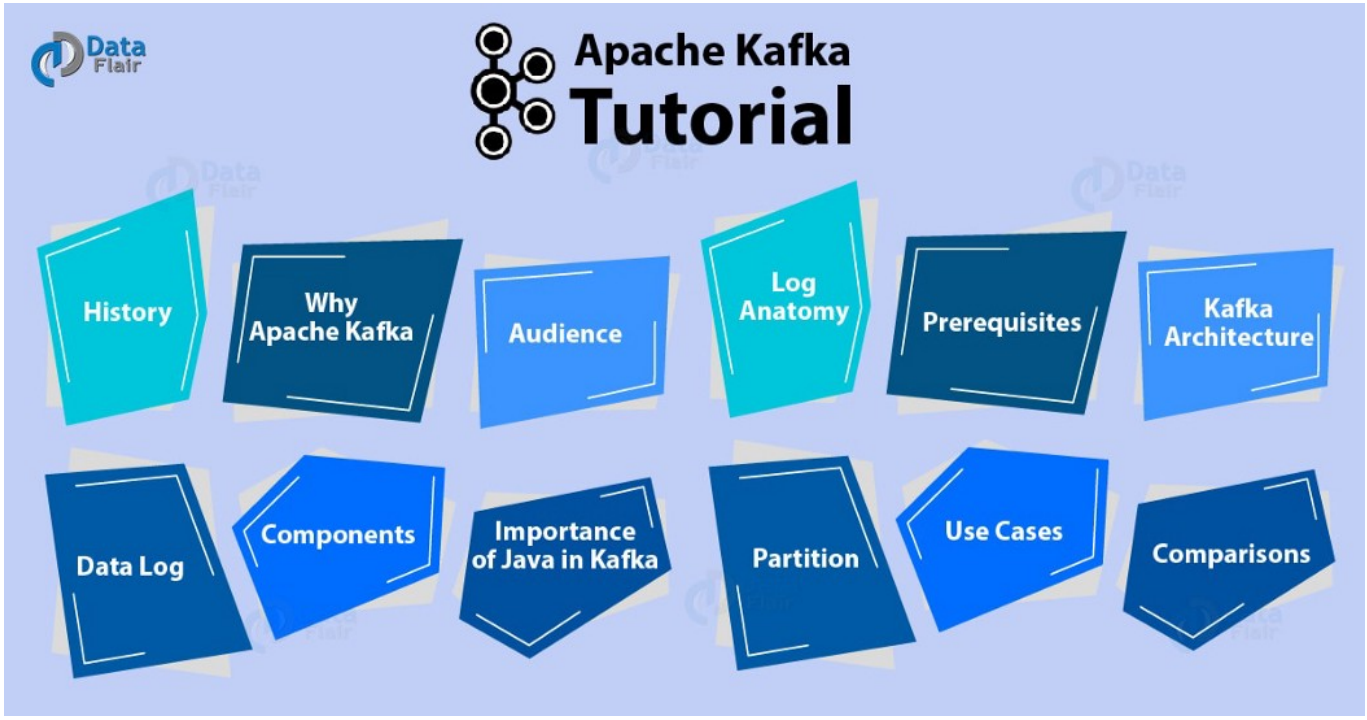
Rinu Gour

Follow 2.3K Followers About

Kafka para principiantes

 Rinu Gour 12 de septiembre de 2018 · 9 min de lectura



¿Qué es Kafka?

Usamos Apache Kafka cuando se trata de permitir la comunicación entre productores y consumidores utilizando temas basados en mensajes. Apache Kafka es unsistema de mensajería de publicación-suscripción rápido, escalable, tolerante a fallos. Básicamente, diseña una plataforma para aplicaciones distribuidas de nueva generación de alta gama.

fallas de los nodos y admite la recuperación automática. Esta característica hace que Apache Kafka sea ideal para la comunicación y la integración entre componentes de sistemas de datos a gran escala en sistemas de datos del mundo real.

Por otra parte, t su tecnología sustituye a los intermediarios de mensajes convencionales, con la capacidad de dar un mayor rendimiento, la fiabilidad, y la replicación como JMS, AMQP y muchos más. Además, Kafka ofrece una abstracción central a un **corredor de Kafka** , un **productor de Kafka** y un **consumidor de Kafka** . El broker de Kafka es un nodo del clúster de Kafka, su uso es para persistir y replicar los datos. Un productor de Kafka inserta el mensaje en el contenedor de mensajes llamado Tema de Kafka. Mientras que un consumidor de Kafka extrae el mensaje del tema de Kafka.

Antes de avanzar en el tutorial de Kafka, comprendamos el significado real del término Sistema de mensajería en Kafka.

a. Sistema de mensajería en Kafka

Cuando transferimos datos de una aplicación a otra, usamos el sistema de mensajería. El resultado es que, sin preocuparse por cómo compartir los datos, las aplicaciones pueden enfocarse solo en los datos. En el concepto de cola de mensajes confiable, se basa la mensajería distribuida. Aunque, los mensajes se ponen en cola de forma asíncrona entre las aplicaciones cliente y el sistema de mensajería. Hay dos tipos de patrones de mensajería disponibles, es decir, sistema de mensajería punto a punto y publicación-suscripción (pub-sub). Sin embargo, la mayoría de los patrones de mensajería siguen a pub-sub.

Apache Kafka - Sistema de mensajería Kafka

- **Sistema de mensajería punto a punto**

Here, messages are persisted in a queue. Although, a particular message can be consumed by a maximum of one consumer only, even if one or more consumers can consume the messages in the queue. Also, it makes sure that as soon as a consumer reads a message in the queue, it disappears from that queue.

- **Publish-Subscribe Messaging System**

Here, messages are persisted in a topic. In this system, Kafka Consumers can subscribe to one or more topic and consume all the messages in that topic. Moreover, message producers refer publishers and message consumers are subscribers here.

History of Apache Kafka

Previously, LinkedIn was facing the issue of low latency ingestion of huge amount of data from the website into a lambda architecture which could be able to process real-time events. As a solution, Apache Kafka was developed in the year 2010, since none of the solutions was available to deal with this drawback, before.

However, there were technologies available for batch processing, but the deployment details of those technologies were shared with the downstream users. Hence, while it comes to Real-time Processing, those technologies were not enough suitable. Then, in the year 2011 Kafka was made public.

Why Should we use Apache Kafka Cluster?

As we all know, there is an enormous volume of data in **Big Data**. And, when it comes to big data, there are two main challenges. One is to collect the large volume of data, while another one is to analyze the collected data. Hence, in order to overcome those

- Tracking web activities by storing/sending the events for real-time processes.
- Alerting and reporting the operational metrics.
- Transforming data into the standard format.
- Continuous processing of streaming data to the topics.

Therefore, this technology is giving a tough competition to some of the most popular applications like ActiveMQ, RabbitMQ, AWS etc. because of its wide use.

Kafka Tutorial — Audience

Professionals who are aspiring to make a career in **Big Data** Analytics using Apache Kafka messaging system should refer this Kafka Tutorial article. It will give you complete understanding about Apache Kafka.

Kafka Tutorial — Prerequisites

You must have a good understanding of **Java**, **Scala**, Distributed messaging system, and **Linux** environment, before proceeding with this Apache Kafka Tutorial.

Kafka Architecture

Below we are discussing four core APIs in this Apache Kafka tutorial:

Apache Kafka — Kafka Architecture

a. Kafka Producer API

This Kafka Producer API permits an application to publish a stream of records to one or more Kafka topics.

b. Kafka Consumer API

To subscribe to one or more topics and process the stream of records produced to them in an application, we use this Kafka Consumer API.

c. Kafka Streams API

In order to act as a stream processor consuming an input stream from one or more topics and producing an output stream to one or more output topics and also effectively transforming the input streams to output streams, this Kafka Streams API gives permission to an application.

d. Kafka Connector API

This Kafka Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

Kafka Components

Using the following components, Kafka achieves messaging:

collection of messages are Topics. In addition, we can replicate and partition Topics. Here, replicate refers to copies and partition refers to the division. Also, visualize them as logs wherein, Kafka stores messages. However, this ability to replicate and partitioning topics is one of the factors that enable Kafka's fault tolerance and scalability.

Apache Kafka — Kafka Topic

b. Kafka Producer

It publishes messages to a Kafka topic.

c. Kafka Consumer

This component subscribes to a topic(s), reads and processes messages from the topic(s).

d. Kafka Broker

Kafka Broker manages the storage of messages in the topic(s). If Kafka has more than one broker, that is what we call a Kafka cluster.

e. Kafka Zookeeper

To offer the brokers with metadata about the processes running in the system and to facilitate health checking and broker leadership election, Kafka uses Kafka **zookeeper**.

messages to the log. One of the advantages is, at any time one or more consumers read from the log they select. Here, the below diagram shows a log is being written by the data source and the log is being read by consumers at different offsets.

Apache Kafka Tutorial — Log Anatomy

Kafka Tutorial — Data Log

By Kafka, messages are retained for a considerable amount of time. Also, consumers can read as per their convenience. However, if Kafka is configured to keep messages for 24 hours and a consumer is down for time greater than 24 hours, the consumer will lose messages. And, messages can be read from last known offset, if the downtime on part of the consumer is just 60 minutes. Kafka doesn't keep state on what consumers are reading from a topic.

Kafka Tutorial — Partition in Kafka

There are few partitions in every Kafka broker. Moreover, each partition can be either a leader or a replica of a topic. In addition, along with updating of replicas with new data,

Apache Kafka Tutorial — Partition In Kafka

Importance of Java in Apache Kafka

Apache Kafka is written in pure **Java** and also Kafka's native API is java. However, many other languages like C++, **Python,** .Net, Go, etc. also support Kafka. Still, a platform where there is no need of using a third-party library is Java. Also, we can say, writing code in languages apart from Java will be a little overhead.

In addition, we can use Java language if we need the high processing rates that come standard on Kafka. Also, Java provides a good community support for Kafka consumer clients. Hence, it is a right choice to implement Kafka in Java.

Kafka Use Cases

There are several **use Cases of Kafka** that show why we actually use Apache Kafka.

- **Messaging**

For a more traditional message broker, Kafka works well as a replacement. We can say Kafka has better throughput, built-in partitioning, replication, and fault-tolerance which makes it a good solution for large-scale message processing applications.

- **Metrics**

operational data.

- **Event Sourcing**

Since it supports very large stored log data, that means Kafka is an excellent backend for applications of event sourcing.

Kafka Tutorial — Comparisons in Kafka

Many applications offer the same functionality as Kafka like ActiveMQ, RabbitMQ, **Apache Flume**, Storm, and **Spark**. Then why should you go for Apache Kafka instead of others?

Let's see the comparisons below:

a. Apache Kafka vs Apache Flume

Kafka Tutorial — Apache Kafka vs Flume

i. Types of tool

Apache Kafka– For multiple producers and consumers, it is a general-purpose tool.

ii. Replication feature

Apache Kafka– Using ingest pipelines, it replicates the events.

Apache Flume- It does not replicate the events.

b. RabbitMQ vs Apache Kafka

One among the foremost Apache Kafka alternatives is RabbitMQ. So, let's see how they differ from one another:

Kafka Tutorial — Kafka vs RabbitMQ

i. Features

Apache Kafka– Basically, Kafka is distributed. Also, with guaranteed durability and availability, the data is shared and replicated.

RabbitMQ– It offers relatively less support for these features.

ii. Performance rate

Apache Kafka — Its performance rate is high to the tune of 100,000 messages/second.

iii. Processing

Apache Kafka — It allows reliable log distributed processing. Also, stream processing semantics built into the Kafka Streams.

RabbitMQ — Here, the consumer is just FIFO based, reading from the HEAD and processing 1 by 1.

c. Traditional queuing systems vs Apache Kafka

Kafka Tutorial — Traditional queuing systems vs Apache Kafka

i. Messages Retaining

Traditional queuing systems — Most queueing systems remove the messages after it has been processed typically from the end of the queue.

Apache Kafka — Here, messages persist even after being processed. They don't get removed as consumers receive them.

ii. Logic-based processing

Apache Kafka — It allows to process logic based on similar messages or events.

So, this was all about Apache Kafka Tutorials. Hope you like our explanation.

Conclusion: Kafka Tutorial

Por lo tanto, en este Tutorial de Kafka, hemos visto el concepto completo de Apache Kafka y hemos visto qué es Kafka. Además, discutimos los componentes de Kafka, los casos de uso y la arquitectura de Kafka. Por último, discutimos la comparación de Kafka con otras herramientas de mensajería. Además, si tiene alguna consulta sobre el tutorial de Kafka, no dude en preguntar en la sección de comentarios. Además, siga visitando Data Flair para obtener más artículos sobre Apache Kafka.

Big Data Kafka Apache Kafka Ciencia de los datos Análisis de los datos