**Introduction**

In this lesson, we will discuss stream processing and its use cases.

**We'll cover the following**

* [Rise of data-driven systems](https://www.educative.io/module/lesson/web-application-architecture-101/qAlxm0M55PG#Rise-of-data-driven-systems)
* [Use cases for data-stream processing](https://www.educative.io/module/lesson/web-application-architecture-101/qAlxm0M55PG#Use-cases-for-data-stream-processing)

**Rise of data-driven systems**[#](https://www.educative.io/module/lesson/web-application-architecture-101/qAlxm0M55PG#Rise-of-data-driven-systems)

Our world today is largely data-driven and is progressing towards becoming entirely data-driven. With the advent of the Internet of Things(IoT), entities have gained self-awareness to a certain degree, and they are generating and transmitting data online at an unprecedented rate. They are capable of communicating with each other and can make decisions without any sort of human intervention.

**Use cases for data-stream processing**[#](https://www.educative.io/module/lesson/web-application-architecture-101/qAlxm0M55PG#Use-cases-for-data-stream-processing)

The primary large-scale use of IoT devices is in sensors in industries and smart cities, electronic gadgets like drones, smartwatches, cellphones, etc., wearable healthcare body sensors, and so on.

To manage the massive amount of data streaming in, we need to have sophisticated backend systems in place to gather meaningful information and archive/purge not so meaningful data.

The more data we have, the better our systems evolve. Today’s businesses rely on data. They need customer data to make plans and projections. They need to understand the user’s needs and their behavior.

All these things enable businesses create better products, make smarter decisions, run more effective ad campaigns, recommend new products to their customers, gain better insights into the market, etc.

Study of this large-scale data eventually results in more customer-centric products and increased customer loyalty.

Another use case of processing streaming-in data is tracking the service efficiency, for instance, getting the “everything is okay” signal from the IoT devices used by millions of customers. This metadata keeps the businesses informed of the correct functionality and uptime of their products.

All these use cases and more make stream processing key to businesses and modern software applications. Time-series database is one tech I discussed that persists and runs queries on real-time data being ingested from the IoT devices.

In the next lesson, let’s explore the components involved in data-stream processing. We will also look at some of the key architectures in the data processing domain.

# Data Ingestion

In this lesson, we will gain insight into the process of data ingestion.

**We'll cover the following**

* [What is data ingestion?](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#What-is-data-ingestion?)
* [Layers of data processing setup](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Layers-of-data-processing-setup)
* [Data standardization](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-standardization)
* [Data processing](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-processing)
* [Data analysis](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-analysis)
* [Data visualization](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-visualization)
* [Data storage and security](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-storage-and-security)

## What is data ingestion?[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#What-is-data-ingestion?)

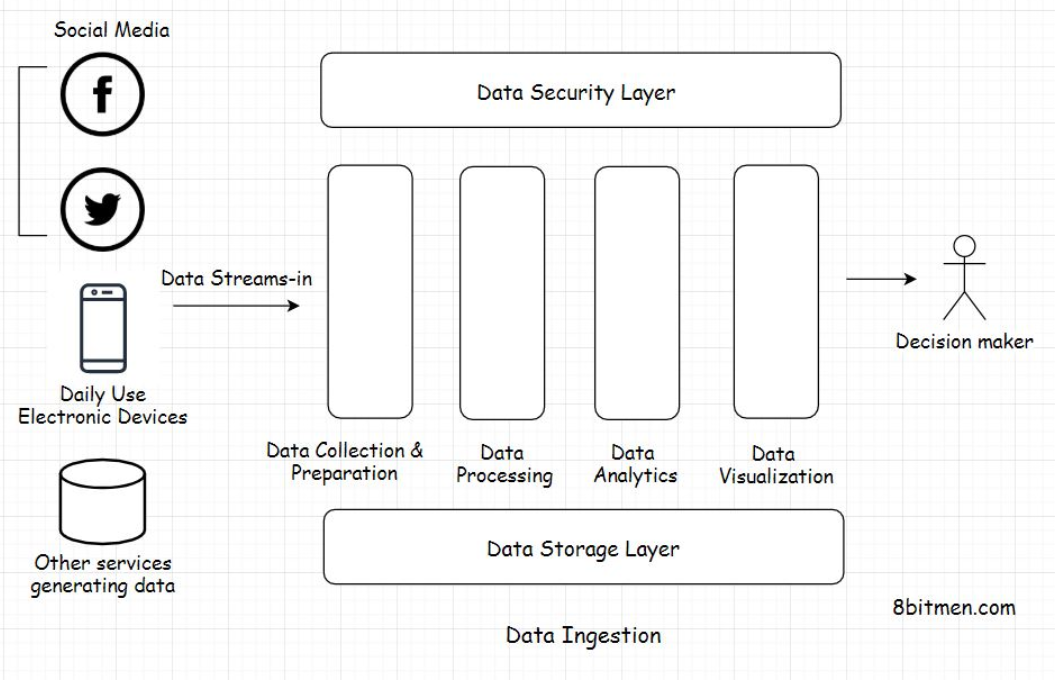
Data ingestion is a collective term for the process of collecting data streaming in from several different sources and making it ready to be processed by the system.

In a data processing system, the data is ingested from the IoT devices and other sources into the system to be analyzed. It is routed to different components/layers through the data pipelines, algorithms are run on it, and is eventually archived.

## Layers of data processing setup[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Layers-of-data-processing-setup)

There are several stages/layers to this whole data processing setup, such as the:

* Data collection layer
* Data query layer
* Data processing layer
* Data visualization layer
* Data storage layer
* Data security layer



As you can see in the diagram, all the data processing layers are pretty self-explanatory.

## Data standardization[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-standardization)

The data, which streams in from several different sources, is not in a homogeneous structured format. We have already gone through different types of data, including structured, unstructured, and semi-structured, in the database lesson. So, you have an idea of what unstructured, heterogeneous data is.

Data streams in into the system at different speeds and sizes from web-based services, social networks, IoT devices, industrial machines, and whatnot. Every stream of data has different semantics.

So, in order to make the data uniform and fit for processing, it has to be first collected and converted into a standardized format to avoid any future processing issues. This data standardization process occurs in the data collection and preparation layer.

## Data processing[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-processing)

Once the data is transformed into a standard format, it is routed to the data processing layer, where it is further processed based on the business requirements. It is generally classified into different flows and routed to different destinations.

## Data analysis[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-analysis)

After being routed, analytics are run on the data. This includes executing different analytics models such as predictive modeling, statistical analytics, text analytics, etc. All the analytical events occur in the data analytics layer.

## Data visualization[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-visualization)

Once the analytics is run and we have valuable intel from it, all the information is routed to the data visualization layer to be presented before the stakeholders, generally in a web-based dashboard.

Kibana is one good example of a data visualization tool widely used in the industry.

## Data storage and security[#](https://www.educative.io/module/lesson/web-application-architecture-101/g2DrLNwY3pl#Data-storage-and-security)

Moving data is highly vulnerable to security breaches. The data security layer ensures the secure movement of data all along. Speaking of the data storage layer, as the name implies, is instrumental in persisting the data.

So, this is the gist of how massive amount of data is processed and analyzed for business use cases. This is just a bird’s eye view of things. The field of data analytics is pretty deep, and an in-depth detailed microscopic view of each layer demands a dedicated data analytics course for itself.

In the next lesson, let’s take a look at the different ways in which data can be ingested.

# Different Ways of Ingesting Data and the Challenges Involved

In this lesson, we will discuss the different ways we can ingest data. We will also cover the challenges involved in the process.

**We'll cover the following**

* [Different ways to ingest data](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Different-ways-to-ingest-data)
* [Challenges with data ingestion](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Challenges-with-data-ingestion)
  + [Slow process](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Slow-process)
  + [Complex and Expensive](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Complex-and-Expensive)
  + [Moving data around is risky](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Moving-data-around-is-risky)

## Different ways to ingest data[#](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Different-ways-to-ingest-data)

There are two primary ways to ingest data: in real-time and in batches that run at regular intervals. Which of the two to pick depends entirely on the business requirements.

Data ingestion in real-time is typically preferred in systems reading medical data, like heartbeat or blood pressure, via wearable IoT sensors. It is also preferred in systems handling financial data like stock market events, etc. These are a few instances where time, lives, and money are closely linked, and we need information as soon as we can get it.

On the contrary, in systems that read trends over time, we can always ingest data in batches. For instance, a system that shows data on the popularity of a sport in a region over a period of time. Real-time data in this use case isn’t critical.

Let’s talk about some of the challenges developers face when ingesting massive amounts of data. I have added this lesson just to give you a deeper insight into the entire process. In the upcoming lesson, I’ll talk about the general use-cases of data streaming in the application development domain.

## Challenges with data ingestion[#](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Challenges-with-data-ingestion)

### Slow process[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Slow-process)

Data ingestion is a slow process. Why? When the data is streamed from several different sources into the system, data coming from each source has a different format, different syntax, attached metadata, etc. The data as a whole is heterogeneous. It has to be transformed into a standard format like JSON or something to be understood well by the analytics system.

This conversion of data is a tedious process. It takes a lot of computing resources and time. Flowing data has to be staged at several stages in the pipeline, processed, and then moved ahead.

Also, data has to be authenticated and verified at every stage to meet the organization’s security standards. With the traditional data cleansing processes, it takes weeks, even months, to get useful information on hand. Traditional data ingestion systems like ETL ain’t that effective anymore.

Okay! But I just said data can be ingested in real-time right? So, how could it be slow?

I want to bring up two things here. First, the modern data processing tech and frameworks are continually evolving to beat the limitations of the legacy, traditional data processing systems. Real-time data ingestion wasn’t even possible with the traditional systems.

Second, analytics information obtained from real-time processing is not that accurate or holistic since the analytics continually runs on a limited set of data. On the contrary, in batch processing, the entire data set is taken into account. The more time we spend studying the data, the more accurate results we get.

You’ll learn more about this when we go through the Lambda and the Kappa architectures of data processing.

### Complex and Expensive[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Complex-and-Expensive)

The entire data flow process is resource-intensive. Much heavy lifting has to be done to prepare the data before being ingested into the system. Also, this isn’t a side process. A dedicated team is required to pull off something like this.

Engineering teams often come across scenarios where the tools and frameworks available in the market fail to serve their needs. They have no option other than to write a custom solution from the bare bones.

Goblin is a data ingestion tool by LinkedIn. At one point, LinkedIn had fifteen data ingestion pipelines running, which created several data management challenges. To tackle this problem, [LinkedIn wrote Goblin](https://engineering.linkedin.com/data-ingestion/gobblin-big-data-ease) in-house. <https://engineering.linkedin.com/data-ingestion/gobblin-big-data-ease>

Today the IoT machines in the industry are evolving at a rapid pace. The semantics of the data coming from external sources also changes sometimes because the data sources are not always under our control, which warrants a change in the backend data processing code.

### Moving data around is risky[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/gxkwJVrGY4Z#Moving-data-around-is-risky)

When data is moved around, it opens up the possibility of a breach. Moving data is vulnerable. It goes through several different staging areas, and the engineering teams have to put in additional effort and resources to ensure their system meets the security standards at all times.

These are some of the challenges developers face when working with streaming data.

# Data Ingestion Use Cases

In this lesson, we will discuss some of the industry's common data ingestion use cases.

**We'll cover the following**

* [Moving Big Data into Hadoop](https://www.educative.io/module/lesson/web-application-architecture-101/m7JnlJkkORR#Moving-Big-Data-into-Hadoop)
* [Streaming data from databases to Elasticsearch server](https://www.educative.io/module/lesson/web-application-architecture-101/m7JnlJkkORR#Streaming-data-from-databases-to-Elasticsearch-server)
* [Log processing](https://www.educative.io/module/lesson/web-application-architecture-101/m7JnlJkkORR#Log-processing)
* [Stream processing engines for real-time events](https://www.educative.io/module/lesson/web-application-architecture-101/m7JnlJkkORR#Stream-processing-engines-for-real-time-events)

This is where I talk about some of the data streaming use cases commonly implemented in the industry.

## Moving Big Data into Hadoop

This is data ingestion’s most popular use. As discussed before, Big Data from IoT devices, social apps, and other sources streams through data pipelines and moves into the most popular distributed data processing framework Hadoop for analysis.

## Streaming data from databases to Elasticsearch server

Elasticsearch is an open-source framework for implementing search in web applications. It is a de facto search framework used in the industry simply because of its advanced features and it being open source. Being open-source enables businesses to write their own custom solutions when they need them.

In the past, I wrote a product search as a service using Java, Spring Boot, and Elastic search. A large amount of data was streamed from the legacy storage solutions to the Elasticsearch server and indexed to make the products come up in the search results.

All the data intended to show up in the search was replicated from the main storage to the Elasticsearch storage. Also, as the new data was persisted in the main storage, it was asynchronously delivered to the Elastic server in real-time for indexing.

## Log processing

If your project isn’t a hobby project, chances are it’s running on a cluster. When we talk about running a large-scale service, monolithic systems don’t cut it. With so many microservices running concurrently, there is a massive number of logs, which are generated over a period of time. Logs are the only way to move back in time, track errors, and study the system’s behavior.

So, to study the behavior of the system holistically, we have to stream all the logs to a central place. All the logs are ingested to a central server to run analytics with the help of solutions like the ELK (Elastic Logstash Kibana) stack, etc.

## Stream processing engines for real-time events

Real-time streaming and data processing are the core components in systems handling LIVE information such as sports. It’s imperative that the architectural setup in place is efficient enough to ingest data, analyze it, figure out the behavior in real-time, and quickly push the updated information to the fans.

Message queues like Kafka and stream computation frameworks like Apache Storm, Apache Nifi, Apache Spark, Samza, Kinesis, etc., are used to implement the real-time large-scale data processing features in online applications.

Here is a good read on the topic: An insight into [Netflix’s real-time streaming platform.](https://netflixtechblog.com/keystone-real-time-stream-processing-platform-a3ee651812a) (<https://netflixtechblog.com/keystone-real-time-stream-processing-platform-a3ee651812a> )

Let’s look into data pipelines in the next lesson.

# Data Pipelines

In this lesson, you will learn about data pipelines.

**We'll cover the following**

* [What are data pipelines?](https://www.educative.io/module/lesson/web-application-architecture-101/YMYNylyyxMp#What-are-data-pipelines?)
* [Features of data pipelines](https://www.educative.io/module/lesson/web-application-architecture-101/YMYNylyyxMp#Features-of-data-pipelines)
* [What is ETL?](https://www.educative.io/module/lesson/web-application-architecture-101/YMYNylyyxMp#What-is-ETL?)

## What are data pipelines?[#](https://www.educative.io/module/lesson/web-application-architecture-101/YMYNylyyxMp#What-are-data-pipelines?)

Data pipelines are the core component of a data processing infrastructure. They facilitate the efficient flow of data from one point to another and enable developers to apply filters on the data streaming-in in real-time.

Today’s enterprise is data-driven, and data pipelines are key in implementing scalable analytics systems.

## Features of data pipelines[#](https://www.educative.io/module/lesson/web-application-architecture-101/YMYNylyyxMp#Features-of-data-pipelines)

Speaking of some more features of the data pipelines -

* They ensure a smooth flow of data.
* They enable the business to apply filters and business logic on streaming data.
* They avert any bottlenecks and redundancy in the data flow.
* They facilitate parallel processing of the data.
* They protect data from being corrupted and so on.

Data pipelines work on a set of rules predefined by the engineering teams, and the data is routed accordingly without any manual intervention. The entire flow of data: extraction, transformation, combination, validation and the convergence of data from multiple streams into one is automated.

Data pipelines facilitate the parallel processing of data via managing multiple streams. I’ll talk more about distributed data processing in the upcoming lesson.

Traditionally we used ETL systems to manage all of the data’s movement across the system, but one major limitation with the technology is that it doesn’t support the management of real-time streaming data, which on the contrary is possible with the new era data processing infrastructure powered by the data pipelines.

## What is ETL?[#](https://www.educative.io/module/lesson/web-application-architecture-101/YMYNylyyxMp#What-is-ETL?)

If you haven’t heard of ETL before, it means Extract Transform Load.

**Extract** means fetching data from single or multiple data sources.

**Transform** means transforming the extracted heterogeneous data into a standardized format based on the rules set by the business.

**Load** means moving the transformed data to a data warehouse or another data storage location for further processing of data.

The ETL flow is the same as the data ingestion flow. The difference is just that the entire movement of data is done in batches as opposed to streaming it through the data pipelines in real-time.

Though real-time data processing offers fast results, it doesn’t undermine the importance of the batch processing approach. Moreover, companies leverage both data processing techniques in their projects to get the best of both worlds.

You’ll gain more insight into it when we go through the Lambda and Kappa architectures of distributed data processing in the upcoming lessons.

In the previous lesson, I brought up a few popular data processing tools, such as Apache Flink, Storm, Spark, Kafka, etc. All these tools have one thing in common they facilitate processing data in a cluster in a distributed environment via data pipelines.

What is distributed data processing? How does it work? We are going to look into this in the next lesson. Stay tuned.

# Distributed Data Processing

In this lesson, we will discuss distributed data processing and the technologies used for it.

**We'll cover the following**

* [What is distributed data processing?](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#What-is-distributed-data-processing?)
* [Distributed data processing technologies](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Distributed-data-processing-technologies)
  + [MapReduce – Apache Hadoop](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#MapReduce-%E2%80%93-Apache-Hadoop)
  + [Apache Spark](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Apache-Spark)
  + [Apache Storm](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Apache-Storm)
  + [Apache Kafka](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Apache-Kafka)

This lesson is all about distributed data processing. I’ll talk about what it is, how different it is compared to a centralized data processing system, the architectures involved in it, and more.

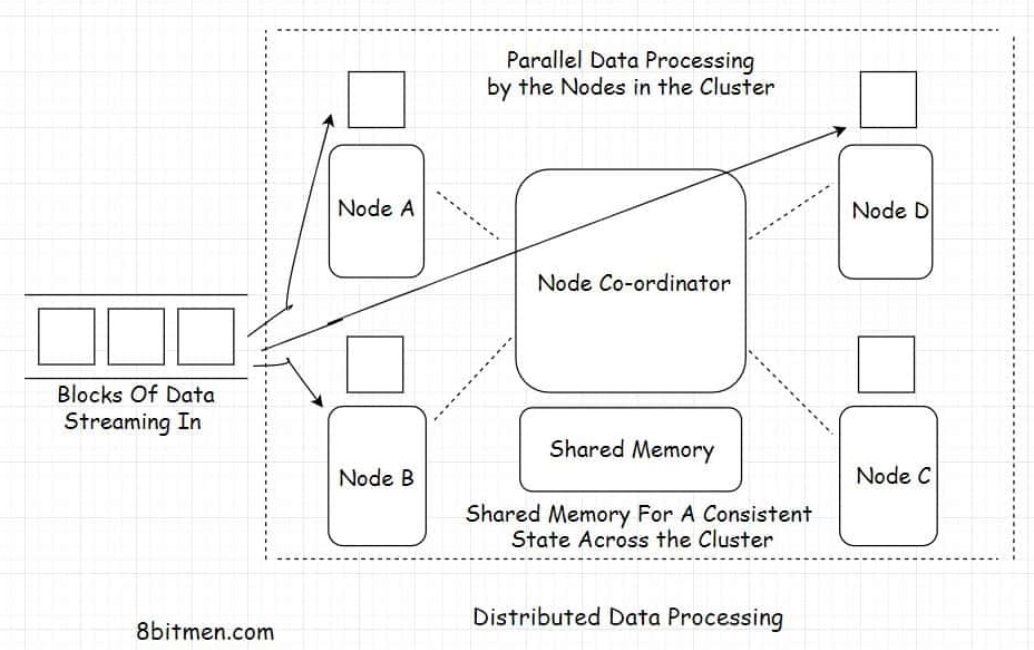
So, without further ado, let’s get on with it.

## What is distributed data processing?[#](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#What-is-distributed-data-processing?)

Distributed-data processing means diverging large amounts of data to several nodes running in a cluster for parallel processing.

All the nodes in the cluster execute the task allotted parallelly, working in conjunction coordinated by a node-coordinator. Apache Zookeeper is one example of a node coordinator widely used in the industry.

Since the nodes are distributed and the tasks are executed parallelly, this makes the entire set-up pretty scalable and highly available. The workload can be scaled both horizontally and vertically. Data is made redundant and replicated across the cluster to avoid any sort of data loss.



Processing data in a distributed environment helps accomplish tasks in significantly less time as opposed to when running it on a centralized data processing system. In a distributed system, the tasks are shared by several nodes. In a centralized system, the tasks are queued to be processed one by one.

## Distributed data processing technologies[#](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Distributed-data-processing-technologies)

Here are some of the popular technologies used in the industry for large-scale data processing.

### MapReduce – Apache Hadoop[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#MapReduce-%E2%80%93-Apache-Hadoop)

MapReduce is a programming model written for managing distributed data processing across several different machines in a cluster. This involves distributing tasks to several machines, running work in parallel, and managing all the communication and data transfer within different parts of the system.

The map part of the programming model involves sorting the data based on a parameter and the reduce part involves summarizing the sorted data.

The most popular open-source implementation of the MapReduce programming model is Apache Hadoop. The framework is used by all big guns in the industry to manage massive amounts of data in their system. It is used by Twitter for running analytics and by Facebook for storing big data.

### Apache Spark[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Apache-Spark)

Apache Spark is an open-source cluster computing framework. It provides high performance for both batch and real-time in-stream processing. It can work with diverse data sources and facilitates parallel execution of work in a cluster.

Spark has a cluster manager and distributed data storage. The cluster manager facilitates communication between different nodes running together in a cluster, whereas the distributed storage facilitates storing Big Data. Spark seamlessly integrates with distributed data stores like Cassandra, HDFS, MapReduce File System, Amazon S3, etc.

### Apache Storm[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Apache-Storm)

Apache Storm is a distributed stream processing framework. In the industry, it is primarily used for processing massive amounts of streaming data. It has several different use cases, such as real-time analytics, machine learning, distributed remote procedure calls, etc.

### Apache Kafka[**#**](https://www.educative.io/module/lesson/web-application-architecture-101/JPvpm3L54lK#Apache-Kafka)

Apache Kafka is an open-source distributed stream processing and messaging platform. It’s written using Java and Scala and was developed by LinkedIn.

The storage layer of Kafka involves a distributed scalable pub-sub message queue. It helps read and write streams of data like a messaging system.

Kafka is used to develop real-time features such as notification platforms, managing streams of massive amounts of data, monitoring website activity and metrics, messaging, and log aggregation.

Hadoop is preferred for batch data processing, whereas Spark, Kafka, and Storm are the right pick for processing real-time streaming data.

By now, I am sure you have a good idea of what data processing is and are aware of its use-cases in modern application development and the associated technologies.

In the lesson up next, let’s take a look at a couple of architectures involved in the process: Lambda and Kappa.

# Lambda Architecture

In this lesson, you will learn about the Lambda architecture of data processing.

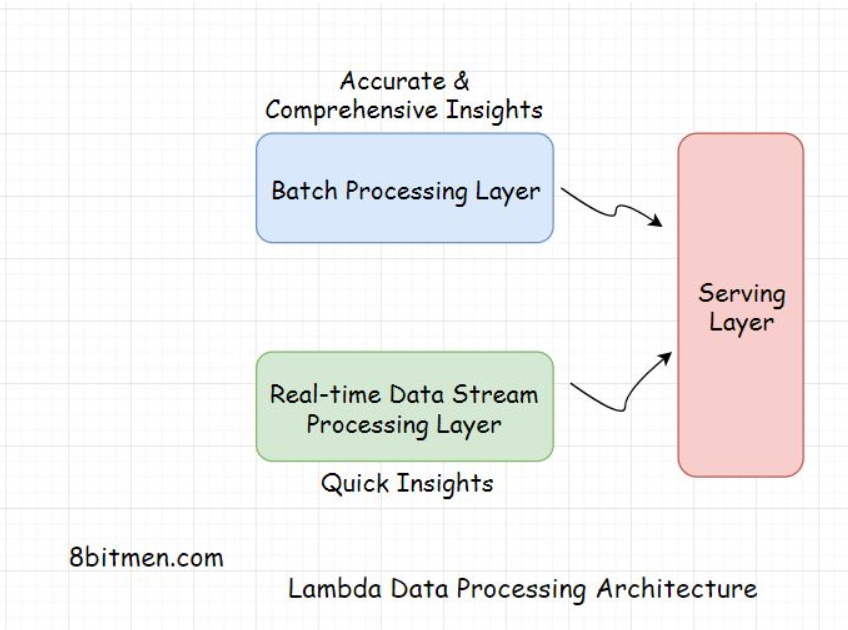
**We'll cover the following**

* [What is lambda architecture?](https://www.educative.io/module/lesson/web-application-architecture-101/YM9PKOWRGz2#What-is-lambda-architecture?)
* [Layers of the Lambda architecture](https://www.educative.io/module/lesson/web-application-architecture-101/YM9PKOWRGz2#Layers-of-the-Lambda-architecture)

## What is lambda architecture?[#](https://www.educative.io/module/lesson/web-application-architecture-101/YM9PKOWRGz2#What-is-lambda-architecture?)

Lambda is a distributed data processing architecture that leverages both the batch and the real-time streaming data processing approaches to tackle the latency issues that arise out of the batch processing approach.

It joins the results from both approaches before presenting them to the end-user. Lambda architecture makes the most of the two approaches.



Batch processing does take time, considering the massive amounts of data businesses have today. However, the accuracy of the approach is high, and the results are comprehensive.

On the contrary, real-time streaming data processing provides quick access to insights. In this scenario, the analytics is run over a small portion of data, so the results are not as accurate or comprehensive as the batch approach.

**Layers of the Lambda architecture**[#](https://www.educative.io/module/lesson/web-application-architecture-101/YM9PKOWRGz2#Layers-of-the-Lambda-architecture)

The architecture has typically three layers:

* Batch layer
* Speed layer
* Serving layer

The batch layer deals with the results acquired via batch processing of the data. The Speed layer gets data from the real-time streaming data processing, and the serving layer combines the results obtained from both the batch and the speed layers.

# Kappa Architecture

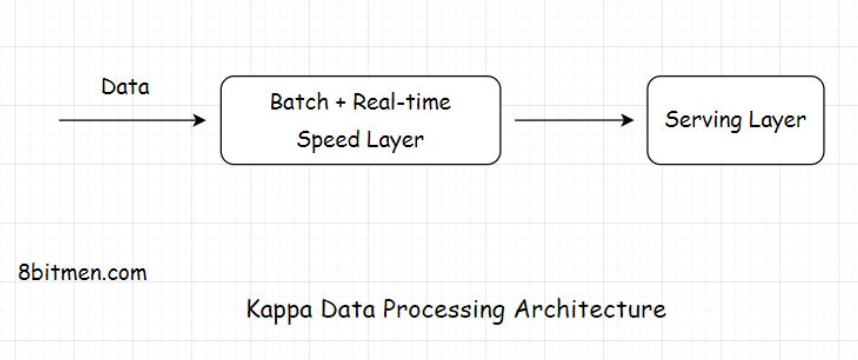
In this lesson, we will discuss the Kappa architecture of data processing.

**We'll cover the following**

* [What is Kappa architecture?](https://www.educative.io/module/lesson/web-application-architecture-101/N7g5JEkgPgp#What-is-Kappa-architecture?)
* [Layers of Kappa architecture](https://www.educative.io/module/lesson/web-application-architecture-101/N7g5JEkgPgp#Layers-of-Kappa-architecture)

## What is Kappa architecture?[#](https://www.educative.io/module/lesson/web-application-architecture-101/N7g5JEkgPgp#What-is-Kappa-architecture?)

In Kappa architecture, all the data flows through a single data streaming pipeline in contrast to the Lambda architecture, which has different data streaming layers that converge into one.



The architecture flows the data of both real-time and batch processing through a single streaming pipeline, reducing the complexity of managing separate layers for processing data.

## Layers of Kappa architecture[#](https://www.educative.io/module/lesson/web-application-architecture-101/N7g5JEkgPgp#Layers-of-Kappa-architecture)

Kappa contains only two layers: Speed, which is the streaming processing layer, and Serving, which is the final layer. Also, Kappa is not an alternative for Lambda. Both the architectures have their use cases.

Kappa is preferred if the batch and the streaming analytics results are fairly identical in a system. Lambda is preferred if they are not.

Well, this concludes the stream processing chapter. Setting up and running a distributed system is something that is not trivial and requires years of work to perfect the system.

With this being said, let’s move on to the next chapter, where I talk about different kinds of architectures involved in the software development domain.