Data Pipeline Supervision: Review and Architectural Proposal

Nawar TOUMI
Amina BOUHAMRA
Soukaina BOUCETTA
Fadwa EL AMRAOUI

November 2025

Contents

Chapter 1

Introduction

1.1 Introduction

Data pipelines have become a fundamental component of modern digital systems. They enable the automatic collection, transformation, validation, and storage of large volumes of data from various sources. These pipelines facilitate feeding analytics platforms, machine learning models, and visualization tools.

However, the increasing complexity of these systems introduces significant risks related to the reliability of data flows. Interruptions, format errors, outliers, or processing delays can directly impact the quality of downstream results. In some cases, these failures may lead to poor decisions or unavailability of critical services.

Given these challenges, active supervision of data pipelines becomes essential. It is no longer just about monitoring system availability, but ensuring deep observability of in-transit data—capable of automatically detecting anomalies and triggering alerts or corrective actions.

Recent studies have explored approaches to strengthen this supervision. For instance, machine learning-based methods have been applied to detect anomalies in complex data flows [4, 3]. Other researchers have proposed more global observability models, relying on the collection of metrics, logs, and traces to assess the health of a pipeline [5]. Finally, some works aim to formalize pipeline design architectures to facilitate end-to-end control [6].

This section aims to review the major contributions in the literature on data pipeline supervision, identifying the approaches used, the tools involved,

and the limitations faced by researchers and practitioners.

Chapter 2

State of the Art

2.1 Introduction

Data pipelines play a fundamental role in distributed data processing systems. They automate data flows from collection to analysis. However, their proper functioning depends on many factors, including data quality, component stability, and the ability to quickly detect malfunctions.

This section explores current approaches to data pipeline supervision, based on a review of recent scientific literature, to identify key monitoring methods, tools used, and the limitations of existing approaches.

2.2 Problem Statement and Research Question

Supervision of data pipelines has become a major challenge due to the growing volume, speed, and variety of data. Even brief interruptions in a pipeline can lead to loss of critical information or distort results. Traditional monitoring tools (logs, system alerts) are no longer sufficient to provide a holistic view of pipeline health and in-transit data quality.

Hence, a central research question arises:

> **How can we design intelligent, modular, and adaptable supervision for data pipelines, enabling efficient anomaly detection while remaining compatible with distributed and heterogeneous environments?**

This question arises in a context where solutions must be both effective and integrable with existing tools such as Apache NiFi, PostgreSQL, or Power BI.

2.3 Literature Review

2.3.1 Supervision in Data Pipelines

In a context where data volumes are constantly increasing, supervision of data pipelines has become essential to ensure reliability and performance. Supervision can be defined as the set of mechanisms put in place to monitor the operational state of the pipeline, detect malfunctions, and enable quick intervention in case of errors.

It includes actions such as validating data format and schema, checking completeness, measuring latency times between stages, and detecting transformation failures. Supervision can occur at several levels: data-level, task-level, or infrastructure-level.

Several studies propose architectures integrating real-time supervision modules. For instance, De Haro-Olmo et al. [1] present an IoT-oriented pipeline including data curation and quality control mechanisms. Similarly, Hernandez et al. [2] design an intelligent platform (ERAIA) that automatically supervises data transmitted between sensors, cloud, and user interfaces.

Other approaches emphasize integrating supervision from the pipeline design phase. For example, Pasupuleti [4] proposes an AI-augmented pipeline including schema verification, performance tracking, and error management modules. This allows automated diagnostics and easier maintenance in production.

In summary, supervision is not limited to passive monitoring—it involves proactive instrumentation of the pipeline to anticipate potential problems and ensure continuous service quality.

2.3.2 Observability and Anomaly Detection

Observability represents a conceptual evolution compared to simple supervision. While supervision mainly consists of monitoring predefined key performance indicators (KPIs) such as latency, throughput, or error rate, observability aims to deeply understand the behavior of a system based on its

internal signals [5]. It relies on the collection and analysis of three complementary types of data: *logs* (event records), *metrics* (numerical system measurements), and *traces* (tracking of calls between components). These data provide a holistic view that allows operators to diagnose the root causes of incidents, even when they are not directly visible via high-level indicators.

In the context of data pipelines, observability helps detect slowdowns, blockages, or inconsistencies in data flows. It becomes essential in distributed or real-time environments where errors may silently propagate between components. Modern platforms such as DataDog, Prometheus, or OpenTelemetry facilitate such data collection, though they require careful configuration and clear understanding of relevant metrics.

Anomaly detection is a direct application of observability. It aims to automatically identify unusual behaviors in data or pipeline operation. Existing methods can be classified into three main categories:

- **Heuristic approaches**, based on manually defined rules (thresholds, business rules);
- **Statistical methods**, which detect significant deviations from normal data distributions (mean, standard deviation, time series);
- **Machine learning techniques**, which learn normal behavior from historical data and detect deviations.

For example, Nasiri et al. [3] propose an intelligent pipeline combining physical laws with a supervised machine learning model to improve anomaly detection in industrial environments with noisy data. Their approach highlights the value of coupling domain knowledge (physics) with automated analysis algorithms.

Therefore, integrating advanced observability layers with intelligent anomaly detection modules is essential to ensure the reliability, traceability, and robustness of modern data pipelines.

2.3.3 Existing Tools and Models (ETL, ELT, etc.)

Data pipelines can be built according to several architectural models, each addressing specific technical and functional constraints. The most common are ETL (Extract-Transform-Load) and ELT (Extract-Load-Transform). The ETL model suits traditional batch processing, where data are extracted,

transformed in an intermediate environment, and then loaded into a target database. Conversely, ELT focuses on rapid loading of raw data into storage systems, followed by deferred transformation, often directly in the data warehouse.

Additionally, streaming-oriented architectures based on technologies such as Apache Kafka, Apache Flink, or Spark Streaming allow real-time data processing with minimal latency. This paradigm is critical for use cases requiring immediate responsiveness, such as anomaly detection or monitoring connected equipment.

Several tools have been developed to implement these models. For instance, Apache NiFi offers a graphical interface for managing data flows, with native support for traceability, conditional routing, and basic monitoring. Apache Airflow focuses on orchestrating complex workflows, mostly in batch mode. StreamSets Data Collector provides a hybrid solution combining batch and real-time processing, with enhanced observability capabilities.

However, these tools do not always fully cover supervision needs. Recent research attempts to formalize best practices through enriched models such as ETLT (Extract-Transform-Load-Transform) and ELTL (Extract-Load-Transform-Load). These design patterns improve readability, modularity, and especially traceability of complex pipeline transformations [6]. The goal is to optimize observability and maintainability while facilitating the integration of third-party components (machine learning, alerting, visualization).

Hence, the choice of pipeline model and associated tools should be guided by requirements in supervision, processing frequency (batch vs real-time), and orchestration flexibility.

2.4 Limitations of Existing Approaches

Despite the variety of research on data pipeline supervision, several important limitations remain. These involve both technical and methodological aspects, showing a gap between theoretical solutions and real-world production needs.

First, many solutions are highly domain-dependent. For example, pipelines designed for the oil industry often include physics-based modules specific to that field [3]. While this improves accuracy in that context, it limits generalizability to other sectors.

Second, several approaches focus on collecting information (logs, metrics, traces) without providing intelligent mechanisms for analysis or action.

Patel's observability model [5], for instance, offers a solid foundation but lacks native alerting or automated incident response. This limits proactive supervision.

Moreover, while tools like Apache NiFi or StreamSets provide built-in monitoring interfaces, they often require advanced manual configuration to handle complex cases (e.g., multivariate anomaly detection or adaptive flow management). Their flexibility usually comes with high operational overhead.

Architecturally, classical pipeline models (ETL, ELT) do not always ensure good traceability of successive transformations, particularly in hybrid batch/streaming contexts. Rucco et al. [6] propose alternative design patterns (ETLT, ELTL), but these are not yet widely adopted or implemented in standard software tools.

Finally, few solutions adopt modular, interoperable architectures. In dynamic environments where data sources, volumes, and tools evolve rapidly, rigid pipelines hinder scalability, adaptability, and integration of emerging technologies such as AI.

These observations highlight the need for a unified, extensible solution that combines supervision, anomaly detection, and real-time visualization, relying on flexible open-source tools.

Table 2.1: Comparison of Data Pipeline Supervision Solutions

Ref.	**Tool / System**	**Approach**	**Identified Limitations**
[4]	AI Pipeline	Integrated supervised machine learning	Requires representative training data; complex deployment
[5]	Maturity model for observability	Collection of logs, metrics, traces; global supervision	Technically expensive implementation; no integrated alerting
[6]	ETLT and ELTL models	Design patterns for traceable pipeline construction	Not implemented in standard tools; lacks automation
[3]	Physics-based pipeline in oil industry	Physical law filtering + ML	Domain-specific; not easily generalizable
[1]	ELI Pipeline	IoT pipeline with data curation and quality control	High IoT complexity; domain dependency
[2]	ERAIA Platform	Intelligent architecture for IoT-based systems	Weak anomaly detection; no integrated alerting

This table highlights the diversity of approaches used for data pipeline supervision. While some focus on anomaly detection (Pasupuleti, Nasiri), others emphasize architecture or modeling (Rucco, Hernandez). However, few provide a comprehensive solution combining acquisition, processing, supervision, and alerting, thus justifying a modular architecture for our proposed solution.

2.5 Conclusion

This chapter highlighted the main approaches to data pipeline supervision, along with the tools and models used. Despite progress, unmet needs remain—particularly in modularity, interoperability, and intelligent anomaly

detection.

These findings pave the way for proposing a supervised pipeline architecture integrating open-source tools and real-time alert mechanisms. The next chapter details this proposed solution.

Chapter 3

Design of the Proposed Solution

3.1 Objectives of the Solution

The proposed solution aims to address the gaps identified in the literature review—specifically the lack of interoperability, absence of intelligent alerting, and integration challenges in industrial environments. The goal is to design a supervised pipeline architecture capable of:

- automating data acquisition, processing, and storage;
- detecting anomalies in near real-time;
- generating alerts;
- and providing intuitive visualization of pipeline health.

3.2 Global Architecture of the Supervision Pipeline

The following figure illustrates the functional architecture of the proposed solution. It integrates components for data collection via APIs, normalization, storage, aggregation, and visualization. It relies on a modular design to facilitate maintenance and extensibility.

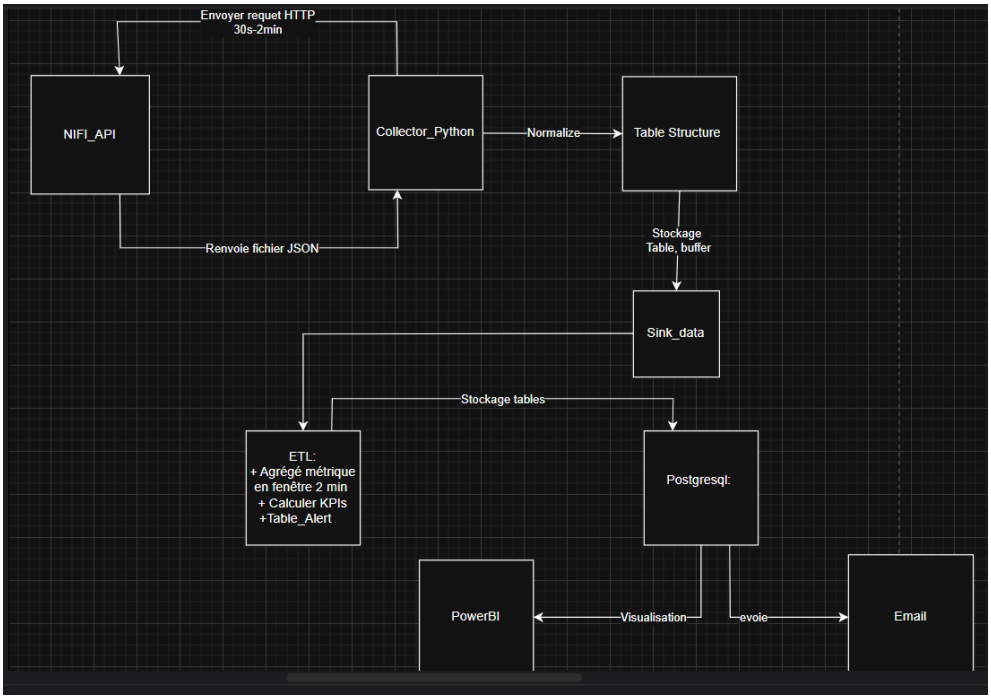

Figure 3.1: Proposed Architecture

3.2.1 Component Description

- **NIFI_API**: periodically sends HTTP requests (every 30s–2min) to API endpoints.
- **Collector_Python**: receives JSON files, normalizes data to unify schema.
- **Table Structure**: defines columns, data types, and normalized table structure.
- **Sink_data (to be defined)**: temporary buffer to prevent loss during write delays.
- **PostgreSQL**: relational database for persistent, historical data storage.
- **ETL**: aggregates metrics (2-minute windows), calculates KPIs, and generates an alert table.

- **Power BI**: visualization tool for monitoring indicators.
- **Email**: automatic alert system based on thresholds defined in the alert table.

3.2.2 Pipeline Operation

The process begins with data collection via NiFi, followed by Python-based structural normalization. After normalization, data are temporarily stored in a buffer (Sink_data), then transferred to PostgreSQL. An ETL module runs every two minutes to: - aggregate data over time windows, - compute KPIs (latency, throughput, errors), - generate alerts when thresholds are exceeded.

Finally, Power BI visualizes results, and alerts are automatically emailed upon anomalies.

3.3 Choice of Technologies

3.3.1 Apache NiFi

Apache NiFi is an open-source dataflow orchestration tool that collects data from various sources (APIs, files, databases). With a graphical interface and logging system, it fits perfectly for this use case.

3.3.2 PostgreSQL

PostgreSQL is a robust relational database ideal for storing historical metrics and ETL results. It supports complex queries and temporal indexing, necessary for KPI calculations.

3.3.3 Power BI and Alerting

Power BI provides dynamic dashboards for real-time monitoring. The alerting process is automated (via script or webhook) that reads from the alert table and sends emails based on predefined rules.

3.4 Contribution Compared to the State of the Art

The proposed solution stands out in several aspects:

- **Interoperability**: uses only open-source, widely compatible tools (NiFi, PostgreSQL, Power BI);
- **Fast Detection**: time-based aggregation and KPI calculations allow precise monitoring and automatic alert generation;
- **Modularity**: independent components simplify future updates or extensions;
- **Integrated Visualization**: enables non-technical users to access key metrics easily.

3.5 Conclusion

This architecture provides a concrete response to the gaps identified in the literature. It ensures data pipeline quality, availability, and transparency. The next chapter will focus on technical implementation and results.

General Conclusion

As data volumes continue to grow, supervising data pipelines has become central to ensuring quality, reliability, and continuity of data flows. This thesis focused on designing a technical solution to detect anomalies in data pipelines, leveraging modern tools and observability principles.

First, a literature review highlighted existing approaches, limitations, and commonly used tools such as Apache NiFi, PostgreSQL, and Power BI. The study revealed a lack of simple, interoperable solutions capable of triggering real-time alerts upon data flow anomalies.

Based on these findings, a modular architecture was proposed. It integrates several components: a data collector (via API), a normalization module, a temporary buffer (*Sink_data*), a relational database for historical storage, a KPI aggregation engine, an automated alert system, and a visualization dashboard. This solution meets requirements for modularity, automated supervision, and traceability.

Operational implementation will allow further testing on concrete use cases. Future improvements may include predictive models for anomaly anticipation, adaptation to Big Data environments, or generalization to other pipeline types (real-time, distributed batch, etc.).

This work thus contributes to data governance by proposing an accessible, robust, and scalable solution for data pipeline supervision in industrial contexts.

Bibliography

[1] De Haro-Olmo, F. J., Ramos-Munoz, J. J., Baena, E. et al., "ELI: An IoT-Aware Big Data Pipeline with Data Curation and Data Quality," *PeerJ Computer Science*, vol. 9, October 2023, p. e1605.

[2] Hernandez, A., Venero, L., et al., "ERAIA - Enabling Intelligence Data Pipelines for IoT-based Application Systems," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, IEEE, 2020, pp. 1–9.

[3] Nasiri, M., Chalvatzis, K., et al., "Enhancing Anomaly Detection in Oil and Gas Rotating Machinery through a Physics-Informed Data Filtration Pipeline," Preprint, 2023.

[4] Pasupuleti, S., "AI-Augmented Data Pipelines: Integrating Machine Learning for Intelligent Data Processing," *Journal of Computer Science and Technology Studies*, vol. 7, no. 11, 2025, pp. 276–283.

[5] Patel, H. R., "A Maturity Model for Observability in Big Data Pipelines," *Journal of Computer Science and Technology Studies*, vol. 7, no. 11, 2025, pp. 195–202.

[6] Rucco, C., Saad, M., and Longo, A., "Formalizing ETLT and ELTL Design Patterns and Proposing Enhanced Variants," Working Paper, University of Salento, 2025.