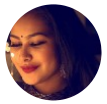
# Apache Kafka Tutorial — Kafka For Beginners



Harshali Patel Follow
Aug 21, 2018 · 9 min read

## 1. Kafka Tutorial

Today, we are starting our new journey, with this Apache Kafka Tutorial. In this Kafka tutorial, we will see what is Kafka, Apache Kafka history, why Kafka. Moreover, we will learn **Kafka Architecture**, components of Kafka and Kafka Partition. Also, we will discuss various comparisons in Kafka and Kafka use cases. Along with this, we will see various terms in this Kafka tutorial such as **Kafka Broker**, Kafka Cluster, Kafka Consumer, Kafka Topics etc.

So, let's begin the Apache Kafka Tutorial.


Apache Kafka Tutorial — Kafka For Beginners

## 2. What is Kafka?

We use Apache Kafka when it comes to enabling communication between producers and consumers using message-based topics. Apache Kafka is a fast, scalable, fault-tolerant, publish-subscribe messaging system. Basically, it designs a platform for high-end new generation distributed applications. Also, it allows a large number of permanent or ad-hoc consumers. One of the best features of Kafka is, it is highly available and resilient to node failures and supports automatic recovery. This feature makes Apache Kafka ideal for communication and integration between components of large-scale data systems in real-world data systems.

Moreover, this technology replaces the conventional message brokers, with the ability to give higher throughput, reliability, and replication like JMS, AMQP and many more. In addition, core abstraction Kafka offers a **Kafka broker**, a **Kafka Producer**, and a **Kafka Consumer**. Kafka broker is a node on the Kafka cluster, its use is to persist and replicate the data. A Kafka Producer pushes the message into the message container called the Kafka Topic. Whereas a Kafka Consumer pulls the message from the Kafka Topic.

Before moving forward in Kafka Tutorial, let's understand the actual meaning of term Messaging System in Kafka.

**a. Messaging System in Kafka**

When we transfer data from one application to another, we use the Messaging System. It results as, without worrying about how to share data, applications can focus on data only. On the concept of reliable message queuing, distributed messaging is based. Although, messages are asynchronously queued between client applications and messaging system. There are two types of messaging patterns available, i.e. point to point and publish-subscribe (pub-sub) messaging system. However, most of the messaging patterns follow pub-sub.

**Do you know about Kafka Cluster**


Apache Kafka — Kafka Messaging System

- **Point to Point Messaging System**

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.

## 3. 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.

## 4. 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 challenges, we need a messaging system. Then Apache Kafka has proved its utility. There are numerous **benefits of Apache Kafka** such as:

- 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.

## 5. Kafka Tutorial — Audience

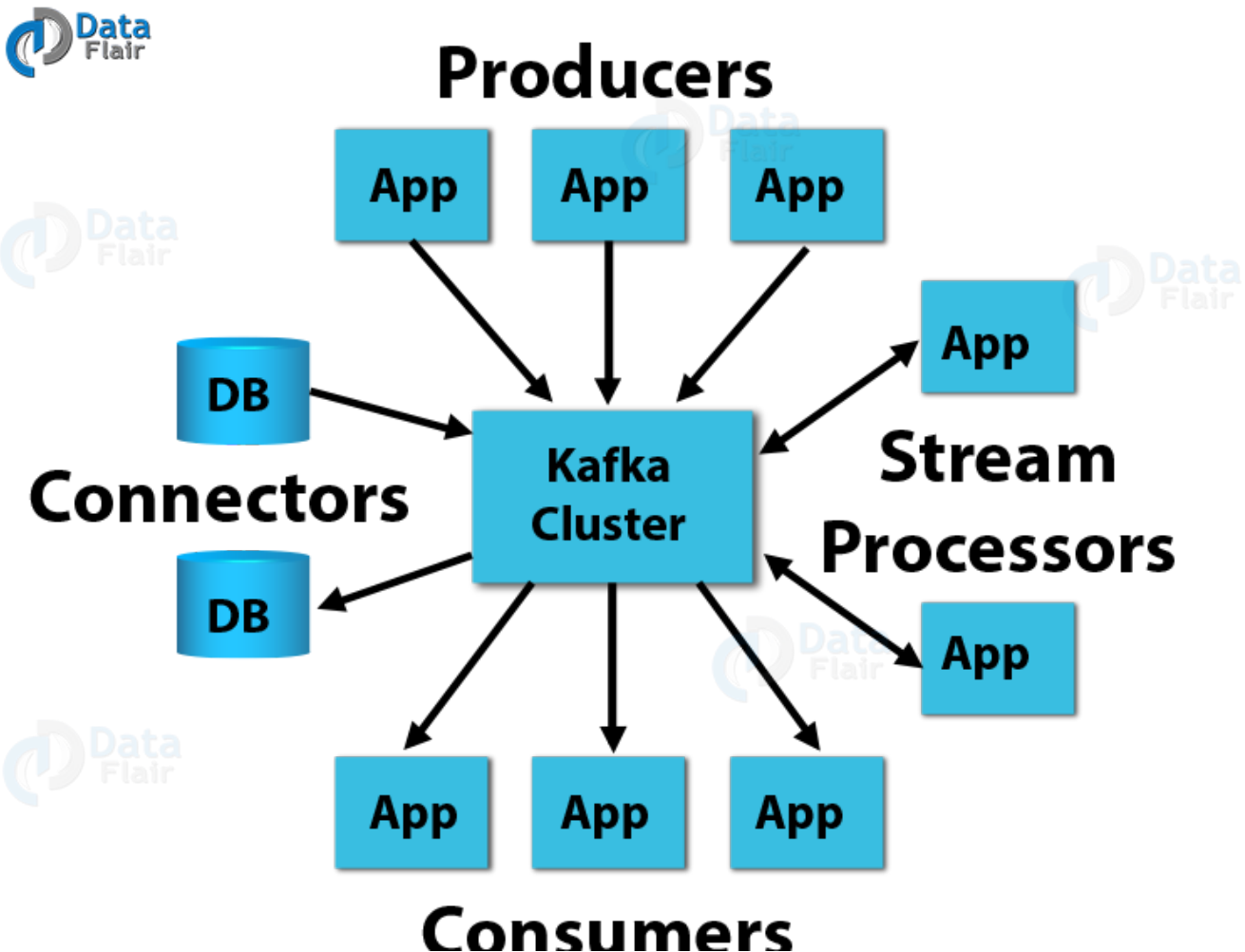
Professionals who are aspiring to make a career in **Big Data** Analytics using Apache Kafka messaging system should refer this Kafka Tutorial article. It will give you complete understanding about Apache Kafka.

## 6. 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.

## 7. 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.

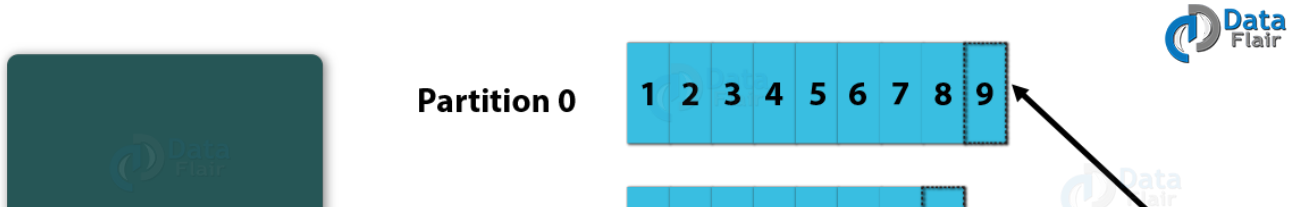
**Apache Kafka Security | Need and Components of Kafka**

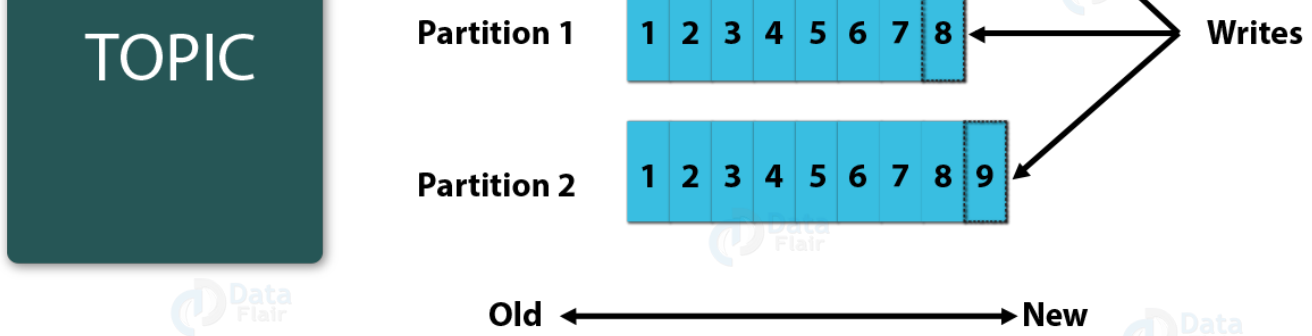
## 8. Kafka Components

Using the following components, Kafka achieves messaging:

## a. Kafka Topic

Basically, how Kafka stores and organizes messages across its system and essentially a 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

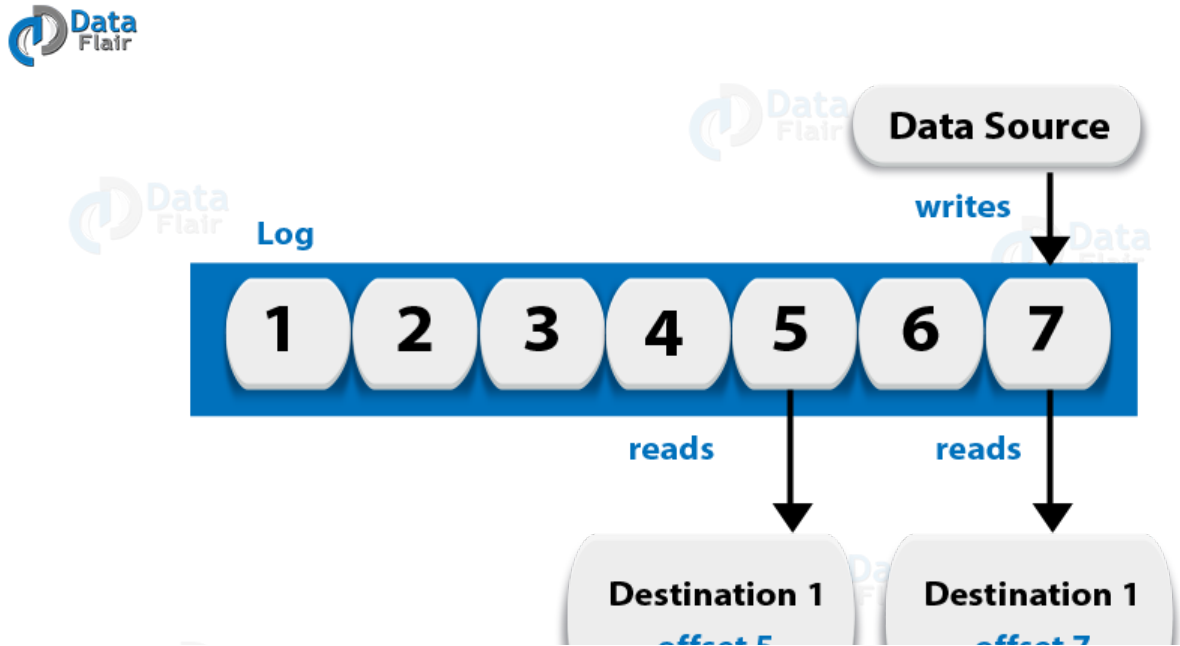
Kafka Broker manages the storage of messages in the topic(s). If Kafka has more than one broker, that is what we call a Kafka cluster.

## 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**.

## 9. Kafka Tutorial — Log Anatomy

We view log as the partitions in this Kafka tutorial. Basically, a data source writes messages to the log. One of the advantages is, at any time one or more consumers read from the log they select. Here, below diagram shows a log is being written by the data source and the log is being read by consumers at different offsets.



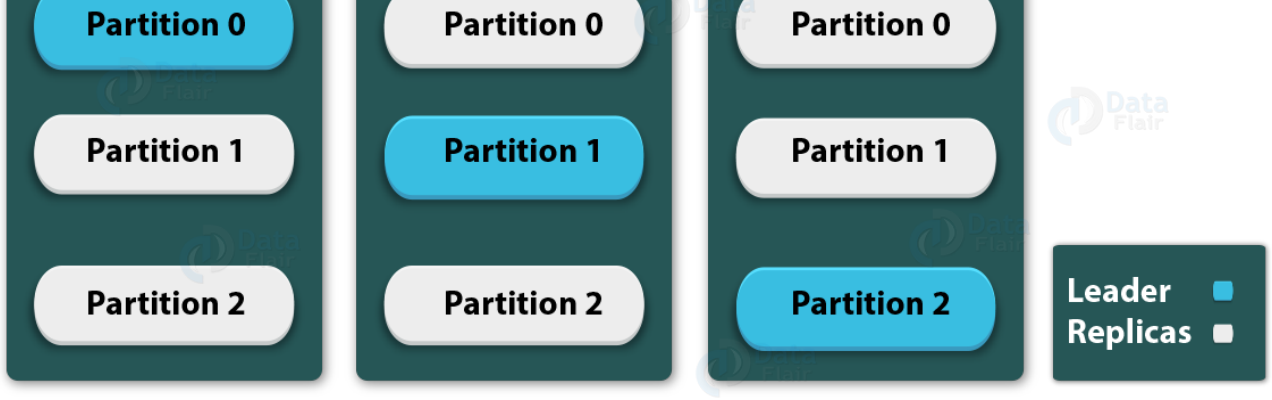

Apache Kafka Tutorial — Log Anatomy

## 10. 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.

## 11. 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, Leader is responsible for all writes and reads to a topic. The replica takes over as the new leader if somehow the leader fails.




Apache Kafka Tutorial — Partition In Kafka

## 12. 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.

## 13. 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**

For operational monitoring data, Kafka finds the good application. It includes aggregating statistics from distributed applications to produce centralized feeds of operational data.

- **Event Sourcing**

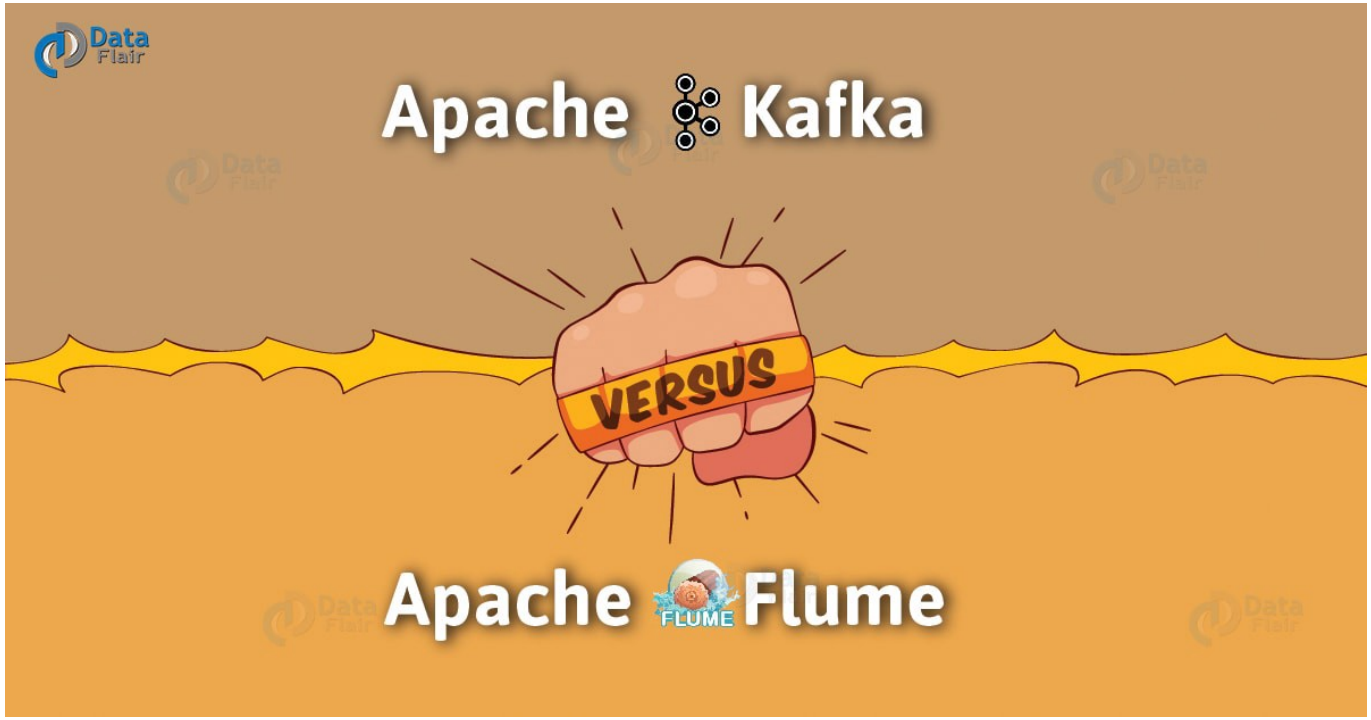
Since it supports very large stored log data, that means Kafka is an excellent backend for applications of event sourcing.

## 14. 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.

**Apache Flume**– Whereas, it is a special-purpose tool for specific applications.

***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.

**RabbitMQ** — Whereas, the performance rate of RabbitMQ is around 20,000 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.

**Let's learn Kafka vs RabbitMQ**

## 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***

**Traditional queuing systems** — It does not allow to process logic based on similar messages or events.

**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.

## 15. Conclusion: Kafka Tutorial

Hence, in this Kafka Tutorial, we have seen the whole concept of Apache Kafka and seen what is Kafka. Moreover, we discussed Kafka components, use cases, and Kafka architecture. At last, we discussed the comparison of Kafka vs other messaging tools. Furthermore, if you have any query regarding Kafka Tutorial, feel free to ask in the comment section. Also, keep visiting Data Flair for more knowledgeable articles on Apache Kafka.

Big Data Kafka Apache Kafka