Data Pipeline Essentials

Strategies for Successful Deployment and Collecting Analytical Insights

CONTENTS

SUDIP SENGUPTA
TECHNICAL WRITER AT JAVELYNN

Modern data-driven applications are based on various data sources and complex data stacks that require well-designed frameworks to deliver operational efficiency and business insights. The result is a flexible, dynamic, and scalable application that enables businesses to predict, influence, and optimize their business outcomes based on real-time recommendations. Based on the numerous benefits that data-driven applications offer to businesses, Gartner predicts that the role of data in achieving agility and collaboration is expected to grow further over the next five years.

In this Refcard, we delve into the fundamentals of a data pipeline and the problems it solves for modern enterprises, along with its benefits and challenges.

WHAT IS A DATA PIPELINE?

A data pipeline comprises a collection of tools and processes for efficient transfer, storage, and processing of data across multiple systems. With data pipelines, organizations can automate information extraction from distributed sources while consolidating data into high-performance storage for centralized access. A data pipeline essentially forms the foundation to build and manage analytical tools for critical insights and strategic business decisions. By building reliable pipelines for the consolidation and management of data flows, development and DataOps teams can also efficiently train, analyze, and deploy machine learning models.

DATA PIPELINE TYPES

Data pipelines are broadly categorized into the following types:

SEE NEXT COLUMN

BATCH PIPELINES

In batch pipelines, data sets are collected over time in batches and then fed into storage clusters for future use. These pipelines are mostly considered applicable for legacy systems that cannot deliver data in streams, or in use cases that deal with colossal amounts of data. Batch pipelines are usually deployed when there's no need for real-time analytics and are popular for use cases such as billing, payroll processing, and customer order management.

STREAMING PIPELINES

In contrast to batch pipelines, streaming data pipelines continuously ingests data, processing it as soon as it reaches the storage layer. Such pipelines rely on highly efficient frameworks that support the ingestion and processing of a continuous stream of data within a sub-second time frame. As a result, stream data pipelines are mostly

suitable for operations that require quicker analysis and real-time insights of smaller data sets. Typical use cases include social media engagement analysis, log monitoring, traffic management, user experience analysis, and real-time fraud detection.

DATA PIPELINE PROCESSES

Though the underlying framework of data pipelines differ based on use cases, they mostly rely on a number of common processes and elements for efficient data flow. Some key processes of data pipelines include:

SOURCES

In a data pipeline, a source acts as the first point of the framework that feeds information into the pipeline. These include NoSQL databases, application APIs, cloud sources, Apache Hadoop, relational databases, and many more.

JOINS

A join represents an operation that enables the establishment of a connection between disparate data sets by combining tables. While doing so, join specifies the criteria and logic for combining data from different sources into a single pipeline.

Joins in data processing are categorized as:

- INNER Join — Retrieves records whose values match in both tables
- LEFT (OUTER) Join — Retrieves all records from the left table plus matching values from the right table
- RIGHT (OUTER) Join — Retrieves all records from the right table plus matching records from the left table
- FULL (OUTER) Join — Retrieves all records, whether there is a match or not in any of the two tables. In SQL tables with star schema, full joins are typically implemented through conformed dimensions to link fact tables, creating fact-to-fact joins.

EXTRACTION

Source data remains in a raw format that requires processing for further analysis. Extraction is the first step of data ingestion where the data is crawled and analyzed to ensure information relevancy before it is passed to the storage layer for transformation.

STANDARDIZATION

Once data has been extracted, it is converted into a uniform format that enables efficient analysis, research, and utilization. Standardization is the process of formulating data with disparate variables on the same scale to enable easier comparison and trend analysis. Data standardization is commonly used for attributes such as dates, units of measure, color, size, etc.

CORRECTION

This process involves cleansing the data to eliminate errors and pattern anomalies. When performing correction, data engineers typically use rules to identify a violation of data expectation, then modify it to meet the organization's needs. Unaccepted values can then be ignored, reported, or cleansed according to pre-defined business or technical rules.

LOADS

Once data has been extracted, standardized, and cleansed, it is loaded into the destination system, such as a data warehouse or relational database, for storage or analysis.

AUTOMATION

Data pipelines often involve multiple iterations of administrative and executive tasks. Automation involves monitoring the workflows to help identify patterns for scheduling tasks and executing them with minimal human intervention. Comprehensive automation of a data pipeline also involves the detection of errors and notification mechanisms to maintain consistent data sanity.

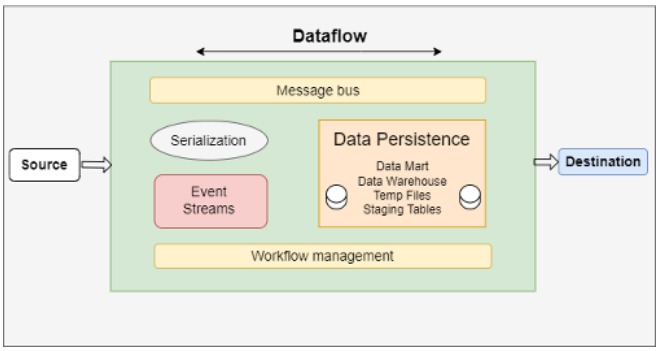
DEPLOYING A DATA PIPELINE

Considered one of the most crucial components of modern data-driven applications, a data pipeline automates the extraction, correlation, and analysis of data for seamless decision-making. When building a data pipeline that is production-ready, consistent, and reproducible, there are plenty of factors to consider that make it a highly technical affair. This section explores the key considerations, components, and options available when building a data pipeline in production.

COMPONENTS OF A DATA PIPELINE

The data pipeline relies on a combination of tools and methodologies to enable efficient extraction and transformation of data. These include:

Common components of a data pipeline



EVENT FRAMEWORKS

Event processing encompasses the analysis and decision-making based on data streamed continuously from applications. These

systems extract information from data points that respond to tasks performed by users or various application services. Any identifiable task or process that causes a change in the system is marked as an event, which is recorded in an event log for processing and analysis.

MESSAGE BUS

A message bus is a combination of a messaging infrastructure and data model that both receives and queues data sent between different systems. Leveraging an asynchronous messaging mechanism, applications use a message bus to instantaneously exchange data between systems without having to wait for an acknowledgement. A well-architected message bus also allows disparate systems to communicate using their own protocols without worrying about system inaccessibility, errors, or conflicts.

DATA PERSISTENCE

Persisting data refers to the ability of an application to store and retrieve information, so it can be processed in batches. Data persistence can be achieved in several ways, such as by storing it on the block, object, or file storage devices that ensure data is durable and resilient in the event of system failure.

Data persistence also includes backup drives that provide readily available replicas for automatic recovery when a server crashes. The data persistence layer creates the foundation for unifying multiple data sources and destinations into a single pipeline.

WORKFLOW MANAGEMENT

In data pipelines, a workflow comprises a set of tasks with directional dependencies. These tasks filter, transform, and move data across systems, often triggering events. Workflow management tools, such as Apache Airflow, structure these tasks within the pipeline, making it easier to automate, supervise, and manage tasks.

SERIALIZATION FRAMEWORKS

Serialization tools convert data objects into byte streams that can easily be stored, processed, or transmitted. Most firms operate with multiple data pipelines built using different approaches and technologies. Data serialization frameworks define storage formats that make it easy to identify and access relevant data, then write it to another location.

KEY CONSIDERATIONS

Key factors to consider when building and deploying a modern data pipeline include:

SELF-MANAGED VS. UNIFIED DATA ORCHESTRATION PLATFORM

Organizations can choose whether to leverage third-party enterprise services or self-managed orchestration frameworks for deploying data pipelines. The traditional approach of doing so is to build in-house data pipelines that require provisioning infrastructure in a self-managed, private data center setup. This offers various benefits, including flexible customization and complete control over the handling of data.

However, self-managed orchestration frameworks rely on a number of various tools and niche skills. Such platforms are also considered less flexible to handle pipelines that require constant scaling or high availability. On the other hand, unified data orchestration platforms are supported by the right tools and skills that offer higher computing power and replication that enables organizations to scale quickly while maintaining minimum latency.

ONLINE TRANSACTION PROCESSING (OLTP) VS ONLINE ANALYTICAL PROCESSING (OLAP)

OLTP and OLAP are the two primary data processing mechanisms. An **OLTP** system captures, stores, and processes user transactions in real-time where every transaction is made up of individual records consisting of multiple fields and columns.

An **OLAP** system relies on large amounts of historical data to perform high-speed, multidimensional analysis. This data typically comes from a combination of sources, including OLTP databases, data marts, data warehouses, or any other centralized data store. OLAP systems are considered ideal for business intelligence, data mining, and other use cases that require complicated analytical calculations.

QUERY OPTIONS

These are a set of query string parameters that help to fine-tune the order and amount of data a service will return for objects identified by the Uniform Resource Identifier (URI). These options essentially define a set of data transformations that are to be applied before returning the result.

These options can be applied to any task except the `DELETE` operation.

Some commonly used query options include:

- `Filter`: Enables the client to exclude a collection of resources addressed by the URI
- `Expand`: Specifies a list of resources related to the data stream that will be included in the response
- `Select`: Allows the client to request a specific set of properties for each resource type
- `OrderBy`: Sorts resources in a specified order

DATA PROCESSING OPTIONS

There are two primary approaches to cleaning, enriching, and transforming data before integration into the pipeline.

In **ETL** (Extract - Transform - Load), data is first transformed in the staging server before it is loaded to the destination storage or data warehouse. ETL is easier to implement and is suited for on-premises data pipelines running mostly structured, relational data.

On the other hand, in **ELT** (Extract - Load - Transform), data is loaded directly into the destination system before processing or transformation. When compared to ETL, ELT is more flexible and scalable, making it suitable for both structured and unstructured cloud workloads.

CHALLENGES OF IMPLEMENTING A DATA PIPELINE

A data pipeline includes a series of steps that are executed sequentially on each dataset in order to generate a final output. The entire process usually involves complex stages of extraction, processing, storage, and analysis. As a result, each stage as well as the entire framework requires diligent management and adoption of best practices. Some common challenges while implementing a data pipeline include:

COMPLEXITY IN SECURING SENSITIVE DATA

Organizations host petabytes of data for multiple users with different data requirements. Each of these users has different access permissions for different services, requiring restrictions on how data can be accessed, shared, or modified. Assigning access rights to every individual manually is often a herculean task, which if not done right, may lead to the access of sensitive information to malicious individuals.

SLOWER PIPELINES DUE TO MULTIPLE JOINS AND STAR SCHEMA

Joins allow data teams to combine data from two separate tables and extract insights. Given the number of sources, modern data pipelines use multiple joins for end-to-end orchestration. These joins consume computing resources, thereby slowing down data operations. Besides this, large data warehouses rely on star schemas to join DIMENSION tables to FACT tables. On account of its highly denormalized state, star schemas are considered less flexible to enforce the data integrity of dynamic data models.

NUMEROUS SOURCES AND ORIGINS

The dynamic nature of data-driven applications requires constant evolution and are often ingesting data from a growing number of sources. Managing these sources and the processes they run is often challenging as these expose data with different formats. A large number of sources also makes it difficult to document the data pipeline's configuration details, which hampers cross-domain collaboration in software teams.

GROWING TALENT GAP

With the growth of emerging disciplines such as data science and deep learning, companies require more personnel resources and expertise than job markets can offer. Combined with this is the fact that a typical data pipeline implementation requires a huge learning curve, thereby requiring organizations to dedicate resources to either upskill existing staff or hire skilled experts.

SLOW DEVELOPMENT OF RUNNABLE DATA TRANSFORMATIONS

With modern data pipelines, organizations are able to build functional data models based on the recorded data definitions. However, developing functional transformations from these models comes with its own challenges as the process is expensive, slow, and error-prone. Developers are often required to manually create executable codes and runtimes for data models, thereby resulting in ad-hoc, unstable transformations.

ADVANCED STRATEGIES FOR MODERN DATA PIPELINES

Some best practices to implement useful data pipelines include:

GRADUAL BUILD USING MINIMUM VIABLE PRODUCT PRINCIPLES

When developing a lean data pipeline, it is important to implement an architecture that scales to meet growing needs while still being easy to manage. As a recommended best practice, organizations should apply a modular approach while incrementally developing functionalities to handle more advanced data processing needs.

INCORPORATE AI CAPABILITIES FOR TASK AUTOMATION

DataOps teams should leverage auto-provisioning, auto-scaling, and auto-tuning to reduce design time and simplify routing. Autoscaling is crucial since big data workloads have data intake requirements that vary dramatically within short durations.

***Example**: The following snippet outlines a sample automation script for the ingestion of logs in a Python-based data pipeline.*

```
=
f_a = open(LOG_FILE_A, 'r')
f_b = open(LOG_FILE_B, 'r')
while True:
    where_a = f_a.tell()
    line_a = f_a.readline()
    where_b = f_b.tell()
    line_b = f_b.readline()
    if not line_a and not line_b:
        time.sleep(1)
```

CODE CONTINUES ON NEXT PAGE

```
        f_a.seek(where_a)
        f_b.seek(where_b)
        continue
    else:
        if line_a:
            line = line_a
        else:
            line = line_b
```

The above script opens log files and reads them one line at a time. Each line is parsed into fields, following which the lines and fields are both parsed to the database. The script also ensures that the database does not accept duplicate lines.

PARAMETERIZE THE DATA PIPELINE

Parameters help data teams to make predictions using data models and estimate the effectiveness of the models in analytics. By referring objects defined within a function and using them as parameters to pass external values, parameterization ensures clean code in data pipelines while maintaining a common standard.

USE NO-CODE/LOW-CODE ETL TO SIMPLIFY DATA OPERATIONS

As a recommended best practice, organizations should leverage low-code or no-code ETL platforms that learn how to transform data from previous data sets based on available boilerplates and formulas. Such platforms typically include built-in actions that eliminate manual coding work, enabling rapid setup of data integration and transformation.

MINIMIZE DEPENDENCIES BY USING ATOMIC WORKFLOWS

Data pipelines undertake multiple data transformations, such as enrichment and format validation. As a best practice, it is recommended that these transformations are broken down into smaller, reproducible tasks with deterministic outputs. This makes it easy to test changes while ensuring data quality and reliability.

IMPLEMENT DATA PIPELINE VERSIONING

Versioning pipelines enables data teams to determine how and when data was ingested, transformed, or modified. It is important for data engineers and operators to know which version was used to create a specific dataset so that incident root causes and action items can be reproduced easily. Versioning also enables rollback while making it easy to evaluate whether new changes to the pipeline are effective.

ENABLE MONITORING AND ALERTS FOR ALL TRANSACTIONS

For proactive security and data consistency, it is crucial to implement end-to-end observability of data pipelines. This also enables data teams to validate data introduced into the pipeline, aiding in faster troubleshooting and vulnerability management.

TRACK AND LOG CHANGES TO THE DATA

Logging and tracking changes to data help data teams to find problems within the pipeline as soon as possible. As an extension of the versioning mechanism, logging helps operations and security teams to optimize strategies for root cause analysis and gather insightful metrics of application usage.

CONCLUSION

Modern data-driven applications need more than just data. As such, niche purposes of emerging tech such as data science and AI rely on a complex framework of data sources, storage, and computing power. The benefits of data pipelines are often considered proportional to the weightage of its accumulated data.

By leveraging the power of these data-driven solutions, businesses can take faster business decisions, maximize their bottom line, and satisfy customer requirements. In this Refcard, we delved into the benefits, challenges, and configuration strategies of data pipelines for emerging businesses. As the technology landscape evolves further, it will be interesting to see how businesses churn, store, and process data in the years to come.

WRITTEN BY SUDIP SENGUPTA,
TECHNICAL WRITER AT JAVELYNN

Sudip Sengupta is a TOGAF Certified Solutions Architect with more than 15 years of experience working for global majors such as CSC, Hewlett Packard Enterprise, and DXC Technology. Sudip now works as a full-time tech writer, focusing on Cloud, DevOps, SaaS, and cybersecurity. When not writing or reading, he's likely on the squash court or playing chess.

DZone

DZone, a Devada Media Property, is the resource software developers, engineers, and architects turn to time and again to learn new skills, solve software development problems, and share their expertise. Every day, hundreds of thousands of developers come to DZone to read about the latest technologies, methodologies, and best practices. That makes DZone the ideal place for developer marketers to build product and brand awareness and drive sales. DZone clients include some of the most innovative technology and tech-enabled companies in the world including Red Hat, Cloud Elements, Sensu, and Sauce Labs.

Devada, Inc.
600 Park Offices Drive
Suite 300
Research Triangle Park, NC 27709

888.678.0399 | 919.678.0300