免数据丢失，

batch.size

生产者发送多个消息到 broker 上的同一个分区时，为了减少网络请求带来的性能开销，通过批量的方式来提交消息，可以通过这个参数来控制批量提交的字节数大小，默认大小是 16384byte,也就是 16kb，意味着当一批消息大小达到指定的 batch.size 的时候会统一发送

linger.ms

Producer 默认会把两次发送时间间隔内收集到的所有 Requests 进行一次聚合然后再发送，以此提高吞吐量，而 linger.ms 就是为每次发送到 broker 的请求增加一些 delay，以此来聚合更多的 Message 请求。 这个有点想 TCP 里面的 Nagle 算法，在 TCP 协议的传输中，为了减少大量小数据包的发送，采用了 Nagle 算法，也就是基于小包的等-停协议。

Ø batch.size 和 linger.ms 这两个参数是 kafka 性能优化的关键参数，很多同学会发现 batch.size 和 linger.ms 这两者的作用是一样的，如果两个都配置了，那么怎么工作的呢？实际上，当二者都配置的时候，只要满足其中一个要求，就会发送请求到 broker 上

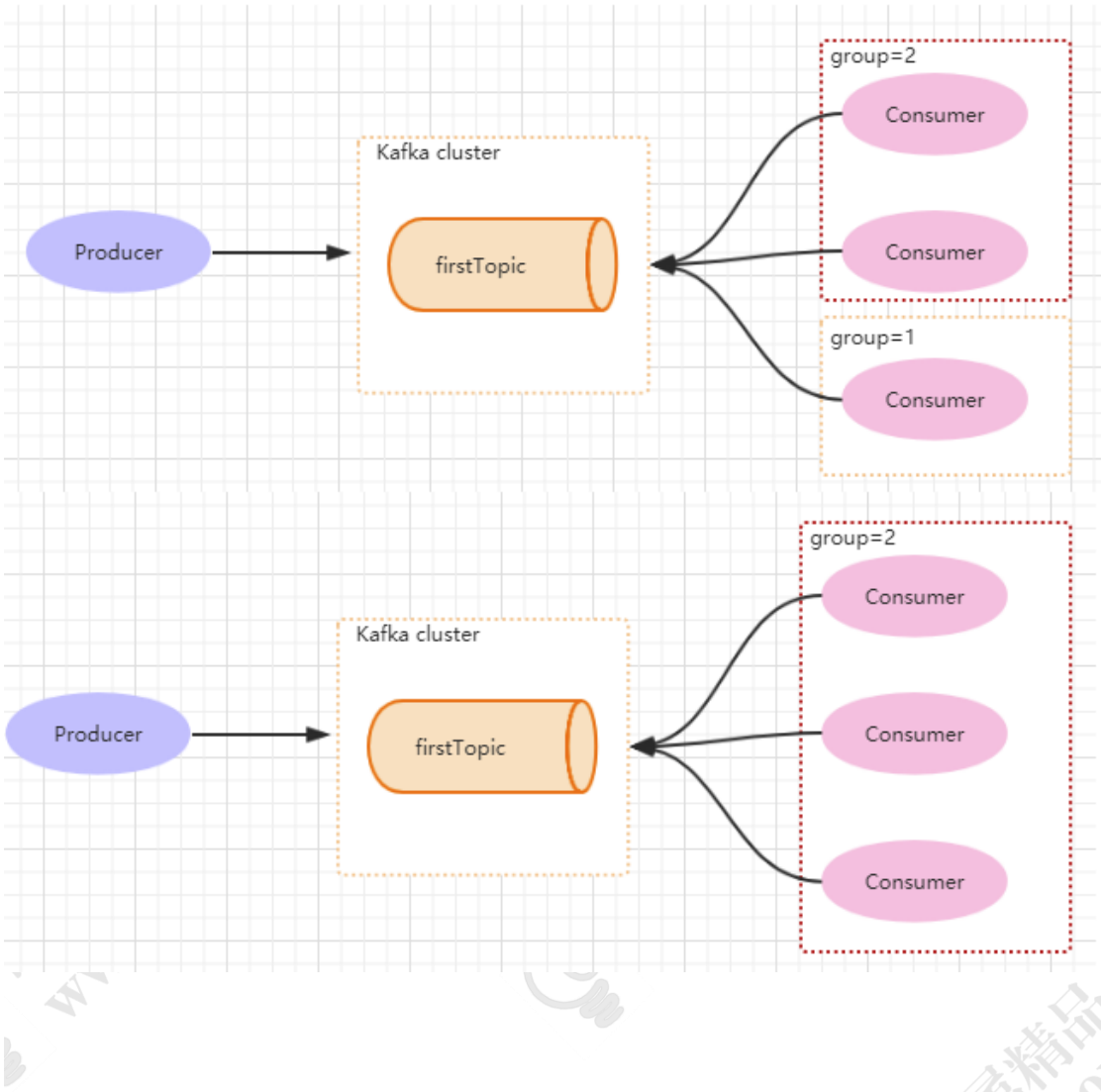
max.request.size

设置请求的数据的最大字节数，为了防止发生较大的数据包影响到吞吐量，默认值为 1MB。

消费端的可选配置分析

group.id

consumer group 是 kafka 提供的可扩展且具有容错性的消费者机制。既然是一个组，那么组内必然可以有多个消费者或消费者实例(consumer instance)，它们共享一个公共的 ID，即 group ID。组内的所有消费者协调在一起来消费订阅主题(subscribed topics)的所有分区(partition)。当然，每个分区只能由同一个消费组内的一个 consumer 来消费.如下图所示，分别有三个消费者，属于两个不同的 group，那么对于 firstTopic 这个 topic 来说，这两个组的消费者都能同时消费这个 topic 中的消息，对于此事的架构来说，这个 firstTopic 就类似于 ActiveMQ 中的 topic 概念。如右图所示，如果 3 个消费者都属于同一个 group，那么此事 firstTopic 就是一个 Queue 的概念



enable.auto.commit

消费者消费消息以后自动提交，只有当消息提交以后，该消息才不会被再次接收到，还可以配合 auto.commit.interval.ms 控制自动提交的频率。

当然，我们也可以通过 consumer.commitSync()的方式实现手动提交

auto.offset.reset

这个参数是针对新的 groupid 中的消费者而言的，当有新 groupid 的消费者来消费指定的 topic 时，对于该参数的配置，会有不同的语义

auto.offset.reset=latest 情况下，新的消费者将会从其他消费者最后消费的 offset 处开始消费 Topic 下的消息

auto.offset.reset= earliest 情况下，新的消费者会从该 topic 最早的消息开始消费

auto.offset.reset=none 情况下，新的消费者加入以后，由于之前不存在 offset，则会直接抛出异常。

max.poll.records

此设置限制每次调用 poll 返回的消息数，这样可以更容易的预测每次 poll 间隔要处理的最大值。通过调整此值，可以减少 poll 间隔

spring-kafka 集成

详见代码-gitlab->vip-project