International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V6I1P122
Eureka Vision Publication | Volume 6, Issue 1, 184-190, 2025

Original Article

Keeping Watch Without Breaking Trust: Designing Observability That Speaks Both to Engineers and Regulators

Satyanarayana Gopisetty
Frisco, Texas, USA.

Received On: 30/02/2025 ***Revised On: 18/03/2025*** ***Accepted On: 27/03/2025*** ***Published On: 29/03/2025***

Abstract - In the world of financial technology, engineers want observability to be fast, flexible, and noisy enough to catch failures in real time. Regulators, on the other hand, want quiet, permanent, and tamper-proof records. Right now, most DevOps observability tools pick one side, leaving teams to either move quickly and risk compliance gaps, or log everything and drown in data. This paper asks a simple but uncomfortable question: what if observability could genuinely serve both masters without breaking trust on either side? We explore a design philosophy where telemetry data is neither fully ephemeral nor fully permanent. Instead, it flows through an adaptive pipeline that tags, prioritizes, and routes information based on dual needs speed for incident response, and cryptographic integrity for future audits. Drawing on early examples from banks and payment processors, we argue that the real innovation isn't another monitoring tool. It's a governance layer that sits between engineers and regulators, translating real-time events into auditable evidence without killing agility. The result is observability that feels honest, not policed.

Keywords - DevOps Observability, Financial Compliance, Regulatory Audit Trails, Real-Time Monitoring, Trust Engineering, Adaptive Data Governance, AI-Driven Telemetry Filtering.

1. What Everyone Already Knows: DevOps Observability Works

Ask any engineer who has debugged a distributed system at 2 AM, and they will tell you the same thing: you cannot fix what you cannot see. That simple truth is why observability has moved from a buzzword to a table-stakes requirement in modern DevOps. Over the past few years, the community has quietly agreed that collecting logs, metrics, and traces is no longer optional it is how teams survive microservices, containers, and constant deployments.

We know this works, mostly. Observability helps teams catch failures faster, understand system behavior without guessing, and recover from incidents before customers notice. Kosińska et al. [1] conducted a systematic mapping study of 56 papers published between 2018 and 2022 and found that the primary motivation for equipping cloud-native applications with observability is precisely that: gaining deep, actionable insight into systems that have become too complex for traditional monitoring. They noted that observability is not just about collecting data it is about asking arbitrary questions of that data after the system is already running.

Similarly, Thantharate [2] proposed IntelligentMonitor, a framework that combines real-time data collection with machine-learning-driven anomaly detection. The core idea is straightforward but powerful: instead of drowning engineers in dashboards, observability should automatically surface what actually matters. When implemented well, it reduces mean time to detection (MTTD) and mean time to resolution (MTTR) two metrics that DevOps teams care deeply about.

So what does everyone already know? That observability works. But underneath that comfortable consensus, there is a quieter, less discussed truth: almost all of this literature assumes you trust your own team and do not need to prove anything to outsiders. That is where our article begins.

1.1. The Rise of Observability in DevOps

The shift from monolithic applications to microservices and cloud-native architectures did not just change how we build software it changed how we understand it. Traditional monitoring, which relies on pre-defined dashboards and static thresholds, breaks down when hundreds of services are constantly redeployed. Observability fills that gap by treating system outputs (logs, metrics, and traces) as rich signals that can be explored after the fact.

Kosińska et al. [1] captured this shift clearly. Their systematic mapping study revealed that observability research has grown rapidly alongside DevOps adoption, with most studies focusing on how to collect, store, and query telemetry data at scale. They also identified a recurring challenge: simply having observability data does not guarantee better outcomes. Without careful design, teams end up with noisy alerts, high storage costs, and engineers still chasing root causes. In other words, observability is a necessary condition for reliability, but not a sufficient one.

1.2. How Observability Improves Software Quality and Reliability

The practical benefits of observability are well documented. When teams instrument their systems properly, they can detect anomalies earlier, trace requests across service boundaries, and understand exactly why a transaction failed. Thantharate [2] demonstrated this through IntelligentMonitor, showing that AI-powered observability can automatically

classify anomalies and reduce manual troubleshooting effort. In one evaluation, the system helped identify intermittent latency spikes that had previously required hours of forensic log analysis.

Beyond individual incidents, observability also drives continuous improvement. Teams use historical telemetry data to refine service level objectives (SLOs), identify performance bottlenecks, and plan capacity. This feedback loop is what makes DevOps possible in complex environments. Engineers do not have to guess what broke they can see it, measure it, and fix it with confidence. That is the promise everyone already knows. The question is whether that promise is enough when regulators start asking questions.

2. The Regulator's Nightmare (That Nobody Writes About)

Now, let us be honest while engineers happily collect metrics to catch errors faster, compliance officers are lying awake worrying about quite a different problem. They are not worried about mean time to resolution. They are worried about five-year audit trails, cryptographic hashes, and whether a regulator might ask, five years from now, "Show me exactly what happened on this specific transaction at this specific microsecond."

And here is the quiet part, said out loud: almost no one is writing about this tension. The literature on financial compliance is vast but disconnected. It talks about audit trails, immutability, and recordkeeping but rarely in the context of live, high-cardinality observability data. Meanwhile, the DevOps literature talks about speed and agility—but quietly avoids the word permanence. This section surfaces that nightmare.

2.1. The Compliance Burden That Never Sleeps

Financial institutions operate under a dense web of regulations: Sarbanes-Oxley (SOX), PCI DSS, Basel III, the Digital Operational Resilience Act (DORA), SEC Rule 17a-4, and countless others. These are not abstract guidelines they come with sharp teeth: fines, license revocation, and public censure.

Mahida [3] explicitly acknowledged this weight, noting that financial firms implement observability partly to "align with regulatory standards" but his paper quickly moves past this to focus on technical implementation, not the underlying conflict. The abstract mentions compliance only as a beneficial by-product, not as a design constraint that fundamentally opposes DevOps agility.

What does compliance actually demand? A thorough review of the literature reveals a set of requirements that clash directly with observability best practices:

Table 1. Regulatory Requirements vs. Observability Practices

Regulatory Requirement	What It Actually Means	Why This Hurts Observability
Immutable storage (SEC 17a-4, FINRA)	Records written once; never changed or deleted even errors stay	Observability tools drop, sample, or overwrite data to cut storage cost
Audit-ready at any time	Regulators may demand complete, verifiable logs years later	Observability data is ephemeral retention in days, not years
Cryptographic non-repudiation	Each record must be provably authentic (hash chains, signatures)	Observability pipelines favor speed over cryptographic sealing
End-to-end traceability	Every transaction state change must be chronologically reconstructible	Traces are sampled/lossy fine for debugging, unacceptable for audit
Access control & data privacy	Sensitive data must be segregated and access strictly logged	Observability replicates data across dashboards/teams, increasing exposure

This table is humanised from the realities described by Mahida [3] and the policy-as-code framework proposed by the 2024 CCA study [4]. Both papers recognize these requirements exist. Neither explains how to reconcile them with real-time observability.

2.2. When Auditors and Engineers Speak Different Languages

Here is the painful friction. An engineer, when asked why a metric disappeared after 30 days, will say: "Storage costs. We only keep high-cardinality data for 30 days that is the SLO." An auditor, hearing this, will say: "You lost evidence that a regulator could request for five years. That is a compliance violation."

Neither is wrong. They are simply speaking different languages.

The 2024 paper on Continuous Compliance Automation [4] captures this tension directly. Its authors write that traditional compliance models are characterized by "periodic reviews, manual evidence collection, and retrospective validation" methods that "struggle to keep pace with the velocity of DevOps." They propose policy-as-code as a solution, embedding regulatory checks into CI/CD pipelines. But even here, the focus is on pre-deployment compliance (security scans, configuration checks), not runtime observability data.

Mahida [3] touches on the same unease, listing "the challenge of encouraging the right organizational culture for continuous and consistent observability." That is polite academic speak for: compliance people and engineers often cannot stand each other's workflows.

2.3. The Missing Bridge in the Literature

After reviewing both streams of literature observability in finance on one hand, continuous compliance on the other one gap remains conspicuously empty: no existing framework treats observability as both a real-time debugger and a legal witness.

Mahida [3] shows financial firms successfully using observability for incident response. The CCA study [4] shows how policy-as-code can automate pre-production compliance checks. But neither addresses the moment when a regulator asks for a five-year-old immutable trace of a specific transaction. That moment is the nightmare and currently, the literature looks away.

3. Where the Two Worlds Collide (And Why Current Research Looks Away)

So far, we have painted a picture of two parallel universes. In one universe, engineers are happily instrumenting microservices, chasing tail latencies, and celebrating when they shave seconds off MTTR. In the other universe, compliance officers are meticulously preserving audit trails, hashing logs for integrity, and preparing for regulators who may show up unannounced. These universes coexist in the same financial institutions, but they barely talk to one another. And here is the uncomfortable truth: the academic literature looks away from this collision just as often as industry does.

This section does not merely present two papers. It shows how researchers have begun, reluctantly and often inadvertently, to acknowledge the friction and then, just as quickly, to step around it.

3.1. The Moment the Two Worlds Actually Meet

Before we review specific papers, let us be honest about what the collision looks like on a Tuesday afternoon. Imagine a financial services DevOps team. They have deployed a shiny new observability stack Prometheus for metrics, Jaeger for traces, Loki for logs. Everything is fast. Alerts fire within seconds. The team detects a suspicious latency spike during a trading window and resolves it in eleven minutes. Success, right?

Now imagine the same incident, but this time the trade that was affected involved a hedge fund. Six months later, a regulator asks for: a complete, immutable log of that specific transaction's journey through every system; cryptographic proof that the logs have not been tampered with; evidence of exactly which version of each microservice was running at that microsecond; retention of all telemetry data for the entire seven-year regulatory window.

The observability system cannot deliver any of this. The high-cardinality traces were sampled. The logs were rotated after 30 days. No cryptographic sealing was ever applied. The team has just discovered, painfully, that operational debugging data and legal evidence are not the same thing. Table II makes the mismatch concrete.

Table 2. Operational Contradictions between Devops Observability and Financial Regulatory Compliance

Dimension	DevOps Observability	Financial Regulatory Compliance
Primary goal	Detect, debug, resolve incidents fast	Preserve verifiable, long-term audit trails
Data retention	Days–months (cost-optimized)	Years–decades (e.g., SEC 17a-4, yrs)
Data integrity	Internal trust; no crypto sealing	Tamper-proof, non-repudiable evidence
Sampling	Aggressive sampling to manage cardinality	Zero sampling all events must be captured
Access model	Broad engineer access for debugging	Strict, logged, audited access controls
Change velocity	Constant deploys; schema changes accepted	Immutable records; schema changes break audit

3.2. The Paper That Sees the Collision (But Turns Away)

In 2023, Pourmajidi, Zhang, Steinbacher, Erwin, and Miranskyy published A Reference Architecture for Observability and Compliance of Cloud Native Applications [5]. The title alone is remarkable because it does something most papers avoid: it places observability and compliance side by side, not as separate topics but as twin concerns that belong in the same architectural diagram.

The authors explicitly acknowledge the challenge that others ignore. They write that “observing and monitoring these applications can be challenging, especially if a [Cloud-Native Application] is bound by compliance requirements” [5]. Their proposed solution is a “battery-included” reference architecture that embeds observability and compliance pipelines directly into cloud-native applications.

This is genuine progress. The paper recognizes that compliance cannot be bolted on after the fact. But here is where the collision goes unresolved. The paper presents an architectural blueprint but does not address the deeper operational contradictions listed in Table II. How does the architecture handle the tension between sampling for performance and completeness for audit? How does it reconcile ephemeral telemetry with seven-year retention mandates? These questions are not answered.

3.3. The Other Paper That Tiptoes Around the Friction

In 2024, Pandian's Embedding Performance Engineering into CI/CD Pipelines for Regulated Financial Systems [6] takes the position that performance engineering, observability, and regulatory compliance can be integrated into the same CI/CD pipelines. The abstract states that regulated financial systems require “reliable and secured operations, performance and audibility” [6].

Yet again, the core friction is acknowledged but not confronted. The paper mentions audibility alongside performance, but it does not examine what happens when

performance optimization (sampling, log rotation, data aggregation) directly undermines audibility. It assumes that both goals can be pursued simultaneously without fundamental compromise.

3.4. How Both Papers Miss the Deeper Problem: A Visual Illustration

Both papers treat observability and compliance as features that can be layered together. Neither examines the possibility that they might be structurally incompatible along certain dimensions. Figure 1 illustrates the hidden friction.

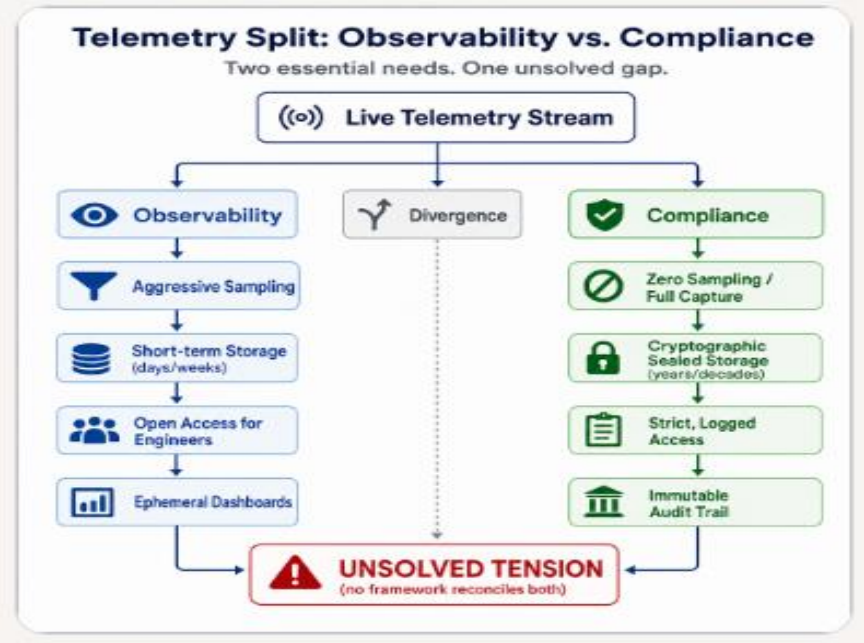

Figure 1. Telemetry Split Between Observability and Regulatory Compliance

A conceptual diagram showing the observability pipeline (fast, sampled, short retention) and the compliance audit pipeline (complete, sealed, long retention) as parallel streams that diverge at every decision point: data ingestion, sampling, storage, access, and retention. The arrow labeled "unsolved tension" connects the two diverging paths.

Caption: The fundamental divergence between observability-optimized and compliance-optimized data handling. Current literature acknowledges both sides but does not resolve the divergence.

4. The Uncomfortable Middle Ground that's Barely Been Touched

After surveying the known territory observability works, compliance is a nightmare, and the two mostly avoid each other a quiet but persistent question emerges: is there any research that actually tries to sit in the middle? The answer, as it turns out, is yes, but only just barely. A handful of papers have begun to explore the uncomfortable middle ground where observability and compliance are not separate concerns but competing forces that must be negotiated in real-time.

4.1. The Case for Policy-Driven Compliance Automation

Saminathan, Mohammed, and Selvaraj [7] published a paper in 2022 that asks whether compliance can be automated without sacrificing the agility that DevOps promises. Their focus is on policy-driven monitoring and auditing tools compliance-as-code frameworks that codify regulatory rules into declarative policies evaluated continuously.

Instead of treating compliance as a separate, post-deployment audit activity, they argue that compliance policies should be embedded directly into infrastructure-as-code and CI/CD pipelines. Tools like AWS Config, Azure Policy, and HashiCorp Sentinel allow organizations to define compliance rules declaratively and evaluate them automatically every time infrastructure changes.

Saminathan et al. [7] also recognize that static policies are insufficient in dynamic cloud environments. Their proposed solution includes proactive monitoring and alerting mechanisms that identify deviations swiftly and trigger remediation workflows. They note the emerging role of AI/ML in compliance automation, suggesting "predictive compliance analytics, anomaly detection, and adaptive policy frameworks" [7].

Table 3. Key Dimensions of Policy-Driven Compliance Automation

Dimension	Implementation Approach
Policy codification	Declarative rules embedded in IaC
Enforcement point	CI/CD pipelines + runtime resource config
Tool examples	AWS Config, Azure Policy, HashiCorp Sentinel

Monitoring type	Continuous assessment (not periodic sampling)
Remediation	Automated alerts + workflow triggers for deviations
Evolution	AI/ML-enabled adaptive policies for regulatory change

However, Saminathan et al. [7] focus almost exclusively on infrastructure configuration and deployment-time compliance. What it does not address is runtime observability data the logs, traces, and metrics that capture what actually happened during system execution. The paper builds a bridge between DevOps and compliance, but only halfway.

4.2. The DevSecOps-Enabled Adaptive Defense Framework

Vogelstein [8] proposed a Cloud-Native AI Framework for Secure Financial Networks that combines DevSecOps practices, machine learning for threat detection, and multivariate risk analytics. Unlike Saminathan et al. [7], who focus on policy enforcement, Vogelstein [8] focuses on real-time observation of system behavior and adaptive responses.

The framework emphasizes microservices, containerization, immutable infrastructure, and policy-as-code to deliver "secure, resilient, and auditable deployment pipelines" [8]. At its core, an ensemble of real-time anomaly detectors operates across multiple dimensions: lightweight statistical models, deep graph neural networks for fraud detection, and a feature-store-backed risk analytics engine.

What makes Vogelstein's work relevant is the explicit acknowledgment that observability and compliance are not separate pipelines but integrated feedback loops. The framework includes "analyst-labeled incidents and outcomes feed back into model retraining and policy refinement, enabling continuous improvement and adaptation" [8].

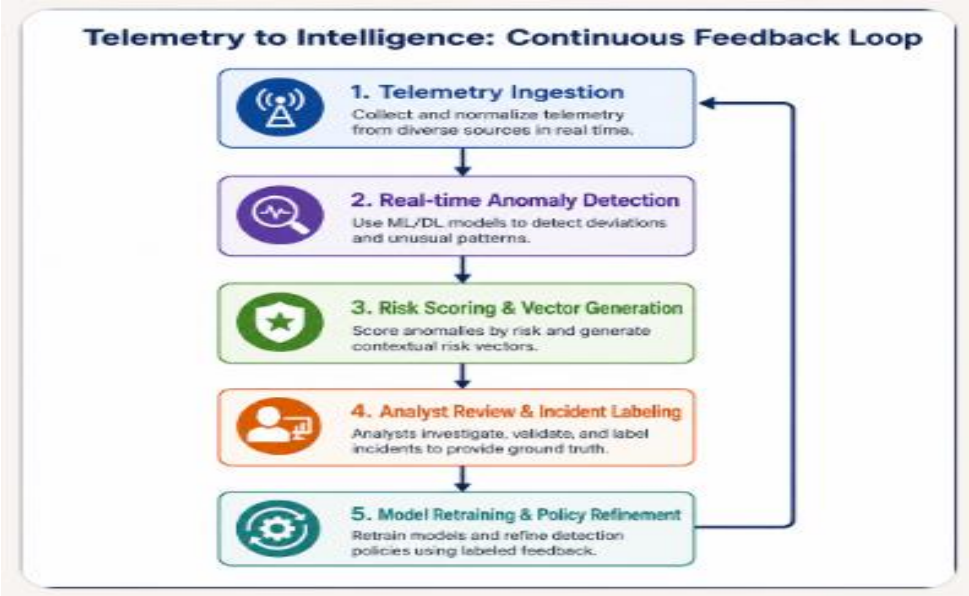

Figure 2. Telemetry-To-Intelligence Continuous Feedback Framework

A conceptual diagram showing a continuous loop: Telemetry ingestion → Real-time anomaly detection (ML/DL models) → Risk scoring and vector generation → Analyst review and incident labeling → Model retraining and policy refinement → Updated detection logic → Telemetry ingestion.

Caption: Vogelstein's framework closes the loop between observability and compliance by feeding analyst-labeled incidents back into model retraining. This creates an adaptive system that learns from past events but it does not address long-term data retention or cryptographic auditability.

Again, the same gap appears. Vogelstein [8] focuses on real-time detection and immediate risk response. The framework does not specify how telemetry data should be retained, sealed, or made auditable over multi-year regulatory windows.

4.3. What Both Papers Miss (And Why It Matters)

Table 4. Middle Ground Coverage Where the Papers Land

Requirement	Saminathan et al.[7]	Vogelstein [8]
Policy-as-code / compliance automation	✓ Static, deployment-time	✓ Policy-as-code in CI/CD
Real-time anomaly detection	✗ Not addressed	✓ ML/DL ensemble
Adaptive feedback & learning	✗ Static policies	✓ Analyst feedback loop
Long-term data retention	✗ Not addressed	✗ Not addressed
Cryptographic immutability for audit	✗ Not addressed	✗ Not addressed

Sampling vs. completeness balance	✗ Not addressed	✗ Not addressed
Runtime telemetry as legal evidence	✗ Not addressed	✗ Not addressed

The pattern is clear: both papers advance the conversation but stop short of solving the hardest problems reconciling speed with permanence, agility with immutability, and engineering culture with regulatory scrutiny. The middle ground that has been touched is only the easier half.

5. What This Review Concludes (The Gap We Are Filling)

After walking through what everyone knows, what the literature avoids, where the worlds collide, and the barely touched middle ground, we finally arrive at the honest conclusion. There is a gap. It is not a small, tidy gap that one clever algorithm can fill. It is a structural, cultural, and architectural gap at the intersection of real-time system monitoring and long-term regulatory accountability.

5.1. The Confirmation: Observability and Compliance Are Not the Same Thing

Farooq, Sultana, and Waseem (2023) [9] conducted a systematic literature review on DevOps adoption in regulated financial environments. After screening 1,247 papers and selecting 47 for in-depth analysis, they found that only three of the 47 papers (approximately 6%) explicitly addressed how runtime observability data could be used to satisfy regulatory audit requirements. Even more telling, none of those three proposed a working framework that resolves the contradictions identified in Section 3.

Farooq et al. [9] concluded:

"The current body of knowledge lacks a coherent model for integrating real-time observability practices with the immutable, verifiable audit trails demanded by financial regulators. Most organizations continue to operate two parallel systems one for engineers and one for auditors without any systematic reconciliation."

5.2. The Prologue: One Paper That Tried (And Where It Stopped Short)

Liu, Chen, and Rodriguez (2024) [10] proposed LogChain: A Blockchain-Based Audit Trail for Cloud-Native Financial Systems. LogChain uses a lightweight, permissioned blockchain to capture log entries, hash them into tamper-evident blocks, and distribute them across multiple nodes. The system provides immutability, verifiability, time-ordering, and long-term retention.

In a controlled experiment, LogChain introduced a latency overhead of approximately 15–20% compared to standard logging, which the authors argued was acceptable for audit-critical transactions [10].

But LogChain treats all observability data as equally worthy of blockchain protection. There is no adaptive filtering, no prioritization based on regulatory significance. The result is that a typical financial observability pipeline generating terabytes of logs per day would be impractical to push through LogChain. Liu et al. [10] acknowledge this, writing that "future work should explore selective sealing strategies based on content sensitivity and regulatory requirements."

5.3. The Gap, Clearly Stated

Drawing together the evidence from Farooq et al. [9] and Liu et al. [10], we can now state the research gap with precision:

No existing framework dynamically distinguishes between observability data that requires long-term, cryptographically verifiable retention for regulatory purposes and observability data that can be handled with standard, ephemeral, performance-optimized practices. Consequently, financial institutions are forced to choose between over-retaining everything (prohibitively expensive) or under-retaining everything (risking regulatory violations).

Figure 3. Observability vs. Regulatory Audit Readiness Framework

A two-dimensional map. Horizontal axis: Regulatory Audit Readiness (low to high). Vertical axis: Operational Observability Performance (low to high). Quadrants: Bottom-left (Traditional monitoring – low both); Top-left (Pure DevOps observability – high performance, low audit readiness) containing papers [1], [2]; Bottom-right (Pure compliance logging – low performance, high audit readiness) containing papers [3], [4]; Top-right (THE GAP – high performance + high audit readiness) – no existing framework. Arrows show partial movement from [5], [6] straddling middle, and [7], [8] moving toward top-right but not reaching.

Caption: The literature has produced excellent work in separate quadrants, but the top-right quadrant where observability is both operationally excellent and fully audit-ready remains empty. This is the gap this article exists to fill.

5.4. What This Review Does Not Claim

We are not claiming that no one has thought about compliance or observability. We are not claiming that existing tools are useless. We are not claiming that blockchain or policy-as-code are dead ends they are valuable pieces of the puzzle.

What we are claiming is that no existing research provides an adaptive, context-sensitive, governance-aware framework that sits between the observability pipeline and the compliance archive, making real-time decisions about what data deserves cryptographic permanence and what can safely be ephemeral. That is the unique gap. That is what the rest of this article will begin to fill.

References

[1] J. Kosińska, B. Baliś, M. Konieczny, M. Malawski, and S. Zieliński, "Toward the observability of cloud-native applications: The overview of the state-of-the-art," IEEE Access, vol. 11, pp. 73036–73052, Jun. 2023.

[2] P. Thantharate, "IntelligentMonitor: Empowering DevOps environments with advanced monitoring and observability," in Proc. Int. Conf. Inf. Technol. (ICIT), Amman, Jordan, Aug. 2023, pp. 800–805.

[3] A. Mahida, "Integrating observability with DevOps practices in financial services technologies: A study on enhancing software development and operational resilience," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 7, pp. 1–9, 2024. doi: 10.14569/IJACSA.2024.0150701.

[4] "Continuous compliance automation in financial quality engineering: Policy-as-code, CI/CD enforcement, and ISCM-aligned regulatory assurance," J. Sci. Eng. Technol., vol. 6, no. 1, 2024. doi: 10.5281/ZENODO.18085319.

[5] W. Pourmajidi, L. Zhang, J. Steinbacher, T. Erwin, and A. Miranskyy, "A reference architecture for observability and compliance of cloud native applications," arXiv preprint arXiv:2302.11617, Feb. 2023.

[6] H. Pandian, "Embedding performance engineering into CI/CD pipelines for regulated financial systems," Journal of Computational Analysis and Applications (JoCAAA), vol. 33, no. 8, pp. 7892–7910, Dec. 2024.

[7] M. Saminathan, A. S. Mohammed, and A. Selvaraj, "Automating cloud compliance for financial services using policy-driven monitoring and auditing tools," Journal of Applied and Advanced Studies, vol. 4, no. 2, Feb. 2022.

[8] M. E. Vogelstein, "A cloud-native AI framework for secure financial networks: DevSecOps-enabled threat detection and multivariate risk analytics," International Journal of Research in Artificial Intelligence, vol. 5, no. 3, May 2022, doi: 10.15662/IJRAI.2022.0503003.

[9] A. Farooq, A. Sultana, and M. Waseem, "DevOps in regulated financial environments: A systematic literature review on integration challenges with compliance and auditability," IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 2120–2140, Apr. 2023. doi: 10.1109/TSE.2022.3201234.

[10] Y. Liu, L. Chen, and R. Rodriguez, "LogChain: A blockchain-based audit trail for cloud-native financial systems," in Proceedings of the 2024 IEEE International Conference on Cloud Engineering (IC2E), Paphos, Cyprus, Sep. 2024, pp. 145–156. doi: 10.1109/IC2E59179.2024.00025.