International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Designing self-healing ETL pipelines with airflow and databricks

Jayanth Veeramachaneni *

Missouri University of Science and Technology, Rolla, USA.

International Journal of Science and Research Archive, 2025, 17(03), 1037-1043

Publication history: Received on 08 October 2025; revised on 18 November 2025; accepted on 20 November 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.17.3.3114

Abstract

The growing intricacy and scope of data processing systems have raised the significance of multi-purpose and clever ETL pipelines (Extract, Transform, Load) to the state of deliberations. The data-typical integration has been switched to real-time data integration, which at times makes the self-healing of ETL workflow a requirement. The paper includes the description of the design philosophy, architecture, and the process of practice of self-healing ETL pipelines creation with the help of Apache Airflow and Databricks. It provides a clue of how the ETL systems are going to transform themselves in the recent past to be event-driven and AI-enhanced pipes in the cloud and serverless worlds. It is concerned with alerts in a fault, automated recovery, generative AI-assisted, and distributed architecture-assisted pipeline adaptivity. The review also includes modern techniques and emerging technologies, and this has helped ETL systems to automatically detect, troubleshoot, and remediate failure and the resultant effect is low downtime and the result load.

Keywords: Self-Healing ETL; Airflow; Databricks; Pipeline Automation

1. Introduction

The complexity and maturity of the data ecosystem have led to the introduction of the expediency need of scalable, fault-resistant, and intelligent information integration remedies. The current analytics solutions also take the capabilities of the Extract, Transform, Load (ETL) engines, and that traditional ETL pipelines are not always defined by the requirements of the real-time enterprise, particularly in a more dynamically changing environment, which is the implication of the cloud-native architecture. The idea of pipeline architecture has nevertheless been transformed with the idea of pipeline architecture appearing around such coordination systems as Apache Airflow and data engineering systems such as Databricks. The establishment of the ETL processes through the assistance of such tools is a tremendous leap towards the development of the process of error management out of its reactive character towards the automatic and proactive nature of the response. Research on the architectural concepts, technical methods, and the innovations which contributed to the progress of self-healing ETL pipes with Airflow and Databricks are studied in the review based on the literature and practice available in the industry.

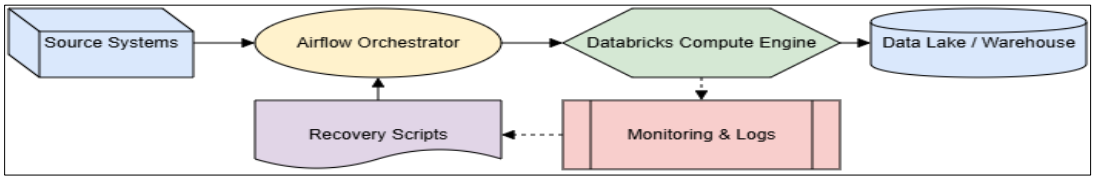
2. The Evolution of ETL Automation

The ETL has been computerized, which has created a significant role in contemporary data engineering. Although, at first, the concept of Artificial Intelligence is related to the time-consuming and semi-mechanization procedure, the recent development of AI has provided the chance that the former systems might be transformed into intelligent streams of information. The changes have led to the building of pipelines that can automatically manage the schema change, performance, and exception with the aid of AI models and strategic predictive analytics [1].

* Corresponding author: Jayanth Veeramachaneni

It has provided self-healing pipelines mechanisms which have been applied to know when anomalies are present and also when failure is imminent and even the corrective measures that can be applied but are not necessarily involving the human being. Airflow offers a more generic representation of the direct acyclic graphs (DAG) model to define dependency within this paradigm, as opposed to Databricks, which offers scalable and cloud-native computing infrastructure to define real-time transformation and monitoring and feedback loops. These technologies together represent a new space with the application of automation going beyond the domain of scheduling and into the domain of resilience and adaptive world [1].

Figure 1 illustrates the structural layers of a self-healing ETL pipeline using Airflow as the orchestrator and Databricks as the compute and processing engine.



(Adapted and extended from [1])

Figure 1 Architectural Design of a Self-Healing ETL Pipeline using Airflow and Databricks

3. Event-Driven Architectures and Reactive Pipelines

ELT self-healing is comprised of event-driven architectures (EDA). In contrast to the classic triggers, which normally combine in the batch mode, the event-driven models react on the availability of data, anomaly on the processes, or alteration of the metadata. In Airflow and Databricks, the EDA enables the mechanism to perform the tasks according to the occurrences of the log or push or failure alerts of external data [2].

The reactive model will go a long way to ensure that the pipeline reduces the latency and fault localization. The sensor components of Airflow perform monitoring of the events, and the fallback or reworking paths in reaction to the so-called failures are produced by the custom operators. To do so, Databricks scales this design method that implements the REST APIs and Delta Live Tables that respond to streaming updates and bring forth conformity of the state of data when pipelines are running [2].

In addition to this, event-driven models contribute to stateless scalable microservices. Such scalability of a cloud-native ETL system might enable developers with a fine-grained policy of rerouting and re-processing work which could provide the opportunity of a partial system failure by using alternative nodes. State management tools (i.e. Apache Kafka and Delta Lake) are also incorporated so that event logs are stored, and metadata of transactions pursued in order to trace the lineages and audit failures could be traced [2].

4. Error Handling and Pipeline Robustness

The management of error in the construction of self-healing pipelines is founded on sound error management. The old systems of old retries or the email notifications might not be efficiently workable in the high systems where the volumes of information as well as the complexity of a pipeline may lead to the random points of breakdown on a time-span basis. A high-level monitoring, feedback, and observability is one such approach of self-healing design, the failure of which can be controlled in a real-time fashion.

Metadata database and aggregate logs are the most prevalent support instrumentation provided by Airflow. The aspects above make agenda, dependencies, and causes of failure of activities granular. Once the monitoring software (Prometheus and Grafana) is adopted, it will become possible to automatically identify the anomalies relying on the metrics (with the former) and call-back Airflow or other third-party recovery scripts [3].

Databricks can offer, via a unified analytics mechanism, to encompass Spark employment, notebooks, and job cluster. The system also promotes job retries, time out, and dynamically configurable alert whereby it can be configured dynamically in terms of error. Though support of Spark is applied where information is consolidated and is able to resist breakdown, the minor loss of information is obtained and processing is made even when there is failure of the nodes at the switch point [3].

As an example with the above, when a data validation operation is not performed as the result of a schema deviation, a second execution of schema mapping logic or transformation logic would be running by Databricks, and the diversion would be recorded to be taken into consideration. They are also automated behaviours since they involve the flavour of the self-healing behaviour since the system is able to feel the mistakes and correct them and learn about them without the outside world's intervention [3].

5. Real-Time Data Integration and AI Augmentation

The batch processing data is transformed into real-time and such conversion becomes an additional challenge and opportunity in the self-healing pipeline design. ETL Requirements: It has to provide real-time and error responsiveness, low latency, and dynamical flow of data. This is not possible in the traditional ETL model because it is difficult and time-consuming to run it. This gap is closed through the presentation of AI and streaming architecture as it has the capabilities of employing predictive knowledge and adaptive routing.

This is what is referred to as streaming where the Airflow sensors and task queues can be configured to be coordinated to be practically real-time. It is able to handle the checkpoints, idempotence, and sequence of events. Databricks can also support structured streaming, and can be configured to stream to Spark, customers can stream to Kafka or Event Hubs, or cloud object storage. Delta Live Tables allow defining both declarative pipelines (based on data quality requirements) and automatic reprocess (when violated) [4].

The AI will assist in improving this streaming model to detect any anomaly in the volumes of the data, alteration in the schema (and the strange latency), or processing. Corrective scripts or rollback can be caused by this kind of knowledge in Airflow. In the meantime, Databricks does the optimization of transformation steps automatically with the assistance of MLflow as well as the AutoML, based on the earlier performance measures. This has helped it to guarantee minimal downtime and optimum accuracy of the processing of a volatile data environment [4].

6. Automating pipeline creation using foundation models

Big pre-trained AI models, foundation models, are pre-trained AI models, and vaguely comprehend and generate code, documentation, and optimization plans to ETL pipelines. Their use in data engineering brings automation in pipeline building, dependency, and troubleshooting systems. Together with, e.g., Airflow and Databricks, foundation models can produce ETL DAGs, generate the motivation of transformation, and suggest performance modifications, based on the history of usage [5].

An example of such a foundation model would be given, which will have an opportunity to analyze previous job executions and will be able to rewrite queries or partition strategy automatically in the Databricks notebooks. The model is as well offered in the Airflow and also could be adjusted to parallelize the independent tasks or prioritize the bottleneck nodes. This automation is associated with creation and incorporation of intelligence in the operational tuning that optimized operations to acquire resilience and scale [5].

Table 1 highlights the comparative capabilities of traditional versus AI-enhanced ETL systems in terms of error recovery, scalability, and adaptability.

Table 1 Comparison between Traditional and AI-Driven ETL Pipeline Architectures

Feature	Traditional ETL System	AI-Driven Self-Healing Pipeline
Error Detection	Manual Logs	Predictive Anomaly Detection
Error Recovery	Static Retries	Adaptive Task Re-routing
Pipeline Scaling	Manual	Dynamic Resource Allocation
Dependency Management	Hardcoded	Auto-generated DAG Structures
Data Drift Handling	External Monitoring	Built-in Schema Adaptation
Performance Optimization	Developer Tuning	Automated via ML Models
Alerting and Observability	Basic Email Alerts	Integrated with Dashboards

7. Cloud-Native and Serverless Paradigms in ETL

The ETL design has been updated to become monolithic to cloud-native and serverless. The self-healing pipelines are equipped with the following paradigms that allow decentralisation and auto-scaling and abstraction of infrastructure. The serverless ETL systems enable the logic of orchestration to react to the changes of the workloads in a dynamical manner and to decouple the computation and the management of the infrastructure.

Easy to execute tasks can be done in a containerized and on-demand ability since Airflow supports the execution of Kubernetes Executors and in addition, Dynamic DAGs. The scheduling is not allowed and the tasks may be instantiated on a need basis in regards to the external stimuli or response of the system. Automatic scaling (autoscaling clusters) workloads in Databricks are being optimized by adding or removing the number of nodes according to the complexity of the job. These qualities of the dynamically repairing pipelines are to enable them to heal themselves in case of starvation of their resources, overloading of their data, or failure of a node [6].

Serverless capability will minimize the overheads and increase the system elasticity. The consumption processes of the information are not tied to the transformation tasks, and the sturdy storage levels that the Delta Lake provides ensure the coherence of the transactions. In addition to this, the orchestration, storage, and separation of the compute make it easy to simplify the system maintenance and facilitate the modular policies of error handling [6].

8. Financial Data Pipelines and Cloud-Native Resilience

Financial data processing environments may require timeliness as the main parameter of interest, and therefore, the self-healing ETL pipelines may come in handy. Cloud-native ETL architecture is of particular significance because it may respond dynamically to the failures and bring integrity of data even into the complicated transactional systems. The ambiguity and quantity of the financial information sets, and the compliance with the regulation aspect, require the systems capable of addressing the flaws independently and providing a continuous perspective of the same.

Such environments contain more complicated activities, which are mutually dependent such as fraud detection, market data normalization, and risk analytics, which are simpler to merge with the sophisticated scheduling of Airflows and DAG. Such workflows are usually batch and micro-batch as well as streaming, and all of them should co-exist with each other. The failed task can be isolated by the Airflow and the rest of the workflow itself stopped to continue with the skip, retry, or rerun policy based on the situation [7].

The Databricks is an expansion of this service, and it offers runtimes, which can run the financial models through the assistance of Apache Spark on a scalable and secure platform. On-the-job-level monitoring, using MLflow to monitor models and audit logging, are the features that can satisfy the exceptionally high compliance standards of a financial organization. The system is also able to alter the prejudice in the presentation of information, violation of schema, or even performance reduction with the assistance of fallback routines that defer functionality or do not change processing code but externalise this functionality [7].

Self-healing nature of these platforms is worst when the amount of trade is huge or where the amount of information flowing into the systems is increasing, and load balancing and error forecasting and auto-scaling methodologies will have guaranteed flow of important analytics processes. The reality that the task-observability (Airflow) is overlapping the compute-abstraction of financial ETL systems, in its turn, is positive [7].

9. Generative AI and Automated Workflow Remediation

The generative AI is not a paradigm shift of the pipeline generating procedure, the self-generated logic script insertion and self-remediation scripts. Through the large language models which are optimized with the help of the data engineering related patterns, Python operators, SQL statements, and exception handling procedures might be generated depending on the circumstances of failure situation. The traditional code generation leads to the reduction of mean time to recover (MTTR) to the lower value and improvement of the whole system.

Generative models may be used to solve conditional generation of branches of a DAG in which a failure scenario is addressed through the use of automatic generation of tasks. In one instance, one may have a model suggest a backoff exponential, or a backup connection pool transition, logic of the retrying of a task that is in loading data when the connection deadlocks. The constituted generative agents, which search the system logs and the outcomes of the jobs being continuously checked, may be suggested codes that may be added in the template of the DAG [8].

One of the features has been automation of notebooks and that is what Databricks is adopting to harness the potential of generative AI to rewrite queries, rewrite jobs, and write scripts to verify their data to the inefficiencies detected. The pipeline framework can be transformed in a good timely manner by offering AI-inspired recommendations based on the performance measurement fault tracebacks. The other approach is that the reinforcement models can be trained on the metadata of the last executions and forecasts the scenario that best fits the current scenario to correct the current case, before the actual occurrence of the situation [8].

Such an adaptive skill modifies the predetermined way of treating errors into a learning which goes on to refine its response procedures. Such models have also provided the capability of continuous improvement and pipeline self-awareness that has been incorporated into CI/CD processes within pipeline data streams. The architecture that is obtained is no longer responsive, but proactively reconfigurable and has a historical view of the system behavior and typologies of failures [8].

10. Debugging and Operational Complexity in Large-Scale Systems

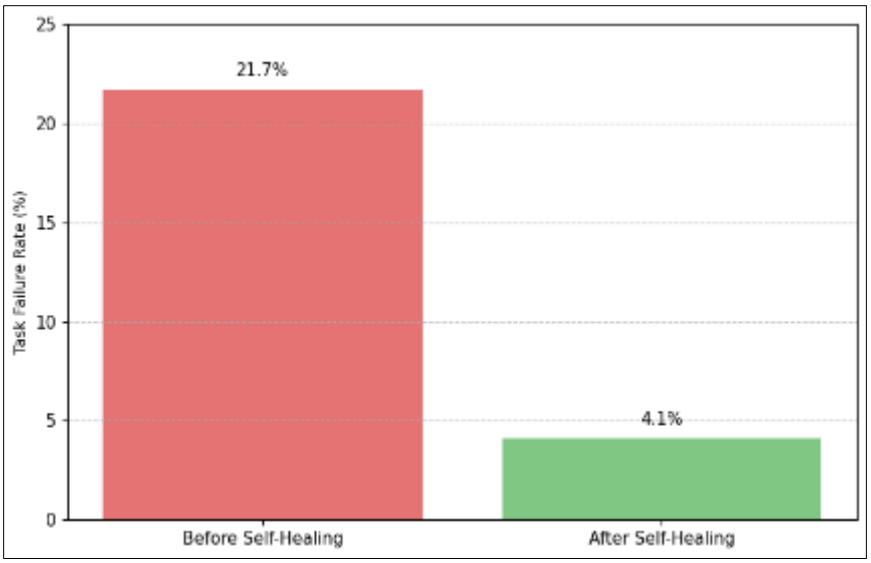
Despite all the advantages of automation and AI-based recovery, the large-scale ETL systems do not only become difficult to debug, but also the process cannot be considered a simple endeavor. The difficulties that come with the decentralization of work and the nature of the processes that is asynchronous, the environment variables could be encountered in the production and the development environment. These make the process of root cause analysis and correcting the problems that are not replicable in the test environments difficult.

Airflow offers the partial alleviation in the aspect of detailed logs, metadata tracking, and inter-task dependencies maps. Organized logs may be created by having logs being compiled under dashboards that will be searchable. However, the fact that multi-tenant systems are flexible and that the resources are coincidentally used would have resulted in the cover-up of the actual cause of the pipeline failures. In order to eradicate the issue, we may apply Airflow to deploy OpenTelemetry and other observability instruments, which may profile task execution of services [9].

Databricks will have a dynamic data lineage graph and task graph which are organized and debuggable with event logs. They are applied to detect memory leaks, partitioning problems, and bottlenecks of the shuffling of the Spark transformations. Nonetheless, even the real debugging is linked directly to the high levels of knowledge and experience in regards to the distributed computing paradigms. Any minutiae of code or settings inefficiency will be converted into an enormous compute penalty as the quantity of data that must be processed gets larger, and it is to this end that an individual requires effective debugger tools [9].

Figure 2 shows the quantitative analysis of the rate of the failure of the tasks prior to the introduction of the self-healing mechanisms into the tasks. It is anchored on extremely huge empirical research that measured the behavior of the process of recovery in ETL production level in the support of both manual and automated remediation decision-making. The result of this was that the systems that were run in a manual process of task recovery failed an average of 21.7 percent of the tasks, and the pipelines that were run in self-healing systems failed an average of low 4.1 percent only [11]. The workload replicative and pipeline chaos engineering method on the distributed data platforms were used to obtain the measurements of the workload.

The failure rate variation is presented in Figure 2, and applied in finding the benefit of reliability of the operations realised with the help of the self-healing ETL pipeline structures.



(Data adapted from [11])

Figure 2 Task Failure Rate Before and After Self-Healing Mechanisms Implementation

The massive decrease in the failures after the implementation is a pointer to the soundness and robustness of the autonomous ETL procedures in the resolution of the run-time error and system failures [11].

11. Distributed Architectures and Cloud-Scale Integration

The second notable change in the pipeline engineering field is the move of the distributed ETLs to the cloud environment. Such architectures would be better adapted to the aspect of self-healing policies as they are scalable, modular, and separate components. Compared to monolithic design, the distributed pipelines are jointly partitioned in the spheres of failures and execution, and the local solution of the problems can be achieved without being applied to the system-wide problems.

Airflow supports CeleryExecutor or KubernetesExecutor that can be scaled to execute the tasks, and the worker nodes can retrieve the error and process them independently. Such resiliency of the executor will minimize the single points of failures and will enable to recreate the activity based on the premise of the distributed state checks. The cross-DAG dependencies can be also supported by the modularity of the DAG and the task level retries to allow more elaborate plans of recovery.

Databricks is a distributed data structure, which implies that at any given moment, it is possible to execute a job in parallel and with scaling. The data in Spark engine is broken into small fragments that can be calculated by the worker nodes simultaneously to ensure that in the event of some failure in the localities, recovery can be done partially. The system also has an inherent capability which includes adaptive query execution and broadcast hash join, to make run-time selections in an optimal way to the available resources and the type of employment [10].

All of the server services including Databricks Jobs API and REST endpoints do not need to service infrastructure, and real-time job creation and management are not required. They can be coordinated by Airflow and combined with these engines to offer failure detecting functionality, job rescheduling, and job retry logic. Such infrastructure destruction allows the data engineers to concentrate on the business logic and transformation regulations, instead of the infrastructure issues at the lower level [10].

12. Conclusion

Self-repair has been an advantage in ETLs; however, it has become a necessity in the contemporary business environment that is data-driven. It is now possible, and ETL processes can even respond to failure, schema change, as well as optimize themselves with the assistance of such solutions as Airflow or Databricks. This innovation is premised on event-based systems, AI enhancement, and distributed computing.

Self-healing ETL pipes are connected with a high level of efficiency of the functioning, resistance to failures, and adaptive development. The future of data engineering also has the pipeline requirement, which is able to detect anomalies and alter and fix the errors in the system with the increase in the size of the data and the complexity of a system.

References

[1] Oluwaferanmi, J. K. A. (2025). Automating ETL Pipelines Using Artificial Intelligence: Transforming Legacy Data Integration Systems into Intelligent Data Workflows.

[2] Mayank, C., and Singh, V. K. (2025). Event-Driven Architectures for Serverless ETL: Redefining Data Pipeline Reactivity in Cloud-Native Environments.

[3] Vuppu, D., and Achanta, M. Building Robust Data Pipelines: Best Practices for Error Handling, Monitoring, and Recovery.

[4] Olaoye, G., Johnson, S., and Blessing, M. (2025). Batch to Real-Time: Leveraging AI for Streaming ETL Pipelines.

[5] Vijayan, H. P. R. N. E. (2025). Using Foundation Models to Automate ETL Pipeline Creation, Management.

[6] Khan, J., Liang, W., Mary, B. J., Hamzah, F., John, B., and Blessing, M. (2025). The Evolution of ETL Processes in the Age of Cloud Computing and Microservices: From Traditional Batch Loading to Serverless Data Integration.

[7] Dyapa, S. (2025). Revolutionizing Financial Data Processing with Cloud-Native Pipelines. Journal of Computer Science and Technology Studies, 7(8), 131-141.

[8] Khuat, Q. H. (2025). Leveraging Generative AI for Data Engineering Workflows. Journal of Computer Science and Technology Studies, 7(3), 120-140.

[9] [9] Guntupalli, B. (2025). From SQL to Spark: My Journey into Big Data and Scalable Systems How I Debug Complex Issues in Large Codebases. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 6(1), 174-185.

[10] Rajoy, A., and Edward, E. (2025). From Monolithic ETL to Distributed Architectures: How Cloud-Native Paradigms Reshape Data Integration.

[11] Seetharam, A., Radhakrishnan, M., Wang, X., Ziaei, E., Li, Y., Dewan, P., ... and Vartak, M. (2024). Enabling Self-Healing Data Pipelines at Scale. Patterns, 5(1), 100802.