Published in Towards Data Science

You have **1** free member-only story left this month.
Sign up for Medium and get an extra one

Mike Shakhomirov Follow

Jan 2 · 9 min read · Member-only · Listen

Data pipeline design patterns

Choosing the right architecture with examples

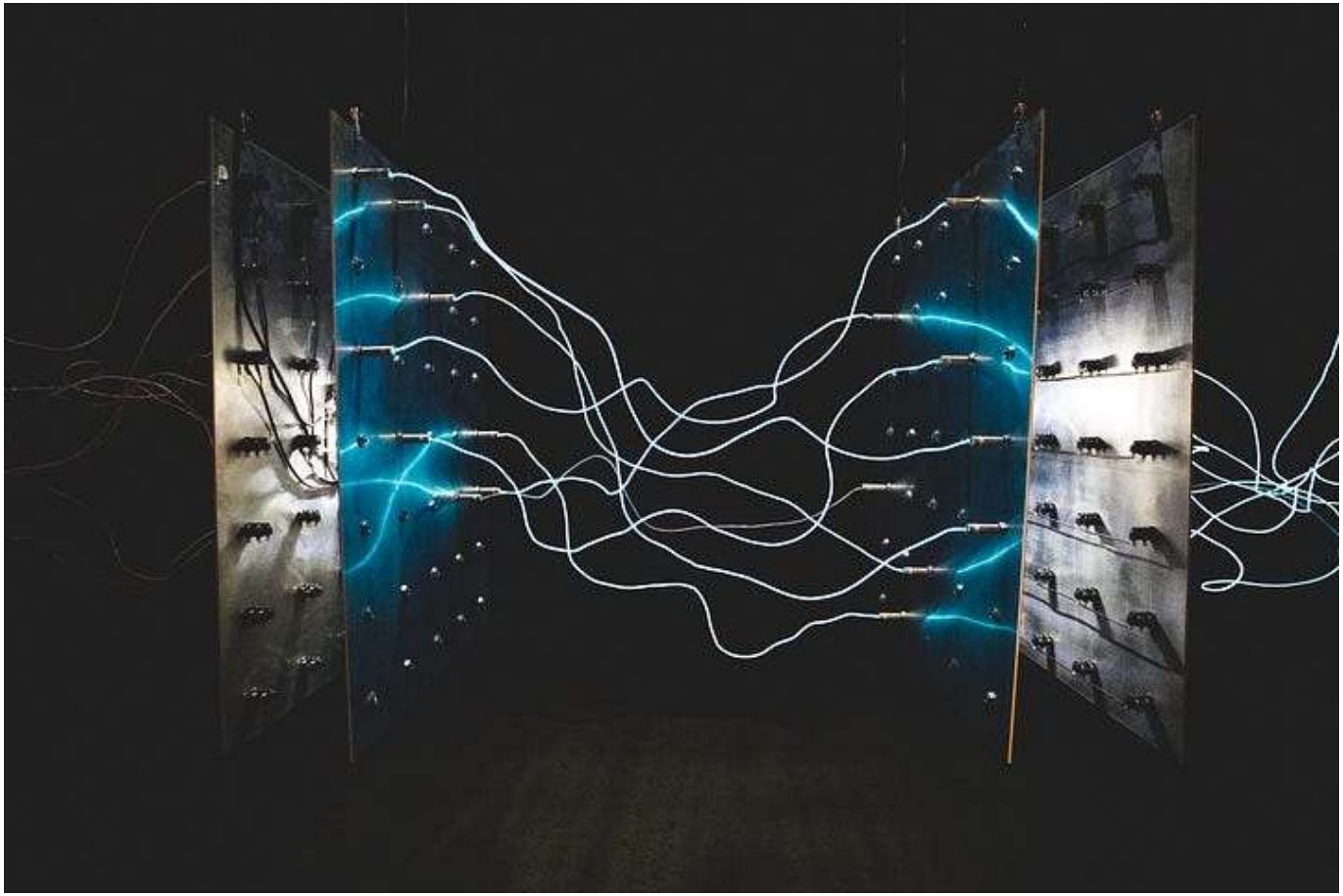


Photo by israel palacio on Unsplash

Typically data is processed, extracted, and transformed in steps. Therefore, a sequence of data processing stages can be referred to as a **data pipeline**.

Which design pattern to choose?

There are lots of things to consider, i.e. **Which data stack to use? What tools to consider? How to design a data pipeline conceptually? ETL or ELT? Maybe ETLT? What is Change Data Capture?**

I will try to cover these questions here.

A data pipeline

So it is a sequence of data processing steps. Due to ***logical data flow connections*** between these stages, each stage generates an **output** that serves as an **input** for the following stage.

> There is a data pipeline whenever there is data processing between points A and B.

A data pipeline's three major parts are a **source, a processing step or steps, and a destination.** Data extracted from an external API (a source) can then be loaded into the data warehouse (destination). This is an example of a most common data pipeline where the source and destination are different. However, it is not always the case, as *destination-to-destination* pipelines also exist.

For example, data can originate as a `reference` table in the data warehouse in the first place and then after some data transformation, it can *land* in a new schema, for example, in `analytics` to be used in reporting solutions.

> There is always a data pipeline when data is processed between source and destination.

Data pipeline. Image by author

Event data created by just one `source` at the back-end, an event stream built with Kinesis Firehose or Kafka stream, can feed a number of various `destinations`.

Consider ***user engagement*** **data** from *Google Analytics* as it flows as an event stream that can be used in both analytics dashboards for user activity and in the Machine learning (ML) pipeline for churn prediction.

Despite using the same data source, both pipelines operate independently and must successfully complete before the user can see the results.

Alternatively, data from two or more `source` locations can be aggregated into just one `destination`. For example, data from different payment merchant providers can be transformed into a revenue report for the Business Intelligence (BI) dashboard.

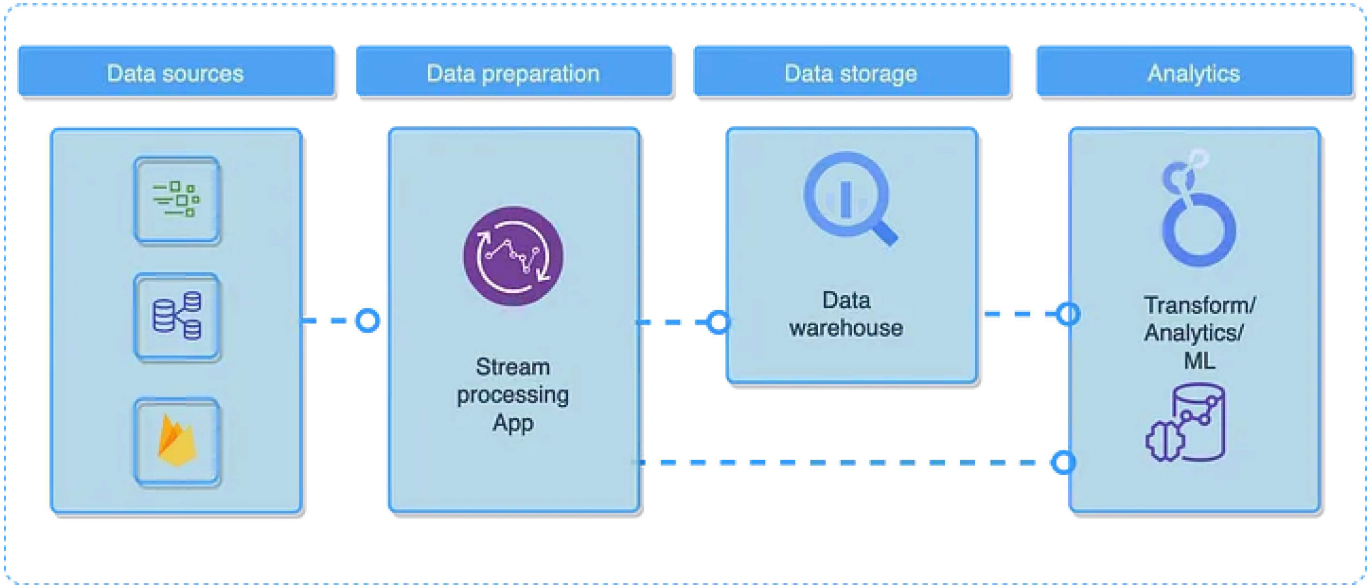
> Data quality checks, data cleansing, transformation, enrichment, filtering, grouping, aggregation, and the application of algorithms to the data are frequent steps in data pipelines.

Architecture types and examples

Data pipeline architecture as a term might mean several things depending on the situation. In general, it can be split into **conceptual** (logical) and **platform** levels or architecture types.

The **conceptually** logical part describes how a dataset is processed and transformed from collection to serving, whereas **platform architecture** focuses on an individual set of tools and frameworks used in a given scenario, as well as the functions that each of them plays.

This is a **logical structure** of a data warehouse pipeline:



Conceptual data pipeline design. Image by author

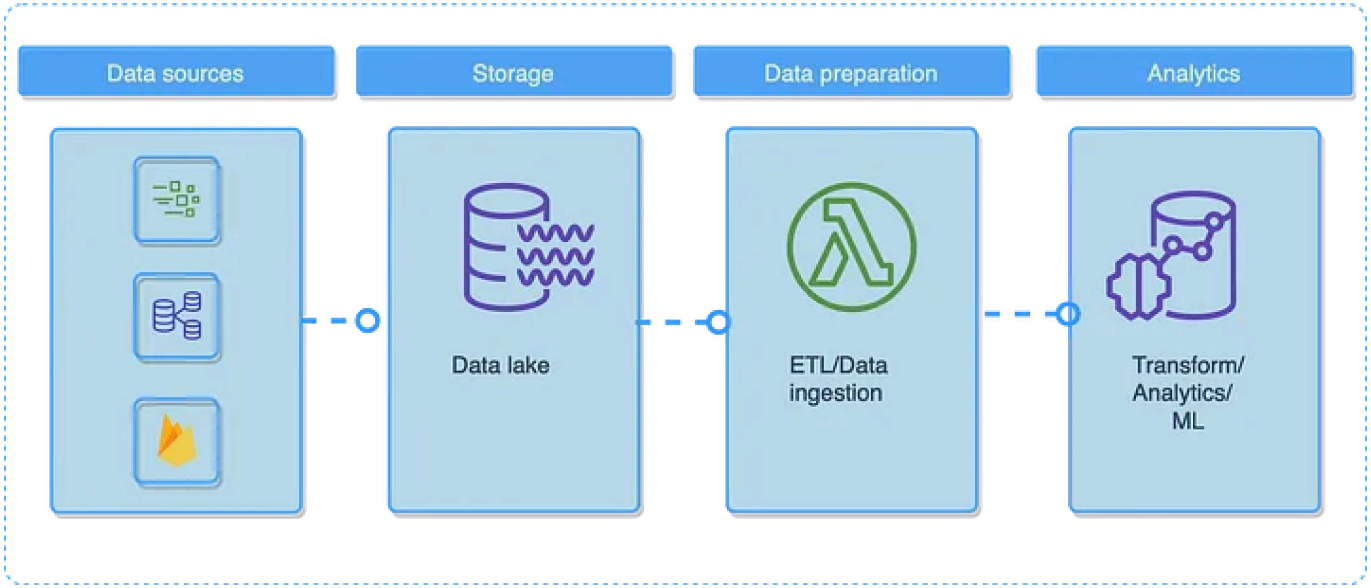
In this article I found a way to extract a real-time data from Firebase/Google Analytics 4 and load it in BigQuery:

How to extract real-time intraday data from Google Analytics 4 and Firebase in BigQuery

And always have an up-to-date data for your custom reports

towardsdatascience.com

And this is a conceptual data lake pipeline example:



Conceptual data pipeline design. Image by author

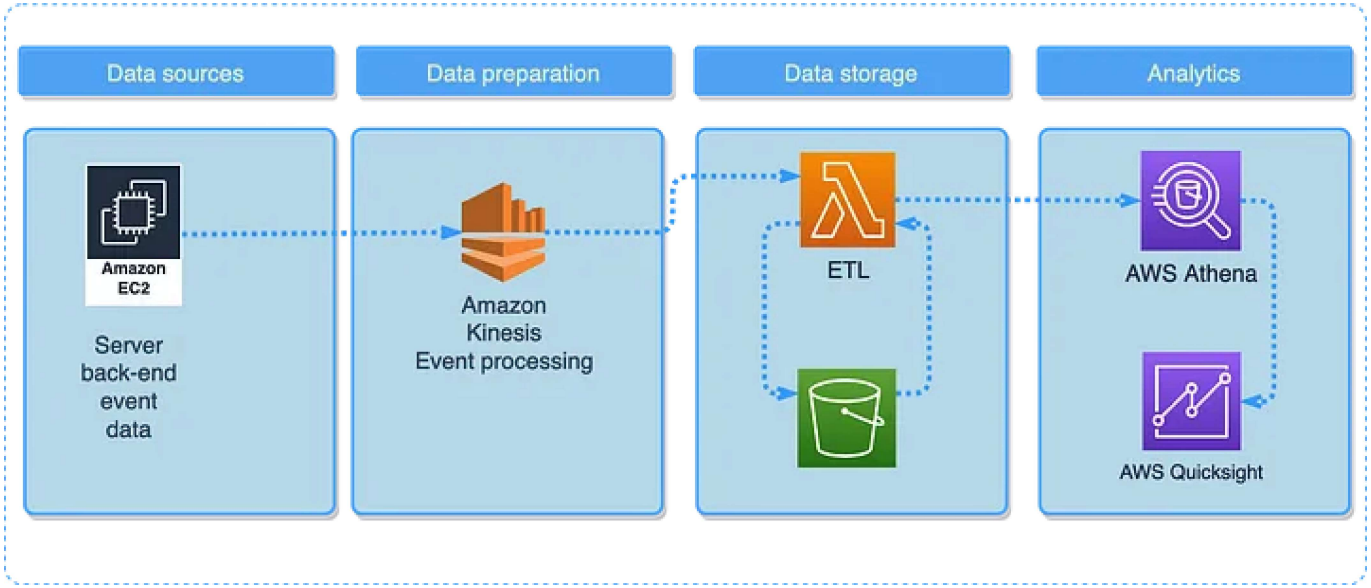
For example, in this post I previously wrote how to extract data MySQL databases, save it in Cloud Storage so later it could be used for analysis later:

MySQL data connector for your data warehouse solution

How to build one and export millions of rows in chunks, stream, capture real-time data changes or extract data and save...

towardsdatascience.com

This is a platform level architecture example:



Platform level data pipeline design. Image by author

This is a very common pattern for many lake house architecture solutions. In this blog post I created a bespoke data ingestion manager that is triggered by new object events when they are created in Cloud Storage:

How to Handle Data Loading in BigQuery with Serverless Ingest Manager and Node.js

File formats, yaml pipe definitions, and transform and event triggers for your simple and reliable data ingestion…

towardsdatascience.com

Streaming

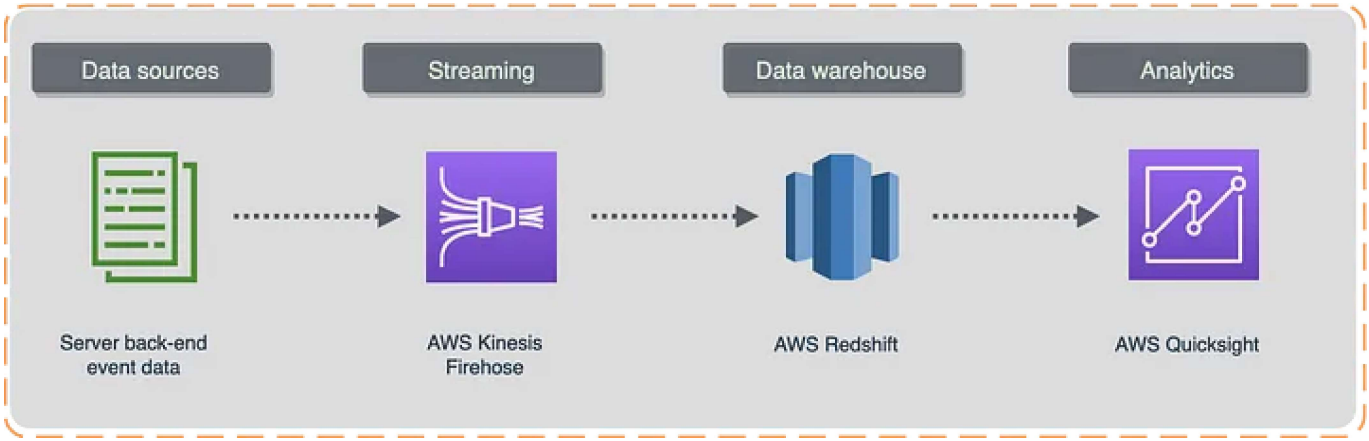
Applications can trigger immediate responses to new data events thanks to stream processing.

> Streaming is a “must-have” solution for enterprise data.

Sign up Sign In

Search Medium Write

Common use cases are **anomaly detection and fraud prevention**, **real-time personalisation and marketing** and **internet of things.** Data and events are often produced by a “publisher” or “source” and transferred to a “stream processing application,” where the data is processed immediately before being sent to a “subscriber.” Very often, as a `source`, you can meet streaming applications built with **Hadoop, Apache Kafka, Amazon Kinesis,** etc. The "**Publisher/subscriber**" pattern is often referred to as `pub/sub`.



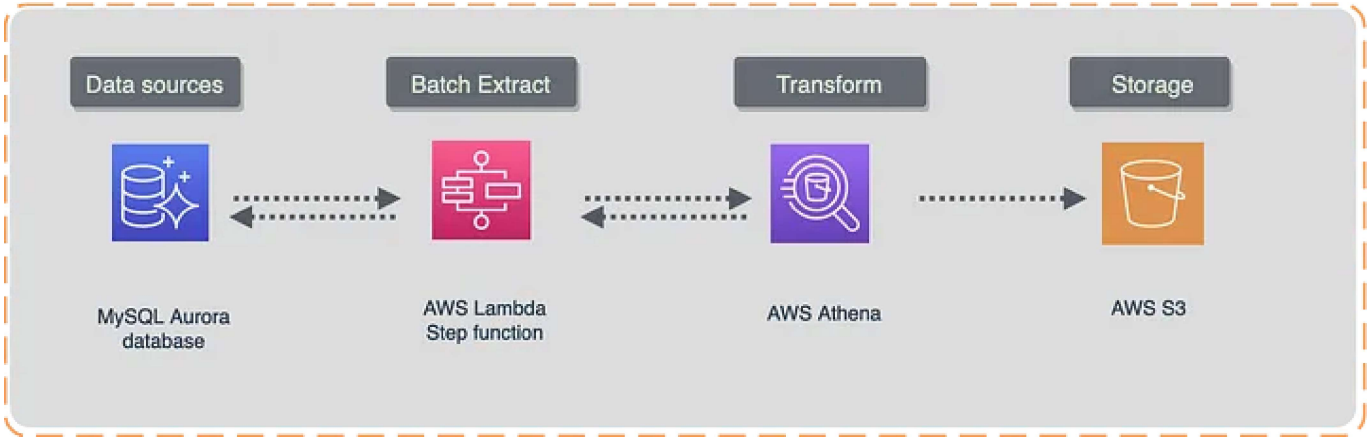
In this example we can set up an **ELT streaming** data pipeline to **AWS Redshift.** AWS Firehose delivery stream can offer this type of seamless integration when streaming da 170 2 led directly into the data warehouse table. Then data will be transformed to create reports with **AWS Quicksight** as a BI tool.

Batch processing

Batch processing is a model where data is gathered according to a predetermined threshold or frequency in both micro-batch processing and conventional batch processing, and then processing takes place. Historically workloads were primarily batch-oriented in data environments. However, modern applications continuously produce enormous amounts of data, and a business leans to micro-batch and streaming processing, where data is

being processed immediately to maintain a competitive advantage. Technologies for **micro-batch** loading include **Apache Spark Streaming, Fluentd, and Logstash**, and it is very similar to **conventional batch processing**, where events are processed on a schedule or in small groups.

> It's a good choice where the accuracy of your data is not relevant at this moment precisely.

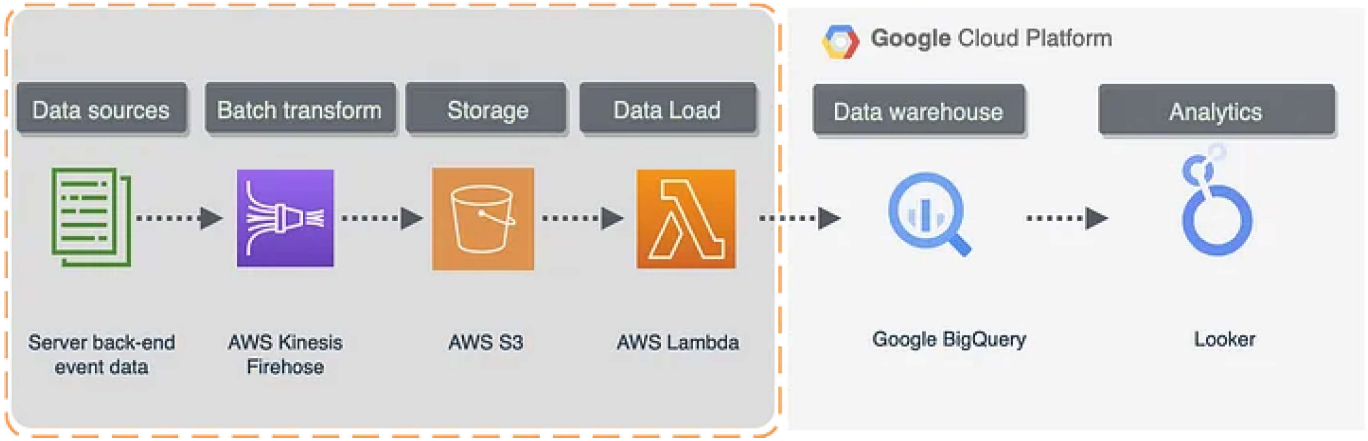


This data pipeline design pattern works better with smaller datasets that require ongoing processing because Athena charges based on the volume of data scanned.

> Let's say, you don't want to use `change log` on MySQL database instance. That would be an ideal case because payments dataset is not huge.

Lambda/Kappa architecture

This architecture combines batch and streaming methodologies. It combines the best of both worlds and advises that raw data must be retained, for example, in a data lake in case you would want to use it again to build a new pipeline or investigate an outage. It has both **batch** and **streaming** (speed) layers which helps to respond instantly to shifting business and market conditions. Lambda architectures can sometimes very complicated with multiple code repositories to maintain.



Lambda data pipeline design. Image by author

Transform first then Load?

ETL is considered to be a conventional approach and the most widely used historically. With the rise of data warehousing **ELT** becomes increasingly popular.

> Indeed, why do we need to transform first if we can centralise it for all data pipelines in the data warehouse?

Vurtualisation is another popular approach for data warehouses where we create **views** on data instead of meterialised tables. New requirements to business agility put cost-effectiveness on the send plan and data users qould query **views** instead of **tables**.

Change Data capture is another approach to update the data exactly when the changes occur. When used typically latent data pipelines, CDC technology can recognize data changes as they happen and offer information about those changes. Changes are usually pushed to message queue or provided as a stream.

How to choose the data pipeline architecture?

In recent years, data architecture components such as data pipelines have developed to support massive volumes of data. The term “Big Data” can be described as having three traits known as volume, variety and velocity. Big data can open up new opportunities in a variety of use cases, including predictive analytics, real-time reporting, and alerting. Architects and developers have had to adapt to new “big data” requirements because of the substantially increased volume, diversity, and velocity of data. New data processing frameworks emerged and kept emerging. Due to the high velocity of modern data streams, we might want to use *streaming* data pipelines. Data may then be collected and analysed in real-time, allowing for immediate action.

> However, streaming data pipeline design pattern is not always the most cost-effective.

For example, in the majority of data warehouse solutions batch data ingestion is free. However, streaming, might go with a price. Same statement would be relevant for data processing. **Steaming** is the most expensive way to process the data in the majority of cases.

Big Data `volumes` require the data pipeline to be able to process events concurrently as very often they are sent simultaneously at once. Data solutions must be scalable.

The `variety` implies that data might come through the pipelines in different formats, very often unstructured or semi-structured.

Architecture type depends on various factors, i.e. `destination` type and where data has to be in the end, **cost considerations** and your team's **development stack** and certain data processing skills you and your colleagues already have.

> Do your data pipelines have to be managed and cloud-based, or would you rather want to deploy it on-premises?

In reality, there are numerous combinations of variables that can help to select the best data platform architecture. **What would be the velocity or data flow rate in that pipeline? Do you require real-time analytics, or is near-real-time sufficient?** This would resolve the question of whether or not you require "streaming" pipelines.

For example, there are services that can create and run both **streaming** and **batch** data pipelines, i.e. Google Dataflow. So how is it different from any other pipeline built in the data warehouse solution? The choice would depend on existing infrastructure. For example, if you have some existing **Hadoop** workloads, then GCP DataFlow would be a wrong choice as it will not let you to re-use the code (it is using Apachec Beam). In this case you would want to use **GCP Dataproc** which works on **Hadoop/Spark** code.

> The rule of thumb is that if the processing is dependent on any tools in the Hadoop ecosystem, Dataproc should be utilized.

It is basically a Hadoop extension service.

On the other hand, if you are not limited by existing code and would like to reliably process ever-increasing amounts of **streaming** data then **Dataflow** is the recommended choice. You can check these Dataflow templates if you like to do things in **JAVA**.

There is a system design guide from Google

Conclusion

BigData posed new challenging data architecture requirements for data developers. A constantly increasing variety of data formats and data sources increased the importance of data integration without disrupting production applications. Businesses are increasingly aiming to automate data integration procedures, process streaming data in real-time, and streamline the lifetime of data lakes and warehouses. This becomes a challenging task indeed, taking into account the increased data volume, velocity and variety of data formats over the last decade. Now data pipeline design must be robust and, at the same time, flexible to enable new data pipeline creation in a simplified and automated way. The increasing trend of using `data mesh` / `data mart` platforms requires data catalogues to be created. To create controlled, enterprise-ready data and give data consumers an easy method to find, examine, and self-provision data, this process should ideally be automated too. Therefore, choosing the right data pipeline architecture can solve these issues effectively.

Originally published at https://mydataschool.com.

Recommended read:

Work with Data Pipelines | Cloud Dataflow | Google Cloud

You can report Dataflow Data Pipelines issues and request new features at Note: google-data-pipelines-feedback." You...

cloud.google.com

Cloud Architecture Guidance and Topologies | Cloud Architecture Center | Google Cloud

Discover reference architectures, guidance, and best practices for building or migrating your workloads on Google...

cloud.google.com

GitHub - GoogleCloudPlatform/DataflowTemplates: Google-provided Cloud Dataflow template pipelines...

These Dataflow templates are an effort to solve simple, but large, in-Cloud data tasks, including data...

github.com

Tutorials

Tutorials - AWS Data Pipeline The following tutorials walk you step-by-step through the process of creating and using...

docs.aws.amazon.com

AWS Data Pipeline documentation [https://docs.aws.amazon.com/data-pipeline/]

Data Engineering Data Science Big Data Business Intelligence Data Pipeline

Enjoy the read? Reward the writer.[Beta]

Your tip will go to Mike Shakhomirov through a third-party platform of their choice, letting them know you appreciate their story.

Give a tip

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.

Get this newsletter

About Help Terms Privacy