Apache Druid and Data Rivers

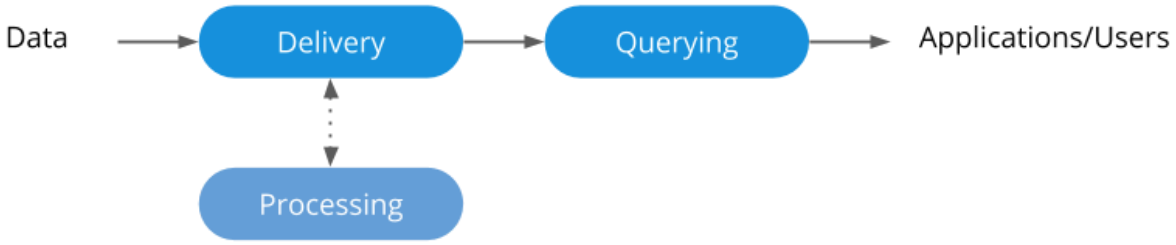
Enterprise Reference Architecture

Data Rivers

The data river architecture consists of 3 primary components:

- A delivery system to deliver events from one location to another
- A processing system to clean or transform data
- A querying system to store and analyze data (Druid/Imply)

Each piece of the stack works natively with streams and is designed to do a specific set of things very well. The combination of technologies is flexible enough to handle a wide variety of processing requirements and query loads.



In order to provide flexibility, preserve data integrity, guarantee availability, and easily scale, each of the 3 functions (delivery, processing, and querying) must have a dedicated system behind it. Each function is distinct enough where a single system is not sufficient to handle the workload.

Data Delivery

Data delivery systems are responsible for delivering events from one location to another. In the streaming world, data delivery systems are also known as message buses or stream hubs. Modern technologies here include Apache Kafka, RabbitMQ, RocketMQ, AWS Kinesis, and many others.

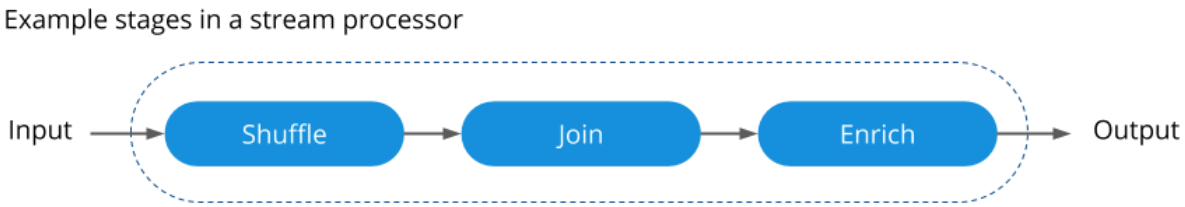
To better understand stream hubs, consider Apache Kafka (one of the most popular stream hubs today) as an example. Kafka has a notion of producers and consumers. Producers write events into Kafka brokers, which retain events for a period of time. Consumers, typically other downstream systems, read events from the Kafka brokers. By isolating

systems that produce data from those that consume it, we minimize the impact that disruptions or downtime in one place can have on the other.

Data Processing (Optional)

Stream-focused data processing systems are known as stream processors or CEP engines. Modern technologies include Apache Flink, Spark Streaming, Kafka Streams, and many others. Although some of these systems can support simple analytic queries, stream processors typically do not store data in formats that are optimized for such queries. However, their core architectures are very well suited for data processing.

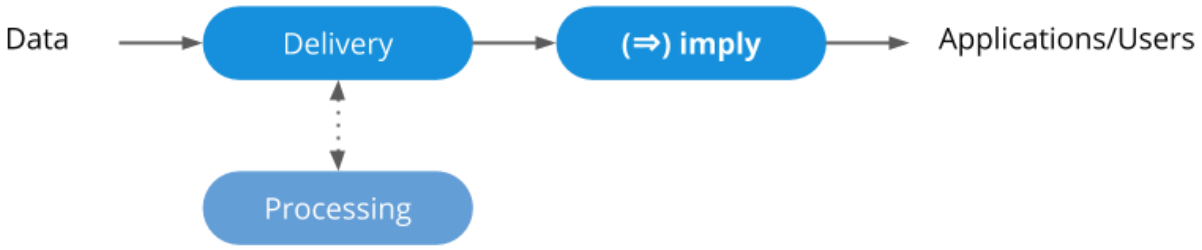
Data processing involves cleaning or transforming data so that it is more useful for downstream query systems. Steps may include denormalizing data, joining different streams together, enriching data, and many other useful functions.



Data processors are an optional part of the data river architecture. If your data is simple and clean enough that it does not need much complex processing, you can write events from the delivery system directly into the querying system.

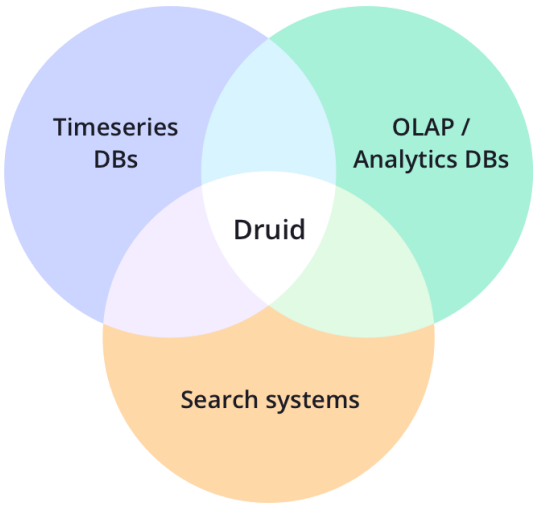
Data Querying

Data querying systems are primarily responsible for storing and analyzing data. While a traditional data warehouse can be considered to be a query system, data warehouses are a poor fit for streaming data. Data warehouses are not optimized for streaming ingestion, ad-hoc analysis, or workloads that require immediacy. A system such as Imply, powered by Druid, is built natively to work with streams in several different ways.



Druid can ingest data exactly once from stream hubs such as Kafka, and data is available for queries milliseconds after it is first seen in the delivery system. In the event the delivery system is Kafka, or a similar architecture such as AWS Kinesis or RocketMQ, Druid acts as a Kafka consumer, and automatically handles deduplication of events from Kafka. Furthermore, Druid is able to handle out of order events and delayed events while maintaining high availability and data consistency.

As a query technology, Druid's core architecture is designed for ad-hoc slice and dice queries (OLAP queries), which enables end users to dynamically explore and understand data as it is happening in real-time. Druid's architecture combines ideas from analytics databases, timeseries databases, and search engines. Similar to analytics databases, Druid stores data in a column orientation. This allows Druid to provide fast scans of data, making it ideal for numerical aggregations. Druid is also designed for complex, multi-dimensional groupBys. Similar to timeseries databases, Druid partitions data by time. This allows queries that involve time to be completed much faster than in any traditional database system. Similar to a search engine, Druid creates inverted indexes for very fast search and filter queries. This makes Druid ideal for queries that involve WHERE clauses.



Druid can not only stream in real-time data, it can also batch load historical data as well, and retain data indefinitely. This enables workloads where current trends can be immediately compared against historical ones.

Imply, as the primary company behind Apache Druid, can help you build data rivers in your organization. The architecture is a powerful solution for a variety of use cases across user experience, QoS, network analysis, IoT workloads, and much more.