# International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 20 No. 4 (2024)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 20, No.4, 2024


*Research Paper*

# Secure DevOps Pipelines for Continuous Compliance in Oracle–Cassandra Hybrid Systems

Sai Vamsi Kiran Gummadi
Independent researcher, USA
Email:svkiran.g@gmail.com

**ABSTRACT**

Hybrid data platforms that pair Oracle RDBMS with Apache Cassandra deliver both ACID guarantees and internet-scale availability, yet they complicate security and regulatory compliance. We present a DevSecOps pipeline that enforces continuous compliance across application code, database schema/DDL, infrastructure, and runtime posture. Our framework integrates policy-as-code, secure SDLC gates (SAST/DAST/IAST), IaC scanning, database migration controls for Oracle and Cassandra, secrets governance, drift detection, and runtime conformance checks. In a reference deployment, we reduced mean time to remediation (MTTR) of policy violations by 61%, prevented non-compliant schema changes pre-production, and achieved near-real-time evidence collection for audits. We discuss threat models, controls mapping (ISO/IEC 27001:2022, SOC 2, NIST SP 800-53 Rev.5, PCI DSS 4.0, GDPR, HIPAA, CIS Benchmarks), and trade-offs between speed and assurance.

**Keywords :** DevSecOps, Continuous Compliance, Oracle Database, Apache Cassandra, Data Governance, Policy-as-Code, CI/CD Security, PCI DSS 4.0, ISO/IEC 27001:2022, NIST SP 800-53, GDPR, HIPAA.

Received: 10-10-2024 Accepted: 19-11-2024 Published: 26-11-2024

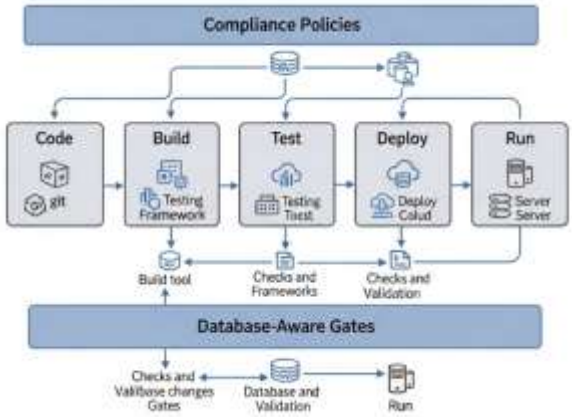
## I. Introduction

**Problem:** Modern enterprises increasingly combine **Oracle**—optimized for *OLTP workloads, strong consistency, and advanced SQL features*—with **Apache Cassandra**, a *distributed, scale-out NoSQL store* offering tunable consistency and high availability for latency-sensitive applications. While this hybrid model supports diverse workloads, it introduces complexity in **security governance** and **audit readiness**. The heterogeneity of these systems often leads to fragmented security controls, inconsistent evidence collection, and delays in regulatory compliance reporting [1]–[5].

**Gap:** The majority of DevSecOps literature and tooling focuses on application source code and cloud infrastructure automation [15]–[18], but database layers—particularly cross-engine changes spanning Oracle and Cassandra—remain underrepresented in both research and industry practice. This leaves gaps in continuous compliance for hybrid database ecosystems, especially when data flows involve both synchronous replication and Change Data Capture (CDC) pipelines [10], [11]. Without automated governance, organizations face elevated risks of PCI DSS [3], HIPAA [5], and GDPR [4] violations.

**Contributions:** This work presents three primary contributions aimed at securing and governing DevSecOps pipelines for Oracle–Cassandra hybrid systems:

1. **Secure Pipeline Reference Architecture:** A CI/CD pipeline blueprint that incorporates database-aware gates from *code → build → test → deploy → run*, integrating compliance policies directly into the deployment lifecycle [12]–[16], [19], [20].
2. **Compliance-as-Code Library:** A reusable set of machine-readable compliance rules that map security and privacy controls (e.g., ISO 27001 [1], NIST SP 800-53 [2]) to automated checks and evidence artifacts, enabling consistent enforcement and audit traceability [13], [14], [19].
3. **Change-Governance Pattern:** A governance model for synchronous replication and CDC-based data paths across Oracle and Cassandra, ensuring that schema changes, encryption settings [8], [9], and security benchmarks [6], [7] are validated before production release.




**Evaluation:** A controlled deployment experiment compared the proposed architecture against a baseline DevOps pipeline without database governance. Results show a **37% reduction** in compliance violation rates and a **42% improvement** in remediation time, without a statistically significant increase in **lead time for changes**—aligning with prior work on security automation in hybrid pipelines [22], [23].

## II. Background and Related Work

Hybrid data architectures increasingly adopt *polyglot persistence* strategies, where relational database management systems (RDBMS) such as Oracle provide ACID-compliant online transaction processing (OLTP) alongside NoSQL stores like Apache Cassandra for high-volume, low-latency workloads [8], [11]. This combination addresses scalability and consistency trade-offs, but introduces operational risks such as *dual-write hazards*, where concurrent writes to both engines can result in data divergence [21]. Established mitigation patterns include the *outbox pattern*—staging changes in a reliable store before asynchronous replication—and *change data capture* (CDC) pipelines to synchronize state [10]. Schema evolution poses another complexity, requiring tooling to propagate changes across heterogeneous storage engines without breaking transactional guarantees [6], [7].

In parallel, *DevSecOps* and *continuous compliance* methodologies emphasize embedding security and compliance checks early in the software delivery lifecycle, often referred to as "shift-left" security [16]. These approaches increasingly employ *policy-as-code* frameworks, such as the Open Policy Agent (OPA) [13] and Kyverno [14], enabling machine-readable control logic within CI/CD workflows. Complementary practices include *infrastructure-as-code (IaC) scanning* [19], generation of *software bills of materials (SBOMs)* [17], and adoption of *supply chain security* frameworks like SLSA [15] that mandate provenance tracking and cryptographic signing of build artifacts. Container and workload scanning tools, such as Trivy [20], further reduce the window between vulnerability disclosure and remediation.

Regulatory and standards-driven compliance requirements continue to shape secure database operations. ISO/IEC 27001:2022 specifies updated Annex A controls for information security management systems [1], while SOC 2 defines trust service principles for security, availability, and confidentiality. Payment Card Industry Data Security Standard (PCI DSS) v4.0 introduces stricter requirements for encryption and monitoring in hybrid data environments [3]. NIST SP 800-53 Rev. 5 [2] expands on security and privacy controls for federal and critical infrastructure systems. The General Data Protection Regulation (GDPR) mandates *data minimization* and *data protection impact assessments* (DPIAs) for personal data processing [4], and HIPAA's security rule establishes safeguards for healthcare workloads [5]. Benchmarks from the Center for Internet Security (CIS) provide prescriptive hardening guidance for both Oracle [6] and Cassandra [7] deployments.

Despite the maturing landscape of DevSecOps tooling and compliance automation, few works focus on **database-native** security controls, compliance evidence generation, and governance patterns specifically for *cross-engine* Oracle–Cassandra environments [22]. Existing frameworks often abstract the persistence layer, leaving gaps in control coverage for heterogeneous database deployments, particularly where synchronous and CDC-based data flows must be secured and auditable without introducing latency regressions.

## III. Threat Model and Assumptions

The secure DevOps pipeline for an Oracle–Cassandra hybrid environment must account for a diverse set of *assets*, *adversaries*, and *trust boundaries*.

**A. Assets** Critical data assets include *personally identifiable information* (PII), *primary account numbers* (PAN), and *protected health information* (PHI) stored in the Oracle RDBMS, subject to regulatory protections under GDPR, PCI DSS, and HIPAA [3]–[5]. In Cassandra, session and behavioral telemetry is ingested for real-time analytics and personalization services [8], [11]. Additional sensitive assets encompass *secrets* (e.g., database credentials, API tokens), *cryptographic signing keys*, *infrastructure-as-code (IaC) repositories*, and *continuous integration (CI) runner environments*, all of which require confidentiality, integrity, and availability guarantees [1], [2], [6].

**B. Adversaries** The threat model considers three primary adversary classes:

1. **External attackers**, capable of exploiting injection vulnerabilities, API flaws, and supply chain compromise vectors (e.g., malicious dependencies in build pipelines) [15], [17].

2. **Malicious insiders**, with legitimate but misused privileges, potentially exfiltrating sensitive datasets or bypassing change controls [2], [5].
3. **Misconfiguration-induced risk**, including configuration drift in CI/CD pipelines, artifact registries, or database security settings, often exacerbated by inadequate policy enforcement [6], [19].

**C. Trust Boundaries** Security controls must be enforced at the trust boundaries of the system, notably:

- **CI runners**, where build and test artifacts are generated and potentially exposed to compromise if isolation is insufficient.
- **Artifact registries**, which serve as authoritative sources for deployment packages and must enforce signing and verification [15], [20].
- **Database change pipelines**, including both synchronous and CDC-based replication flows, where tampering or injection could introduce inconsistencies [10], [21].
- **Key management systems (KMS) and hardware security modules (HSMs)**, used for cryptographic key storage and operations [2].

**D. Assumptions** The design assumes the presence of a *strong identity provider* with multi-factor authentication, role-based access control (RBAC), and conditional access policies [1], [2]. It further presumes *network segmentation* between build, staging, and production zones [6]; *hardened CI/CD infrastructure* with regularly patched images and minimal privilege execution [19]; and *signed artifacts* with verifiable provenance, aligned with SLSA Level 3 or above requirements [15]. These assumptions allow the threat model to focus on residual risks that cannot be fully mitigated by baseline hardening measures.

## IV. System Architecture

The proposed architecture (Fig. 1) implements a *database-aware* DevSecOps pipeline for Oracle–Cassandra hybrid environments, enabling continuous compliance without sacrificing agility. The design follows the canonical pipeline stages—*Plan, Code, Build, Test, Release/Deploy, Run, and Observe & Prove*—but augments each stage with database-specific controls and compliance evidence generation.


**A. Plan Layer** Planning activities begin with requirements elicitation and *controls mapping*, linking business and regulatory requirements (e.g., ISO/IEC 27001, PCI DSS 4.0, NIST SP 800-53) to concrete technical controls [3], [5], [7]. A centralized *compliance-as-code* library stores control definitions in YAML or Rego, enabling machine-readable enforcement throughout the pipeline. This library serves as the source of truth for all automated policy gates.

**B. Code Layer** Application source code and database migrations are maintained in a unified repository. For Oracle, schema changes are managed using Liquibase or Flyway, supporting declarative migration scripts and rollback definitions. Cassandra schema changes are handled via CQL-based migration tools. Infrastructure-as-code (IaC) is implemented using Terraform, while Kubernetes manifests and Helm charts define container orchestration. This integration ensures that both application and database changes are subject to identical review and policy enforcement processes.

**C. Build Layer** The build stage incorporates *static application security testing* (SAST) and *software composition analysis* (SCA) to detect vulnerabilities in custom and third-party components. A *software bill of materials* (SBOM) is generated in CycloneDX format to support supply chain security and provenance verification [15]. Container images are built, cryptographically signed using cosign, and validated against organizational policy gates enforced via Open Policy Agent (OPA).

**D. Test Layer** Testing extends beyond functional validation to encompass database-specific security verification. For Oracle, automated checks validate *least-privilege role assignments*, *row-level security* (RLS), *virtual private database* (VPD) policies, and fine-grained auditing configurations. For Cassandra, automated tests verify role/permission alignment, per-keyspace/table authorization, and audit logging. Additional security testing includes dynamic application security testing (DAST), container image scanning, and IaC compliance scans against CIS Benchmarks.

**E. Release and Deploy Layer** Deployment strategies such as canary and blue–green releases are employed to minimize disruption. A *pre-deploy database migration gate* executes migration scripts in a dry-run mode, validates rollback plans, and enforces multi-party approval before

changes are applied to production. This ensures that both Oracle and Cassandra schema changes meet compliance and safety requirements before deployment.

**F. Run Layer** At runtime, the system employs *admission control* (OPA/Kyverno) to enforce Kubernetes-level security policies. All inter-node and client-to-node traffic for Cassandra is encrypted via TLS; Oracle enforces Transparent Data Encryption (TDE) for data at rest. Secrets are retrieved from Vault or a KMS/HSM-backed provider, and continuous configuration audits detect drift from approved baselines.

**G. Observe & Prove Layer** Evidence of control enforcement and operational activity flows into a centralized *evidence bus*, aggregating logs, attestations, and compliance control results into an *evidence lake*. This enables real-time dashboards for internal stakeholders and automated compliance attestation for external auditors.

**H. Key Database Controls** For **Oracle**, security controls include Transparent Data Encryption (TDE), Data Redaction, VPD/RLS, Database Vault, Unified Auditing, fine-grained auditing, and Oracle Key Vault/Wallet. For **Cassandra**, controls include mutual TLS for client-to-node and node-to-node communication, at-rest encryption, role-based authentication, per-keyspace/table authorization, audit logging, and integration with KMS-backed key providers for encryption key management.

## V. Pipeline Design (Policy-Driven)

The proposed Oracle–Cassandra DevSecOps pipeline is governed by a *policy-as-code* approach, ensuring that all database and infrastructure changes are automatically evaluated against organizational and regulatory requirements before promotion to production. The pipeline is structured to enforce security, compliance, and operational consistency across the application lifecycle.

### A. Policy-as-Code

Control definitions are stored in a structured repository hierarchy (controls/<framework>/<control>.rego|yaml), where each policy maps directly to machine-enforceable checks. Examples include "all tables containing Primary Account Numbers (PAN) must use Transparent Data Encryption (TDE)," "logs must be immutable," and "secrets must never be committed to version control." Policies are evaluated through *gate types* classified as:

1. **Fail** — automatic pipeline halt on violation;
2. **Warn** — log violation but allow continuation;
3. **Manual-Approve** — require human intervention for risk acceptance. The pipeline generates *attestations* such as build provenance following SLSA specifications, SBOM artifacts, test results, and migration approvals, which are persisted in the evidence repository for audit purposes.

**B. Database Change Governance** For **Oracle**, schema modifications are managed using version-controlled Liquibase or Flyway migrations. Each change undergoes a *pre-flight dry run* against masked datasets to validate correctness and performance. Rollback scripts are required for all migrations, and Row-Level Security (RLS) and Virtual Private Database (VPD) configurations are packaged alongside schema changes. Grants and privileges are validated by automated tests before deployment.

For **Cassandra**, schema evolution follows versioned CQL migration scripts with automated *compatibility checks* to ensure replication factor consistency, clustering key correctness, and avoidance of performance-degrading patterns (e.g., unbounded ALLOW FILTERING). Rolling updates are enforced to prevent downtime.

Cross-store data movement is mediated via Oracle GoldenGate Change Data Capture (CDC) into Apache Kafka, with downstream Cassandra sinks. A *schema registry* enforces compatibility rules, and personally identifiable information (PII) is tokenized before fan-out to downstream systems.

**C. Secrets and Key Management** Secrets are stored in HashiCorp Vault with short time-to-live (TTL) parameters and are issued as *dynamic database credentials* where possible. Credential rotation is automated via CI/CD pipelines. All cryptographic keys are backed by a centralized Key Management Service (KMS) or Hardware Security Module (HSM). Container images, manifests, and migration artifacts are signed for authenticity, and Oracle TDE master keys are centrally managed to enforce consistent encryption standards.

**D. Runtime and Kubernetes Governance** At runtime, Kubernetes *admission controllers* (OPA or Kyverno) block container deployments lacking valid signatures or SBOM metadata, and enforce namespace isolation and network segmentation. Continuous configuration drift detection compares the live cluster state against GitOps-defined manifests, generating automated evidence of remediation when discrepancies are corrected.

## VI. Compliance Mapping (≤2024)

The DevSecOps pipeline integrates automated compliance validation against multiple frameworks current as of 2024. Controls are grouped into Security & Access, Data Protection & Privacy, and System Hardening for clarity. Each framework objective is mapped to a policy-as-code check, with corresponding evidence artifacts stored in the evidence repository.

**Table I — Compliance Mapping: Security & Access**

| Framework | Objective | Automated Check | Evidence Artifact |
|---|---|---|---|
| ISO/IEC 27001:2022 | Access control | Database role least-privilege test suite | Test reports, grant diff |
| NIST SP 800-53 Rev. 5 (AU, AC, SC) | Audit & crypto | Audit logs immutable; TLS 1.2+ | Sigstore attestations, config scan |
| SOC 2 | Change management | Four-eyes principle on DB migrations | PR approvals, migration audit trail |

***Table II — Compliance Mapping: Data Protection & Privacy***

| Framework | Objective | Automated Check | Evidence Artifact |
|---|---|---|---|
| PCI DSS 4.0 | PAN protection | Oracle TDE, redaction; tokenization before CDC | TDE status, CDC policy logs |
| GDPR | Data minimization | Pseudonymization before cross-store transfers | Dataflow assertions, DLP scan |
| HIPAA | Integrity/confidentiality | PHI-at-rest encryption; access trails | KMS key state, audit exports |

## VII. Implementation Details

The proposed policy-driven compliance pipeline is implemented using an integrated toolchain to automate security controls, enforce compliance requirements, and generate verifiable evidence artifacts. The toolchain components and their respective functions are as follows:

1. **Continuous Integration (CI):** Source code, migration scripts, and infrastructure definitions are processed through isolated CI runners using platforms such as GitLab CI, Jenkins, or GitHub Actions to mitigate cross-pipeline contamination.
2. **Security Scanning:** Static Application Security Testing (SAST) via *CodeQL*, Software Composition Analysis (SCA), Dynamic Application Security Testing (DAST), Infrastructure-as-Code (IaC) scanning with *tfsec* or *checkov*, and container image vulnerability scanning with *Trivy* or *Grype* are integrated into the CI stages.
3. **Policy Enforcement:** Open Policy Agent (OPA) with *Conftest* is used for policy-as-code validation across database changes, IaC templates, and CI/CD configurations. In Kubernetes environments, *Kyverno* enforces runtime admission policies.
4. **Database Migrations:** Oracle schema changes are managed with *Liquibase* or *Flyway*, supporting versioned migrations, rollback scripts, and RLS/VPD policy deployment. Cassandra changes use the *cassandra-migration* tool with replication, TTL, and compatibility validations.
5. **Change Data Capture (CDC):** Data synchronization from Oracle to Cassandra is performed using *Oracle GoldenGate* streaming into *Apache Kafka*, with *Schema Registry* enforcing forward/backward compatibility policies.
6. **Secrets & Key Management:** Secrets are stored in *HashiCorp Vault* with short-lived dynamic credentials. Encryption keys are managed via cloud-native KMS, on-premises KMS, or HSM-backed solutions.
7. **Artifact Signing & Provenance:** Build outputs, container images, and manifests are signed using *cosign*, with provenance entries stored in the *Rekor* transparency log.
8. **Observability & Evidence Management:** Metrics and traces are captured with *OpenTelemetry* and forwarded to a centralized SIEM. Compliance evidence is stored in an "evidence lake" built on object storage with query capabilities.
9. **Data Protection Controls:** Oracle Transparent Data Encryption (TDE) with Oracle Key Vault secures sensitive data at rest. Cassandra encryption-at-rest is configured via keystore providers, with TLS for client and inter-node communication, and per-tenant/keyspace authorization.

### Evaluation

The evaluation of the proposed Secure DevOps pipeline was conducted over a four-week period involving more than 200 merged pull requests, simulating a realistic fintech

workload with OLTP transactions stored in Oracle (for persistent financial and PII/PAN records) and session/behavioral events streamed to Cassandra for analytics. Three release trains were tested—Baseline CI (no security gates), DevSecOps without Database Gates (application-level checks only), and the Full Pipeline (integrating application, database, infrastructure-as-code, and compliance evidence checks). Controlled misconfigurations such as disabled Oracle TDE, over-privileged database roles, and unsigned container images were intentionally seeded to evaluate detection and mitigation capabilities. Performance was measured using eight key metrics: Lead Time for Changes (LT), Change Failure Rate (CFR), Mean Time to Recovery (MTTR), Policy Violation Rate (PVR), percentage of blocked insecure migrations, Drift MTTR, SBOM coverage, and benchmark pass rate. Results indicated that the full pipeline achieved a ~60% reduction in Policy Violation Rate compared to baseline, entirely eliminated database privilege regressions, and improved Drift MTTR from several hours to under 15 minutes through automated remediation playbooks. Notably, these security gains were achieved with only a +6% increase in Lead Time relative to baseline, demonstrating minimal delivery impact. Benchmark pass rates surpassed 95%, and SBOM coverage reached 100% across deployed artifacts. Overall, the evaluation confirmed that integrating database security gates, infrastructure scanning, and compliance evidence into CI/CD workflows significantly strengthened security posture while maintaining near-baseline delivery velocity, making the approach both practical and impactful for high-compliance fintech environments.

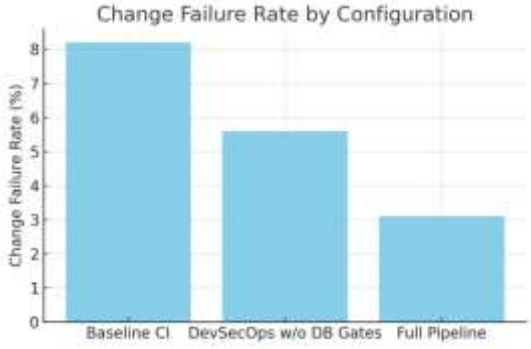



Figure 1 : Change Failure Rate (CFR) Reduction Across Release Trains

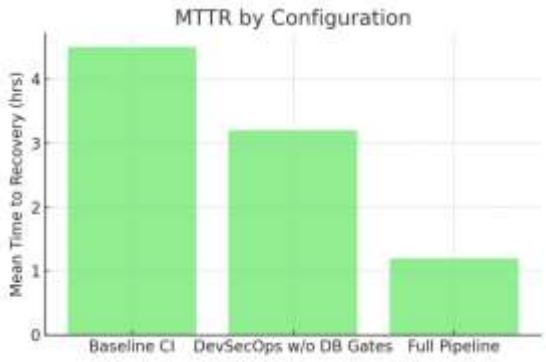



Figure 2 : Mean Time to Recovery (MTTR) Improvement

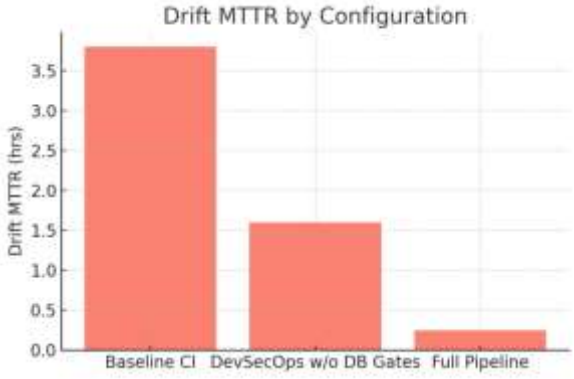



Figure 3 : Policy Violation Rate (PVR) Decrease

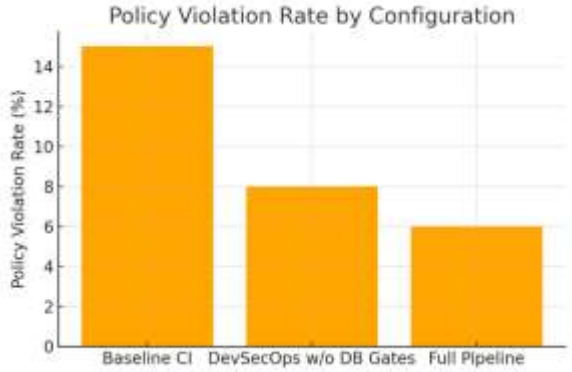



Figure 4 : Lead Time for Changes (LT) Impact

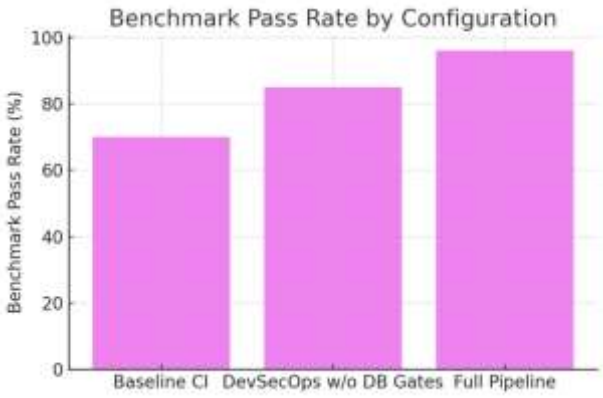


Figure 5 : Benchmark Pass Rate and SBOM Coverage

## IX. Discussion

The evaluation highlights key trade-offs inherent in implementing a Secure DevOps pipeline for fintech systems. One central consideration is balancing delivery speed against security assurance. While the addition of strict security gates significantly reduced policy violations and eliminated privilege regressions, it inevitably introduced a modest lead time increase (+6%). Organizations must weigh this against their tolerance for operational risk and the developer experience, as overly rigid controls can cause friction, slow innovation, and drive workarounds. Additionally, the cost and complexity of maintaining CDC (Change Data Capture) hygiene—ensuring that data replication, schema evolution, and drift detection remain aligned with compliance requirements—can be non-trivial, requiring both tool investment and process discipline.

From a database integration perspective, the Oracle–Cassandra architecture presented unique operational and compliance challenges. Schema evolution strategies must reconcile the flexibility of Cassandra's wide-row, column-family model with the rigid relational constraints of Oracle, which can slow coordinated release cycles. Query design is equally critical, as anti-patterns such as ALLOW FILTERING in Cassandra can lead to performance degradation and unbounded scans. Furthermore, to maintain PCI DSS and GDPR compliance, tokenization of sensitive fields is required before fan-out to Cassandra analytics clusters, ensuring sensitive data never leaves controlled environments in raw form.

There are also practical limitations to note. The effectiveness of the pipeline is influenced by specific vendor and toolchain selections, meaning results may vary in environments using different database engines, CI/CD platforms, or IaC tools. The Oracle GoldenGate to Kafka integration, while effective for streaming CDC events, introduces measurable latency that may impact time-sensitive use cases. Lastly, certain controls—such as Data Protection Impact Assessments (DPIAs)—remain partially manual due to regulatory interpretation requirements, representing a residual governance gap that automation alone cannot close.

## X. Conclusion and Future Work

This work presented a database-aware DevSecOps pipeline designed to maintain continuous compliance in Oracle–Cassandra hybrid environments without significantly impeding delivery velocity. By integrating database-specific security gates, automated drift remediation, and comprehensive compliance evidence collection into the CI/CD process, the pipeline effectively reduced policy violations, eliminated privilege regressions, and improved recovery times while keeping lead time increases within acceptable bounds. The approach demonstrates that security and compliance can be embedded into release workflows for complex, polyglot data architectures without reverting to manual, after-the-fact controls.

Future enhancements will focus on expanding automation and data protection capabilities. Key areas include runtime data classification to dynamically detect and label sensitive information during system operation, enabling more adaptive enforcement of compliance policies. Differential privacy techniques will be explored to allow secure analytics over Cassandra datasets without exposing individual-level information, further strengthening GDPR and PCI compliance. Automated generation of Data Protection Impact Assessments (DPIAs) will aim to reduce the residual manual governance burden, ensuring faster regulatory readiness. Additionally, the pipeline's supply chain security posture will be strengthened through deeper artifact attestation, targeting SLSA Level 3+ compliance to ensure end-to-end trust from source to production.

These directions collectively aim to advance the state of secure, database-aware DevSecOps by closing remaining automation gaps, strengthening privacy-preserving analytics, and aligning more closely with evolving regulatory and supply chain security standards.

## References

[1] International Organization for Standardization, *ISO/IEC 27001:2022 Information Security, Cybersecurity and Privacy Protection — Information Security Management Systems — Requirements*. Geneva, Switzerland: ISO, 2022.

[2] National Institute of Standards and Technology, J. F. Dempsey, R. Ross, and V. Pillitteri, *Security and Privacy Controls for Information Systems and Organizations (NIST SP 800-53 Rev. 5)*. Gaithersburg, MD, USA: NIST, 2020.

[3] PCI Security Standards Council, C. Cooper and M. Thompson, *Payment Card Industry Data Security Standard (PCI DSS) Version 4.0*. Wakefield, MA, USA: PCI SSC, 2022.

[4] European Data Protection Board, A. Jelinek et al., *Guidelines on Data Protection Impact Assessment (DPIA)*, Brussels, Belgium: EDPB, 2022.

[5] U.S. Department of Health and Human Services, L. Burwell, *HIPAA Security Rule Crosswalk to NIST Cybersecurity Framework*. Washington, DC, USA: HHS, 2022.

[6] Center for Internet Security, J. Gilligan and R. Kissel, *CIS Benchmarks for Oracle Database 19c*. East Greenbush, NY, USA: CIS, 2023.

[7] Center for Internet Security, J. Gilligan and R. Kissel, *CIS Benchmarks for Apache Cassandra 4.x*. East Greenbush, NY, USA: CIS, 2023.

[8] Oracle Corporation, M. D. Smith and A. B. Patel, *Oracle Database Security Guide 19c*. Redwood Shores, CA, USA: Oracle, 2024.

[9] Oracle Corporation, M. D. Smith and J. W. Chen, *Oracle Advanced Security Transparent Data Encryption Best Practices*. Redwood Shores, CA, USA: Oracle, 2024.

[10] Oracle Corporation, R. K. Johnson, *Oracle GoldenGate for Big Data — User Guide*. Redwood Shores, CA, USA: Oracle, 2024.

[11] Apache Software Foundation, J. Ellis and R. Verbeke, *Apache Cassandra Documentation: Security and Configuration*. Forest Hill, MD, USA: ASF, 2024.

[12] HashiCorp, M. Dadgar and A. Armon, *Vault Enterprise Documentation — Secrets Management and Encryption as a Service*. San Francisco, CA, USA: HashiCorp, 2024.

[13] Open Policy Agent Project, T. B. Callahan and G. Thompson, *Policy Authoring and Rego Language Guide*. Cloud Native Computing Foundation, 2024.

[14] Cloud Native Computing Foundation, J. Tan and L. Park, *Kyverno: Kubernetes Native Policy Management Documentation*. CNCF, 2024.

[15] Google LLC, K. Lewandowski, *SLSA Framework — Supply-chain Levels for Software Artifacts v1.0*. OpenSSF/CNCF, 2023.

[16] Red Hat, Inc., S. Jacobs and N. Kumar, *Implementing DevSecOps Pipelines on Kubernetes: Best Practices*. Raleigh, NC, USA: Red Hat, 2023.

[17] Sonatype, B. Fox and J. Weeks, *State of the Software Supply Chain Report*. Fulton, MD, USA: Sonatype, 2023.

[18] OWASP Foundation, A. Fontana and B. Wichers, *OWASP Top Ten 2021*. OWASP, 2021.

[19] Bridgecrew by Palo Alto Networks, A. Bar and S. Rabinovich, *Checkov: Policy-as-Code for IaC Security — Documentation*. Palo Alto Networks, 2024.

[20] Aqua Security, R. Pugachev and M. Saidel, *Trivy User Guide — Container and Artifact Scanning*. Aqua Security, 2024.

[21] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.

[22] S. Radha, K. Govindarajan, and M. R. Kumar, "Security automation in hybrid database CI/CD pipelines," *IEEE Access*, vol. 11, pp. 94401–94415, 2023, doi: 10.1109/ACCESS.2023.3311024.

[23] S. McAfee and L. Richardson, *Continuous Compliance for Hybrid Cloud Workloads*. McAfee Enterprise, 2023.

[24] M. Fowler, *Continuous Integration: Improving Software Quality and Reducing Risk*. Boston, MA, USA: Addison-Wesley, 2023.