APACHE KAFKA

After Apache Spark, there was another BigData tool, Kafka, which was widely used. Our internship coach said that it was also important to learn Kafka. So I will research and apply Kafka for a while.First of all, if you want to talk a little bit about:

Apache Kafka is a structure that holds records in a structure similar to the log record and presents these records in the form of a messaging queue (MQ) to other systems.It provides real-time data flow with high performance and low latency.

Messaging System

The messaging system allows the data to be transferred from one application to another. The concept called Distributed Messaging is actually called a reliable queue structure (Message Queue). The messaging systems can be 2 buildings:

**1. Point-to-Point**

Although there may be more than one recipient in this structure, a message is only transmitted to a recipient

**2. Publisher-Subscriber**

In this structure, messages are grouped under one category. Buyers can take multiple categories of data. In this system, the application that sends the message is named Publisher and the recipient application is called Subscriber.

Apache Spark’tan sonra sırada diğer bir bigData tool’u olan ve çokça kullanılan Kafka vardı.Staj koçumuz Kafka öğrenmemizin de önemli olduğunu söyledi.O yüzden bir süre Kafka üzerine araştırma ve uygulamalar yapacağım.Öncelikle biraz kafkadan bahsetmek gerekirse :

Apache Kafka temelde log kaydına benzer bir yapıda kayıtları tutan ve bu kayıtları diğer sistemlere mesajlaşma kuyruğu(messaging queue – MQ ) şeklinde sunan bir yapıdır. Yüksek performansı ve düşük gecikme zamanı(latency) ile gerçek zamanlı veri akışı sağlar.

**Mesajlaşma Sistemi**

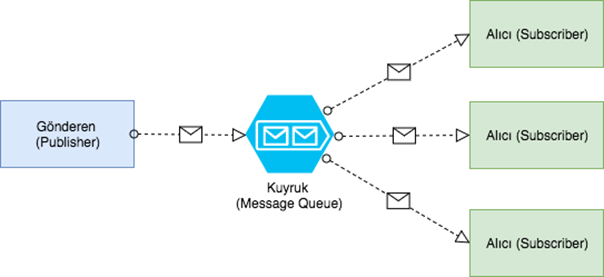
Mesajlaşma sistemi, verinin bir uygulamadan diğerine aktarılmasını sağlar. Distributed Messaging denilen kavram aslında güvenilir bir kuyruk yapısı (Message Queue) olarak da adlandırılır. Mesajlaşma sistemleri 2 yapıda olabilir:

**1. Point-to-Point**

Bu yapıda birden fazla alıcı olabilmesine rağmen, bir mesaj sadece bir alıcıya iletilir. Mesaj bir alıcıya iletildikten sonra siliniyor şeklinde bir benzetme yapabiliriz.

**2. Publisher-Subscriber**

Bu yapıda ise mesajlar bir kategori altında toplanırlar. Alıcılar birden fazla kategoriden veri çekebilir. Bu sistemde mesaj gönderen uygulamalara **Publisher**, alıcı uygulamalara da **Subscriber** adı verilir.



* Kafka mesajları “**topic**”lerde tutar.
* “**Topic**”lere mesaj gönderenler “**producer**”lardır.
* “**Topic**”leri okuyanlar ise “**consumer**”lardır.
* Kafka’nın çalıştığı her bir sunucu ise “**broker**”dır.
* “**Broker**”ların hepsi birlikte bir “**cluster**” oluşturur.

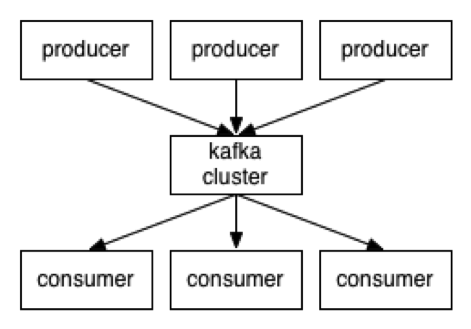
• Kafka keeps messages "topic".

• The "producer" is the sender of messages to "Topic".

• "Topic" readers are "consumers".

• Each server that Kafka runs is a "broker".

• "Brokers" all together form a "cluster".



Topic Nedir? What Is Topic?

The place where messages coming to Kafka are held is a topic and always has a name. Basically each topic consists of one or more log files that are partitioned. Each partition has a portion of the existing topic. Each chapter contains a sequence in itself. This order is called "offset"

Every new message that comes to the topic is added to the end of the chapter and offset numbers are increasing one by one. All messages are kept in the offset information for the time specified according to the config (config).

Kafka’ya gelen mesajların tutulduğu yer bir topic ve her zaman bir ismi var. Temelde her topic bölümlendirilmiş (partitioned) bir ya da birden fazla log dosyasından oluşuyor. Her bir bölüm(partition) mevcut topicin bir kısmına sahip oluyor. Her bölüm kendi içinde bir sıra ihtiva ediyor. Bu sıraya “**offset**” deniyor

Topice gelen her yeni mesaj bölüm sonuna ekleniyor ve offset numarası birer birer artıyor. Bütün mesajlar yapılan ayara (config) göre belirtilen süre boyunca offset bilgi değişimeden tutuluyor.

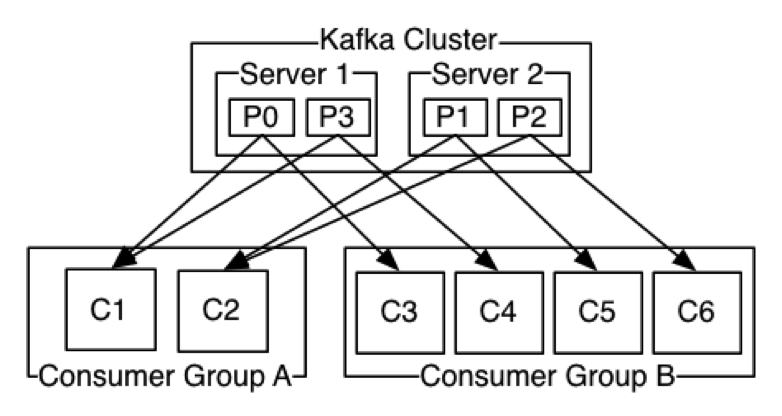
Producer KullanımıProducer Usage

Producer can decide which message to go to which topic to split. It does this either in round-robin fashion or with semantic partitioning. If you are sending a top message and you have a key (message key) and the corresponding topic has a logical partitioning function, the message is written to a gore partition.

Producer hangi mesajın ilgili topicte hangi bölüme gideceğine karar verebiliyor. Bunu ya round-robin şeklinde ya da mantıksal bölümlendirme (semantic partitioning) ile yapıyor. Bir topice mesaj gönderirken eğer bir anahtar(message key) verirseniz ve ilgili topicin mantıksal bölümlendirme fonksiyonu varsa, mesaj ona gore bir partitiona yazılır.

**Partition**

The Topic are divided into Partitions called Partitions.



**Broker**

A single kafka server is called a Broker. One of the nodes in the cluster is defined as the "leader" and It is responsible for all read and write operations for the Partition concerned. Other Nodes are called "followers"

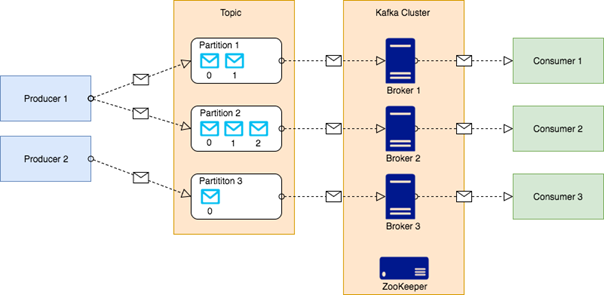
and works similarly to Consumer. If the leader node does not work, one of the follower nodes serves as the leader node.

Tek bir kafka sunucusuna Broker adı verilir. Bir diğer deyişle de Kafka Cluster’daki her bir Node’a Broker adı verilir. Cluster içerisinde yer alan Node’lardan biri “lider” olarak tanımlanır ve ilgili Partition için bütün okuma ve yazma işlerinden sorumludur. Diğer Node’lar ise “takipçi” olarak tanımlanır ve Consumer ile benzer şekilde çalışırlar. Eğer lider Node çalışmazsa takipçi Node’lardan birisi lider Node olarak görev alır.

**Consumer**

Consumer also corresponds to Subscriber from Publisher-Subscriber. Read data from the broker. It can read data from more than one topic at a time. Kafka has a concept called "consumer group. Each consumer belongs to a "consumer group".If this consumer grub is connected to a single consumer group then it will display queuing behavior. The incoming messages for that consumer group are read one by one from all sections. If more than one consumer is connected to a consumer grub then a topic called publish / subscribe will be a topic. That is, multiple processes in multiple consumer groups read incoming messages in more than one partition.

Consumer da Publisher-Subscriber kelimelerinden Subscriber’a karşılık gelir Broker’dan veri okur. Bir yada birden fazla Topic üzerinden veri okuyabilir. Kafka’da “**consumer group**” diye bir kavram vardır. Aslında her consumer bir “**consumer group**”a aittir. Bu consumer gruba eğer tek consumer group bağlı ise o zaman mesajları okuma şekli kuyruk (queue) davranışı sergiler. O consumer group için gelen mesajlar bütün bölümlerden tek tek okunur. Birden fazla consumerın bir consumer gruba bağlı olması durumda ise o zaman publish/subscribe denilen yapıda bir topic olur (Kafka’daki topic ile karıştırmayın J) Yani gelen mesajları birden fazla consumer group içindeki birden fazla proses birden fazla bölümden okur.



Alttaki komutla **topic** oluşturulurken bir mesajın kaç partition üzerinden gönderileceğini belirtebiliriz , bununla beraber aynı mesajın birden fazla kopyasının gönderilmesini sağlayan **replication-factor**parametresini de vermiş oluruz.

We can specify how many partitions a message will be sent to when the topic is created with the command below. Along with the replication-factor parameter, which allows multiple copies of the same message to be sent.

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1

--partitions 1 --topic Hello-Kafka

### Start Producer to Send Messages

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic Hello-Kafka

**Output**

$ bin/kafka-console-producer.sh --broker-list localhost:9092

--topic Hello-Kafka[2016-01-16 13:50:45,931]

WARN property topic is not valid (kafka.utils.Verifia-bleProperties)

Hello

My first message

My second message

### Start Consumer to Receive Messages

bin/kafka-console-consumer.sh --zookeeper localhost:2181 —topic topic-name

--from-beginning

**Output**

Hello

My first message

My second message