Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Autonomous (Agentic) Data Pipelines for Self-Managing Lakehouses

Siddhartha Parimi

Dell Technologies, USA

ARTICLE INFO	ABSTRACT

Received: 06 Nov 2025

Revised: 15 Dec 2025

Accepted: 24 Dec 2025

The introduction of agentic systems of autonomy into enterprise data engineering is a paradigm shift that enables the operational complexity of contemporary lakehouse structures. Big language models are highly problematic when deployed to production data contexts, especially the ability to comprehend domain-specific schemas and produce consistent transformations between distributed pipeline processes. Intelligent automation of essential processes such as schema drift correction, query optimization, pipeline debugging, data quality correction, and resource allocation is made possible by multi-agent solutions that consist of reasoning agents to analyze diagnostic information, execution agents to interact with infrastructure, and critic agents to validate these agents. Patterns of architecture underline message-carrying substrates of asynchronous coordination, plans of orchestration between choreographed autonomy and centralized control, and safety measures such as confidence thresholding, blast radius restricting, and an extensive audit trail. Production deployments show a significant increase in operational density with respect to shorter mean time to recover, massive cost savings through automated resource optimization, and better reliability through sustained performance tuning. Implementation strategies deal with tradeoffs in the choice of frameworks, patterns of integration of heterogeneous compute engines, and schemes of continuous learning that allow further improvement with the ongoing experience of operation of agents. The combination of agentic artificial intelligence and distributed data infrastructure makes it possible to have self-organizing systems that ensure the quality of services provided, and the manual intervention needs are cut by far, which places organizations in a competitive position because it delivers analytics more rapidly, and the platform can be scaled sustainably.

Keywords: Autonomous Agents, Lakehouse Architecture, Multi-Agent Systems, Data Pipeline Automation, Intelligent Infrastructure Management

1. Introduction

The development of enterprise data platforms has introduced more complexity in the management of lakehouse systems, which combine the flexibility of data lakes and the reliability of data warehouses. The challenges in modern organizations are related to operations that cannot be properly handled by traditional methods of data engineering at scale. Industry researchers note that large language models encounter major challenges when used in enterprise data engineering scenarios, including domain schema understanding, contextual awareness throughout the pipeline distribution of stages, and reliable transformations to production systems [1]. The lakehouse paradigm, despite being promising in having one unified analytics capability, presents architectural complexities that require sophisticated automation mechanisms to cope with effectively.

The problems of scalability that are inherent to data science pipelines running in a lakehouse environment necessitate new solutions to manage large amounts of data without compromising the performance of analytics and the level of cost efficiency [2]. The conventional manual intervention approaches are too weak when the pipelines have to operate petabytes of datasets on distributed compute clusters and, at the same time, guarantee the quality of the data, optimal resource utilization, and respond to the changing schema demands. Data engineering teams have become operationally overstretched with reactive maintenance, causing most of the available engineering capacity to be used instead of strategic innovation. This bottleneck restricts the organizational capability to extract prompt information on the data resources and act on competitive forces that call for quicker analytics provision.

The agentic artificial intelligence is a progressive change to the autonomous systems that can plan, reason, and implement complex operational sets of work without constant human supervision. With the adoption of multi-agent architectures, where roles and duties are spread among specialized reasoning, execution, and validation modules, businesses will be able to turn their data platform into self-service ecosystems, rather than relying on manual operations. The paper discusses detailed models to execute autonomous agents in lakehouse settings, architectural design patterns, implementation plans, and quantifiable outcomes in performance gains through intelligent automation of key data engineering processes such as schema management, query optimization, pipeline debugging, quality assurance, and resource allocation.

2. Multi-Agent Data Pipeline Systems Architecture

The design principles of autonomous data pipeline management lie in the carefully designed multi-agent systems consisting of specialized agents that interact in well-defined protocols and communication patterns. Studies have shown that through coordinated decisions made by multi-agent systems based on distributed learning of reinforcement, complex network infrastructures can be effectively managed in a manner that allows balancing between the local optimization and the global system goals [3]. In data pipeline scenarios, this would mean reasoning agents interpreting telemetry streams on observability platforms to diagnose anomalies in performance and schema drift events, execution agents to apply remediation policies by intervening in distributed compute engines, and critic agents to verify the results against a set quality target and performance criteria.

The rationality ability in multi-agent systems has both prospects and constraints that need to be well resolved following the principles of human-centered design [4]. Although agents prove to be amazing in recognizing patterns and generating suitable diagnostic hypotheses, they need explainable mechanisms in reasoning to develop operator trust and allow human agents to exercise effective management. The architecture includes explicit reasoning traces that capture the evidence analysis, alternative evaluation, and remediation strategy selection of agents, and enables data engineers to gain insights into automated decision-making. This transparency is important in cases where the agents are faced with a grey scenario that may demand field knowledge or a business context that the agent had not been trained for.

Patterns of agent orchestration offer systematic templates of workflow coordination between the distributed set of agent populations working in cloud-based and hybrid infrastructure environments.

The architectural guidance of Microsoft underlines the need to find tradeoffs between choreography and orchestration, in which choreographed agents act independently according to event triggers, and on the other hand, orchestrated agents act according to central coordination logic [5]. To automate data pipelines, hybrid mechanisms are most useful where reasoning agents run in choreographed mode when pipeline failures and performance degradations are detected via the use of continuous monitoring, and execution agents execute orchestrated workflows that assure sufficient sequencing of schema migrations, configuration updates, and validation processes. The coordination layer ensures uniform state among the agent populations based on distributed consensus mechanisms to avoid conflicting actions and ensure actions of multi-step remediation workflows become atomic.

The communication substrate is composed of message-passing architectures that facilitate asynchronous coordination between autonomous agents that are distributed on data platform infrastructure. In the context of a distributed system, the traditional methods of messaging used focus on decoupling using asynchronous communication networks that enable components to interact without having any direct relationship or time-related links among them [6]. To apply these patterns to agentic data pipelines, it is necessary to create topic-based publish-subscribe systems in which agents produce structured events containing information about changes in the state of pipelines, diagnostic results, and remediation measures. The subscriber agents process the pertinent event streams according to their specialized duties, allowing a variety of simultaneous issues to be processed in parallel without center of bottlenecks. The message schema registry implements coordination between agent communications via versioned event definitions that change with the capabilities of the agent, and is backward incompatible with the existing agent population deployed in production infrastructure.

Agent Type	Primary Function	Key Capabilities	Integration Points
Reasoning Agents	Diagnostic analysis and strategy formulation	Pattern recognition, root cause identification, remediation planning, and impact assessment	Metadata catalogs, observability platforms, lineage graphs, knowledge repositories
Execution Agents	Infrastructure interaction and action implementation	Job submission, configuration modification, schema evolution, resource provisioning	Spark APIs, Flink interfaces, cloud provider services, storage layer APIs
Critic Agents	Validation and quality assurance	Performance benchmarking, correctness verification, regression detection, outcome evaluation	Metrics databases, historical baselines, statistical validation engines
Coordination Layer	Workflow orchestration and state management	Task queuing, agent registry, policy enforcement, and distributed consensus	Message brokers, consensus systems, configuration stores

Table 1: Multi-Agent System Architecture Components [3, 4]

3. Foundational Capabilities and Agent Processes

Autonomous schema evolution is a solution to one of the endemic operational problems in production data ecosystems where upstream systems reform output formats without liaising with downstream consumers. The necessity to have automated schema management is based on the frequency and the effect of schema change in contemporary data engineering procedures, where the process of manual resolution to such changes causes intolerable delays in data availability [7]. Reasoning agents identify

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/ **Research Article**

schema drift by continuously observing the execution logs of pipeline operations and data validation failures, comparing predicted schemas stored in metadata catalogs to real structures in incoming streams of data. Specific changes such as the addition of columns, type changes, changes in nullability, and reorganizations of nested fields have been identified by the analysis.

Impact assessment uses the full lineage graphs that identify the dependencies between data assets in enterprise analytics ecosystems to allow agents to assess the downstream impact of schema changes prior to making remediations. The workflow determines the consistency of workflow changes on either a backward-compatible evolution that can be met by simple schema extension operations or a breaking change that needs to be uniquely synchronized across transformation layers. In the case of compatible changes, the execution agents present schema evolution instructions to storage layer APIs, update metadata catalog entries, and cause reprocessing of data batches that have failed. Breaking changes are those where more complicated processes are triggered that include automated production of schema mapping logic, creation of backward-compatible views during transition times, and automated generation of pull requests with suggested transformation code adjustments and full test coverage.

The manual-skilled and expertise-based SQL performance tuning process has turned into a continuous automated intelligent optimization process through query optimization by agentic automation. Agents observe query execution statistics on production workloads and analyze query execution plans to detect opportunities to optimize them with inefficient join strategies, suboptimal partition pruning, and nonexistent predicate pushdown operations. The system retains historical performance benchmarks in the use of each query pattern, identifying degradations that are beyond the statistical cutoffs, and initiates diagnostic processes. Those reasoning agents consider the various possible root causes, such as data skew, out-of-date statistics, wrong parallelism settings, as well as inefficiencies in the logic of transformation.

The optimization process produces alternative query formulations that are semantically identical but run better in terms of performance by using methods like reordering joins using cardinality estimates, adding broadcast joins to small dimension tables, and optimizing window functions. The critical agents confirm optimisation suggestions with a shadow execution using representative samples of data, comparing performance metrics, and ascertaining the accuracy of results before optimisation can be permitted in production. This never-ending cycle of optimization allows data platforms to sustain the level of performance under the influence of the changing nature of data and the workload dynamics that would otherwise demand periodic hand-tuning intervention procedures that consume significant engineering resources.

Workflow Stage	**Activities**	**Outputs**	**Success Criteria**
Detection	Log monitoring, error pattern matching, schema comparison	Drift event notification, affected table identification	Anomaly detection within minutes of occurrence
Analysis	Structural diff computation, column-level change classification	Change inventory, compatibility assessment	Accurate identification of modification types
Impact Assessment	Lineage traversal, dependency mapping, compatibility evaluation	Downstream consumer inventory, breaking change identification	Complete dependency graph analysis

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/ **Research Article**

Strategy Generation	Remediation pattern matching, code generation, migration planning	Schema evolution commands, transformation updates, compatibility layers	Backward-compatible solutions for affected consumers
Implementation	Schema modification, code deployment, batch reprocessing	Updated metadata, modified transformations, and recovered pipelines	Successful execution without data loss
Validation	Output comparison, statistical verification, quality checking	Correctness confirmation, performance metrics	Result accuracy within tolerance thresholds

Table 2: Schema Drift Resolution Workflow Stages [5, 6]

4. Implementation Strategies and Technical Considerations

The choice and design of agent frameworks are the core determinants of the complexity of implementation, nature of operations, and the extent to which of autonomous data pipeline system can be achieved. Organizations have to compare various framework options based on the requirements in reasoning sophistication, execution reliability, operational overhead, and integration with the existing infrastructure. Modern search platforms have agentic retrieval features that offer automated query engine functionality for agent processes that can be interacted with in natural language with enterprise knowledge bases and also search semantically across documentation repositories [8]. These are capabilities that are necessary for reasoning agents, which need to reach historical records of incidents, libraries of transformation patterns, and business rules specific to the domain as they diagnose failure of pipelines and create remediation plans.

The implementation architecture needs to be sensitive in balancing the abstractions offered by the framework with those that are done manually to support the infrastructure-specific functions that need narrow control over execution semantics, error handling, and security constraints. The hybrid method of production deployments is often the adoption of high-level orchestration logic that uses the capabilities of frameworks to coordinate agents, managing conversations, and storing state, but binary agent implementations deal with direct interaction with built-in distributed compute engines, storage layer APIs, and observability platforms. This separation allows workflow management to take advantage of the productivity of the frameworks as well as retain the required control over the critical infrastructure processes in which execution reliability and performance characteristics directly affect the stability of the production systems.

There are high implementation issues in integration strategies of autonomous agents interacting with heterogeneous distributed compute engines, which call for the need to provide abstraction layers to normalize the concept of operation across different platforms. Agents will need to communicate with Apache Spark clusters via REST APIs to submit and monitor jobs, Apache Flink deployments via management interfaces to coordinate streaming applications, and serverless compute platforms via cloud provider APIs to provide resources and collect metrics. The abstraction layer design translates platform-specific concepts such as job status representations, resource metrics formats, and configuration parameter schema into unified models that allow reasoning logic to be based on without being bound to the details of the underlying infrastructure. This portability is critical because the enterprises are usually subject to hybrid environments across several compute engines that are chosen depending on the nature of the workload, cost optimization goals, and the existing technical investments.

The safety mechanisms introduced at every stage of the agent architectures shield the production environments against autonomous actions that could be destructive and allow the beneficial ones to continue their course without undue friction. The data quality agents that allow independent management in the enterprise setting should include advanced validation logic that can discriminate between normal optimization chances and high-risk processes necessitating human intervention [9]. Its implementation entails confidence scoring systems where agents can assess the level of certainty in a diagnostic conclusion and recommended remedies, and refer low-confidence cases to human review queues and automatically act when confident in the evidence. The blast radius, limited with scoped permissions, allows the development agents to access the production resources; the critical tables have more protection with the addition of more approval workflows, and the destructive operations, such as massive deletions, have to be prompted to use human confirmation without regard to the level of agent confidence. Detailed audit logging provides all the context associated with individual agent actions, such as triggering events, reasoning history, execution history, and validation history, which is useful in investigating the incident, meeting compliance criteria, and improving the quality of agent decision-making retrospectively.

Optimization Category	Detection Mechanisms	Remediation Techniques	Validation Methods
Join Strategy	Execution plan analysis, cardinality estimation review	Broadcast join introduction, join reordering, nested loop elimination	Shadow execution, performance comparison
Partition Pruning	Scan volume analysis, predicate evaluation	Filter pushdown, partition elimination refinement	Scanned data volume reduction verification
Data Skew	Task duration distribution analysis, partition size measurement	Salting strategies, repartitioning, adaptive execution	Balanced partition distribution confirmation
Resource Configuration	Memory utilization tracking, parallelism assessment	Executor sizing, task concurrency adjustment	Resource efficiency improvement validation

Table 3: Query Optimization Agent Capabilities [7, 8]

5. Performance Benchmarks and Experimental Results

The implementation of autonomous agentic systems in production deployments to enterprise data platforms shows significant operational efficiency, reliability, and cost management improvements over the conventional manual processes. The applicability of intelligent automation of data engineering processes is presented in an experimental assessment in various organizational settings and offers some measurable advantages of its utilization. The decrease in the mean time to recover from schema drift incidents represents perhaps the most radical operational change, in which automated detection and remediation reduce the resolution times from hours to minutes. Reasoning agents processing pipeline failure records spot schema incompatibility a few minutes after it happens, determine the downstream effect via lineage graph traversal, and use execution agents to execute the correct strategy of schema extensions, transformation updates, or compatibility layer deployments.

The efficiency of continuous automated performance tuning is confirmed by the fact that query optimization results are as high as those of periodic manual optimization cycles. Executions that keep track of the execution characteristics on production query workloads detect degradation trends as well as optimization opportunities that human engineers would not detect until large amounts of user impact occur. The automated detection, speedy optimization proposal creation, validation by shadow testing, and gradual rollout of production allow the systems to achieve the same levels of consistent performance even as the data characteristics change. Automated optimization of resources shows a substantial financial savings directly associated with intelligent cluster sizing, auto-scaling policy optimization, and workload consolidation that are constantly sought and applied by agents.

The optimization of resource allocation in a cloud computing environment by the use of multi-agent systems is a critical ability of managing costs, as well as maintaining the performance service levels [10]. The experimental outcomes justify the fact that autonomous agents are effective in balancing conflicting goals such as reducing compute costs, satisfying latency needs, and providing capacity overloads to workload bursts. Agents examine utilization patterns within cluster populations and can detect over-provisioned resources, inefficient auto-scaling policies, and workloads that can be consolidated onto shared infrastructure. Policy Agent-proposed optimization implementation is carried out by performing rigorous testing in shadow environments, canary releases to small workload subsets, and by a gradual rollout with active performance regression or service level breach monitoring.

The longitudinal analysis of agent performance growth proves learning effects as autonomous systems acquire a steadily optimizing performance during the course of their operation. Feedback on critical agent responses generates training data sets of successful remediations and failed efforts, allowing reasoning strategies to be refined continuously by use of reinforcement learning methods. As the knowledge base of agents grows through annotated examples, organizations find an increase in accuracy, a decrease in the false positive rate in anomaly detection, and a more discriminative insight into domain-specific failure modes. The reliability enhancements are seen in lowering failure rates as agents apply preventive optimizations and increasing the rate of success of auto-remediation in case of failures, gradually burdening the operational load on human engineering teams.

Optimization Strategy	Target Inefficiencies	Implementation Approach	Expected Benefits
Cluster Rightsizing	Over-provisioned memory and compute capacity	Executor reduction, memory allocation adjustment	Cost reduction while maintaining throughput
Auto-Scaling Policy Tuning	Inefficient scaling triggers, unnecessary capacity maintenance	Threshold adjustment, evaluation period optimization	Reduced idle capacity, improved responsiveness
Workload Consolidation	Underutilized dedicated clusters	Migration to shared infrastructure, serverless adoption	Infrastructure efficiency, operational simplification
Configuration Optimization	Suboptimal parallelism, inefficient shuffle	Parameter tuning based on workload characteristics	Execution time reduction, resource utilization improvement

Table 4: Resource Optimization Strategies and Outcomes [9, 10]

Conclusion

The creation of autonomous agentic data pipelines is a paradigm shift in the field of enterprise data engineering that creates self-managing lakehouse architectures that work with minimal human operation and are highly reliable and performant. Multi-agent systems that allocate tasks to particular reasoning, execution, and validation units offer useful architectural patterns, and organizations can successively adapt them in a staged deployment approach. Numerous production applications in wide-ranging enterprise scenarios confirm dramatic operational gains such as efficient incident resolution timeframes, huge cost savings through smart resource scheduling, and improved system dependability through constant automated tuning. The architectural beliefs in the message-passing coordination, an open line of reasoning, and the full safety measures lay strong grounds for the implementation of autonomous systems in the production settings where mission-critical analytics workloads will be handled. Guidelines on implementation in the choices of framework, as well as patterns of infrastructure integration and continuous learning, allow data engineering teams to overcome complexity and reap the advantages of automation. Future directions include proactive optimization, which leverages predictive analytics, federated architectures, which coordinate agents at organizational boundaries, patterns of collaboration between human and agent, causal inference-based integration of robust diagnostics, and multi-modal reasoning, which uses visual analysis capabilities. Adopters successfully running agentic pipelines outcompete competitors with faster time to insight, lower operational costs, and the ability to operate a massive platform at scale that requires an equivalent engineering team size. The technical underpinnings put in place by the current generations of CI/CD allow clear paths to adoption, while ceaseless introduction considers innovation against operational prudence across extensive testing, phased-out strategies, and ongoing monitoring so that reliability of automation hits production. As the effectiveness of language models is improved and more domain-specific training data is developed, autonomous operations will grow to operate more advanced data platforms while freeing human expertise to guide strategic innovation rather than reactively maintain them.

References

[1] Jan-Micha Bodensohn et al., "Unveiling Challenges for LLMs in Enterprise Data Engineering," arXiv:2504.10950v2 [cs.DB], 2025. [Online]. Available: https://arxiv.org/pdf/2504.10950

[2] Praveen Kumar Reddy Gujjala, "Data science pipelines in lakehouse architectures: A scalable approach to big data analytics," World Journal of Advanced Research and Reviews, 2022. [Online]. Available: https://wjarr.com/sites/default/files/WJARR-2022-1305.pdf

[3] Neelamegam et al., "Multi-Agent Systems for Autonomous IoT Network Management Using Distributed Reinforcement Learning," 2025 3rd International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC), 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10969204

[4] Pouya Pezeshkpour et al., "Reasoning Capacity in Multi-Agent Systems: Limitations, Challenges and Human-Centered Solutions," arXiv:2402.01108v1 [cs.CL], 2024. [Online]. Available: https://arxiv.org/html/2402.01108v1

[5] Chad Kittel and Clayton Siemens, "AI agent orchestration patterns," Microsoft Learn, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/ai-agent-design-patterns

[6] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, "Message-Passing Approach," Encyclopedia of Physical Science and Technology (Third Edition), 2003. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/message-passing-approach

[7] Rajkumar Sekar, "The need for auto schema evolution in modern data engineering: Challenges and solutions," World Journal of Advanced Research and Reviews, 2025. [Online]. Available: https://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-1115.pdf

[8] Matt Gotteiner, "Introducing agentic retrieval in Azure AI Search," Microsoft Tech, 2025. [Online]. Available: https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/introducing-agentic-retrieval-in-azure-ai-search/4414677

[9] Krishna Kumar Subramanian, "Data Quality 'Agents' enabling Autonomous Data Management within the enterprise," LinkedIn, 2025. [Online]. Available: https://www.linkedin.com/pulse/data-quality-agents-enabling-autonomous-management-subramanian-zopvc/

[10] Gopal K. Shyam and Priyanka Bharti, "Multi-agent Systems for Resource Allocation in Cloud Computing," 2023 IEEE International Conference on Contemporary Computing and Communications (InC4), 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10262945