Event Streaming vs. Related Trends

There are several ways to view Event Streaming. This page relates Kafka to other technology trends you may be familiar with.

Feedback

There is an old parable about an elephant and a group of blind men. None of the men had come across an elephant before. One blind man approaches the leg and declares, “It’s like a tree.” Another man approaches the tail and declares, “It’s like a rope.” A third approaches the trunk and declares, “It’s like a snake.” So each blind man senses the elephant from his particular point of view, and comes to a subtly different conclusion as to what an elephant is. Of course the elephant is like all these things, but it is really just an elephant!

Likewise, when people learn about Apache Kafka® and Event Streaming they often come to it biased by their own previous experience. The perspectives they draw are still valid but tend to focus on some subsection of the whole event streaming platform. Here we’re going to look at some common points of view.

Compared to an Enterprise Service Bus?

PDFmyURL converts web pages and even full websites to PDF easily and quickly.
PDFmyURL

If we consider Kafka as a messaging system—with its Kafka Connect interface, which continuously imports data from and continuously exports data to a wide range of interfaces and datastores, and streaming APIs that can manipulate data in flight—it does look a little like an Enterprise Service Bus (ESB). The difference is that ESBs focus on the integration of legacy and off-the-shelf systems, using an ephemeral and comparably low-throughput messaging layer, which encourages request-response protocols.

Kafka, however, is a streaming platform, and as such puts emphasis on high-throughput events and stream processing. A Kafka cluster is a distributed system at heart, providing high availability, storage, and high performance with linear scale-out as comparison benchmarks show. This is quite different from traditional messaging systems, which are limited to a single machine, or if they do scale outward, those scalability properties do not stretch from end to end. Tools for Kafka like its Kafka Streams library and the streaming database ksqlDB allow you to write applications that manipulate events as they move and evolve. These make the processing capabilities of a database available in the application layer, via an API, and outside the confines of the shared broker. This difference to ESBs is quite important.

Why is that? ESBs are criticized in some circles. This criticism arises from the way the technology has been built up over the last 15 years, particularly where ESBs are controlled by central teams that dictate schemas, message flows, validation, and even transformation. In practice centralized approaches like this can constrain an organization, making it hard for individual applications and services to evolve at their own pace. ThoughtWorks have called this out, encouraging users to steer clear of recreating the issues seen in ESBs with Kafka. At the same time, they encourage users to investigate event streaming as a source of truth for an organization's data. Both of these represent sensible advice.

So Kafka may look a little like an ESB, but it is very different as you can learn on this Confluent Developer website. Kafka provides a far higher level of throughput, availability, storage, and processing, and there are hundreds of companies routing their mission-critical data through a single Kafka cluster. Beyond that, streaming encourages services to retain control, particularly of their data, rather than providing orchestration from a single, central team or system. So while having one single Kafka cluster at the center of an organization is quite common, the pattern works because it is simple—nothing more than data transfer and storage provided at scale and high availability. This is emphasized by the core mantra of event-driven services: Centralize an immutable stream of facts. Decentralize the freedom to act, adapt, and change.

Compared to RPC/REST/etc.?

PDFmyURL converts web pages and even full websites to PDF easily and quickly.

Kafka provides an asynchronous protocol for connecting applications together. But it is undoubtedly a bit different from, say, TCP, HTTP, or an RPC protocol. The difference is the presence of the Kafka broker. A broker is a separate piece of infrastructure that broadcasts messages to any applications that are interested in them, as well as storing them for as long as is needed. So it's perfect for streaming as well as for fire-and-forget messaging.

Other use cases sit further from its home ground. A good example is request-response. Say you have a service for querying customer information. So you call a `getCustomer()` method, passing a `CustomerId`, and get a document describing a customer in the reply. You can build this type of request-response interaction with Kafka using two topics: one that transports the request and one that transports the response. People build systems like this, but in such cases, the broker doesn't contribute all that much. There is no requirement for broadcast. There is also no requirement for storage. So this leaves the question: would you be better off using a stateless protocol like HTTP?

So Kafka is a mechanism for programs to exchange information, but its home ground is event-based communication, where events are business facts that have value to more than one service and are worth keeping around.

Compared to a Database?

Some people like to compare Kafka to a database. It certainly comes with similar features:

- Kafka provides storage. Production topics with hundreds of terabytes are not uncommon.
- It has a SQL interface with ksqlDB that lets users define queries and execute them over the data held in the log. These can be piped into views that users can query directly.
- It also supports transactions.

These are all things that sound quite "database-y" in nature! So many of the elements of a traditional database are there, but if anything, Kafka is a database turned inside out, a tool for storing data, processing it in real-time, and creating views. And while you are perfectly entitled to put a dataset in Kafka, run a ksqlDB query over it, and get an answer—much like you might in a traditional database—the streaming database ksqlDB and the Kafka Streams application library are optimized for continuous, streaming computation rather than batch processing.

PDFmyURL converts web pages and even full websites to PDF easily and quickly.

So while the analogy is not wholly inaccurate, it is a little off the mark. Kafka is designed from the ground up for data in motion, operating on that data as it is flowing through streams, tables, applications, systems, and clouds. It's about real-time processing first, long-term storage second.

Kafka as an Event Streaming Platform

Kafka is an event streaming platform. At its core sits a cluster of Kafka brokers. You can interact with the cluster through a wide range of client APIs in Go, Scala, Python, REST, and more.

There are two popular technologies to build real-time applications and microservices that work with data in Kafka: the Kafka Streams application library and the streaming database ksqlDB. They support all the common operations for stream processing (and more), allowing users to filter streams and tables, join them together, aggregate, store state, and run arbitrary functions over data in motion. Both technologies support stateful computations, including the ability to hold data tables much like a regular database.

For data integration and setting data in existing systems in motion, there is Kafka Connect. This has a whole ecosystem of connectors that interface with a plethora of cloud services (like AWS S3, MongoDB Atlas, Salesforce), relational databases (like Postgres, MySQL, Oracle), and other endpoints, both to continuously import and continuously export data into/from Kafka. Finally, there is a suite of utilities: for instance, Confluent Replicator and Kafka's own Mirror Maker enable geo-replication and interconnecting multiple clusters. For data governance, tools like Schema Registry enforce data contracts by managing and validating schemas for the data that flows through a Kafka-powered infrastructure.

A streaming platform brings these tools together with the purpose of turning data at rest into data in motion. The analogy of a central nervous system is often used. Kafka's ability to scale, to store and process data reliably and efficiently, and to operate without interruption makes it a unique tool for connecting many disparate applications and services across a department or entire organizations. Kafka connectors make it easy to evolve away from legacy systems, by unlocking siloed datasets and turning them into event streams. Stream processing lets applications and microservices analyze and react immediately over these resulting streams of events.

PDFmyURL converts web pages and even full websites to PDF easily and quickly.

Kafka vs. Other Systems

Kafka is the most established event streaming system, but it is not the only one. There exist other, less well-known event streaming systems including Pulsar and Pravega as well as traditional messaging systems like RabbitMQ and ActiveMQ. If you'd like to know more about the difference between these, please read the detailed comparison.

Confluent Cloud is a fully managed Apache Kafka service available on all three major clouds. Try it for free today.

Try It For Free

PDFmyURL converts web pages and even full websites to PDF easily and quickly.

Terms & Conditions | Privacy Policy | Do Not Sell My Information | Modern Slavery Policy | Cookie Settings

Copyright © Confluent, Inc. 2014-2023. Apache, Apache Kafka, Kafka, Apache Flink, Flink, and associated open source project names are trademarks of the Apache Software Foundation

PDFmyURL converts web pages and even full websites to PDF easily and quickly.