DATA PIPELINE

A data pipeline is a series of data processing steps.

If the data is not currently loaded into the data platform, then it is ingested at the beginning of the pipeline. Then there are a series of steps in which each step delivers an output that is the input to the next step. This continues until the pipeline is complete. In some cases, independent steps may be run in parallel.

Data pipelines consist of three key elements: a source, a processing step or steps, and a destination. In some data pipelines, the destination may be called a sink. Data pipelines enable the flow of data from an application to a data warehouse, from a data lake to an analytics database, or into a payment processing system,

*For example*, Data pipelines also may have the same source and sink, such that the pipeline is purely about modifying the data set. Any time data is processed between point A and point B (or points B, C, and D), there is a data pipeline between those points.

End-to-end data pipelines will properly equip organizations to source, collect, manage, analyse, and effectively use crucial data to generate new market opportunities and deliver cost-saving business processes.

Data generated in one source system or application may feed multiple data pipelines, and those pipelines may have multiple other pipelines or applications that are dependent on their outputs.

Common steps in data pipelines include data transformation, augmentation, enrichment, filtering, grouping, aggregating, and the running of algorithms against that data.

What Is a Big Data Pipeline?

As the volume, variety, and velocity of data have dramatically grown in recent years, architects and developers have had to adapt to “big data.”

The term “big data” implies that there is a huge volume to deal with. This volume of data can open opportunities for use cases such as predictive analytics, real-time reporting, and alerting, among many examples.

Like many components of data architecture, data pipelines have evolved to support big data. Big data pipelines are data pipelines built to accommodate one or more of the three traits of big data.

The velocity of big data makes it appealing to build streaming data pipelines for big data. Then data can be captured and processed in real time so some action can then occur.

The volume of big data requires that data pipelines must be scalable, as the volume can be variable over time.

The variety of big data requires that big data pipelines be able to recognize and process data in many different formats—structured, unstructured, and semi-structured.

Data Pipeline vs. ETL

ETL refers to a specific type of data pipeline. ETL stands for “extract, transform, load.”

It is the process of moving data from a source, such as an application, to a destination, usually a data warehouse.

“Extract” refers to pulling data out of a source; “transform” is about modifying the data so that it can be loaded into the destination, and “load” is about inserting the data into the destination.

Data Pipeline Considerations

Data pipeline architectures require many considerations.

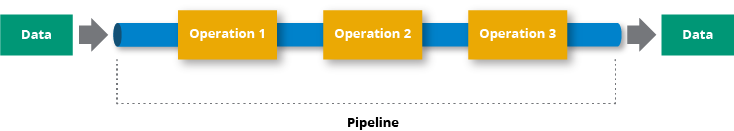
For example,

* Does your pipeline need to handle streaming data?
* What rate of data do you expect?
* How much and what types of processing need to happen in the data pipeline?
* Is the data being generated in the cloud or on-premises, and where does it need to go?
* Do you plan to build the pipeline with micro services?
* Are there specific technologies in which your team is already well-versed in programming and maintaining?

Architecture Examples

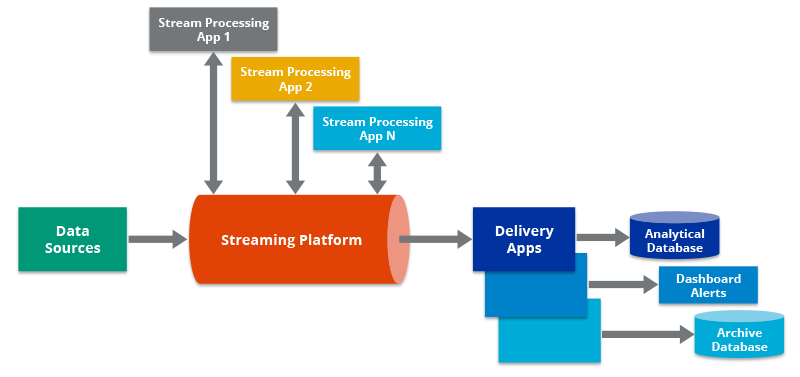
One common example is a batch-based data pipeline. In that example, you may have an application such as a point-of-sale system that generates a large number of data points that you need to push to a data warehouse and an analytics database. Here is an example of what that would look like:

Picture - A basic example of a data pipeline.



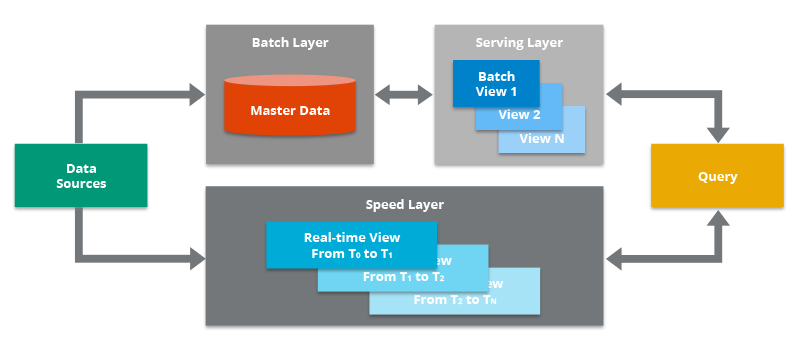
Another example is a streaming data pipeline. In a streaming data pipeline, data from the point of sales system would be processed as it is generated. The stream processing engine could feed outputs from the pipeline to data stores, marketing applications, and CRMs, among other applications, as well as back to the point of sale system itself.

Picture - This diagram models a streaming data pipeline. The data stream is managed by the stream processing framework where it can be processed and delivered to apps and/or solutions.



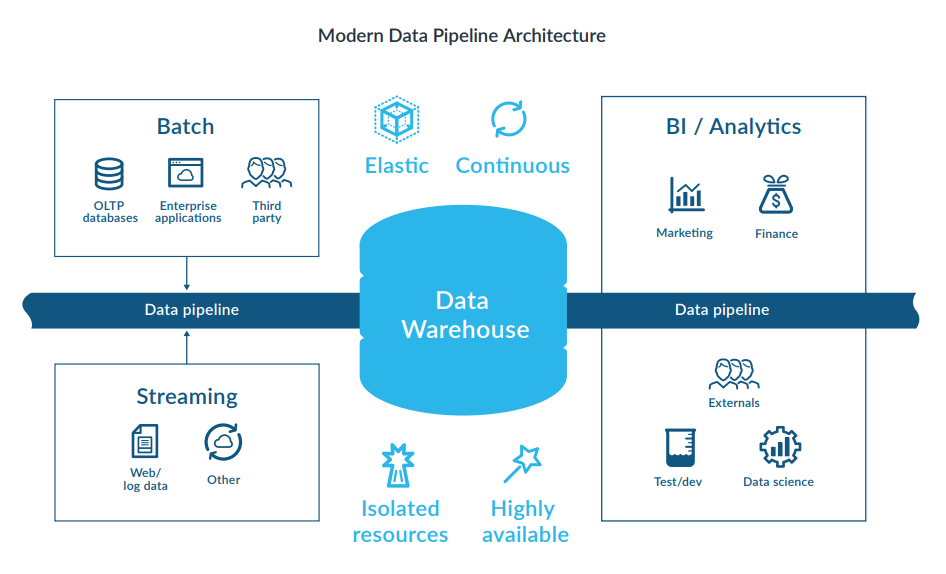
A third example of a data pipeline is the Lambda Architecture, which combines batch and streaming pipelines into one architecture. The Lambda Architecture is popular in big data environments because it enables developers to account for both real-time streaming use cases and historical batch analysis. One key aspect of this architecture is that it encourages storing data in raw format so that you can continually run new data pipelines to correct any code errors in prior pipelines, or to create new data destinations that enable new types of queries.

Picture -The Lambda Architecture accounts for both a traditional batch data pipeline and a real-time data streaming pipeline. It also has a serving layer that responds to queries.



Traditional Vs. Modern Pipeline

|  |  |
| --- | --- |
| **Traditional data pipeline** | **Modern data pipeline** |
| Traditional data pipelines are rigid, brittle, and difficult to change, and they do not support the constantly evolving data needs of today’s organizations.  They present many challenges, by:  • Taking significant time and cost to design and build  • Comprising multiple tools that are not compatible and require unnecessary integration  • Requiring that only qualified IT professionals, who have skills in short supply, build data pipelines, thereby creating work bottlenecks  • Introducing avoidable latency, causing delayed data extraction or transport through the pipeline  • Ignoring the demands of streaming data, handling batch-only data loading  • Being rigid, making it difficult to change and manage over time  • Relying on schema-dependent data loading processes | Modern data pipelines make it faster and easier to extract information from the data you collect.  They start with extracting raw data from a number of different sources. The data is then collected and transported to a common place, typically a data repository in the cloud.  From there, the data undergoes various transformations until it’s usable for analytics and produces business value.  The data is then loaded into a data warehouse, where it is easily managed and accessed by data science workloads, automated actions, and other such computing jobs.  To address these issues, the best data pipelines have these five characteristics:  • Continuous and extensible data processing  • The elasticity and agility of the cloud  • Isolated and independent resources for data processing  • Democratized data access and self-service management  • High availability and disaster recovery. These characteristics enable organizations to leverage their data quickly, accurately, and efficiently to make quicker and better business decisions. |



Principles

Capture the Data from the Right Person at the Right Place at the Right Time -

Data quality can be improved as the actors providing the data (the right person) access it at its origin (the right place) and so are able to provide reliable and detailed data elements to the Pipeline Data Exchange Structure (PDES).

Capture Once and Use Many Times in the Supply Chain -

A second important principle is that not only is pipeline data collected early on in the movement of goods along the supply chain, but once a data element is included in the pipeline data exchange structure, it does not need to be re-entered. This reflects the single window principle of sharing electronic data.

Data re-entry is reserved exclusively for data elements that have changed or need correcting. The pipeline data exchange structure makes it possible for the data elements to be used by multiple cross-border agencies, avoiding the need to resubmit data for each agency.

For certain types of data such as parties (buyer, seller) the use of external trusted data source(s) could be included to provide certainty.

Capturing Information at Source-

Data captured from the source of the information provides more certainty that the information provided is correct at that point in the process.

The consignor is a key actor to start the data capture in the transport process. The consignor knows more about the goods than any other actor in the supply chain at that point in time and in most cases, they are the last actor to physically touch the goods before being sealed for transport.

This data used for customs declarations (export or import) will be more faithful to reality as declarants can provide information coming from the source.

Participating businesses which are prepared to voluntarily provide advanced data can and will also provide input data back to the pipeline.

For example, ‘goods receipt’ at their warehouse and confirmation of package totals received. It is beneficial for them to do this as the data could be fed back through output waypoints to the customs authorities to correct or amend the import customs declaration from the simplified declaration made before the container is opened or pallet unpacked.

Output Waypoints: Extracting Information from the Pipeline -

Data contained in the pipeline can be extracted by the authorized actors by using two methods, push or pull.

• Push Methodology - Some regulatory agencies desire data to be pushed to their data repositories. They then store this data and use it to assess risk and provide historical retrospective in their own internal systems.

• Pull Methodology - Some regulatory agencies desire a pull process where and when they can request data using a secure access token or similar. They plan to evaluate risk internally in their own systems. If a shipment is selected for inspection they can request the data and its history to re-evaluate the risk and make a decision without the need for a physical inspection or manual intervention to provide documentation.

Both methods have their advantages and drawbacks; however, commonly both require the use of a structured, standardized data model. This reduces the development effort and the data analytics energy required to make sense of the data and provide certainty of the data fields of interest to the local regulatory agencies.

This push and pull methodology can be adapted for output waypoints between commercial actors as well. In these cases, the relevant privacy and data ownership aspects would need to be addressed bilaterally or multilaterally. These methods could also be used for data exchange between data pipeline providers

Building a Pipeline Carrier - Basic Principle

It is important to identify the actors who will be providing the input waypoint data ahead of time and enable them to contribute to the PDES.

Use of Standards -

The standard is important on several levels, including:

i. Firstly, the data element level to ensure that the semantics are understood to be the same across all participants.

ii. It is recommended to codify information when possible in order to avoid Misunderstandings due to language or spelling differences; such code lists and data types must also be standardized in order to be mutually understood.

iii. The message structure should also be standardized as information is very rarely flat, but it is often hierarchal bringing information into relation with other information (for = example, a seal should be associated to the container on which it is affixed; or the postal code should be associated to the party to which it is related).

iv. The standardization of the exchange protocol can also facilitate the exchange of information, though this can be less critical than the three other levels of standards.

Input Waypoints: Adding Data into the Carrier

Information can be integrated into the PDES as a submission of a full document. However, in This case, information may be redundant as documents often repeat information that was established earlier in the data exchange. In this case, redundant information should not overwrite previously submitted information unless it could be considered the data source

Change of Information -

Data captured through the pipeline should in almost all cases be actual data that is true at that point in time. The amending of data should not as a rule take place; however, some waypoints can capture and provide the ‘planned’ route or ‘estimated’ arrival dates, which by nature are variable. At each input waypoint, the latest advised estimates for these events or factors could be provided.

Resources –

<https://hazelcast.com/glossary/data-pipeline/>

<https://unece.org/fileadmin/DAM/cefact/GuidanceMaterials/WhitePaperDataPipeline_Eng.pdf>

<https://www.snowflake.com/wp-content/uploads/2020/02/5-characteristics-of-a-modern-data-pipeline.pdf>