Available online at www.HighTechJournal.org

ISSN: 2723-9535

HighTech and Innovation Journal

Vol. 7, No. 2, June, 2026

A Governed Lakehouse DataOps Architecture: Design and Evaluation in Healthcare

Aymen Fannouch [1*], Jihane Gharib [1], Youssef Gahi [1]

[1] *Laboratory of Engineering Sciences, National School of Applied Sciences, Ibn Tofail University, Kenitra 14000, Morocco.*

Received 26 February 2026; Revised 21 May 2026; Accepted 25 May 2026; Published 01 June 2026

Abstract

Healthcare organizations increasingly require secure, governed and AI-ready data pipelines capable of handling heterogeneous and sensitive data sources. This study aims to design and evaluate a unified DataOps reference architecture that operationalizes the full data lifecycle through a governed Medallion Lakehouse model. Methodologically, the proposed architecture integrates data-centric CI/CD, Infrastructure as Code, workflow orchestration, governance and metadata management, monitoring, and explicit promotion contracts across Bronze, Silver, and Gold layers. The framework was implemented and evaluated in a controlled healthcare testbed using approximately 3.5 GB of multi-source clinical data over a 25-day workload. The findings show that the proposed architecture achieved a DataOps Operational Excellence Index (DOEI) of 0.92, an ingestion throughput of approximately 100 MB/s, a data quality score of 97.87% and a 72% reduction in infrastructure provisioning time, from 3 hours to 50 minutes. The main novelty of this work lies in combining a governed Lakehouse-based DataOps architecture with explicit promotion contracts and a composite benchmarking index for assessing operational maturity. This improvement provides a reproducible, auditable, and scalable framework for secure data operations in regulated environments such as healthcare.

Keywords: DataOps; Agile Data Management; Data Driven Enterprise; Data Governance; Medallion Lakehouse; Data Pipeline; Healthcare.

1. Introduction

In recent years, data has evolved from a by-product of operational systems into a strategic asset, for organizations. Decisions at operational, managerial, and strategic levels are increasingly supported by analyses derived from large, heterogeneous, and continuously generated datasets. This transformation has intensified the challenges, associated with data volume, velocity, variety, quality, governance, and real-time accessibility [1]. In healthcare, these challenges are even more critical, because data pipelines must handle sensitive clinical information, heterogeneous sources, regulatory constraints and the need for timely decision support [2].

The Big Data era has further amplified these issues. Organizations now collect data, from multiple sources, including sensors, enterprise systems, social media platforms, Internet of Things devices, and domain-specific operational systems [3, 4]. These data may be structured [5], semi-structured [6] or unstructured [7] making their integration, validation, governance, and transformation increasingly complex [8]. In response, modern data platforms and architectural paradigms including cloud-native data platforms and Lakehouse architectures, have been proposed to improve scalability, flexibility, and analytical readiness [9]. However, architectural modernization alone does not fully address the operational challenges of continuously delivering reliable, governed, and reproducible data products [10].

* Corresponding author: aymen.fannouch@uit.ac.ma

https://doi.org/10.28991/HIJ-2026-07-02-021

DataOps has emerged as a response to these limitations by adapting principles from Agile, DevOps, Lean, and continuous delivery to the data lifecycle [11-14]. Its objective is to improve collaboration, automation, monitoring, and reliability across data pipelines. Previous studies have examined DataOps from different perspectives. Some contributions have focused on DataOps adoption and lifecycle models in healthcare [15], while others have addressed DataOps for cyber-physical systems, labour market intelligence, semantic integration, data quality automation, or enterprise big data pipelines [16-19] etc... These studies demonstrate the growing relevance of DataOps and confirm its potential for improving the operational management of data-intensive systems.

Nevertheless, the existing literature remains fragmented. Many contributions, address isolated aspects of the DataOps lifecycle such as ingestion, quality control, semantic integration, governance, orchestration, or monitoring, without providing a unified end-to-end architecture that operationalizes the full path from data sources to consumption. Moreover, several important capabilities are rarely integrated within a single coherent framework, including Medallion Lakehouse storage, data-centric CI/CD, Infrastructure as Code, explicit promotion contracts, metadata-driven governance, and standardized operational evaluation. This limitation is particularly significant in healthcare, where data operations require not only scalability and automation, but also traceability, reproducibility, privacy protection, and auditable governance.

To address these gaps, this study proposes a governed Lakehouse-based DataOps reference architecture designed for regulated healthcare environments. The proposed architecture integrates workflow orchestration, data-centric CI/CD, Infrastructure as Code, governance and metadata management, monitoring, and explicit promotion contracts across bronze, silver, and gold layers. The architecture is implemented and evaluated in a controlled healthcare testbed to assess its operational performance, governance effectiveness, and reproducibility. In addition, this work introduces the DataOps Operational Excellence Index (DOEI), a composite evaluation framework designed to support more structured benchmarking of DataOps implementations.

The contributions of this work are summarized as follows:

- C1: Governed end-to-end reference architecture. We propose a unified and modular DataOps architecture that operationalizes the source-to-consumption lifecycle with explicit phase boundaries, artefacts, and promotion contracts aligned with a Medallion Lakehouse progression.
- C2: Data-centric CI/CD with governance gates. We define a data-centric CI/CD process in which dataset promotion is regulated by contract checks, schema validation, data quality tests, and compliance controls, with governance enforced through integrated metadata, lineage, and access policies.
- C3: Reproducible tool justification via the Spine Model. Because existing DataOps studies, seldom justify tool choices systematically, we operationalize the Spine Model to structure tool selection through a value-driven chain linking needs, values, principles, practices, and tools.
- C4: Standardized benchmarking through the DOEI framework. We address the lack of standardized evaluation practices in DataOps, by introducing the DataOps Operational Excellence Index which aggregates processing efficiency, governance, agility and resource utilization into a composite operational maturity score.
- C5: Reproducibility assets. We provide a reproducible deployment approach based on Infrastructure as Code, containerization, and orchestration pseudo-code, enabling adaptation of the pipeline in similar regulated environments.

In this work, C1–C4 represent the main scientific contributions, while C5 captures the engineering integration and reproducibility contribution. Prior DataOps studies often emphasize lifecycle guidance, partial implementations, or specific technical capabilities, but they generally do not operationalize explicit promotion contracts, IaC-driven reproducibility, data-centric CI/CD gates, and Lakehouse-based governance within a single validated architecture. Conversely, Lakehouse-oriented works primarily focus on storage-layer guarantees, rather than pipeline-wide automation and governance across the full data lifecycle. By integrating these strands into one governed and reproducible framework, and validating it empirically in a healthcare setting, this study contributes an auditable end-to-end DataOps implementation that addresses a documented gap in the existing literature.

To guide the study the following research questions are formulated:

- RQ1: What are the key architectural components and open-source tools required for implementing a governed DataOps pipeline?
- RQ2: How can their integration be optimized through a value-based tool selection framework?
- RQ3: How does the proposed DataOps architecture improve operational performance in a healthcare data environment?

RQ1 is addressed in Section 5 through the proposed architecture, RQ2 in Section 6 through the Spine Model-based implementation process, and RQ3 in Section 7 through the healthcare case study and performance evaluation. The remainder of the paper is organized as follows. Section 2 presents the conceptual background of DataOps and Lakehouse

architectures. Section 3 describes the research methodology. Section 4 reviews existing contributions and identifies the main gaps. Section 5 introduces the proposed modular DataOps architecture. Section 6 details tool selection and implementation. Section 7 presents the healthcare application and empirical results. Section 8 discusses the findings, limitations, and future work. Section 9 concludes the paper.

2. Background

DataOps is a collaborative paradigm that integrates the iterative delivery of agile methodology [11], the continuous-delivery of DevOps [12], and advancing continuous-improvement maturity in Lean Manufacturing [13]. Its objective is to optimize the data lifecycle from ingestion to consumption by fostering cross-functional collaboration, automating repetitive tasks and reducing friction [14]. Together, these perspectives provide the conceptual basis for the DataOps principles that guide the design and operation of practical data pipelines.

Building on this conceptual foundation, Table 1 synthesizes the core principles of DataOps ensuring iterative improvement to pipelines. These principles enable key benefits like quicker time to insight, improved data quality and scalability. The table also highlights the primary technical elements that underpin these practices. These are particularly critical in domains like healthcare, where timely, accurate and well-governed data can directly impact patient outcomes.

Table 1. DataOps Principles, Advantages, Technical Components

Aspect	Description
Core Principles	Agility: Iterative development for a continuous improvement and feedback loops. Automation: to reduce the manual tasks and streamlining the workflow of the data. Collaboration: Between functional teams including data engineers with scientists and operations. Continuous Integration and Delivery: Frequent, reliable updates to data pipelines.
Key Advantages	Short time to insight through real time analytics. Improved data quality by the continuous testing and validation. Scalability to handle data with its diverse formats and large volumes. Enhanced governance and compliance by the integration of security and lineage for tracking.
Technical Components	Data Ingestion: Tools like Apache NiFi or Apache Kafka. Processing Frameworks: Apache Spark, Flink and DBT. Storage: HDFS, Delta Lake, Lakehouse architectures. Orchestration: Apache Airflow, Prefect. CI/CD Pipelines: GitLab CI/CD or Jenkins. Infrastructure as Code: Terraform, Ansible. Monitoring/Observability: ELK Stack, Prometheus.

Furthermore, the choice of underlying data architecture strongly influences the effectiveness of the pipeline, and to motivate the Medallion Lakehouse choice in a DataOps pipeline, we compare it with its predecessors. Table 2 explores the key characteristics of data lakes, data warehouses, and data Lakehouse across a number of dimensions. It also shows how the data Lakehouse combines the flexibility and scalability of data lakes with the structured reliability of data warehouses, bringing together the strengths of both. This hybrid capability provides a solid foundation for the DataOps pipeline that we propose below.

Table 2. Architectural Comparison: Data Lake vs. Data Warehouse vs Data Lakehouse

Comparison Criteria	Data Lake	Data Warehouse	Data Lakehouse
Data Types	Supports raw data in various formats: structured, semi structured, and unstructured.	Handles primarily structured and pre-processed data.	Supports a mix of structured, semi structured, and unstructured data.
Data Format	Utilise open-source formats (e.g., Parquet, ORC).	Uses closed or vendor specific formats	Relies on open data formats
Purpose	Designed for advanced analytics, AI, and machine learning.	Mainly used for reporting and traditional business intelligence.	Built to support diverse workloads including BI, ML, and AI.
Cost	Offers a low cost, scalable storage solution.	Typically, more expensive due to compute and storage integration.	Balances cost efficiency with broad functionality.
Scalability	Easily scalable due to decoupled storage and compute layers.	Scalability is limited due to tightly integrated architecture.	Provides seamless scalability like a data lake.
Agility	Offers flexibility with minimal configuration required.	Rigid setup with extensive configuration needs.	Combines the flexibility of lakes with structured management.
Ease of Use	Schema is applied on read; setup and data prep require technical expertise	Schema on write simplifies querying but requires upfront design effort	Supports schema on read with improved usability and simplified data preparation.
Processing	Storage and processing are separated allowing flexibility.	Combines storage and processing reducing flexibility.	Separates compute from storage while enabling efficient operations.
ACID Compliance	Lacks native ACID compliance external tools are needed.	Fully ACID compliant due to tightly integrated components.	Natively supports ACID transactions over flexible data storage formats.

To move beyond theoretical principles, it is important to consider how these practices have translated into measurable outcomes in real-world settings. Beyond these conceptual foundations, the practical impact of DataOps is also evidenced by industry surveys and statistics. For instance, according to 451 Research, among 300 North American firms, almost four in 5 respondents agreed that the implementation of DataOps led to greater success. Over half of companies are unable to produce any valuable insights without DataOps. A quarter of employees spend more than half of their effort simply accessing data. At the same time, companies that are maturing in DataOps report lower compliance risks (44% less), faster legal response times (35% quicker), and better collaboration overall- all of which can be traced to their operational efficiency as well as efforts in cloud integration, with its contribution toward agility and competitiveness [20]. These advantages were also highlighted by other surveys. The first, according to a 2021 IDC survey, successful implementation of DataOps has led to a reduction by 49% in how often data analytics products are delivered late [21]. However, the second highlights a survey of the working environment without the adoption of DataOps, based on 600 data engineers, including 100 who are managers, found that nearly all (97 %) were suffering from burnout. Over 70% indicated they plan to leave their current company within the next year, and 79% have even thought about leaving data altogether [22]. These findings underscore the urgent need for structured DataOps methodologies to improve both operational efficiency and employee well-being.

In response to these challenges, leading technology companies like Facebook, LinkedIn, eBay, and Netflix have increasingly integrated DataOps into their data operations practices [14]. For example, Netflix uses DataOps for conducting real-time analysis and making personalized suggestions based on tremendous streams of data [23]. Following this, the DataOps market is experiencing rapid growth. Global revenue in 2023 reached $3.9 billion, and analysts forecast a compound annual growth rate (CAGR) of 23% from 2023 to 2028, when it will have grown to approximately $10.9 billion at the end of that period [21]. Platform-based solutions will be the mainstay in the year of 2023, accounting for more than 65 percent of total income worldwide. Its ascendency is thanks to be growing demand for real-time (or close to real-time) data processing and analysis capabilities. Such capabilities help an enterprise make faster decisions and keep ahead in competition [24].

To meet these evolving needs, many vendors have entered the market in recent years. In its 2024 DataOps Buyers Guide, Information Services Group (ISG), a leading technology research and advisory firm, evaluated 49 solution providers across several different categories, including Data Observability, Orchestration, Pipelines, and Data Products. Notable brands here included Cloudera, IBM, DataKitchen, Collibra, and Informatica, among others [25]. Even though the advantages of implementing DataOps are becoming more apparent in many industries, there are still shortcomings when these solutions are used in practical situations, because most contributions are still disjointed and only address discrete issues like data quality or ingestion, failing to offer a cohesive, end-to-end architecture. These shortcomings are especially critical in healthcare, where patient record management and heterogeneous clinical data require seamless integration, governance, and automation. To systematically address these gaps, the next section presents the research methodology, which is structured to analyze existing practices, identify their limitations, and design a coherent framework capable of overcoming these healthcare-specific integration challenges.

3. Research Methodology

This research follows a structured methodology composed of six interconnected stages, as depicted in Figure 1. Rather than a linear sequence, the process forms an integrated workflow that progressively moves from knowledge consolidation to practical validation, ensuring that each step builds logically on the previous one.

Figure 1. Research Methodology Process

It begins with a systematic literature review reported in accordance with PRISMA 2020, a widely adopted reporting guideline that promotes transparent and reproducible study identification and selection [26]. Specifically, the PRISMA flow diagram as illustrated in Figure 2 documents the number of records retrieved, screened, excluded (with reasons) and ultimately included, thereby making the evidence base supporting our gap analysis and framework design auditable. This stage consolidates existing knowledge by refining an initially broad scope into a curated body of relevant contributions, laying the intellectual foundation of the study.

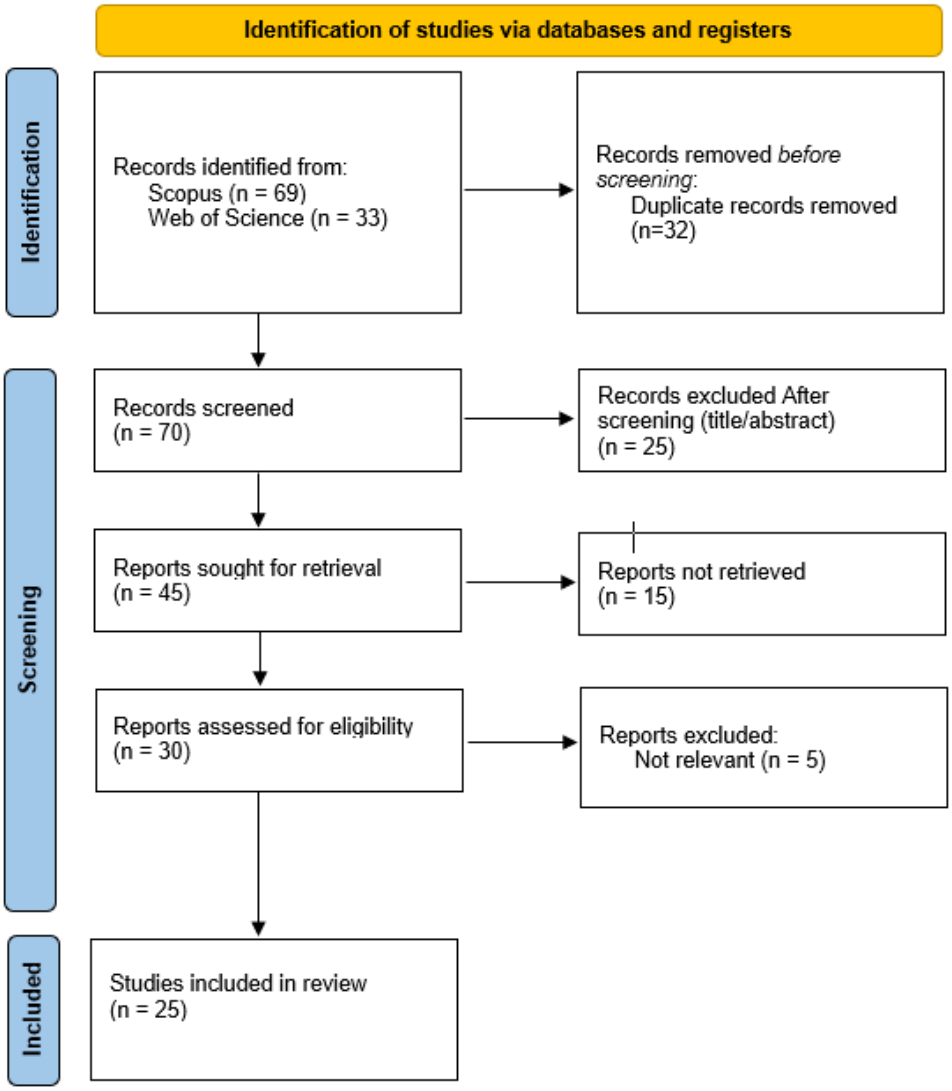

Figure 2. PRISMA flow Diagram

The second stage carries out a gap analysis, identifying limitations in current contributions and solutions that are particularly critical in healthcare and where challenges persist. From these insights, the third stage develops a conceptual framework, synthesizing findings into the key design principles of a modular architecture. The fourth stage translates this conceptual model into practice through the selection and integration of tools, guided by the Spine Model, a value-driven tool selection framework. The fifth stage operationalizes the framework in a realistic healthcare environment, testing both technical feasibility and organizational adaptability. Finally, evaluates performance using Key Performance Indicators (KPIs), offering verifiable evidence of the pipeline's capability.

This methodology ensures that our proposed solution is not only theoretically designed but also demonstrably effective in addressing healthcare's data challenges.

4. Literature Review

This section begins with an overview of the DataOps evolution, followed by a critical analysis of existing solutions, examining strengths, limitations, and gaps, while identifying the gaps that serve as the starting point for our work.

As Figure 3 shows, the evolution of DataOps reflects this holistic vision, passing through four fundamental phases: from the first challenges of Big Data integration (2004-2010), to the adoption of DevOps principles in data operations (2011-2015), the formalisation of DataOps frameworks and methodologies (2016-2020), and the current phase (2021-2025), marked by the adoption of CI/CD and industrial deployments in areas such as healthcare [15], and cyber physical systems (CPS) [16]. This evolution underlines growing emphasis on agility, automation, and governance as fundamental principles of modern data management. This trajectory is especially relevant to healthcare, where the complexity of patient records and medical imaging highlights the need for robust and adaptive data management approaches.

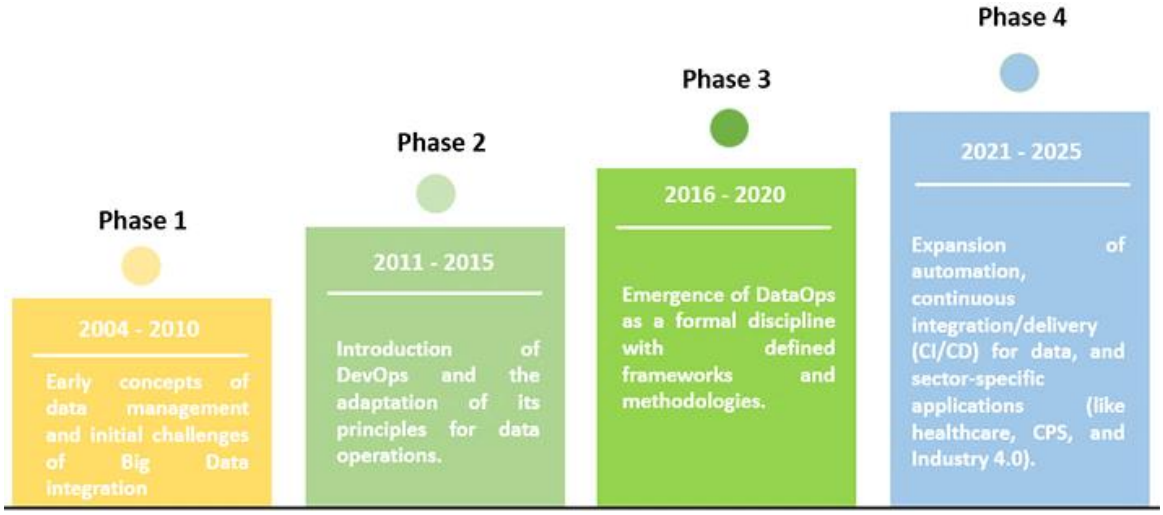

Figure 3. DataOps Evolution

Complementing this conceptual timeline, Figure 4 provides a bibliometric overview of the reviewed corpus (2015–2026), capturing the period during which DataOps emerged and consolidated as a distinct research stream. A striking pattern is the recency of the field: more than 90% of the reviewed contributions are concentrated in 2020–2025, signalling a rapidly accelerating, trend-driven research agenda rather than a mature and stabilised body of knowledge. This surge culminates in 2025, consistent with the recent push toward industrial-grade adoption and the convergence of DataOps with CI/CD, governance, and scalable architectures.

At the same time, the distribution of document types indicates that the journal articles account for only ≈27%. This imbalance suggests that, despite fast growth, much of the evidence is still oriented toward early-stage proposals, modular solutions, and conceptual frameworks, whereas fewer studies provide the depth typically expected in journal publications namely rigorous end-to-end validation, reproducible implementation details, and comprehensive governance and quality evaluation.

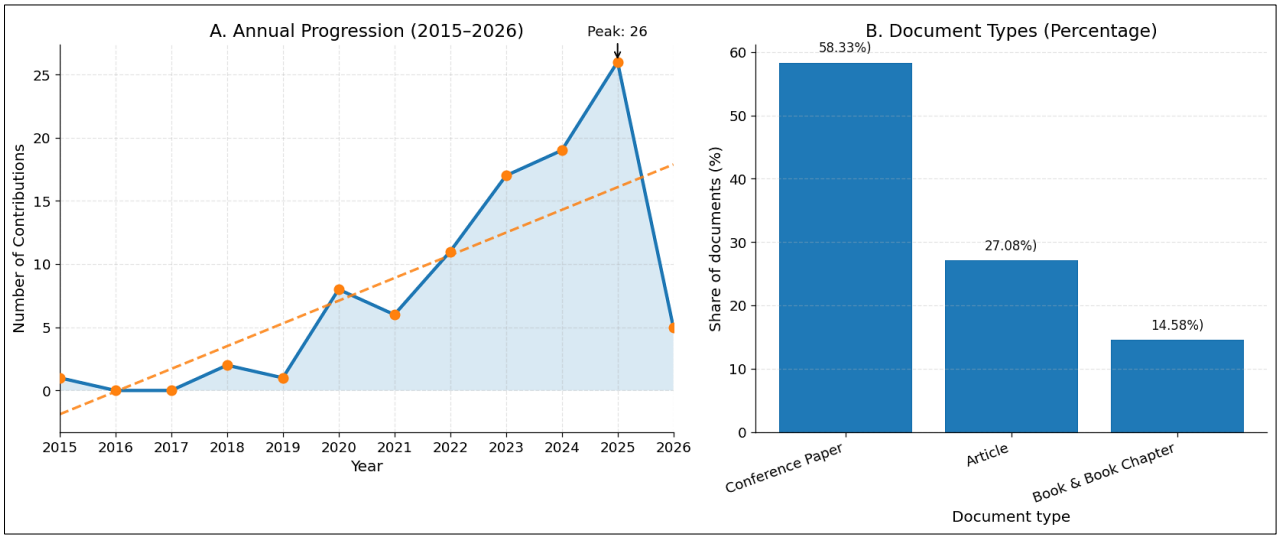

Figure 4. Bibliometric overview of the DataOps literature considered in this review (2015–2026): (A) annual publication trend; (B) distribution of document types

Building on this historical and bibliometric perspective, we now examine current DataOps practices and their limitations. Existing DataOps practices draw on multiple frameworks and methodologies that prioritise automation, collaboration, and adaptability to optimise end-to-end data workflows in different sectors, such as financial, healthcare, and software development. Structured lifecycle models incorporate Agile, DevOps, and Lean principles. They often rely on workflow orchestration, deployment automation, and governance to enhance operational efficiency. Table 3 presents a comparative evaluation of key contributions in the field.

Table 3. Evaluation of DataOps Contributions by Functional Dimensions and Domains

Contribution	End-to-end	Governance	CI/CD	Lakehouse	IaC	Monitoring /Testing	Workflow Orchestration & Containerisation and Scalability	Domain
Tamburri et al. [17]	x					x		Labor Market
Bahaa et al. [15]	x	x				x		Healthcare
Scrocca et al. [18]	x		x			x		Healthcare + Manufacturing + Mobility
Garriga et al. [16]	x	x				x		Airport Passenger Flow
Pestana et al. [19]	x	x				x		Ceramic and Glass Industry
Zahid et al. [27]	x	x	x			x	x	Telecommunications / Enterprise Big Data
Mat Rejab et al. [12]	x	x						Generic IT / Conceptual for IT MLEs
Pinkel et al. [28]	x							Generic Enterprise Semantic Data Integration
Tu et al. [29]	x	x				x		Generic / Enterprise ML and BI Pipelines
Bayram et al. [30]	x					x		Industrial Manufacturing

Despite the growing number of publications on DataOps, our content review and the bibliometric analyses in Figure 5 indicate that the literature remains fragmented. Most contributions address isolated capabilities such as data ingestion, data quality, governance, or data reduction rather than proposing an end-to-end reference architecture that unifies the DataOps lifecycle. In addition, several components that are increasingly relevant for operational DataOps are still rarely integrated within a single coherent solution, notably the adoption of Medallion Lakehouse storage patterns, the use of Infrastructure as Code (IaC) to support reproducible deployments, and the consistent enforcement of a comprehensive governance framework across pipeline stages. Beyond architectural coverage, the implementation approach is often insufficiently specified to enable replication and adaptation across different use cases. Many studies remain conceptual, and technically oriented works frequently provide only limited deployment and integration details. Finally, evaluation practices are not standardised: existing studies do not converge on a clear evaluation protocol or comparable metrics that could serve as a baseline for benchmarking and cumulative improvement. Altogether, these gaps motivate the need for a unified and operational DataOps framework that is reproducible and evaluable in domain-specific settings.

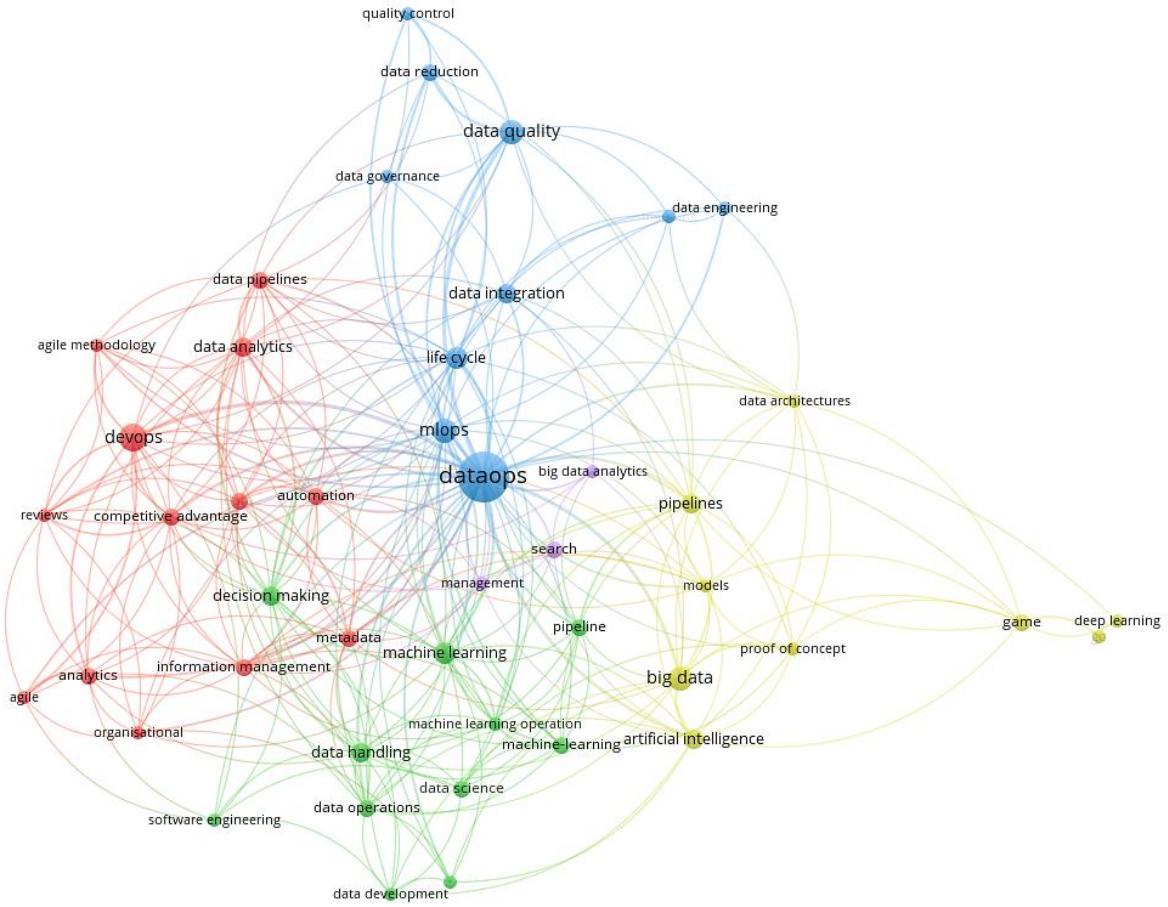

Figure 5. Conceptual Landscape of DataOps Related Contributions

Similarly, Figure 6 shows a significant gap in DataOps implementation within healthcare, representing less than 2% of all contributions. This gap is evident globally, particularly in healthcare implementation addressed by our architecture in Section 6.

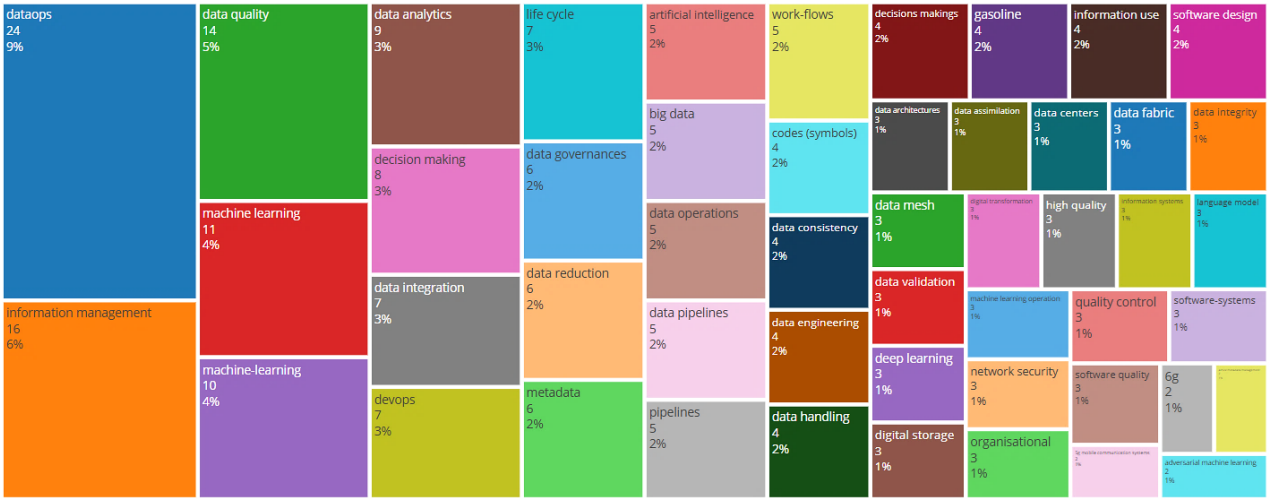

Figure 6. Tree map of Key Research Themes Related to DataOps

5. Modular DataOps Architecture: Phases and Components

This section presents the proposed DataOps architecture, designed to operationalise the principles described above. It begins with an overview of the pipeline, followed by a detailed examination of each component, giving a specific framework for each module.

5.1. Overview

An effective DataOps pipeline requires an architecture, that encompasses all stages of the data lifecycle starting with an ingestion layer that automates and validates incoming data. This layer is integrated with a Medallion Lakehouse, which combines the benefits of Lakehouse and Medallion concepts: The Lakehouse benefits from the flexibility of data lakes which can handle large volumes and various data formats, while also providing the transactional guarantees typical of data warehouses. The Medallion layer builds on this, by organizing data into distinct stages (Bronze, Silver, Gold), enhancing data quality and governance across the pipeline.

However, a seamless flow from raw data to insights requires more than stage-to-stage movement. The architecture must be anchored by pillars that ensure stability and continuous evolution. As illustrated in Figure 7, these include:

- CI/CD: Automation and rapid iteration.
- Data Governance: Secure monitoring and management.
- Workflow Orchestration: Dynamic resource allocation.
- Testing & Monitoring: Real-time validation
- Infrastructure as Code: Reproducibility and security.

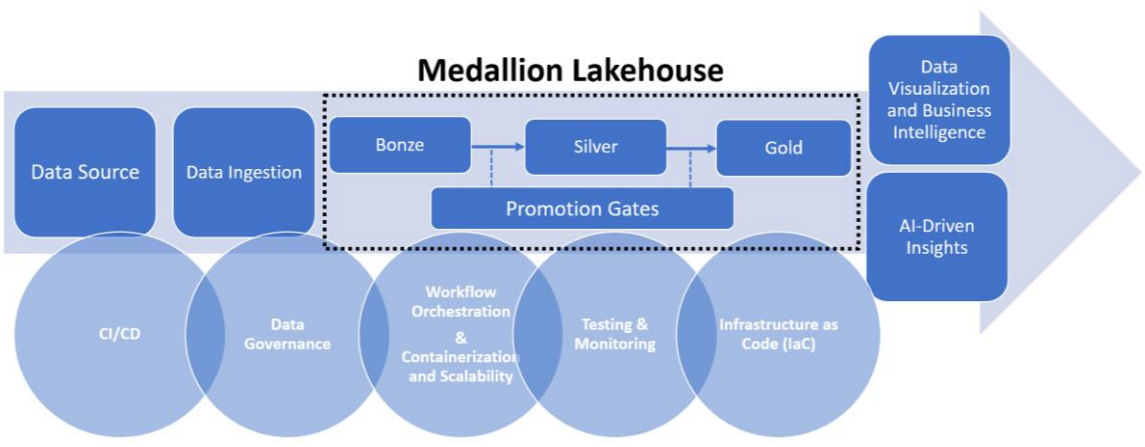

Figure 7. End-to-End Conceptual DataOps Reference Architecture Integrating a Governed Medallion Lakehouse with Promotion Contracts

To make phase boundaries and promotion contracts explicit, Table 4 summarizes the seven canonical phases (Source to Consumption) with inputs to outputs, artefacts, promotion quality gates, and KPIs. Having presented an overview of the proposed architecture with its pillars, the following sections provide a detailed exploration of each core component.

Table 4. Canonical DataOps Phases: Inputs to Outputs, Artefacts, Promotion Contracts and KPIs

Phase	Inputs → Outputs	Artefacts	Quality Gates / Policies (Promotion Contracts)	KPIs
Source & Contracts	Business specs → **Data contracts** (schema, PII tags, SLAs)	Versioned contract, schema registry, sample sets	Contract validation, breaking-change policy, PII tagging policy	% sources under contract; % changes caught pre-prod
Ingestion	Streams/Batch → **RAW/Bronze**	Manifests, flow logs, **DLQ**	Idempotence, retry, watermarking, DLQ policy	Ingestion latency; % to DLQ; backfill time
Bronze	RAW landing → **Working/Silver inputs**	Partitions, **checksums**, immutable files	**WORM**/immutability; duplicate detection	% rewrites (=0); landing completeness
Silver	RAW → **Curated/Silver**	Spark jobs, DQ rules, unit tests, schema evolution files	DQ tests (completeness, uniqueness, validity), backfill policy, schema-compact rules	DQ score; test pass rate; cost/GB
Gold	Curated/Silver → **Views/Marts/Features**	Versioned views, marts, feature store entries	Versioning, freshness SLOs, reproducibility	P95 query latency; TPS; freshness lag
Consumption	Gold → **BI/API/ML**	Dashboards, APIs, model endpoints	Throttling, client SLAs, drift/rollback policy	Uptime; 4xx/5xx error rate; adoption (#active users)
Governance/Metadata	Any ↔ **Catalog**	**Lineage graphs**, glossary, tags, ACLs	RBAC/ABAC, GDPR/PHI policy, approval workflow	% catalogued objects; policy coverage; approval lead time

5.2. Source and Ingestion

Every data pipeline's origin is specified in the Source phase, it creates data contracts between consumers (the pipeline) and producers (such as devices, clinical systems, sensors and external APIs and applications). Prior to ingestion, this stage ensures that inputs are precisely specified. The ingestion stage is the starting point of the pipeline where the raw data, streams or batches are collected, validated and sent into the RAW/Bronze zone. The primary goal of this stage is to ensure the reliability, idempotence and observability of data movement, while maintaining low latency and traceability of each record.

The ingestion stage can be summarised through a set of core capabilities that ensure alignment with DataOps principles (Figure 8).

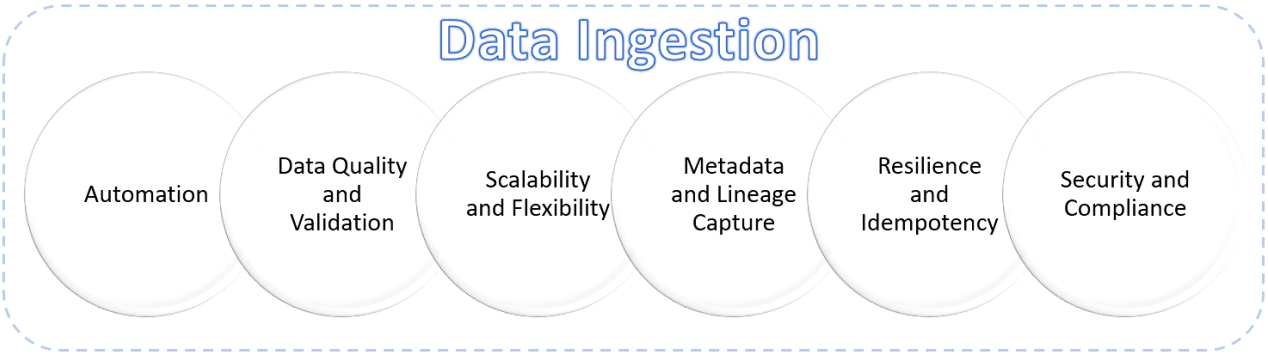

Figure 8. Phase Capability Pattern

5.3. Bronze Layer

During this stage the collected data is kept in its original format. It serves as the pipeline's unchangeable landing zone, preserving data fidelity and ensuring that any changes remain auditable.

Before data is promoted from the Bronze layer to the Silver layer, it must meet specific standards in data quality, governance, and operational efficiency. These dimensions are combined into a single composite score, referred to as the Trustworthiness Data Score (TDS). A TDS ≥70% is required, for the data to pass through the gate and be promoted. This gate serves as the promotion contract between these two phases to ensure that data meets all required specifications.

5.4. Silver Layer

The raw data from the Bronze zone is cleaned up, made more consistent, and improved so that it may be used for analysis during the Transform/Working/Silver phase. At this point, both technology (schema alignment, type casting, and deduplication) and business processes (adding reference data and checking against healthcare standards) need to be better. The goal is to make sure that datasets that are well-organised and reliable can be conveniently used in later steps. Before promotion to the Gold the data must meet a Trustworthiness Data Score (TDS) $\geq 95\%$. This ensures that the data has passed all necessary quality, governance and operational checks and can be trusted for final consumption in AI-driven analytics, machine learning models or business intelligence applications.

5.5. Gold Layer

The Serving/Gold phase delivers high-quality datasets that are ready for consumption. At this stage, data has passed all validation and transformation steps and is optimised for consumption. In a healthcare context, this means exposing trusted and standardized datasets.

5.6. Consumption

The Consumption phase is the last step in the pipeline where trusted data is sent to end users. This layer makes it possible for machine learning model endpoints, business intelligence dashboards, regulatory reporting and API-based services. In healthcare, this means that clinicians, administrators and policymakers can use it to get useful information, such as real-time dashboards for monitoring patients and predictive analytics for tracking the progress of diseases.

Beyond the description of each modular component, the pipeline relies on a CI/CD backbone that enforces automatic validation, monitoring, and rollback mechanisms. Figure 9 illustrates this data-centric CI/CD process, highlighting how gates regulate the promotion of datasets across stages.

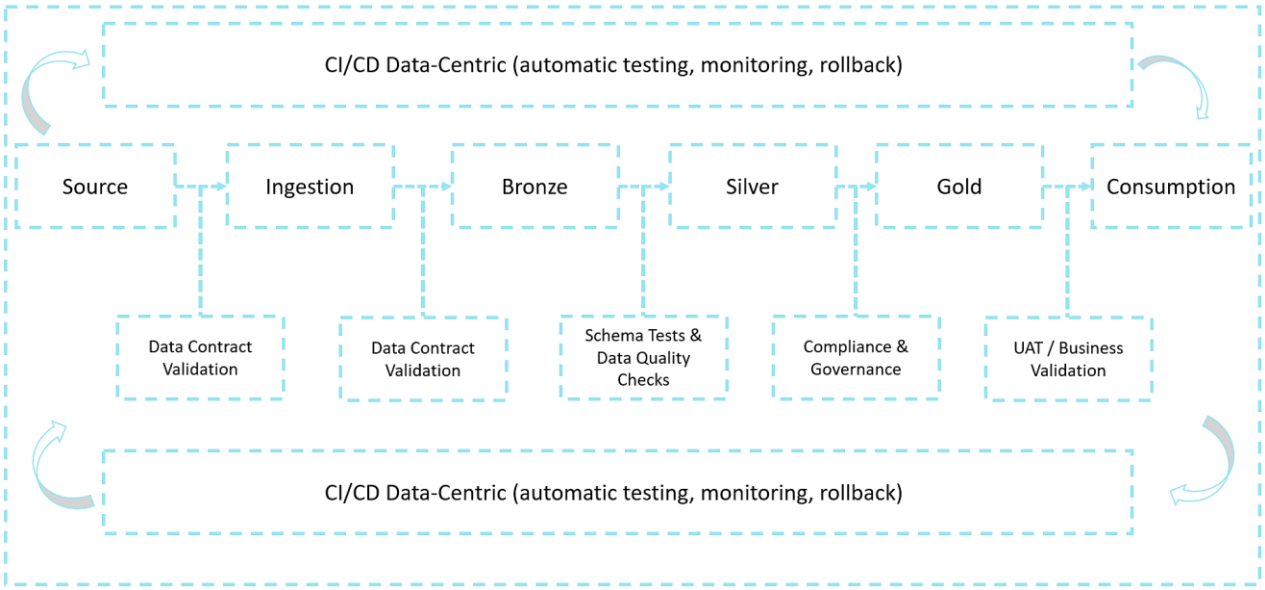

Figure 9. Data-Centric CI/CD pipeline with validation gates

5.7. Governance / Metadata

The Governance and Metadata phase ensures that data flowing through the pipeline is properly controlled, catalogued and classified, enabling traceability, compliance and security. This layer integrates governance into every pipeline stage. In a healthcare context, this is particularly critical for meeting regulatory requirements such as GDPR and HIPAA, which ensure that patient data is not only secure but also auditable and ethically managed.

To show how governance integrates seamlessly into the pipeline, Figure 10 explores the four fundamental governance capabilities, which are as follows: catalogue and lineage, followed by policy and access, as well as privacy and compliance, and quality with management. These capabilities are anchored at critical control points in the pipeline, supported by precise and well-defined RACI roles (owner, manager, consumer) and entry points (contracts, promotions, audits).

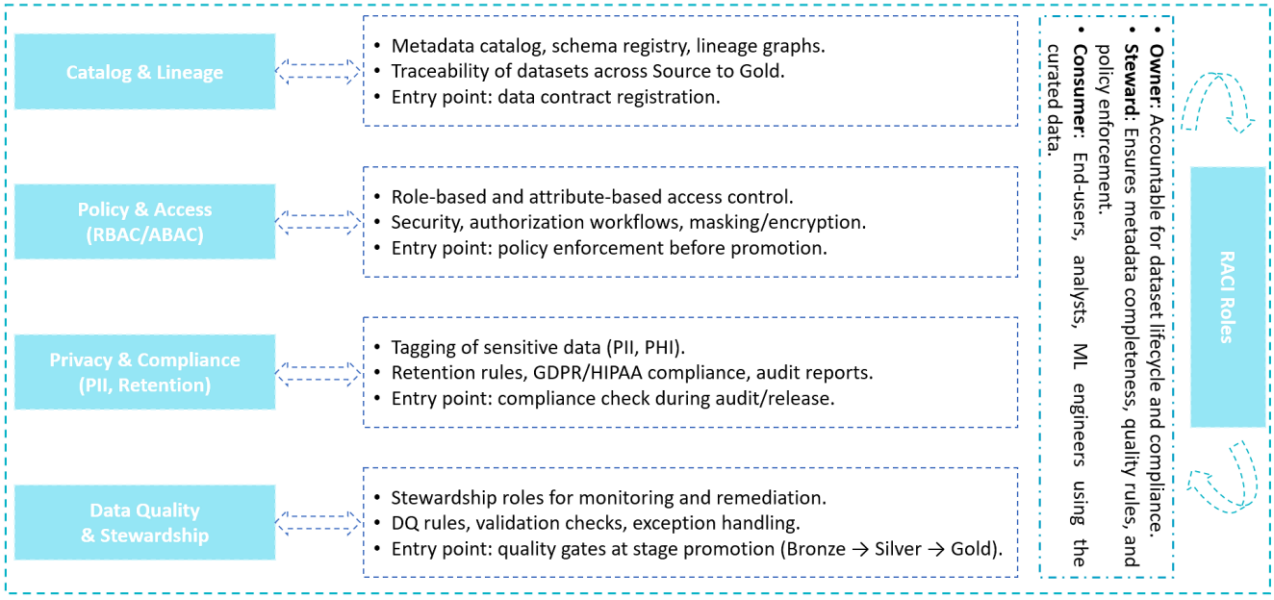

Figure 10. Governance framework highlighting four core capabilities

5.8. Governed Pipeline Orchestration Logic

To synthesize the interaction between the architectural phases described above, Algorithm 1 formalizes the end-to-end data lifecycle within the proposed framework. While Sections 5.1 through 5.7 detail the individual components, this section defines the dynamic execution flow and the enforcement of promotion contracts

By utilizing tool-agnostic pseudo-code, this logic serves as a blueprint for the automated orchestration and governance gates that regulate the transition from raw ingestion to AI-ready gold datasets. This procedural definition ensures that the architecture remains reproducible regardless of the underlying technical stack.

Algorithm 1. Governed DataOps Pipeline Orchestration

```
    Input: Multi-source Healthcare Data
    Output: Datasets for BI or AI-Ready Business, Auditable Lineage, and Performance Metrics
1   BEGIN
2       /* Stage 1: Ingestion & Data Contract Validation */
3       FOR each dataSource DO:
4           IDENTIFY source_type (Stream | Batch);
5           VERIFY data_contract (schema_registry, PII_tags)
6           IF contract_valid THEN
7               INGEST raw_data TO RAW/Bronze_Zone
8               GENERATE ingestion_manifest (timestamp, record_count)
9           ELSE
10              SEND TO Dead_Letter_Queue (DLQ);
11              TRIGGER alert ("Contract Violation")
12          END IF
13      ENDFOR
14      /* Stage 2: Validation & Metadata Tagging */
15      IF RAW_data_available THEN
16          EXECUTE data_cleaning (null_handling, type_casting)
17          RUN quality_checks (completeness, uniqueness, validity)
18          APPLY metadata_tags (GDPR/HIPAA compliance, sensitivity_level)
19          UPDATE lineage_graph (Atlas_Catalog)
20      END IF
```

```
    /* Stage 3: Storage & Transformation (Silver Layer) */
    IF quality_score >= 70% THEN
        TRANSFORM data (aggregations, joins, enrichment);
        CONVERT TO Delta_Format (ACID_compliance, schema_evolution)
        PROMOTE TO Silver_Zone;
    ELSE
        HALT promotion
        NOTIFY Data_Steward
    END IF
    /* Stage 4: Analytics & Machine Learning (Gold Layer) */
    IF Silver_data_ready THEN
        EXTRACT features (MLlib_processing)
        EXECUTE ML_model_training (versioned_experiment)
        VALIDATE compliance & reproducibility
        PUBLISH Serving_Views TO Gold_Zone
    END IF
    /* Stage 5: Monitoring & Governance Stage */
    LOG operational_KPIs (latency, throughput, resource_usage)
    AUDIT access_logs (Ranger_policies);
    IF anomaly_detected THEN
        TRIGGER auto_rollback()
    END IF
END
```

Promotion contracts in dynamic environments are enforced as executable controls rather than static documentation, each contract is versioned and linked to automated validation checks including: schema compatibility rules, mandatory field verification, data quality tests, metadata tagging and governance policies. When schema drift, or source evolution occurs, the pipeline distinguishes between compatible and breaking changes.

Compatible changes, such as the addition of optional fields, can be handled through schema evolution and contract versioning, however breaking changes, such as missing attributes, unexpected type changes, renamed fields or newly detected sensitive attributes, trigger a gate failure. The affected data is then, redirected to a quarantine zone or dead-letter queue while alerts are sent, to the responsible data steward or pipeline owner. Promotion to the next layer is suspended until the contract is updated, validated and approved. This mechanism ensures that evolving data sources, can be integrated, without compromising quality, traceability, compliance or downstream reliability.

5.9. Evaluation Framework: The DataOps Operational Excellence Index (DOEI)

To ensure that our reference architecture achieves its goals of the efficiency and reproducibility, we define a formal evaluation framework. Rather than observing isolated metrics, we define the DataOps Operational Excellence Index (DOEI), a composite score that synthesizes individual KPIs into a unified measure of pipeline maturity.

5.9.1. Key Performance Indicators (KPIs) Selection

The evaluation is built upon eight fundamental KPIs, as defined in Table 5. These metrics cover four critical dimensions:

- **Processing Efficiency:** Ingestion Latency, Ingestion Throughput, and Pipeline Throughput.
- **Data Governance & Quality:** Data Quality Score (DQS) and Alert Resolution Time.
- **Agility:** Deployment Frequency and Provisioning Time.
- **Resource Management:** Resource Utilisation Efficiency.

For clarity, the notation used in the KPI formulas is defined as follows: $n$denotes the number of observations or events over the evaluation window; $x$denotes the raw value of a KPI; $\hat{x}$denotes its normalized score on a $[0, 1]$scale; and *Baseline* denotes the reference value obtained from repeated pre-optimization runs under the same experimental conditions. In the DOEI formulation, $\bar{P}$, $\bar{G}$, $\bar{A}$, and $\bar{E}$ represent the normalized scores of the processing, governance, agility, and resource-efficiency dimensions, respectively, while ωdenotes the weight assigned to each dimension, with $\sum\omega = 1$.

Table 5. Key Performance Indicators (KPIs)

KPI	Definition (Formula)	Unit	Where Measured	Method (Eq.)	Observed Value (Mean ± Std)	Time Window	Baseline/ Reference
Ingestion Latency	$\frac{\sum_{i=1}^{n}(T_{ingest,i} - T_{source,i})}{n}$	ms	NiFi ingestion layer	(1)	≈ 9.8 ± 1.2 ms	25 days, ~3.5 GB data	< 50 ms
Ingestion Throughput	$\frac{Total\ Data\ Volume\ Ingested\ (MB)}{Total\ Processing\ Time\ (s)}$	MB/s	NiFi → HDFS	(2)	≈ 100 ± 5 MB/s	25 days	20–30 MB/s
Pipeline Throughput	$\frac{Total\ Records\ Processed\ (records)}{Total\ Processing\ Time\ (s)}$	records/s	Spark + Delta Lake	(3)	≈ 30,584 ± 820 records/s	25 days	10k–15k rec/s
Data Quality Score	$\omega_c.C + \omega_A.A + \omega_F.F + \omega_U.U$ where: • $C, A, F, U \in [0, 100]$ are respectively Completeness, Accuracy, Freshness, and Uniqueness • $\sum \omega = 1$	%	Silver transformation	(4)	≈ 97.87 ± 0.8 %	25 days	≥ 95%
Deployment Frequency	$\frac{Number\ of\ Successful\ Deployments}{Time\ Period}$	deployments/5d	GitLab CI/CD	(5)	≈ 3 ± 0.5	25 days (15 deploys)	1/week
Provisioning Time	$T_{end} - T_{start}$	min	IaC layer (Terraform/Ansible)	(6)	≈ 50 ± 3 min	10 runs	2–3 h
Alert Resolution Time	$\frac{\sum_{i=1}^{n}(T_{resolved,i} - T_{alert,i})}{n}$	min	ELK monitoring	(7)	≈ 12 ± 2 min	25 days	> 30 min
Resource Utilisation Efficiency	$\frac{Actual\ Resource\ Usage}{Actual\ Resource}$	%	Spark + Kubernetes	(8)	≈ 40 ± 4 %	25 days	30–35 %

Baseline values were established from repeated non-automated or pre-optimization runs conducted under the same experimental environment. For each evaluated task the baseline corresponded to the conventional execution setting used prior to the proposed framework, including the relevant preparation, configuration, execution and validation steps. Measurements were collected across repeated runs, and the reported reference values represent the average observed performance, expressed where applicable as mean ± standard deviation. These baseline references were then used to assess the relative gains achieved by the proposed automated and policy-driven approach.

It is important to clarify that the reported improvements were validated against a controlled pre-implementation baseline rather than multiple independent baseline systems. This baseline corresponds to the conventional non-automated execution setting used before deploying the proposed DataOps framework, under the same hardware, workload, and experimental conditions. The use of repeated baseline runs was intended to reduce measurement variability and isolate the performance gains attributable to the proposed automated, governed, and policy-driven architecture. Therefore, the reported improvements should be interpreted as within-environment gains over the pre-implementation setup, while comparison against multiple alternative DataOps architectures is identified as a relevant direction for future benchmarking studies.

5.9.2. Normalization and Composite Logic

To ensure mathematical consistency across heterogeneous units (ms, MB/s, %) each raw KPI value x is transformed into a normalized score $\hat{x} \in [0,1]$ relative to its baseline. We apply a scaling logic based on the nature of the metric:

For Higher-is-better metrics (like Throughput, Quality): $\hat{x} = \frac{x}{Basline}$

For Lower-is-better metrics (like Latency, Resolution Time): $\hat{x} = \frac{Basline}{x}$

To address the lack of standardized benchmarking in DataOps literature, the DOEI aggregates these metrics into a single reference value, each KPI x is first normalized $\hat{x}$ to a $[0, 1]$ scale relative to the baseline. The index is then calculated as a weighted sum:

$$DOEI = \omega_{efficiency} \cdot \bar{P} + \omega_{governance} \cdot \bar{G} + \omega_{agility} \cdot \bar{A} + \omega_{ressource} \cdot \bar{E} \quad (9)$$

where, $\sum \omega = 1$. For this healthcare implementation, weights are skewed toward Governance ($\omega = 0.40$) to prioritize clinical data safety and quality. This framework allows for a reproducible assessment that can be compared against future DataOps implementations.

6. Implementation

This section highlights the tool selection methodology, the translation of the conceptual framework into an operational pipeline, the deployment process, and the definition of the performance indicators that will subsequently be calculated.

6.1. Spine Model Methodology

In the context of the proposed DataOps architecture, tool selection is not a one-size-fits-all exercise, but a strategic process guided by the functional and organisational alignment of each component. DataOps, emphasises process design and continuous experimentation rather than rigid adherence to specific tools or platforms. The real value lies in how these tools support the broader ecosystem enabling collaboration, automation, scalability, and governance throughout the pipeline. To avoid the pitfalls of tool centric architectures, such as vendor lock in or poor interoperability, our approach follows a value driven logic based on the Spine Model introduced by [31] as illustrated on Figure 11. This hierarchical framework suggests that tools should be the last component defined, built upon clear practices, grounded in principles, aligned with organisational values, and responsive to concrete needs. This top-down awareness ensures that every technical choice made reinforces the strategic direction of the pipeline.

The Spine Model is also adaptable when organizational constraints limit tool choices such as in environments with legacy systems, existing vendor platforms or restricted infrastructure policies. In such cases, the model should not be interpreted as prescribing a fixed tool stack. Instead, it first identifies the required capabilities, values, principles, and practices and then maps them to the tools that are technically and organizationally feasible within the target context. Legacy systems can therefore be incorporated through connectors, APIs, wrappers or hybrid integration layers, provided that they support the required DataOps principles such as automation, traceability, governance, monitoring, and reproducibility. This makes the Spine Model useful not only for greenfield implementations but also for incremental modernization scenarios where organizations progressively align existing assets with practices of DataOps.

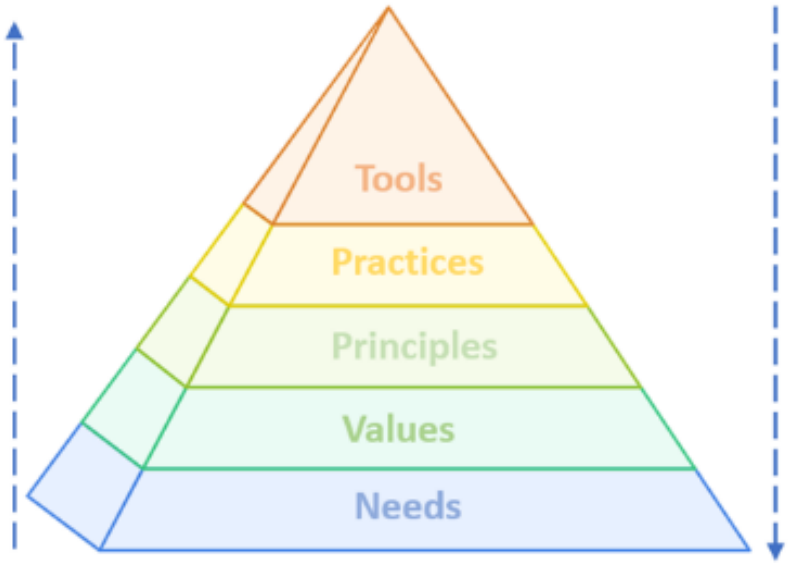

Figure 11. The Hierarchical Structure of the Spine Model

The practical application of this model to direct the specific tool selection for the suggested architecture is demonstrated in the section that follows.

6.2. Application of the Tool Selection Framework in Practice

Building on this hierarchical approach, this study places particular emphasis on selecting open-source tools for all components. Table 6 maps each component of the DataOps architecture according to the Spine Model aligning needs, values, principles, and practices with appropriate tools. It is important to note that each row of the table outlines a set of eligible tools capable of addressing the functional requirements of a specific pipeline layer. However, only one or a combination of these tools will be retained in the final architecture.

Table 6. DataOps Enabling Toolchain

	Need	Values	Principles	Practices	Tools
Data ingestion	Collect heterogeneous data from multiple sources. It should support both batch and real time flows.	We value openness, ease of use, auditability, and adaptability to different volumes and formats.	Visual or low code interfaces are encouraged. Tools must integrate well with the rest of the architecture. Containerisation is required.	We use declarative, versioned pipelines. Quality checks are applied early. Failures are monitored. Data is standardised before processing.	• Apache NiFi • Apache Kafka • Apache Flume • Logstash
Data orchestration	Coordinate complex workflows across ingestion, transformation, and analysis stages. Handle task dependencies, retries, scheduling, and conditional execution.	We value automation, repeatability, observability, and flexibility in pipeline management.	Orchestration must support modular, reusable. It must integrate with and easy to monitor.	Workflows are versioned and executed on schedule or event. Logs and status are tracked. Orchestration is decoupled from data logic.	• Apache Airflow • Prefect • Dagster
Version Control	Track changes to pipeline code, configurations and infrastructure scripts. Enable rollback, collaboration and auditability across teams.	We value transparency, reproducibility, team collaboration and traceability of every change.	Use distributed, open-source systems with CI/CD integration. Must support branching, tagging, and history tracking.	Every change is committed and reviewed. Branching strategies and pull requests are used. Integrated with pipeline automation.	• GitLab • GitHub • Bitbucket
Containerisation	Ensure portability, scalability, and environment consistency across all pipeline components and stages.	We value reproducibility, resource efficiency, isolation, and flexibility across development and production environments.	Prefer open-source tools with strong ecosystem support. Must integrate with orchestration and IaC. Should enable fast deployment and easy scaling.	Each service runs in its own container. Images are versioned. Containers are deployed via CI/CD and scaled automatically.	• Docker • Podman • LXC
Scalability	Scale processing and storage resources automatically based on workload demand. Support elastic infrastructure and workload distribution.	We value elasticity, high availability, performance under load, and cost efficiency.	Use open-source solutions that support horizontal scaling. Must integrate with containerisation and orchestration systems.	Workloads are distributed across nodes. Scaling is automated. Resources are monitored and adjusted dynamically.	• Kubernetes • Docker Swarm • Apache Mesos
Security	Control access to data assets across the pipeline. Enforce fine grained security policies and ensure auditability of data usage.	We value compliance, transparency, traceability, and security across all data layers.	Access control must be centralised, role based, and adaptable to different tools (Spark, Hive, HDFS...). Open source and enterprise grade is preferred.	Policies are defined by role or group. Access logs are stored for audits. Integration with metadata and lineage tools is required.	• Apache Ranger • Apache Sentry
Governance	Ensure data traceability, compliance, and controlled access across the entire pipeline. Provide metadata management, lineage tracking, and role-based access control.	We value transparency, auditability, compliance with data regulations, and secure data access.	Use open source, scalable tools supporting integration with big data platforms. Data lineage and metadata must be automated. Access control should be centralised and role based.	Metadata is captured at each pipeline stage. Policies are defined per role. Lineage is visualised for audit. Security and governance are embedded via APIs and UI based interfaces.	• Apache Atlas • DataHub
Infrastructure as Code	Automate infrastructure provisioning and configuration. Ensure consistency across environments and support versioning of infrastructure changes.	We value repeatability, modularity, security, and traceability of infrastructure deployment.	Infrastructure must be versioned, tested, and integrated into CI/CD workflows. IaC must be cloud agnostic and support hybrid deployments.	Infrastructure is defined as code, stored in Git, and deployed through pipelines. Configuration is automated. Secrets are managed securely.	• Terraform • Ansible • Pulumi
Monitoring	We value transparency, real time feedback, anomaly detection, and fast incident resolution.	Use open-source tools with modular dashboards. Monitoring must be centralised and integrated with CI/CD and orchestration layers. Logs and metrics must be query able and visualizable.	Monitoring must be end-to-end, centralised, and actionable.	All pipeline components emit logs and metrics. Dashboards are built for key KPIs. Alerts are configured for failures or threshold breaches.	• ELK Stack • Prometheus
Data Processing and Data Lakehouse	Enable unified storage and efficient processing for raw and structured data. Support both batch and streaming, with ACID transactions, schema evolution, and analytics integration.	Emphasise scalability, cost efficiency, and flexibility. Ensure consistency and support for diverse analytics use cases.	Adopt open formats with separation of storage and compute. Ensure compatibility and reliable transaction handling.	Use layered zones (raw, curated, structured). Support time travel, schema evolution, and versioning. Pipelines are scheduled and support joins, filtering, and aggregations.	• Delta Lake • Apache Iceberg • Apache Hudi • Apache Spark • Apache Flink • HDFS • Apache Hive
Data Visualisation, BI, and AI	Deliver actionable insights through interactive dashboards, reports, and machine learning outputs. Support both business intelligence and AI driven analytics for decision making.	We value interpretability, accessibility of insights, performance, and integration with upstream data workflows.	Use open source, scalable tools that support modern BI and data science. Must connect directly to the data Lakehouse and offer real time or near real time interaction.	Dashboards are automatically refreshed. Visualisations are connected to structured layers of the Lakehouse. ML models are trained and served within the same pipeline.	• Apache superset • Apache Spark MLlib • Metabase • MLflow

The proposed pipeline integrates a curated set of open-source tools, selected for their cost efficiency, flexibility and strategic alignment with key DataOps principles such as automation, scalability, governance, and end to end observability, as illustrated in Figure 12. The architecture is structured into modular layers ranging from data acquisition to insight generation supported by infrastructure components that enhance resilience, adaptability and transparency. This toolset translates the conceptual architecture outlined into a concrete operational and robust implementation as detailed below.

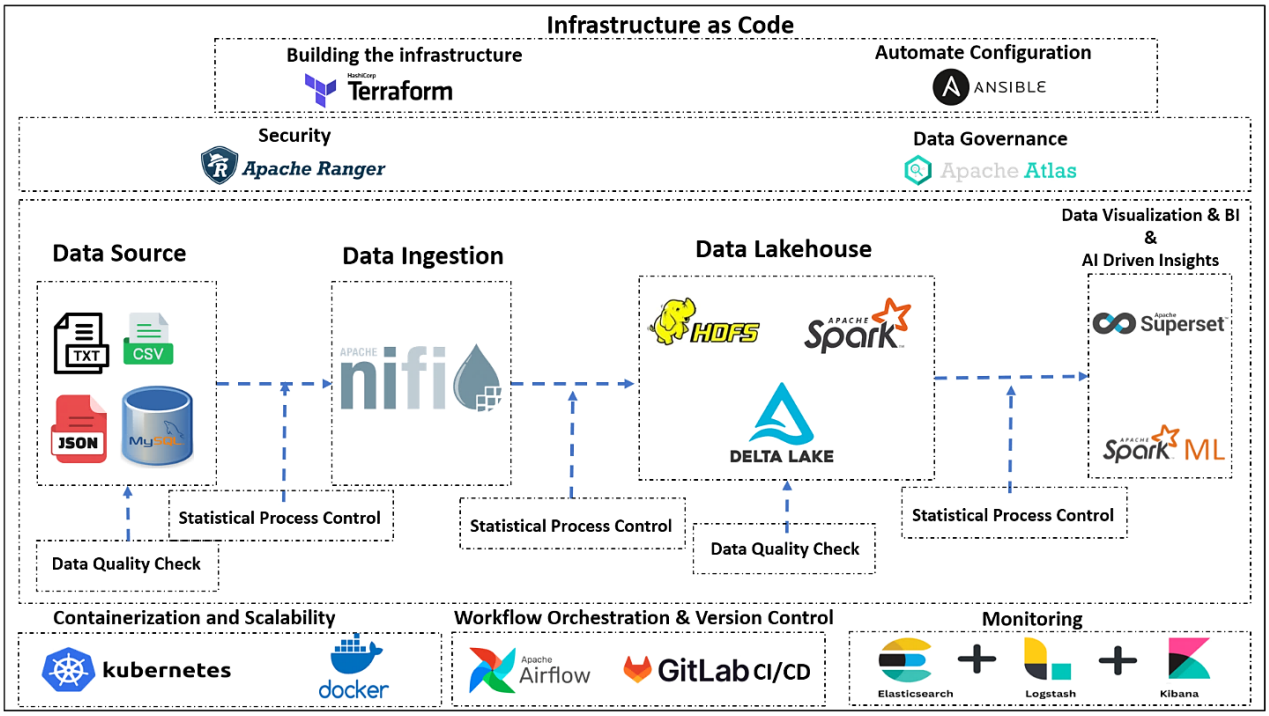

Figure 12. DataOps Pipeline Architecture: Tools and Process

Data ingestion in the proposed pipeline is ultimately managed by Apache NiFi, selected as the most suitable tool after evaluating all choices through the Spine Model framework. While Apache Kafka, Apache Flume, and Logstash are recognised in data ingestion, NiFi supports both streaming and batch ingestion, fine-grained flow control and built-in governance features. It also supports data routing, transformation, enrichment and metadata tagging natively, which is essential for heterogeneous healthcare datasets. Unlike Kafka, which excels in high-throughput streaming but requires additional components for transformation and metadata tracking, NiFi offers visual low-code orchestration with its intuitive GUI that accelerates pipeline deployment and maintenance. These capabilities make it a strong choice for the ingestion layer. Compared to Flume, which is mostly optimised for log collection and lacks versatile processors, NiFi ensures bidirectional data flows, backpressure handling and data provenance tracking, enabling full observability and auditability critical in regulated sectors like healthcare. Lastly, while Logstash is powerful at log ingestion, it lacks NIFI's flow-based orchestration and dynamic prioritisation features that NiFi offers. These characteristics, combined with native integration with the Hadoop ecosystem, containerised deployments and security support make it the most robust choice for the ingestion layer [32].

HDFS, Apache Spark and Delta Lake are combined in the Data Lakehouse layer to provide scalable storage, fast processing, and dependable governance. In contrast to Hive, which is restricted to structured data and does not have native ACID transactions, HDFS is chosen as the storage backbone because it can handle both structured and unstructured data. While Apache Flink is primarily designed for streaming and less appropriate for intricate analytical pipelines, Apache Spark is used as the unified batch and streaming engine, offering support for ETL and ML.

Because Delta Lake ensures data consistency and regulatory compliance in the healthcare industry through native Spark integration, ACID transactions, schema evolution and time travel, it is preferred over Hudi and Iceberg [33, 34]. Apache Superset and Spark MLlib are chosen for the analytics and machine learning stage in order to guarantee interactive visualisation, sophisticated analytics, and smooth pipeline integration. While Metabase is less scalable, Apache Superset provides richer visualisation options, interactive dashboards based on SQL, and native integration with big data backends for complex, multi-source healthcare data. Because Spark MLlib enables distributed in-memory computation on large datasets without requiring additional data movement, it is selected for training, evaluating, and deploying ML models directly within the Spark ecosystem. MLlib is the ideal option for healthcare DataOps pipelines because of its scalability, speed, and native Spark compatibility. MLflow is helpful for model tracking and lifecycle management, but it lacks native distributed training and would require additional integration [35].

Apache Ranger is selected to enforce security and fine-grained data governance because it provides centralised policy management, dynamic role-based access control (RBAC), and integration with the Hadoop ecosystem. Ranger supports multi-component coverage (HDFS, Hive, HBase, Kafka, and more) and provides audit logging, REST APIs, and tag-based policies, in contrast to Apache Sentry, which is mainly concentrated on SQL-based authorisation in Hive and Impala. Because of these features, Ranger is more complete, scalable, and appropriate for a diverse healthcare DataOps

pipeline where traceability and regulatory compliance are crucial [36]. For data classification, lineage tracking, and metadata management all essential for healthcare compliance and auditability Apache Atlas is chosen. Atlas provides fine-grained lineage tracking, policy-based classification, and native Hadoop ecosystem integration that seamlessly integrate with Apache Ranger for governance, in contrast to DataHub, which is superior at metadata discovery and search. Atlas is the go-to option for creating a traceable and legally compliant DataOps pipeline because of these features. On the other hand, supporting the scalability and deployment flexibility the use of Kubernetes for orchestration and scalability and Docker for lightweight containerisation, offering elastic resource management, automated deployment, and portability. Docker is favoured over Podman and LXC. Because of its robust auto scaling, fault tolerance, and rolling updates, as well as its robust community and enterprise adoption. Kubernetes is preferred for orchestration over Docker Swarm or Apache Mesos [37].

To ensure repeatable and auditable pipeline execution, Apache Airflow is chosen for workflow orchestration due to its robust task dependency management, rich integration with big data tools, and sophisticated DAG-based scheduling. While Dagster focuses on data-aware pipelines and Prefect provides a more Pythonic interface, Airflow is more tried-and-true due to its robust scheduling, strong community support, and native integration with Spark, Hadoop, and Kubernetes, making it the most dependable option for intricate, multi-stage DataOps workflows [38], On the other hand, for CI/CD, Because GitLab CI/CD offers a fully integrated DevOps platform with integrated CI/CD pipelines, a container registry, and robust Kubernetes/Docker integration, it is chosen to automate continuous integration and delivery. GitLab provides smooth on-premise deployment, granular access control, and strong artifact management in contrast to Bitbucket Pipelines, which is less adaptable for complex self-hosted environments, and GitHub Actions, which is primarily cloud-centric. It is the most dependable option for automating deployments in a safe healthcare DataOps pipeline because of these features [39].

To ensure consistency and repeatability across environments, the pipeline uses Ansible for automated configuration management and Terraform for infrastructure provisioning. Ansible is superior in agentless configuration, deployment automation, and CI/CD integration, whereas Terraform is favored for its declarative methodology, extensive multi-cloud support, and strong state management. Although Pulumi is code-centric and modern, it adds complexity and is less popular in hybrid on-premise setups. For scalable, secure, and repeatable DataOps infrastructures, the Terraform + Ansible combination is the most reliable and popular option [40]. Throughout the pipeline, the ELK Stack is used to offer real-time anomaly detection, centralised logging, and end-to-end monitoring. Kibana provides interactive dashboards for analytics and visualisation, Logstash manages log aggregation and transformation, and Elasticsearch guarantees quick indexing and search. Prometheus performs exceptionally well in metrics-based monitoring and alerting, but it struggles to manage and query unstructured logs, which are crucial in diverse healthcare pipelines. In order to ensure traceability, troubleshooting, and compliance, the ELK Stack provides a more complete and log-focused observability solution [41]. All these tools form a cohesive backbone empowers the pipeline to operate with transparency, efficiency, and resilience across its entire lifecycle. To consolidate the description of the health data pipeline, Table 6 provides a summary of the main components deployed in the infrastructure, including ingestion, storage, orchestration, governance, and analysis.

Having defined a coherent set of tools aligned with architectural objectives and operational requirements, the aim of the next section is to translate this selection into a functional deployment, detailing how these tools are used and integrated with a structured and repeatable implementation strategy.

6.3. Deployment and Runtime (IaC, Containers, Kubernetes)

Building a functional pipeline starts with setting up the infrastructure, where tools such as Terraform and Ansible play a fundamental role in automating the configuration and ensuring that the environment is consistent and well prepared. To ensure modularity of the Infrastructure as Code scripts, setup for the proposed DataOps pipeline is organised into a structure under the root directory “DataOps IaC/” as illustrated in Figure 13, with separate directories for “Terraform/” and “Ansible/”. Since the infrastructure relies on a preexisting local server, Terraform is used not to provision the hardware, but to declare and expose infrastructure metadata such as the server IP address, SSH credentials and connection parameters. They are defined in variables.tf and populated with “terraform.tfvars” and exported using “outputs.tf”. The “main.tf” file is used as a placeholder for infrastructure state and version management resources. Terraform loads and processes all ".tf" files in the same directory as a single configuration unit by generating unified execution script.

The “Ansible/” directory manages post deployment configuration. It is composed of an “inventory.ini” file that references the IP address and SSH user of the target server written manually, the hosts are grouped under a class called [all], which serves as a target classification for automation scripts. This modular structure facilitates the addition of new resources on the need and allows the easy targeting of specific servers based on their assigned group. The “setup.yml” file is where the Ansible playbook install Docker, Kubernetes and GitLab Runner. Both the Ansible and Terraform files are integrated with GitLab for version control.

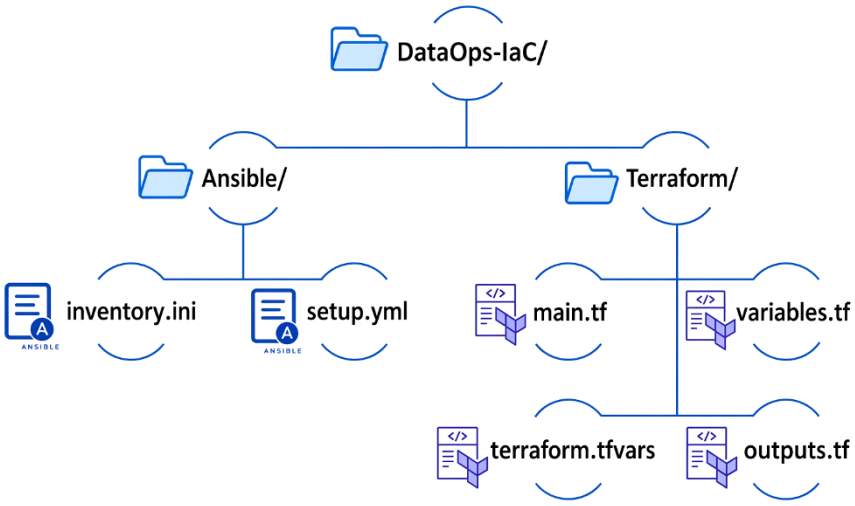

Figure 13. DataOps IaC: Terraform & Ansible

In addition to the previous installations, the setup.yml file is responsible for launching the environment using the "docker-compose.yml" script. After installing the required dependencies, the playbook copies the necessary runtime files including "docker-compose.yml", ".env", and Dockerfiles, as detailed in Figure 14 from the local machine to the target server. It then triggers the deployment of services with Docker Compose, which builds the Docker images from their respective Dockerfiles and orchestrates the containerised components. The "docker-compose.yml" file defines the configuration of each container, including network connections and exposed ports.

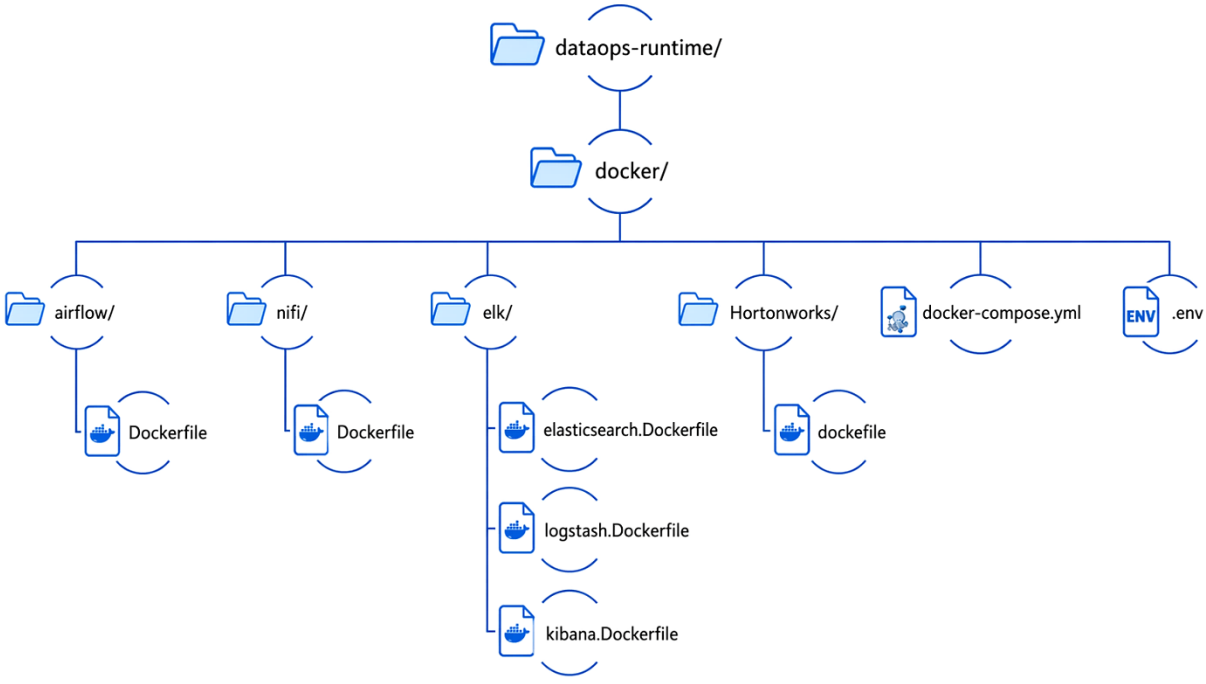

Figure 14. Containerisation Setup for DataOps Architecture

And to support scalability and fault tolerance of the DataOps architecture, a Kubernetes based deployment strategy is implemented and organised under the "DataOps runtime/" directory, as shown in Figure 15. This setup includes two main components: the "kubernetes/" directory for deployment scripts and the volumes/ directory for persistent storage management. Each script within the "kubernetes/" folder serves in managing containerised services. The "deployment.yaml" file defines the pods and replica sets for each microservice, enabling horizontal scaling by adjusting the number of replicas. The service.yaml file exposes these deployments internally within the Kubernetes cluster or externally via NodePort or LoadBalancer, depending on the configuration. To handle external access, routing, and load balancing, the ingress.yaml file is used to define ingress rules for HTTP(S) traffic, ensuring seamless access to services such as Apache Airflow and Superset. Configuration parameters for environment variables, default settings, or tool specific properties are managed using configmap.yaml, while sensitive credentials such as tokens or passwords are securely stored in secret.yaml.

The "volumes/" directory provides separated persistent storage for each service. It includes subdirectories “nifi data/”, “hdfs data/”, “airflow logs/” and “elk storage/”, ensuring data durability and container state persistence during service and pod rescheduling.

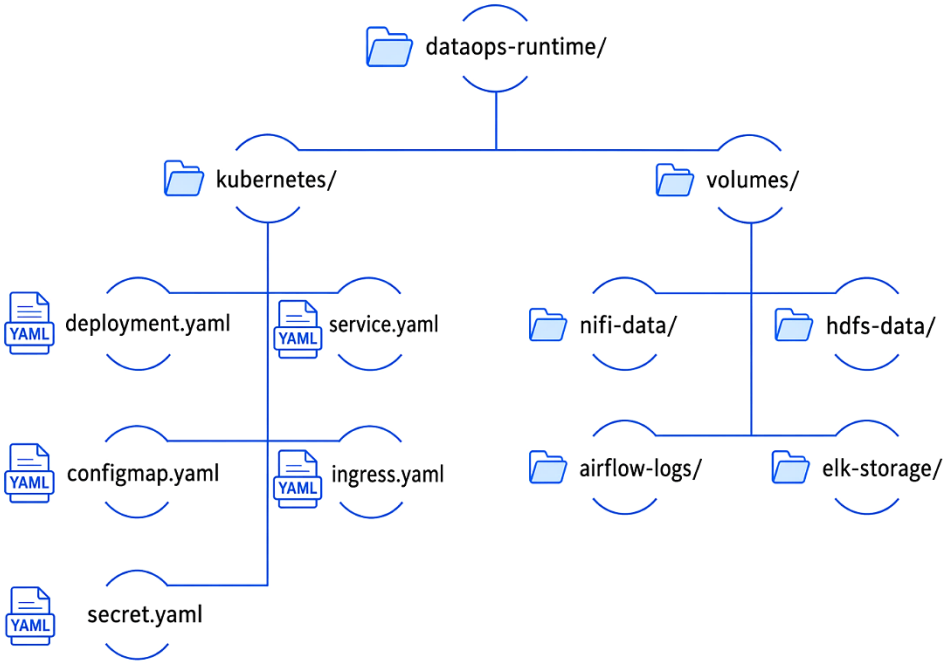

Figure 15. Scalability and Volume Setup for the DataOps Architecture

6.4. Workflow Orchestration (Airflow / DAGs)

After the infrastructure and deployment layers are configured, the orchestration of the workflow is managed by Apache Airflow, where each stage of the pipeline is modelled as a Directed Acyclic Graph (DAG) as illustrated in Figure 16.

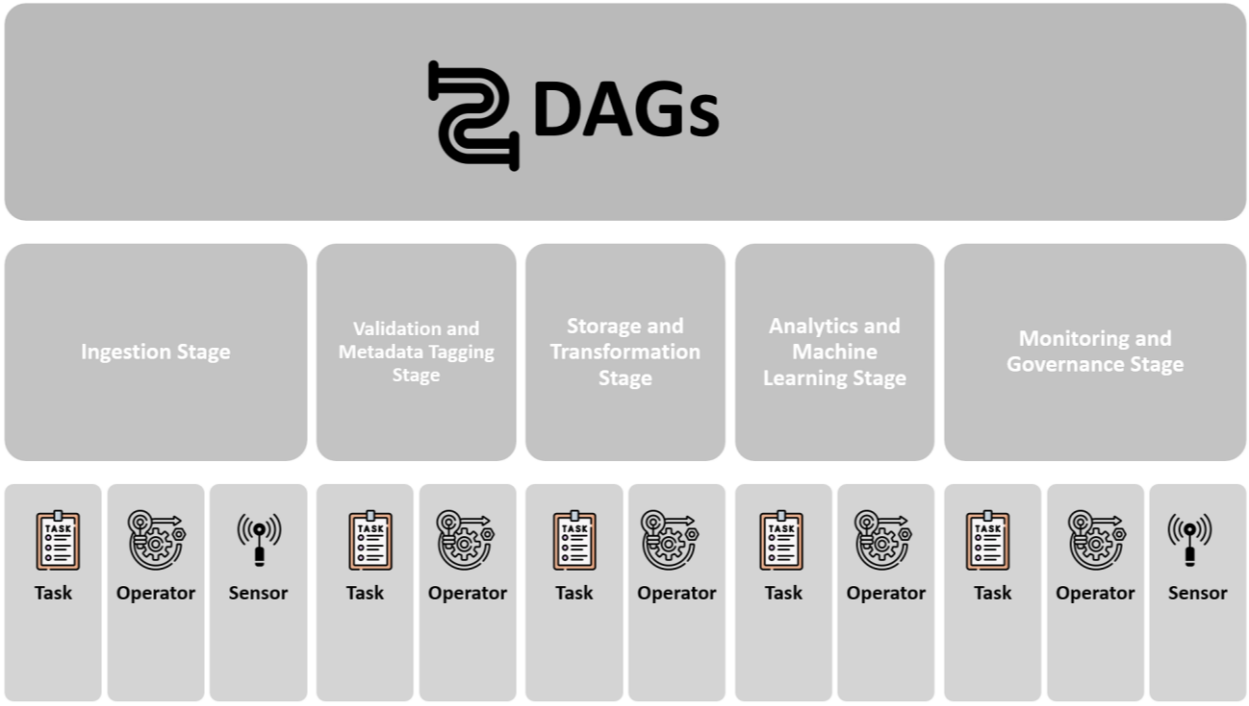

Figure 16. Orchestration of the data pipeline with Apache Airflow

To capture this orchestration more systematically, the pipeline has been divided into five stages Ingestion, Validation & Metadata Tagging, Storage & Transformation, Analytics & Machine Learning, and Monitoring & Governance as summarized in Table 7.

To make the behaviour of these stages explicit and reproducible, the dataset lifecycle implemented in the proposed architecture is formalised below using tool-agnostic pseudo-code. This specification abstracts from implementation describes the workflow that governs all pipeline across the Lakehouse layers.

Table 7. Orchestration of the Data Pipeline with Apache Airflow

Stage	Tasks	Operators	Sensors	Gate
Ingestion Stage	Load raw data, schema validation	PythonOperator, BashOperator	FileSensor, HttpSensor	Data contract validation
Validation & Metadata Tagging	Data cleaning, tagging, DQ checks	PythonOperator, CustomOperator	ExternalTaskSensor	Schema tests & quality checks
Storage & Transformation Stage	Transformations, aggregations	SparkSubmitOperator, SQLExecOp	TimeSensor	Promotion to Silver layer
Analytics & Machine Learning	Model training, feature extraction	MLflowOperator, PythonOperator	ExternalTaskSensor	Compliance & reproducibility validation
Monitoring & Governance Stage	Logs, alerts, lineage, auditing	EmailOperator, SlackAPIOperator	S3KeySensor, DatabaseSensor	Governance & compliance checks before output

With the implementation strategy and tools now in place, the next section presents the real-world application in the healthcare sector

7. Healthcare Domain Application

7.1. Motivation

The growing complexity and volume of health data, combined with the urgency of rapid and accurate decision-making, make this field a suitable validation environment for the proposed solution. More specifically, three factors points that justify the choice of this sector as a validation environment are as follows: firstly, the highly sensitive nature of medical data, which requires strict compliance with privacy regulations such as GDPR and HIPAA and the protection of personally identifiable information (PII); secondly, the heterogeneity of data sources, ranging from structured electronic health records (EHR) and laboratory results to semi-structured imaging reports and continuous signals, making integration and interoperability a major challenge; and third, the need for near real-time decision support in clinical contexts, where latency in data processing has a direct impact on patient care outcomes. By addressing these challenges, the proposed architecture not only validates its robustness but also demonstrates its potential to improve efficiency in critical operations in this sector.

7.2. Data Characteristics

The healthcare dataset used for this study simulates real data flows from multiple sources in the medical ecosystem. It includes data generated by patients, doctors, hospitals, and medical laboratories. This multi-source data demonstrates the heterogeneity and complexity of this sector, making it ideal for validating the solution's performance. The characteristics of these sources including their formats, volumes, frequencies, sensitivity levels, and representative fields are summarised in Table 8, which highlights the diversity and compliance constraints that the pipeline must handle to ensure quality, governance, and analytical readiness.

Table 8. Dataset overview

Source	Format	Volume	Frequency	Sensitivity / PII	Examples of fields
Hospital (EHR)	CSV / DB	1.4 GB	Daily	PII (Names, IDs)	encounter_id, diagnosis_code
Laboratory (Lab)	CSV	0.9 GB	Hourly	Pseudo-PII	test_code, result_value
Imaging	Files	0.7 GB	Weekly	High (linked to patient ID)	modality, path
Patient App	CSV / JSON	0.5 GB	Stream	Medium (sensor + biometric)	device_id, vital_sign

7.3. Experimental Setup

To validate the pipeline, it was deployed on a server equipped with an Intel Xeon W3-2423 processor (6 cores, up to 4.2 GHz Turbo, 120 W), 32 GB DDR5 ECC memory (4800 MT/s), a 1 TB high-performance SSD and an NVIDIA RTX A1000 GPU (4 GB GDDR6) to support parallel processing and machine learning workloads. The system operated as a single-node Kubernetes cluster where the control plane and worker node were co-located on the same machine. All pipeline components were containerised and executed as Kubernetes pods, with persistent storage backed by the local SSD. The server was connected via a 100 Mbps fibre network link, ensuring stable data transfer performance during ingestion and processing. The workload spanned approximately 25 days, totalling approximately 3.5 GB of multi-source healthcare data.

The proposed architecture was designed with scale-out deployment in mind. Although the empirical validation was conducted in a controlled healthcare testbed, the main components of the pipeline, including Apache NiFi, Apache Spark, Delta Lake/HDFS, Kubernetes, Airflow, Apache Atlas, Ranger, and the ELK stack, are compatible with distributed and horizontally scalable deployments. Therefore, extending the pipeline to TB-scale workloads or multi-hospital systems would not require a redesign of the architecture, but rather an adaptation of the deployment configuration, including multi-node Kubernetes clusters, distributed Spark execution, partitioned Lakehouse storage, replicated ingestion services, and optimized metadata and governance services. In such settings, performance would depend on cluster sizing, data partitioning, workload concurrency, network bandwidth, and governance-policy overhead. This makes large-scale deployment technically feasible within the proposed framework, while future work will further quantify its performance under TB-scale and multi-site healthcare conditions.

Figure 17 provides, the process begins with environment setup, where Terraform and Ansible are employed as scripting tools for infrastructure provisioning and automated configuration. This setup initiates the installation of key dependencies, including Docker, Kubernetes, and the GitLab Runner agent, which is responsible for executing CI/CD jobs. The remaining tools are then pulled and configured using Docker Compose and Dockerfile scripts. At this stage, each tool is containerised and deployed with duplication for good scalability, and each is contained in a Kubernetes pod. Finally, the data pipeline workflow is orchestrated using Apache Airflow, where DAGs define all tasks and their interdependencies, ensuring a modular, automated, and scalable execution of the pipeline.

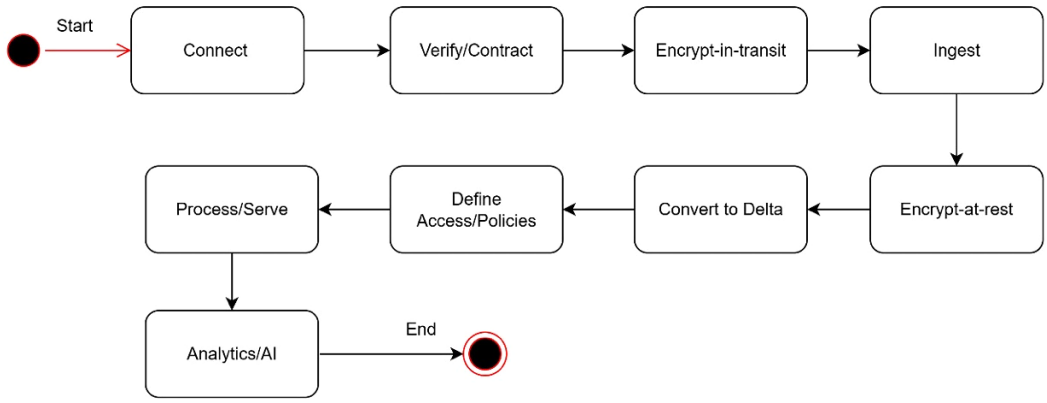

Figure 17. Secure and Optimised Data Pipeline Flow for Analytics and AI

7.4. Results: Performance Synthesis via the DOEI Framework

To provide the answer to RQ3, we synthesize the empirical measurements using the DOEI framework defined in Section 5.9. This approach allows us to evaluate the pipeline not as a set of isolated metrics, but as a balanced and governed system.

- ***KPI Normalization and Dimension Scoring***

We apply the normalization logic to the observed values (Table 5) against task-specific baselines derived from repeated pre-implementation runs under the same experimental environment. Table 9 a summarizes the transition from raw operational data to normalized dimensional scores ($\bar{P}$, $\bar{G}$, $\bar{A}, \bar{E}$).

Table 9. Observed KPI Normalization and Dimensional Performance

Dimension	Key Performance Indicators	Observed Value (Mean)	Baseline	Normalized Score
Governance (G)	Data Quality Score (DQS)	97.87%	$\geq$ 95%	0.98
	Alert Resolution Time	12 min	$>$ 30 min	
Processing (P)	Ingestion Throughput	100 MB/s	25 MB/s	0.91
	Pipeline Throughput	30,584 rec/s	12.5k rec/s	
	Ingestion Latency	9.8 ms	$<$ 50 ms	
Agility (A)	Deployment Frequency	3 / 5 days	1 / week	0.88
	Provisioning Time	50 min	150 min	
Efficiency (E)	Resource Utilisation Efficiency	40%	32.5%	0.85

- ***Composite DOEI Calculation***

Using the weighted formula established in Section 5.9.2, and because no standardized weighting scheme currently exists for composite DataOps evaluation, the DOEI weights were derived through a lightweight expert elicitation adapted to the healthcare context. A small consultation was conducted with domain-relevant stakeholders involved in pipeline design, governance, and operations. Participants were asked to rate the relative importance of the four DOEI dimensions governance, processing, agility, and resource efficiency for a regulated healthcare data pipeline. The normalized average ratings resulted in the following weights: governance = 0.40, processing = 0.25, agility = 0.20 and efficiency = 0.15. The higher weight assigned to governance reflects the centrality of compliance, data quality, traceability, and patient-data protection in healthcare environments.

The final index is calculated as follows:

$$DOEI = (0.40 \times 0.98) + (0.25 \times 0.91) + (0.20 \times 0.88) + (0.15 \times 0.85) = 0.923$$

- ***Results Interpretation***

The obtained DOEI value of 0.923 indicates that the proposed governed Lakehouse DataOps architecture achieved a high level of operational maturity under the controlled healthcare testbed conditions. More importantly, this result should not be interpreted only as an aggregate performance score, but as evidence of a balanced trade-off between processing efficiency, governance enforcement, deployment agility, and resource utilisation. In regulated healthcare environments, such a balance is particularly important because high throughput alone is insufficient if it is obtained at the expense of traceability, data quality, or compliance control.

The governance dimension achieved the highest normalized score (0.98) due to the high data quality score of 97.87% and the reduced alert resolution time, this result suggests that the integration of data contracts, promotion gates, metadata tagging, lineage tracking and access control mechanisms was effective in maintaining data reliability across the pipeline. The result indicates that governance was not implemented as an external verification applied after processing but as an embedded mechanism operating across the Bronze to Gold stages. This is relevant, in healthcare where datasets may contain sensitive identifiable information.

The processing dimension, also showed strong performance, with an ingestion throughput, of approximately 100 MB/s and a pipeline throughput exceeding 30,000 records/s, these values indicate that the introduction of governance controls and validation gates did not create a major operational bottleneck. This is an important finding because one of the common concerns in data architectures, is that compliance and quality checks may slow down the pipeline. The results suggest that when governance rules are integrated into the pipeline design rather than added as isolated post-processing controls but it is possible to preserve both operational velocity and data trustworthiness.

The agility score of 0.88 reflects the contribution of Infrastructure as Code and CI/CD automation to deployment. The reduction of provisioning time, from approximately three hours, to around 50 minutes indicates that environment setup, configuration and deployment became reproducible and less dependent on manual intervention. This improvement is operationally meaningful, because healthcare data platforms often require controlled, repeatable and auditable deployment. By reducing manual configuration effort, the proposed architecture limits configuration drift, improves reproducibility, and supports faster adaptation to pipeline updates.

The resource-efficiency dimension, obtained a normalized score of 0.85, which indicates a stable but still improvable level of resource utilisation. The observed utilisation level suggests that the architecture avoided excessive resource saturation while maintaining sufficient capacity for ingestion, transformation, monitoring, and governance services. However, this result also highlights a potential area for future optimisation particularly through more advanced autoscaling strategies based on real-time workload indicators, such as ingestion latency, throughput variation and CPU or memory pressure.

Overall, the results provide empirical support for the central assumption of this study: a DataOps architecture, can combine automation, Lakehouse-based storage, governance and reproducible deployment without necessarily compromising operational performance. The findings also answer RQ3 by showing that the proposed architecture improves operational performance not through a single isolated component, but through the combined effect of data contracts, promotion gates, CI/CD automation, Infrastructure as Code, monitoring, and governed Lakehouse progression. The validation was conducted in a controlled healthcare testbed; consequently, the findings provide robust within-environment evidence of the architecture's operational effectiveness. Further evaluations on larger-scale, and multi-site healthcare deployments would help assess the generalisability and robustness of the observed gains under broader operational conditions.

- ***Weight Sensitivity and Generalizability***

To assess the sensitivity of the DOEI score to weight selection, we performed a simple scenario-based sensitivity check. The healthcare-oriented weighting scheme gives higher importance to governance because compliance, traceability, and data quality are central in clinical environments. However, when equal weights are assigned to the four dimensions, the DOEI remains high at 0.905. Alternative weighting scenarios prioritizing processing efficiency, agility, or resource efficiency also keep the score within a narrow high-performance range. This relative stability is explained by the fact that all four normalized dimensions obtained strong scores, ranging from 0.85 to 0.98. Therefore, moderate changes in the weighting scheme do not substantially alter the overall interpretation of the pipeline's operational maturity.

The DOEI is not intended to be limited to healthcare. Its dimensions, are generic enough to be applied to other DataOps contexts, including finance, manufacturing, public administration or industrial IoT. However, the weighting scheme should be adapted to the priorities of each domain. For example, a financial or healthcare setting may assign greater weight to governance and compliance while an industrial IoT environment may prioritize processing throughput and latency and a business intelligence context may emphasize agility and deployment frequency. Thus, the DOEI should be interpreted, as a configurable evaluation framework rather than a fixed healthcare-specific metric.

7.5. Comparative Evaluation of DataOps Solutions

The comparative evaluation should be interpreted in light of the current maturity of the DataOps literature. Although comparative assessment is expected in engineering-oriented studies, the reviewed corpus does not yet provide a stabilized reference pipeline, a shared benchmarking protocol, or a consensual set of pipeline-level metrics that would enable strict value-to-value comparison across studies. This limitation is even more pronounced in healthcare, where only a very small number of contributions report concrete implementations, and where the available studies differ substantially in scope, objectives, architectural depth, and reported outcomes. As a result, the comparison in Table 10 is necessarily focused on architectural completeness, implementation depth, governance coverage, reproducibility, and evaluation logic rather than on direct one-to-one numerical benchmarking. Far from weakening the analysis, this situation highlights a structural gap in the field and justifies the introduction of the DOEI framework in this study as an initial reference for future cumulative and quantitatively comparable DataOps evaluations.

Table 10. Comparative analysis

Aspect	Our Contribution	Contribution [15]	Contribution [18]
Scope	Ingestion proposes an end-to-end DataOps framework that includes an architecture, a technological stack, and an industrial case for practical validation.	lacks a unified architecture and instead concentrates on a broad review of DataOps adoption and challenges with illustrative use cases.	Presents a toolbox solution for continuous integration of heterogeneous edge–cloud devices and validated through multiple use cases including.
Methodology	presents the SPINE Model, a structured tool selection methodology, and uses it to build and execute a full pipeline.	explains the trends in tool adoption without providing a standardised methodology or selection procedure.	Uses a declarative, rule-based approach where mapping rules are separated from execution and composed into modular pipelines.
Technical Depth	offers performance metrics, implementation strategies, and a thorough architecture. comprises KPIs and tangible outcomes.	discusses high-level issues and advantages without providing an implementation architecture or a quantitative performance evaluation.	Provides implementation details on Camel-based pipelines, Chimera KG operations, and pipeline generation/customisation mechanisms.
Innovation	combines pipeline automation, data lakehouse, infrastructure as code, and observability etc. to propose a reproducible framework that exemplifies applied innovation.	strong emphasis on reducing data complexity by identifying and using only the four most impactful and relevant features, instead of relying on the full dataset.	Combines semantic interoperability (RDF/KG), declarative mappings, low-code pipeline design (Karavan), and reusable deployment templates for edge/cloud.
Clarity & Structure	Clear logical flow: from literature to framework → implementation → performance → discussion. Well-documented pipeline steps.	Descriptive and rich in references, but lacks a structured empirical or practical section.	Clearly structured around three artefacts: modular components, low-code tools to define pipelines, and deployment templates for multiple environments.
Added Value	Operationalises DataOps and demonstrates measurable results, bridging the gap between theory and practice.	Though theoretical, it adds to the conceptual conversation and raises awareness of DataOps trends.	Enables reusable and customisable semantic pipelines; rules can be iteratively updated by stakeholders (with some learning curve).
Healthcare	Yes, applied in healthcare, specifically validating data pipelines with healthcare data.	Yes, applied in healthcare.	Yes, includes a smart health rehabilitation use case using a knowledge graph repository, recipe↔device matching, and JSON export for orchestration/AI monitoring.
End-to-End	Yes, provides an end-to-end DataOps framework with integrated automation, governance, and scalability.	No, lacks a unified end-to-end architecture	Partial focused on semantic integration, rather than a full lifecycle.
Governance / CI/CD /Lakehouse/IaC	Yes, includes Governance, CI/CD pipeline, Lakehouse architecture, and Infrastructure as Code (IaC).	No integration of governance, CI/CD, Lakehouse, or IaC; focuses on general challenges and adoption trends.	Focuses solely on declarative configuration and deployment and does not integrate these components (governance, CI/CD, IaC and Lakehouse).
Artefacts de Replication	Yes, offers fully replicable artefacts and a complete set of tools for DataOps pipeline implementation.	No replicable artefacts provided.	Provides reusable deployment templates and low-code/exportable pipeline.
KPIs Reported	Ingestion latency: ~100 MB/s, Throughput: ~30,584 records/s, Data quality score: 97.87%, Resource utilisation: ~40%.	Sensitivity: 87.94%, Specificity: ~85%, Accuracy: 82.32%, F1 Score: 86.21% (Logistic Regression)	Reports system-level metrics such as latency < 100 ms and ~100 MB memory (CPU also measured) for deployment options.

Overall, Table 10 shows that the main contribution of this work lies not in outperforming previous studies on a single isolated metric, but in providing an integrated, governed, and reproducible DataOps pipeline whose operational behaviour is explicitly measured and therefore reusable as a benchmark for future comparative studies.

8. Discussion

This section summarises the key findings from the implementation of the pipeline, including how it addresses the research questions, it also discusses the implementation challenges and solutions encountered during deployment and concludes with future research aimed at enhancing the pipeline.

8.1. Key Findings

This study demonstrates that our modular, value-driven DataOps approach can significantly improve the agility, scalability, and reliability of data pipelines in a domain known for its complexity (the healthcare sector). Our study which is based on an analysis of three guiding research questions, has produced the following results:

Regarding RQ1: The research confirms that an effective DataOps architecture must integrate end-to-end automation, governance, and scalability from the design stage rather than as isolated add-ons. The proposed solution, structured around phases and stages from source to consumption, has proven essential to ensuring continuity across the entire pipeline, while also placing transparency and reproducibility as important considerations. This finding highlights that architecture design principles and the process of orchestration are more decisive for success than the specific tools employed. In response to RQ2: The application of the Spine Model demonstrates that the value-based integration ensures coherence between organisational needs and technical implementation, rather than assembling disconnected components and aligning tools with principles. It minimised technical debt and facilitated CI/CD automation while also strengthening governance continuity.

Finally, in response to RQ3: the empirical validation in the healthcare domain confirmed that the proposed architecture achieved performance improvements. The pipeline sustained an ingestion rate of approximately 100 ± 6 MB/s, and a processing latency of 120 ± 15 ms, with P95 and P99 latencies of 150 ms and 180 ms, respectively. Processing throughput a processing throughput exceeding 30,000 records/s and a data quality score of 97.87 % ± 0.5 % while maintaining balanced resource utilisation (≈ 40 %). These results demonstrate that technological advances now make it possible to perform near real-time analyses with reliable data in strict compliance environments. This greatly enhances the potential of data operations solutions to deliver scalable and secure high-quality data flows.

Together, these findings validate the relevance of a modular, value-oriented DataOps framework capable of transforming healthcare data management into a continuously automated, governed, and adaptive process. The following section provides a comparative assessment of our suggested solution against current DataOps frameworks and contributions in order to further contextualise these findings.

8.2. Implementation Challenges and Solutions

Implementing the proposed DataOps architecture in a real-world healthcare setting presented both technical and organisational challenges. To make them actionable, we summarized them in Table 11, structured by challenge, impact, mitigation measure applied, and evidence/measure. By addressing these issues through phased deployment, governance policies, training programmes, and automated monitoring, the implementation has delivered measurable improvements in performance, quality, and resilience.

Table 11. Implementation Challenges, Impact, Mitigation and Supporting Evidence

Challenge	Impact	Mitigation Applied	Evidence / Measure
Challenge Impact Integration of heterogeneous tools (protocols, dependencies, updates)	Pipeline orchestration complexity, risk of failures	Standardised APIs, modular microservices, message queues for decoupling	Stable ingestion throughput (100 MB/s), no critical integration failures
Maintaining performance across components	Latency increase, resource bottlenecks	Continuous benchmarking, autoscaling with Kubernetes & Spark optimisation	Achieved 30,584 ± 820 records/s processing throughput
Employee resistance & lack of DataOps know-how	Slow adoption, operational friction	Training programs, workshops, internal pilot projects	Increased adoption rate, 3 successful CI/CD deployments every 5 days
Fragmented governance & compliance policies	Metadata silos, inconsistent access control	Early definition of metadata standards, IaC-based enforcement with Atlas & Ranger	97.87% data quality score; uniform lineage across Silver layer
Security & regulatory compliance with IaC	Slower provisioning due to constraints	Infrastructure-as-Code with Terraform & Ansible, automated CI/CD	Provisioning reduced to 50 ± 3 min vs 2–3 h manual baseline
Testing & rollback under failures	Risk of cascading errors, downtime	Automated monitoring, rollback & schema validation mechanisms	Alert resolution time reduced to 12 ± 2 min
Resource allocation inefficiency	Over-provisioning or under-utilisation	Automated scaling, KPI-driven monitoring	Resource utilisation efficiency of 40 ± 4 %

Although the proposed architecture is intentionally comprehensive. Its implementation does not require all components to be deployed at full scale from the outset, in small-scale, or resource-constrained environments, the architecture can be adopted progressively through a modular deployment strategy. A minimal configuration may focus first on core ingestion, storage, basic quality validation and monitoring, while more advanced capabilities such as full CI/CD automation, fine-grained governance, Kubernetes-based scalability and advanced observability can be introduced incrementally as operational maturity and resource availability increase. This staged adoption reduces initial complexity, limits infrastructure overhead and allows organizations to preserve the main DataOps principles automation, traceability, quality control, and reproducibility without imposing unnecessary technical burden. Therefore, the proposed framework should be interpreted as a scalable reference architecture rather than a rigid deployment blueprint.

9. Conclusion

In this study, we proposed and evaluated a governed DataOps reference architecture designed to support secure, reproducible, and AI-ready data operations in regulated environments. The architecture integrates a Medallion Lakehouse progression with data-centric CI/CD, Infrastructure as Code, workflow orchestration, monitoring and governance mechanisms. A central contribution is the definition of explicit promotion contracts between pipeline stages, enabling data to move from raw ingestion to curated and consumption-ready layers only after meeting predefined quality, governance, and operational requirements. In addition, the DataOps Operational Excellence Index (DOEI) was introduced as a composite evaluation framework to synthesize heterogeneous performance dimensions, including processing efficiency, governance, agility and resource utilization. This combination provides a structured way to connect architectural design choices with measurable operational outcomes.

The empirical evaluation in a controlled healthcare testbed demonstrated that the proposed architecture can improve operational performance while maintaining strong governance requirements. Using approximately 3.5 GB of multi-source healthcare data over a 25-day workload, the pipeline achieved a DOEI score of 0.92, an ingestion throughput of approximately 100 MB/s, a data quality score of 97.87%, and a reduction in infrastructure provisioning time from 3 hours to 50 minutes. These findings indicate that combining Lakehouse principles with automated deployment, governance gates and continuous monitoring can provide a practical foundation for scalable and auditable DataOps implementation. They also show that governance-oriented controls do not necessarily create operational bottlenecks when they are embedded directly into the pipeline design. However, the evaluation remains limited to a controlled, single-domain healthcare scenario and a moderate dataset size. Future research should extend the architecture to larger, multi-site environments, evaluate alternative deployment configurations and incorporate policy-based self-healing mechanisms to further enhance resilience, compliance, and operational autonomy.

9.1. Future Work

While the current implementation of the proposed solution validates its feasibility and performance, further research is needed to extend its scalability and compliance. The three main avenues being explored are

- *The first is policy-based automatic repair and restoration*

where predefined policies would automatically trigger restoration or rerouting mechanisms in the event of a failure, for example, or anomalies, schema drifts, or deployment failures. This will reduce downtime and operational risks by ensuring the continuity of the service that has been started.

- *Scalability based on performance indicators*

Based on the current monitoring layer, an advanced automatic scalability strategy will need to be tested, in which scaling decisions through tools such as Kubernetes HPA or KEDA will be guided by key KPIs such as ingestion latency, throughput, or resource efficiency, with dynamic thresholds will enable adaptive scalability in real time, allowing for both performance and cost optimisation.

- *Testing data contracts and confidentiality from the design stage onwards*

For better governance and data compliance, future work will need to incorporate data contract testing to ensure schema consistency, metadata traceability and policy enforcement at all stages of the pipeline. At the same time, privacy by design will be implemented through the use of data generators that enable secure testing and validation of models. All of these guidelines are aimed at evolving the pipeline from a system with demonstrable quality to a more robust and advanced system that integrates other innovative elements.

10. Declarations

10.1. Author Contributions

Conceptualization, A.F.; methodology, A.F.; software, A.F.; validation, Y.G. and J.G.; formal analysis, A.F.; investigation, Y.G. and J.G.; resources, A.F., Y.G., and J.G.; data curation, A.F.; writing—original draft preparation, A.F.; writing—review and editing, J.G. and Y.G.; visualization, A.F.; supervision, J.G. and Y.G.; project administration, Y.G. and J.G. All authors have read and agreed to the published version of the manuscript

10.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

10.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

10.4. Acknowledgments

The authors would like to thank the Laboratory of Engineering Science at Ibn Tofail University for providing the technical environment and research support necessary to conduct this study. The authors also acknowledge the valuable academic discussions and feedback that contributed to improving the architecture design and evaluation methodology.

10.5. Institutional Review Board Statement

Not applicable.

10.6. Informed Consent Statement

Not applicable.

10.7. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

11. References

[1] Oktavia, T., & Wijaya, E. (2025). Strategic Metadata Implementation: A Catalyst for Enhanced BI Systems and Organizational Effectiveness. HighTech and Innovation Journal, 6(1), 21–41. doi:10.28991/HIJ-2025-06-01-02.

[2] Alexandrov, I. A., Kuklin, V. Z., Chervyakov, L. M., & Sheptunov, S. A. (2024). Development of a Technique for Discrete-Logical Decision-Making in Medical Information Systems. HighTech and Innovation Journal, 5(4), 1008–1023. doi:10.28991/HIJ-2024-05-04-010.

[3] Fannouch, A., Gahi, Y., & Gharib, J. (2024). Unified Data Framework for Enhanced Data Management, Consumption, Provisioning, Processing and Movement. ACM International Conference Proceeding Series, 3659836. doi:10.1145/3659677.3659836.

[4] Alselami, N., Aati, K., Mutnbak, M., Alrasheed, K. A., & Basit Khan, M. (2025). Impact of the Application of Smart Sensor Networks for the Construction Management of Geotechnical Activities. Civil Engineering Journal, 11(1), 346–368. doi:10.28991/CEJ-2025-011-01-020.

[5] Mishra, S., & Misra, A. (2018). Structured and Unstructured Big Data Analytics. International Conference on Current Trends in Computer, Electrical, Electronics and Communication, CTCEEC 2017, 740–746. doi:10.1109/CTCEEC.2017.8454999.

[6] Hamouda, S., & Zainol, Z. (2019). Semi-structured data model for big data (SS-DMBD). DATA 2019 - Proceedings of the 8th International Conference on Data Science, Technology and Applications, 348–356. doi:10.5220/0007957603480356.

[7] Ntinopoulos, V., Rodriguez Cetina Biefer, H., Tudorache, I., Papadopoulos, N., Odavic, D., Risteski, P., Haeussler, A., & Dzemali, O. (2025). Large language models for data extraction from unstructured and semi-structured electronic health records: A multiple model performance evaluation. BMJ Health and Care Informatics, 32(1), 101139. doi:10.1136/bmjhci-2024-101139.

[8] Saleem, A., Shah, S., Iftikhar, H., Zywiołek, J., & Albalawi, O. (2025). A Comprehensive Systematic Survey of IoT Protocols: Implications for Data Quality and Performance. IEEE Access, 13, 196206–196235. doi:10.1109/ACCESS.2024.3486927.

[9] Yu, S., Chen, T., Han, L., Demartini, G., & Sadiq, S. (2023). DataOps-4G: On Supporting Generalists in Data Quality Discovery. IEEE Transactions on Knowledge and Data Engineering, 35(5), 4668–4681. doi:10.1109/TKDE.2022.3151605.

[10] Yin, Z., Zhou, S., Zhou, J., Tian, M., Lin, M., & Liu, S. (2023). Research on DataOps Capability - Practice and Development. Proceedings - 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom/BigDataSE/CSE/EUC/iSCI 2023, 2170–2174. doi:10.1109/TrustCom60117.2023.00303.

[11] Morjane, W., Bannari, R., & Gharib, J. (2023). Overview of Project Management Methodologies: Traditional Versus Agile Approach. Proceedings of the 5th European International Conference on Industrial Engineering and Operations Management, 783–794. doi:10.46254/eu05.20220157.

[12] Mansour, I. J. S., Mat Rejab, M. B., & Mahdin, H. Bin. (2024). Review in Adoption of DevOps, AIOps, DataOps, GitOps, MLOps in IT MLEs in Germany. International Journal of Engineering Trends and Technology, 72(12), 64–76. doi:10.14445/22315381/IJETT-V72I12P106.

[13] Chung, E. S., & Molléri, J. S. (2025). A Multivocal Literature Review on DataOps—Concepts, Benefits, and Challenges. Lecture Notes in Networks and Systems, 1255, 213–226. doi:10.1007/978-981-96-1747-0_18.

[14] Fannouch, A., Gharib, J., & Gahi, Y. (2025). Enhancing DataOps practices through innovative collaborative models: A systematic review. International Journal of Information Management Data Insights, 5(1), 100321. doi:10.1016/j.jjimei.2025.100321.

[15] Bahaa, S., Ghalwash, A. Z., & Harb, H. (2023). DataOps Lifecycle with a Case Study in Healthcare. International Journal of Advanced Computer Science and Applications, 14(1), 136–144. doi:10.14569/IJACSA.2023.0140115.

[16] Garriga, M., Aarns, K., Tsigkanos, C., Tamburri, D. A., & Heuvel, W. Van Den. (2021). DataOps for Cyber-Physical Systems Governance: The Airport Passenger Flow Case. ACM Transactions on Internet Technology, 21(2), 1–25. doi:10.1145/3432247.

[17] Tamburri, D. A., Heuvel, W. J. Van Den, & Garriga, M. (2020). DataOps for Societal Intelligence: A Data Pipeline for Labor Market Skills Extraction and Matching. Proceedings - 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science, IRI 2020, 391–394. doi:10.1109/IRI49571.2020.00063.

[18] Scrocca, M., Grassi, M., Carenini, A., Anicic, D., Calbimonte, J. P., & Celino, I. (2026). A DataOps Toolbox Enabling Continuous Semantic Integration of Devices for Edge-Cloud AI Applications. Lecture Notes in Computer Science: Vol. 16141 LNCS, 379–397. doi:10.1007/978-3-032-09530-5_22.

[19] Pestana, G., Almeida, M., & Martins, N. (2025). Tracking Secondary Raw Material Operational Framework—DataOps Case Study. Ceramics, 8(1), 12. doi:10.3390/ceramics8010012.

[20] Mishra, T. (2025). 451 Research: DataOps unlocks the value of data. Hitachi Digital Services, California, United States. Available online: https://www.hitachids.com/vn-english/pdf/451-research-dataops-unlocks-the-value-of-data/ (accessed on May 2026).

[21] Chia, J. (2026). What is DataOps? Definition, principles, and benefits. Alation, California, United States. Available online: https://www.alation.com/blog/what-is-dataops/ (accessed on May 2026).

[22] DataKitchen. (2021). Eight top DataOps trends for 2022. DataKitchen Marketing Team, Lexington, United States. Available online: https://datakitchen.io/eight-top-dataops-trends-for-2022/ (accessed on May 2026).

[23] Thusoo, A., & Sarma, J. S. (2017). Creating a data-driven enterprise with DataOps: Insights from Facebook, Uber, LinkedIn, Twitter, and eBay. O'Reilly Media, California, United States.

[24] Grand View Research. (2024). DataOps platform market size & share report, 2024–2030. Grand View Research, California, United States. Available online: https://www.grandviewresearch.com/industry-analysis/dataops-platform-market-report (accessed on May 2026).

[25] ISG. (2025). AI adoption drives interest in DataOps, ISG study finds. Information Services Group, Stamford, United States. Available online: https://ir.isg-one.com/news-market-information/press-releases/news-details/2025/AI-Adoption-Drives-Interest-in-DataOps-ISG-Study-Finds/default.aspx (accessed on May 2026).

[26] Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. BMJ, 372. doi:10.1136/bmj.n71.

[27] Zahid, H., Mahmood, T., & Ikram, N. (2018). Enhancing dependability in big data analytics enterprise pipelines. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11342 LNCS, 272–281. doi:10.1007/978-3-030-05345-1_23.

[28] Pinkel, C., Schwarte, A., Trame, J., Nikolov, A., Bastinos, A. S., & Zeuch, T. (2015). DataOps: Seamless End-to-End anything-to-RDF data integration. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9341, 123–127. doi:10.1007/978-3-319-25639-9_24.

[29] Tu, D., He, Y., Cui, W., Ge, S., Zhang, H., Han, S., Zhang, D., & Chaudhuri, S. (2023). Auto-Validate by-History: Auto-Program Data Quality Constraints to Validate Recurring Data Pipelines. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 4991–5003. doi:10.1145/3580305.3599776.

[30] Bayram, F., Ahmed, B. S., Hallin, E., & Engman, A. (2023). DQSOps: Data Quality Scoring Operations Framework for Data-Driven Applications. ACM International Conference Proceeding Series, 32–41. doi:10.1145/3593434.3593445.

[31] Spine Model. (2026). The Spine Model. Spine Model Documentation and Wiki. Available online: https://spinemodel.info/ (accessed on May 2026).

[32] Carthen, C., Zaremehrjardi, A., Le, V., Cardillo, C., Strachan, S., Tavakkoli, A., Harris, F. C., & Dascalu, S. M. (2023). Orchestrating Apache NiFi/MiNiFi within a Spatial Data Pipeline. Proceedings - 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications, SERA 2023, 366–371. doi:10.1109/SERA57763.2023.10197731.

[33] Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., Torres, J., van Hovell, H., Ionescu, A., Łuszczak, A., Świtakowski, M., Szafrański, M., Li, X., Ueshin, T., Mokhtar, M., Boncz, P., Ghodsi, A., Paranjpye, S., Senster, P., … Zaharia, M. (2020). Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. Proceedings of the VLDB Endowment, 13(12), 3411–3424. doi:10.14778/3415478.3415560.

[34] Papadopoulos, A. N., Sioutas, S., Zaroliagis, C., & Zacharatos, N. (2019). Efficient distributed range query processing in apache spark. Proceedings - 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2019, 569–575. doi:10.1109/CCGRID.2019.00073.

[35] Soares, G. H., & Brito, M. A. (2023). Business Intelligence over and above Apache Superset. Iberian Conference on Information Systems and Technologies, CISTI, 2023-June, 1–6. doi:10.23919/CISTI58278.2023.10211907.

[36] Lucas, C. (2023). Evaluate Apache Ranger to provide comprehensive security across the CERN Hadoop ecosystem. CERN Repository, CERN Repository. Available online: https://repository.cern/records/hsh6d-2xh86 (accessed on May 2026).

[37] Lambert, F., Odier, J., Fulachier, J., Jaume, M., & Delsart, P. A. (2024). Deploying the ATLAS Metadata Interface (AMI) stack in a Docker Compose or Kubernetes environment. EPJ Web of Conferences, 295, 1017. doi:10.1051/epjconf/202429501017.

[38] Tian, L., Sedona, R., Mozaffari, A., Kreshpa, E., Paris, C., Riedel, M., Schultz, M. G., & Cavallaro, G. (2023). End-To-End Process Orchestration of Earth Observation Data Workflows With Apache Airflow on High Performance Computing. International Geoscience and Remote Sensing Symposium (IGARSS), 711–714. doi:10.1109/IGARSS52108.2023.10283416.

[39] Fairbanks, J., Tharigonda, A., & Eisty, N. U. (2023). Analyzing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab. Proceedings - 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications, SERA 2023, 176–181. doi:10.1109/SERA57763.2023.10197778.

[40] Gupta, M., Chowdary, M. N., Bussa, S., & Chowdary, C. K. (2021). Deploying Hadoop Architecture Using Ansible and Terraform. 2021 5th International Conference on Information Systems and Computer Networks, ISCON 2021, 1–6. doi:10.1109/ISCON52037.2021.9702299.

[41] Ahmed, F., Jahangir, U., Rahim, H., Ali, K., & Agha, D. E. S. (2020). Centralized Log Management Using Elasticsearch, Logstash and Kibana. ICISCT 2020 - 2nd International Conference on Information Science and Communication Technology, 1–7. doi:10.1109/ICISCT49550.2020.9080053.