



Let us know how you like to work in this quick survey!

**Take the Survey** ▶

DZone > Big Data Zone > What Is Kafka? Everything You Need to Know

# What Is Kafka? Everything You Need to Know

by Jean-Paul Azar · Aug. 09, 17 · Big Data Zone · Presentation

Get \$10 for a ten-minute survey. DZone is working with [GLG](#) to gather insights for their clients from IT Professionals. [S](#)

Presented by GLG



*Apache Kafka is a distributed streamina nlatform.*

Kafka's growth is exploding. More than one-third of all Fortune 500 companies use Kafka. These companies include the top ten travel companies, seven of the top ten banks, eight of the top ten insurance companies, nine of the top ten telecom companies, and much more. LinkedIn, Microsoft, and Netflix process four-comma messages a day with Kafka (1,000,000,000,000).

**Kafka is used for real-time streams of data, to collect big data, or to do real time analysis (or both).** Kafka is used with in-memory microservices to provide durability and it can be used to feed events to CEP (complex event streaming systems) and IoT/IFTTT-style automation systems.

---

**You may also like: Apache Kafka [DZone Refcard]**

---

## Why Kafka?

Kafka is often used in real-time streaming data architectures to provide real-time analytics. Since Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system, Kafka is used in use cases where JMS, RabbitMQ, and AMQP may not even be considered due to volume and responsiveness.

**Kafka has higher throughput, reliability, and replication characteristics,** which makes it applicable for things like tracking service calls (tracks every call) or tracking IoT sensor data where a traditional MOM might not be considered.

Kafka can work with Flume/Flafka, Spark Streaming, Storm, HBase, Flink, and Spark for real-time ingesting, analysis and processing of streaming data. Kafka is a data stream used to feed Hadoop BigData lakes. Kafka brokers support massive message streams for low-latency follow-up analysis in Hadoop or Spark. Also, Kafka Streams (a subproject) can be used for real-time analytics.

## Kafka Use Cases

In short, Kafka is used for stream processing, website activity tracking, metrics collection and monitoring, log aggregation, real-time analytics, CEP, ingesting data into Spark, ingesting data into Hadoop, CQRS, replay messages, error recovery, and guaranteed distributed commit log for in-memory computing (microservices).

## Who Uses Kafka?

A lot of large companies who handle a lot of data use Kafka. LinkedIn, where it originated, uses it to track activity data and operational metrics. Twitter uses it as part of Storm to provide a stream processing infrastructure. Square uses Kafka as a bus to move all system events to various Square data centers (logs, custom events, metrics, and so on), outputs to Splunk, for Graphite (dashboards), and to implement Esper-like/CEP alerting systems.

It's also used by other companies like Spotify, Uber, Tumbler, Goldman Sachs, PayPal, Box, Cisco, CloudFlare, and Netflix.

## Why Is Kafka So Popular?

**Kafka has operational simplicity.** Kafka is easy to set up and use, and it is easy to figure out how Kafka works. However, the main reason Kafka is very popular is its excellent performance. It is stable, provides reliable durability, has a flexible publish-subscribe/queue that scales well with N-number of consumer groups, has robust replication, provides producers with tunable consistency guarantees, and it provides preserved ordering at the shard level (i.e. Kafka topic partition).

In addition, Kafka works well with systems that have data streams to process and enables those systems to aggregate, transform, and load into other stores. But none of those characteristics would matter if Kafka was slow.

---

**The most important reason Kafka is popular is Kafka's exceptional performance.**

---

## Why Is Kafka so Fast?

Kafka relies heavily on the OS kernel to move data around quickly. It relies on the principals of zero copy. Kafka enables you to batch data records into chunks. These batches of data can be seen end-to-end from producer to file system (Kafka topic log) to the consumer.

Batching allows for more efficient data compression and reduces I/O latency. Kafka writes to the immutable commit log to the disk sequential, thus avoiding random disk access and slow disk seeking. Kafka provides horizontal scale through sharding. It shards a topic log into hundreds (potentially thousands) of partitions to thousands of servers. This sharding allows Kafka to handle massive load.

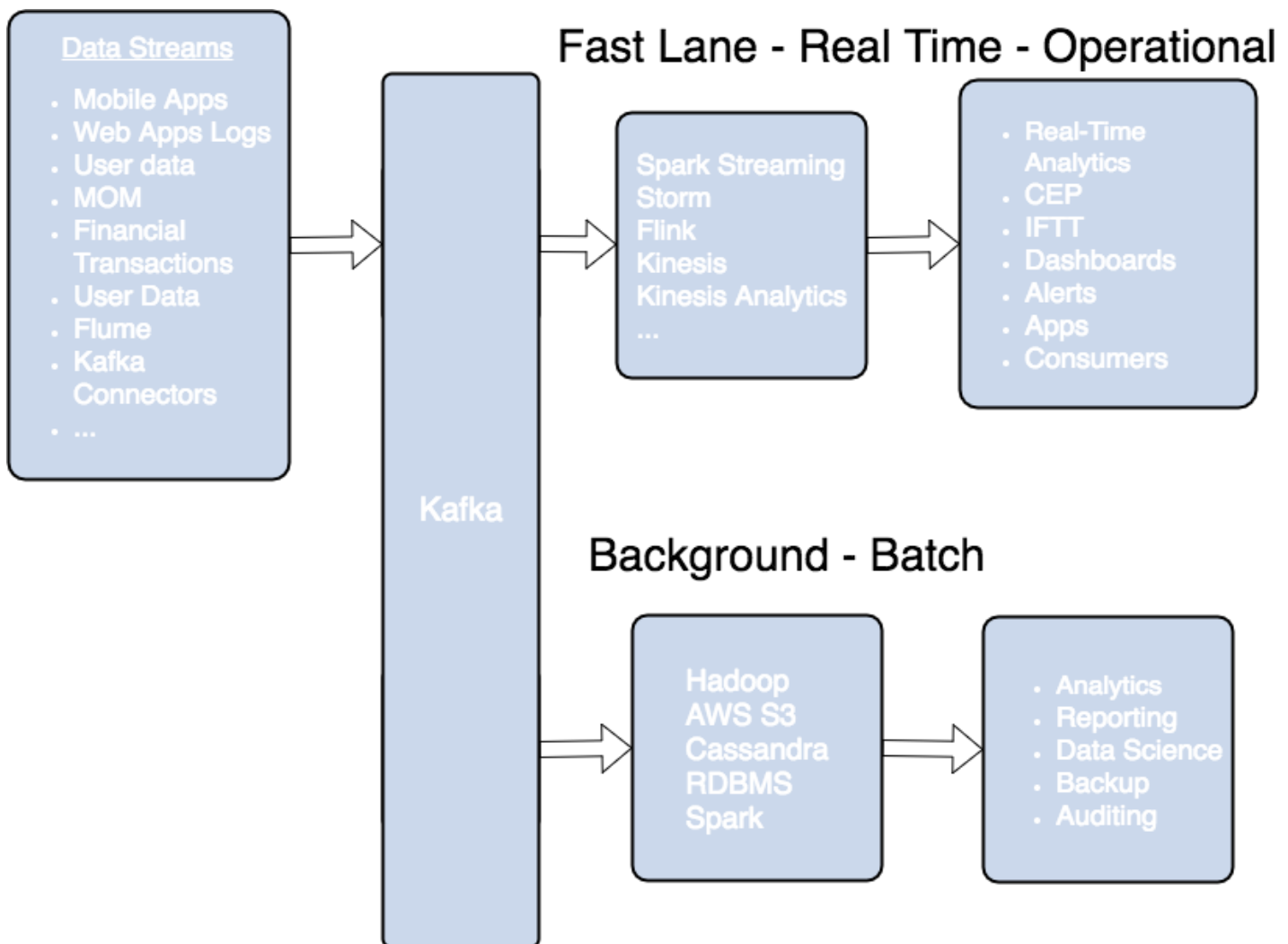
## Kafka: Streaming Architecture

**Kafka is used most often for streaming data in real-time into other systems.** Kafka is a

middle layer to decouple your real-time data pipelines.

Kafka core is not good for direct computations such as data aggregations or CEP. Kafka streaming, which is part of the Kafka ecosystem, provides the ability to do real-time analytics.

Kafka can be used to feed fast lane systems (real-time and operational data systems) like Storm, Flink, Spark streaming, and your services and CEP systems. Kafka is also used to stream data for batch data analysis. Kafka feeds Hadoop. It streams data into your big data platform or into RDBMS, Cassandra, Spark, or even S3 for some future data analysis. These data stores often support data analysis, reporting, data science crunching, compliance auditing, and backups.



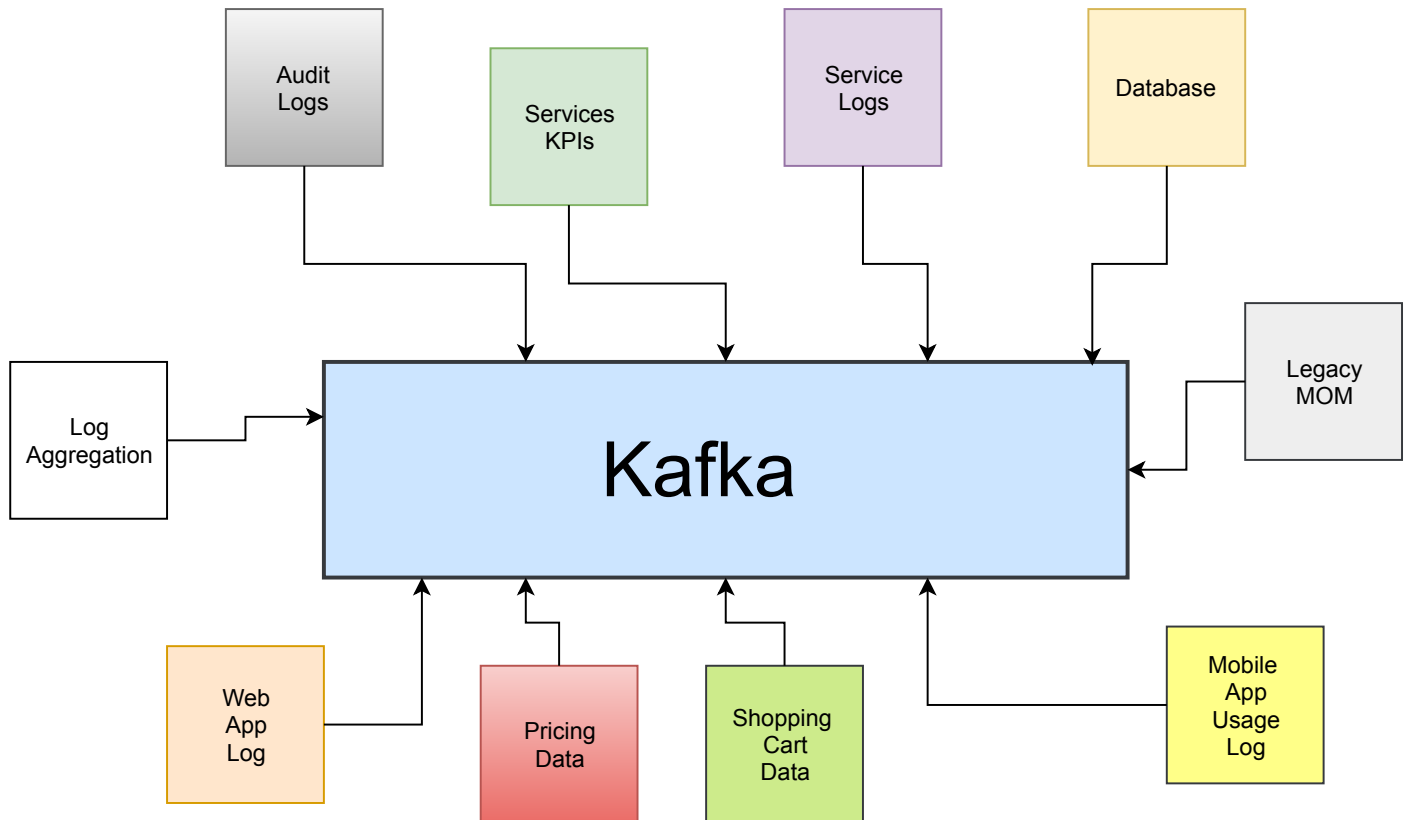
*Kafka streaming architecture diagram*

Now let's truly answer the big question:

## What Is Kafka?

**Kafka is a distributed streaming platform that is used publish and subscribe to**

**streams of records.** Kafka is used for fault tolerant storage. Kafka replicates topic log partitions to multiple servers. Kafka is designed to allow your apps to process records as they occur. Kafka is fast and uses IO efficiently by batching and compressing records. Kafka is used for decoupling data streams. Kafka is used to stream data into data lakes, applications, and real-time stream analytics systems.



*Kafka decoupling data streams*

## Kafka Is Polyglot

Kafka communication from clients and servers uses a wire protocol over TCP that is versioned and documented. Kafka promises to maintain backward compatibility with older clients, and many languages are supported. There are clients in C#, Java, C, Python, Ruby, and many more languages. The Kafka ecosystem also provides REST proxy allows easy integration via HTTP and JSON, which makes integration even easier.

Kafka also supports Avro schemas via the Confluent Schema Registry for Kafka. Avro and the Schema Registry allow complex records to be produced and read by clients in many programming languages and allow for the evolution of the records. Kafka is truly polyglot.

## Kafka Is Useful

Kafka allows you to build real-time streaming data pipelines. Kafka enables in-memory microservices (i.e. actors, Akka, Reactive.io, ORBit, reactors, reactive, Vert.x, PyLava, Spring

microservices (i.e. actors, akka, Datastream, QDIP, Reactors, Reactive, Vert.x, RxJava, Spring Reactor). Kafka allows you to build real-time streaming applications that react to streams to do real-time data analytics, transform, react, aggregate, join real-time data flows, and perform CEP (complex event processing).

You can use Kafka to aid in gathering metrics/KPIs, aggregating statistics from many sources, and implementing event sourcing. You can use it with microservices (in-memory) and actor systems to implement in-memory services (external commit log for distributed systems).

You can use Kafka to replicate data between nodes, to re-sync for nodes, and to restore state. While Kafka is mostly used for real-time data analytics and stream processing, you can also use it for log aggregation, messaging, click-stream tracking, audit trails, and much more.

In a world where data science and analytics are a big deal, capturing data to feed into your data lakes and real-time analytics systems is *also* a big deal. And since Kafka can hold up to these kinds of strenuous use cases, Kafka is a big deal.

## Kafka Is Scalable Message Storage

Kafka is a good storage system for records/messages. Kafka acts like high-speed file system for commit log storage and replication. These characteristics make Kafka useful for all manners of applications.

Records written to Kafka topics are persisted to disk and replicated to other servers for fault-tolerance. Since modern drives are fast and quite large, this fits well and is very useful. Kafka Producers can wait on acknowledgment, so messages are durable as the producer write not complete until the message replicates. The Kafka disk structure scales well. Modern disk drives have very high throughput when writing in large streaming batches.

Also, Kafka clients and consumers can control read position (offset), which allows for use cases like replaying the log if there was a critical bug (i.e. fix the bug and the replay). And since offsets are tracked per consumer group, which we talk about in this Kafka architecture article, consumers can be quite flexible (i.e. replay the log).

## Kafka Has Record Retention

Kafka clusters retain all published record. If you don't set a limit, it will keep records until it runs out of disk space. You can set time-based limits (configurable retention period), size-based limits (configurable based on size), or compaction (keeps the latest version of record using key). You can, for example, set a retention policy of three days or two weeks or a month. The records in the topic log are available for consumption until discarded by time, size or compaction. The consumption speed not impacted by size as Kafka always writes to

size, or compression. The consumption speed not impacted by size as kafka always writes to the end of the topic log.

## Further Reading

Introduction to Apache Kafka [Tutorial]

How to Use the Kafka Streams API

Understanding When to Use RabbitMQ or Apache Kafka

---


Get \$10 for a ten-minute survey. DZone is working with GLG to help their clients learn how IT orgs deploy innovative tech so


[\\$10 And Start Survey](#)

Presented by GLG


---

## Like This Article? Read More From DZone


 **An Overview of the Kafka Distributed Message System (Part 2)**

 **An Overview of the Kafka Distributed Message System (Part 1)**

 **Apache Storm: Architecture**

 **Free DZone Refcard Understanding Apache Spark Failures and Bottlenecks**

Topics: STREAMING , APACHE KAFKA , BIG DATA , TUTORIAL , WHAT IS KAFKA , REAL-TIME STREAMING , DATA ARCHITECTURE , REAL-TIME ANALYTICS , MESSAGING SYSTEM

Published at DZone with permission of Jean-Paul Azar . [See the original article here.](#)   
Opinions expressed by DZone contributors are their own.