upsolver

Whitepaper:

4 Building Blocks of a Streaming Data Architecture

Streaming data is becoming a core component of enterprise data architecture. Streaming technologies are not new, but they have considerably matured over the past year. The industry is moving from painstaking integration of technologies like Kafka and Storm, towards full stack solutions that provide an end-to-end streaming data architecture.

What is Streaming Data Architecture?

A streaming data architecture can ingest and process large volumes of streaming data from multiple sources. While traditional data solutions focused on writing and reading data in batches, a streaming data architecture consumes data

immediately as it is generated, persists it to storage, and may perform real-time processing, data manipulation and analytics.

Why Streaming Data Architecture? Benefits of Stream Processing

Stream processing is becoming an essential data infrastructure for many organizations. Typical use cases include clickstream analytics, which allows companies to track web visitor activities and personalize content; eCommerce analytics which helps online retailers avoid shopping cart abandonment and display more relevant offers; and analysis of large volumes of streaming data from sensors and connected devices in the Internet of Things (IoT).

Stream processing provides several benefits that other data platforms cannot:

- **Able to deal with never-ending streams of events**—some data is naturally structured this way. Traditional batch processing tools require stopping the stream of events, capturing batches of data and combining the batches to draw overall conclusions. In stream processing, while it is challenging to combine and capture data from multiple streams, it lets you derive immediate insights from large volumes of streaming data.
- **Real-time or near-real-time processing**—most organizations adopt stream processing to enable real time data analytics. While real time analytics is also possible with high performance database systems, often the data lends itself to a stream processing model.
- **Detecting patterns in time-series data**—detecting patterns over time, for example looking for trends in website traffic data, requires data to be continuously

processed and analyzed. Batch processing makes this more difficult because it breaks data into batches, meaning some events are broken across two or more batches.

- **Easy data scalability**—growing data volumes can break a batch processing system, requiring you to provision more resources or modify the architecture. Modern stream processing infrastructure is hyper-scalable, able to deal with Gigabytes of data per second with a single stream processor. This allows you to easily deal with growing data volumes without infrastructure changes.

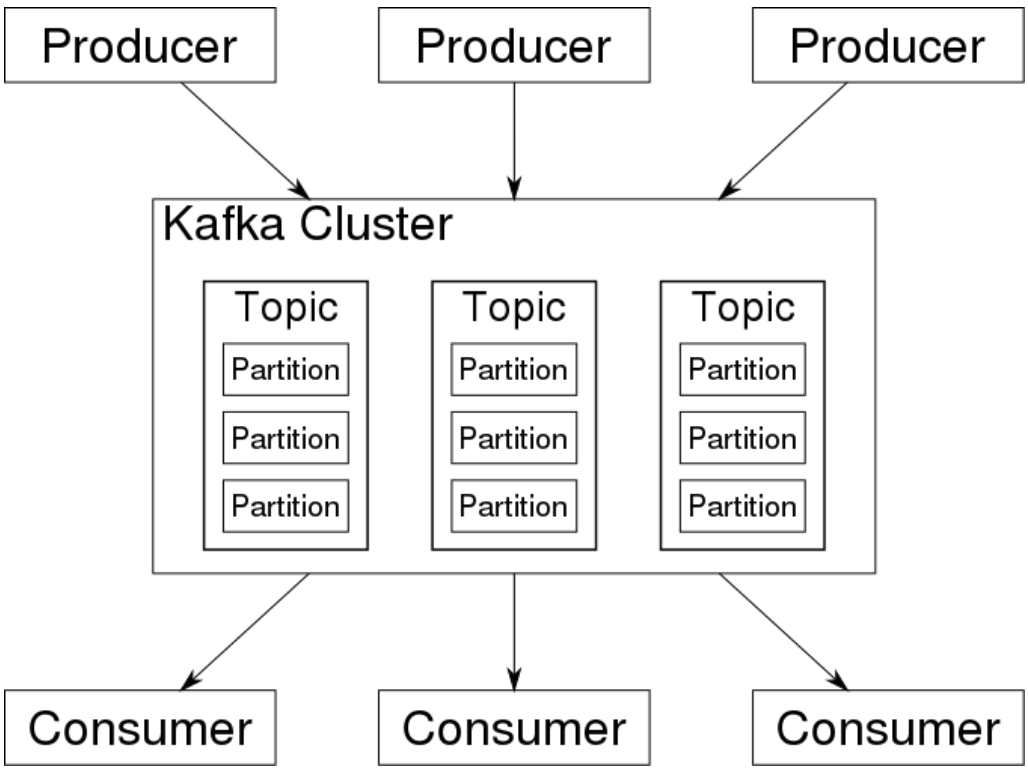
The Components of a Traditional Streaming Architecture

1. The Message Broker

This is the element that takes data from a source, called a producer, translates it into a standard message format, and

streams it on an ongoing basis. Other components can then listen in and consume the messages passed on by the broker.

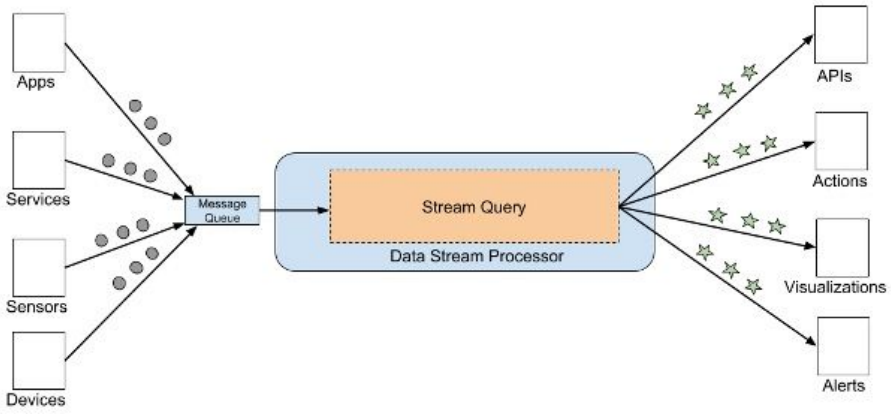
The first generation of message brokers, such as RabbitMQ and Apache ActiveMQ, relied on the Message Oriented Middleware (MOM) paradigm. Later, hyper-performant messaging platforms emerged which are more suitable for a streaming paradigm. Two popular streaming brokers are Apache Kafka and Amazon Kinesis Data Streams.



Unlike the old MoM brokers, streaming brokers support very high performance with persistence, have massive capacity of a Gigabyte per second or more of message traffic, and are tightly focused on streaming with no support for data transformations or task scheduling. You can learn more about message brokers in our article on analyzing Apache Kafka data.

2. Stream Processor / Streaming Data Aggregator

The stream processor collects data streams from one or more message brokers. It receives queries from users, fetches events from message queues and applies the query, to generate a result. The result may be an API call, an action, a visualization, an alert, or in some cases a new data stream.



A few examples of stream processors are Apache Storm, Spark Streaming and WSO2 Stream Processor. While stream processors work in different ways, they are all capable of listening to message streams, processing the data and saving it to storage. Some stream processors, including Spark and WSO2, provide a SQL syntax for querying and manipulating the data.

3. Data Analytics Engine

After streaming data is prepared for consumption by the stream processor, it must be analyzed to provide value. There are many different approaches to streaming data analytics. Here are some of the tools most commonly used for streaming data analytics.

Analytics Tool	Streaming Use Case	Example Setup
Amazon Athena	Distributed SQL engine	Streaming data is saved to S3. You can setup ad hoc SQL queries via the AWS Management Console, Athena runs them as serverless functions and returns results.
Amazon Redshift	Data Warehouse	Amazon Kinesis Streaming Data Firehose can be used to save streaming data to Redshift. This enables near real-time analytics with BI tools and dashboard you have already integrated with Redshift.
Elasticsearch	Text Search	Kafka Connect can be used to stream topics directly into Elasticsearch. If you use the Avro data format and a schema registry, Elasticsearch mappings with correct datatypes are created automatically. You can then perform rapid text search or analytics within Elasticsearch.
Cassandra	Low latency serving of streaming events to apps	Kafka streams can be processed and persisted to a Cassandra cluster. You can implement another Kafka instance that receives a stream of changes from Cassandra and serves them to applications for real time decision making.

4. Streaming Data Storage

With the advent of low cost storage technologies, most organizations today are storing their streaming event data. Here are several options for storing streaming data, and their pros and cons.

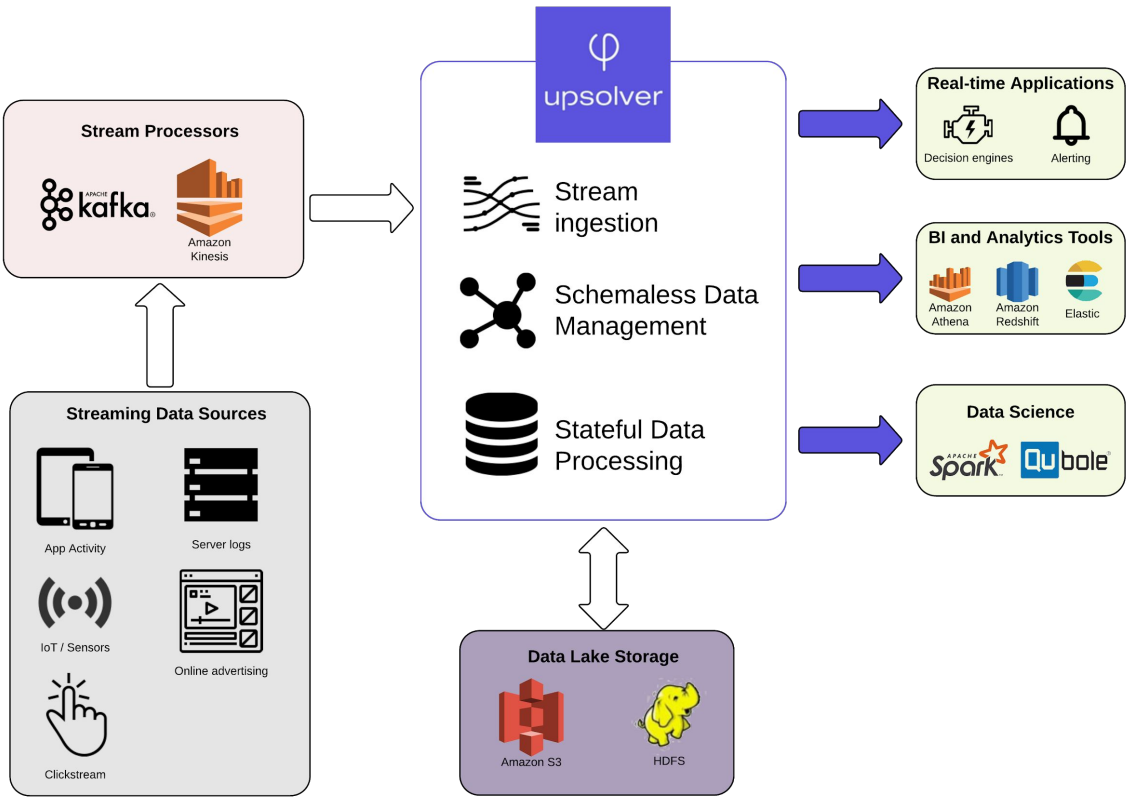
Streaming Data Storage Option	Pros	Cons
In a database or data warehouse—for example, PostgreSQL or Amazon Redshift	Easy SQL-based data analysis.	Hard to scale and manage. If cloud-based, storage is expensive.
In the message broker—for example, using Kafka persistent storage	Agile, no need to structure data into tables. Easy to set up, no additional components.	Data retention is an issue since Kafka storage is up to 10X more expensive compared to data lake storage. Kafka performance is best for reading recent (cached) data (cached).
In a data lake—for example, Amazon S3	Agile, no need to structure data into tables. Low cost storage.	High latency, makes real time analysis difficult. Difficult to perform SQL-based analysis.

A data lake is the most flexible and inexpensive option for storing event data, but it has several limitations for streaming

data applications. Upsolver provides a data lake platform that ingests streaming data into a data lake, creates schema-on-read, and extracts metadata. This allows data consumers to easily prepare data for analytics tools and real time analytics.

Modern Streaming Architecture

In modern streaming data deployments, many organizations are adopting a full stack approach. Vendors are providing technology solutions, most of them based on Kafka, which can take streaming data and perform the entire process, from message ingestion through ETL, storage management and preparing data for analytics.



Benefits of a modern streaming architecture:

- Can eliminate the need for large data engineering projects
- Performance, high availability and fault tolerance built in
- Newer platforms are cloud-based and can be deployed very quickly with no upfront investment
- Flexibility and support for multiple use cases

The Future of Streaming Data in 2019 and Beyond

Streaming data architecture is in constant flux. Three trends we believe will be significant in 2019 and beyond:

- **Fast adoption of platforms that decouple storage and compute**—streaming data growth is making traditional data warehouse platforms too expensive and cumbersome to manage. Data lakes are increasingly used, both as a cheap persistence option for storing large volumes of event data, and as a flexible integration point, allowing tools outside the streaming ecosystem to access streaming data.
- **From table modeling to schemaless development**—data consumers don't always know the questions they will ask in advance. They want to run an interactive, iterative process with as little initial setup as possible. Lengthy table modeling, schema detection and metadata extraction are a burden.

- **Automation of data plumbing**—organizations are becoming reluctant to spend precious data engineering time on data plumbing, instead of activities that add value, such as data cleansing or enrichment. Increasingly, data teams prefer full stack platforms that reduce time-to-value, over tailored home-grown solutions.

You can read more of our predictions for streaming data trends here.

Want to enhance your streaming architecture? Upsolver's streaming data platform processes event data and ingests it into data lakes, data warehouses, serverless platforms, elasticsearch, and much more. Furthermore, it enables real time analytics, using low-latency consumers that read from a Kafka stream in parallel. It is a fully integrated solution that can be set up within hours.

By using Upsolver, you get the best of both worlds—low cost storage on a data lake, easy transformation to tabular formats, and real time support. Begin your free trial to start building a next-gen streaming data architecture.