STM JOURNALS

Journal of
Artificial Intelligence
Research & Advances

ISSN: 2395-6720
Volume 13, Issue 1, 2026
January–April
DOI Journal: 10.37591/JoAIRA

https://journals.stmjournals.com/joaira

Research JoAIRA

AI-Driven DevSecOps Automation: An Intelligent Framework for Continuous Cloud Security and Regulatory Compliance

Roshan Kakarla[1,*], Sai Bharath Sannareddy[2]

Abstract

Cloud-native systems, microservices, and infrastructure-as-code (IaC)–oriented CI/CD pipelines have accelerated the pace of software delivery, yet they have also introduced new layers of operational complexity and widened the overall security exposure of modern applications. Traditional DevSecOps workflows still depend heavily on isolated scanners, manual reviews, and static governance processes that are not well-suited for the elasticity and constant change characteristic of multi-cloud environments. To address these limitations, this paper introduces an AI-Driven DevSecOps Automation Framework (AID-DF) that integrates several complementary forms of artificial intelligence. Machine Learning models are used to identify behavioural anomalies within application and infrastructure telemetry, Natural Language Processing enables automated interpretation and mapping of regulatory and policy requirements, and Reinforcement Learning guides adaptive remediation actions and risk-aware deployment strategies. The framework was validated across a distributed environment consisting of 100 microservices deployed on combined AWS and Azure Kubernetes clusters. Results indicate notable gains in detection precision, regulatory coverage, Mean-Time-to-Detect (MTTD), Mean-Time-to-Respond (MTTR), and overall audit efficiency. These findings suggest that AI-enhanced DevSecOps pipelines can deliver a continuously compliant and highly scalable security posture for contemporary cloud-native ecosystems.

Keywords: Artificial intelligence, machine learning, reinforcement learning, natural language processing, cloud security, compliance automation, hybrid cloud, CI/CD; threat detection

INTRODUCTION

Changing Landscape of Cloud-Native Security

Modern enterprises increasingly depend on distributed architectures, containerized services, IaC-driven provisioning, serverless workloads, and high-frequency CI/CD pipelines. These environments routinely support hundreds of deployments per week, with ephemeral workloads, elastic scaling, and rapidly evolving dependencies. While these characteristics improve agility and availability, they introduce security and compliance challenges that traditional perimeter-based models cannot address effectively. Misconfigurations, over-privileged identities, and unmonitored changes are leading contributors to cloud breaches, and studies report that a majority of incidents could be prevented by continuous validation and automated checks [1, 2].

***Author for Correspondence**
Roshan Kakarla
E-mail: ~~djawale870@gmail.com~~rkakarla1041@gmail.com

[1]DevOps Engineer, Department of Information Technology, Techcorp LLC, Delaware, USA.
[2]Senior Cloud Infrastructure Engineer, Department of Computer Science, Abbott Laboratories, Illinois, USA.

Received Date: November 20, 2025
Accepted Date: December 05, 2025
Published Date: December 11, 2025

Citation: Roshan Kakarla, Sai Bharat Sannareddy. AI-Driven DevSecOps Automation: An Intelligent Framework for Continuous Cloud Security and Regulatory Compliance. Journal of Artificial Intelligence Research & Advances. 2026; 13(1): 1–15p.

In cloud-native environments, the attack surface is fluid rather than static. Network policies, IAM roles, container images, and service interactions change continuously, often driven by automated workflows. Manual audits or infrequent security reviews cannot reliably detect drift, policy violations, or emerging threats. Short-lived containers and serverless functions further complicate forensic analysis and incident response. As a result, organizations require security mechanisms that continuously learn from operational data, adapt in real time, and maintain compliance across multiple cloud providers and runtime platforms.

Limitations of Traditional DevSecOps Practices

DevSecOps was introduced to incorporate security earlier in the software lifecycle, embedding checks into CI/CD pipelines and promoting collaboration between development, operations, and security teams [1, 3]. In practice, however, many implementations remain heavily dependent on static tools such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Software Composition Analysis (SCA), container scanners, and IaC linters.

Key Limitations Include

- *Rule-based detection:* Static scanners rely on signatures and predefined patterns, making them effective mainly for known vulnerabilities and less capable of identifying behavioural anomalies or zero-day threats [4].
- *High false positives:* Conservative rule sets generate noisy findings; developers learn to ignore alerts and suppress checks, reducing overall protection [4, 5].
- *Fragmented visibility:* Telemetry from CI/CD logs, runtime metrics, network flows, and cloud events is typically siloed across multiple tools, hindering end-to-end correlation [6].
- *Manual compliance mapping:* Frameworks such as NIST SP 800-53 and ISO/IEC 27001 require human interpretation and manual policy translation, consuming significant effort and increasing error rates [7].
- *Slow incident response:* Many DevSecOps pipelines stop at detection; remediation remains manual, leading to high MTTD and MTTR.

These limitations make security reactive and inconsistent, particularly in high-velocity environments where configuration drift and dependency changes are constant.

Opportunity for Artificial Intelligence in DevSecOps

Artificial Intelligence (AI) specifically ML, NLP, and RL offers capabilities that align naturally with DevSecOps objectives:

- ML can learn behavioural baselines from logs, metrics, and traces, detecting anomalous activity without relying exclusively on predefined rules [7, 8].
- NLP can process unstructured regulatory documents and organizational policies, extracting obligations and mapping them to technical controls to automate compliance enforcement [9, 10].
- RL can optimize sequential decision-making in dynamic environments, enabling automated deployment gating, rollback selection, and remediation strategies based on observed outcomes [11–13].

By embedding these AI techniques throughout the CI/CD pipeline and runtime infrastructure, DevSecOps can evolve from a static toolchain into a continuously learning and adaptive security governance layer.

Research Motivation and Objectives

Enterprises operating large-scale cloud-native systems require a security model that is:

- Predictive rather than purely reactive
- Context-aware rather than rule-only
- Continuously compliant rather than audit-driven
- Autonomously governed rather than manually tuned

To address these needs, this paper introduces the AI-Driven DevSecOps Automation Framework (AID-DF) with the following objectives:

1. Develop ML models that analyze source-level and runtime behaviours to detect anomalies and emerging threats.
2. Apply NLP to convert regulatory standards into machine-verifiable controls and policy-as-code artefacts.
3. Employ RL to drive risk-aware deployment and remediation decisions in CI/CD and runtime environments.
4. Integrate ML, NLP, and RL components into a unified DevSecOps architecture with continuous feedback loops.
5. Evaluate the framework in a realistic multi-cloud microservice environment and quantify its impact.

Structure of the Paper

The remainder of the paper is organized as follows. Section II reviews related work in DevSecOps, ML-based security, NLP-driven compliance, and RL for cyber defence. Section III details the proposed methodology and explains all technical concepts used in the framework. Section IV presents the experimental setup, quantitative and qualitative evaluation, and enterprise implications. Section V concludes the work and outlines future research directions. Section VI provides author information, followed by references in Vancouver format.

LITERATURE REVIEW

Evolution of DevSecOps in Cloud-Native Ecosystems

DevSecOps evolved from DevOps to embed security checks into the continuous delivery pipeline and to align security activities with agile development practices [1]. Early DevOps initiatives emphasized automation, collaboration, and rapid release cycles but often treated security as an afterthought. As organizations shifted to microservices, Kubernetes, and serverless platforms, security incidents linked to misconfigurations, exposed secrets, and dependency vulnerabilities increased [2, 3].

Empirical studies show that late-stage security testing can increase remediation costs by a factor of eight due to distributed dependency chains and complex rollback procedures [2]. The proliferation of IaC templates and container images has expanded the supply-chain attack surface. DevSecOps seeks to address these issues by embedding controls closer to code and infrastructure definitions, but current practice still exhibits gaps in automation depth, cross-tool correlation, and adaptive behaviour.

Limitations of Existing DevSecOps Tools

The DevSecOps tooling ecosystem comprises SAST, DAST, SCA scanners, container security tools, IaC analyzers, policy-as-code engines, and runtime monitoring platforms [4–6]. Despite this richness, several limitations persist:

- *Tool fragmentation:* Security results are distributed across dashboards, making end-to-end risk assessment difficult [4].
- *Static detection logic:* Signature-based tools struggle with previously unseen attack vectors and complex multi-step attacks [5, 7].
- *Alert fatigue:* Excessive false positives overwhelm development and security teams.
- *Limited compliance automation:* Mapping regulations such as NIST SP 800-53, ISO/IEC 27001, and SOC 2 to concrete controls remains largely manual, with reported error and rework rates of up to 35% [6, 7].
- *Partial coverage of runtime risk:* Many tools focus on pre-deployment scanning, leaving runtime behaviour and drift only partially monitored.

Overall, existing tools improve visibility but do not provide an intelligent, self-optimizing governance mechanism.

Machine Learning in Security and DevOps

ML has been widely applied to intrusion detection, malware classification, spam filtering, and user behaviour analytics [7, 8]. Traditional algorithms such as Random Forests, Support Vector Machines, and Naive Bayes provide interpretable models but face challenges with high-dimensional, non-stationary data typical of cloud logs and microservice telemetry.

Deep learning models, including Long Short-Term Memory (LSTM) networks and transformers, have shown superior performance on sequential and unstructured data [7–15]. In CI/CD and runtime environments, these models can capture temporal dependencies in build logs, deployment records, and resource metrics, enabling predictive detection of anomalies and failures. Autoencoders and other unsupervised techniques provide additional capability to detect novel or rare patterns without requiring labelled datasets [8].

Natural Language Processing for Compliance Automation

Regulatory frameworks such as NIST SP 800-53, ISO/IEC 27001, GDPR, HIPAA, and SOC 2 encode complex textual requirements that must be translated into policies, procedures, and technical configurations. Manual interpretation is time-consuming and prone to inconsistency.

Recent advances in transformer-based NLP (e.g., BERT, RoBERTa, domain-specific models) have enabled accurate extraction of obligations, classification of control statements, and mapping of regulatory clauses to technical requirements [9, 10]. Research demonstrates that NLP can reach high precision and recall when extracting compliance obligations from legal texts [9]. Frameworks such as FACTOR apply NLP to reason about privacy and security policies in cloud environments and link them to system states [10]. In a DevSecOps context, these capabilities can be used to generate policy-as-code definitions and validate configurations against regulatory baselines continuously.

Reinforcement Learning for Security Optimization

RL has been applied to network defence, intrusion response, adaptive firewall configuration, and malware containment [11–13]. RL agents learn policies that maximize cumulative reward by interacting with an environment and receiving feedback signals. In security, rewards can incorporate successful mitigation, low false-positive rates, and adherence to service-level agreements (SLAs).

Studies show that RL-based security controllers can reduce response times and improve the accuracy of automated defence actions compared to static rule sets [11, 30]. However, deploying RL in production requires careful design of state representations, reward functions, and safety constraints to prevent destabilizing behaviour [12, 13].

Security Challenges in Modern Cloud-Native Architectures

Cloud-native environments exhibit properties such as autoscaling, ephemeral infrastructure, multi-tenant workloads, and service mesh–based communication. Misconfiguration of identity policies, network segmentation, encryption settings, or container registries can lead to high-impact breaches [5, 14]. Supply-chain threats arising from vulnerable base images or compromised third-party libraries further complicate assurance [16, 21].

The IBM Cost of a Data Breach report highlights that misconfiguration and inadequate access controls remain dominant causes of incidents in cloud deployments [14]. These findings reinforce the need for continuous validation of IaC templates, container images, and runtime policies.

Compliance Drift and Governance Failures

Compliance drift occurs when deployed environments deviate from approved baselines between audit cycles. Manual evidence collection, spreadsheet-based control mapping, and ad hoc documentation create gaps that attackers can exploit [7, 17]. Organizations report that a substantial portion of compliance team time is spent assembling evidence rather than improving controls [17].

Automated mapping of controls to configurations, continuous drift detection, and automatic evidence generation can significantly reduce manual workload and improve audit readiness [9, 10].

Summary of Research Gap

A survey of recent literature indicates that:

- ML is primarily used for intrusion and anomaly detection.
- NLP is mainly applied to document classification and information extraction, with limited use in automated governance.
- RL is investigated mostly in experimental network defence settings.
- DevSecOps research focuses on process and tooling, with limited integration of advanced AI techniques [18, 19].
- Very few works propose a unified framework that combines ML, NLP, and RL for end-to-end DevSecOps automation.

AID-DF addresses this gap by integrating all three AI paradigms into a cohesive architecture that spans source code analysis, CI/CD security, runtime monitoring, and compliance automation.

METHODOLOGY

The AI-Driven DevSecOps Automation Framework (AID-DF) integrates ML, NLP, and RL components across the software lifecycle. This section explains the architecture, data sources, model design, decision logic, and operational workflow. All key technical terms are defined for clarity.

System Architecture Overview

AID-DF is structured into five core layers:

1. *Source code intelligence layer:* Analyzes repositories, code complexity, and dependency graphs.
2. *Build & artifact security layer:* Validates build outputs, software bills of materials (SBOMs), and container images.
3. *Deployment governance layer:* Enforces policy-as-code and RL-driven gating in CI/CD pipelines.
4. *Runtime behavioural analysis Layer:* Monitors logs, metrics, and network flows for anomalies.
5. *Autonomous optimization layer:* Implements ML–NLP–RL feedback loops for continuous improvement.

Security is treated as a continuous, learning-based process rather than a collection of static checks.

Data Sources and Preprocessing

AID-DF ingests heterogeneous data from development and operational systems.

Source Code and Repository Data

Collected artefacts include:

- Repository snapshots and file structure
- Abstract Syntax Trees (ASTs)
- Tokenized code representations
- Commit metadata (author, timestamp, change size)
- Dependency graphs
- Secret-leak patterns
- Cyclomatic complexity scores

Abstract Syntax Tree (AST)

A tree-structured representation of source code that captures its syntactic structure. ASTs enable ML models to reason about control flow, nesting depth, and vulnerable patterns beyond raw text [18].

Cyclomatic Complexity

A metric capturing the number of independent execution paths in a function or module. Higher complexity correlates with elevated defect and security risk [16].

Dependency Graph

A graph representing relationships between modules and external libraries, useful for identifying vulnerable or high-risk components [21].

These features are vectorized and fed to ML classifiers for risk prediction.

CI/CD Pipeline Telemetry

CI/CD systems produce rich time-series logs, including:

- Test execution outcomes and durations
- Build times and failure codes
- Error stack traces
- Flaky test behaviour
- Deployment events and rollback history

Sequence Modelling: A class of ML techniques for analysing ordered data such as log sequences. LSTM networks are widely used for sequence modelling because they retain contextual information over time [15].

Artifact and Dependency Metadata

Artifact metadata includes:

- SBOMs
- Library and container image versions
- Known vulnerabilities (CVEs)
- Licence information
- Integrity hashes

SBOM (Software Bill of Materials): A machine-readable inventory of all dependencies in an application, critical for supply-chain security and vulnerability tracking [20].

Runtime Telemetry and Cloud Monitoring Data

Runtime data is collected from Kubernetes clusters, service meshes, and cloud monitoring platforms:

- CPU, memory, and network usage
- Pod lifecycle events
- Service mesh traffic attributes
- IAM logs and access attempts
- Network flows and connection patterns

Temporal patterns in these metrics support behavioural anomaly detection [19, 21].

Regulatory and Compliance Text Corpora

NLP models process regulatory and policy documents such as:

- NIST SP 800-53
- ISO/IEC 27001
- GDPR, SOC 2, HIPAA security rules
- Cloud provider security guidelines [22–27]

Regulatory Clause

A statement in a regulation that prescribes a required security or privacy behaviour (e.g., "Encrypt data at rest"). NLP models classify clauses and associate them with corresponding technical controls [24].

Machine Learning Component

The ML subsystem provides predictive anomaly detection, risk classification, and event forecasting.

Feature Engineering

Features are derived from:

- Code complexity and dependency metrics
- Log embeddings from CI/CD and runtime systems
- Temporal anomaly indicators (e.g., sudden latency spikes)
- Deployment frequency and change size
- User and service behaviour profiles

ML Algorithms

AID-DF employs four primary algorithms:

- *Random Forest Classifier:* An ensemble of decision trees that reduces overfitting and handles mixed-type features well [21]. Used for commit-level risk prediction and dependency risk scoring.
- *Gradient Boosting Machines (GBMs):* Models that iteratively improve performance by focusing on residual errors of previous learners, achieving strong results on tabular data [22]. Used for ranking pipeline failures and estimating compliance drift probability.
- *LSTM Neural Networks:* Recurrent neural networks designed to model long-range temporal dependencies in sequences [15]. Used for CI/CD log anomaly detection and runtime behavioural modelling.
- *Autoencoders:* Neural networks trained to reconstruct normal patterns; deviations in reconstruction error identify anomalies [8]. Used for unsupervised detection of unusual network and runtime activity.

Training Pipeline

The training process includes:

- Train–validation–test splits to avoid overfitting
- Class imbalance handling using oversampling techniques where needed
- Hyperparameter tuning via Bayesian optimization to maximize accuracy and F1-score [22]
- k-fold cross-validation to assess generalization

Bayesian optimization: An approach that builds a probabilistic model over the objective function and iteratively selects parameter combinations that maximize expected improvement [22].

ML Inference in Production

Trained models are deployed as services invoked at four checkpoints:

1. Pre-commit hooks for high-risk code detection
2. Build phase for log and dependency anomaly checks
3. Deployment phase for gating decisions support
4. Runtime phase for continuous behavioural monitoring

Latency budgets are enforced (e.g., <800 ms per inference) to avoid pipeline bottlenecks.

Natural Language Processing Component

The NLP subsystem automates compliance interpretation and mapping.

Preprocessing Regulatory Text

Documents are processed using:

- Tokenization and sentence segmentation
- Lemmatization and part-of-speech tagging
- Phrase chunking and dependency parsing

Embedding: A numerical vector representing the semantic meaning of words or sentences. Transformer-based models generate contextual embeddings that capture subtle legal nuances [9, 24].

Control Extraction and Classification

The model identifies:

- Obligation clauses
- Security and privacy requirements
- Technical control statements
- Verification and evidence expectations

Transformer-based models achieve high accuracy on control extraction and classification tasks [9, 10].

Semantic Compliance Mapping

Semantic similarity: A measure of how closely two pieces of text are related in meaning. Embedding similarity is used to map regulatory clauses (e.g., "Encrypt data at rest") to concrete configurations (e.g., storage encryption settings in AWS or Azure).

Policy-as-Code Generation

The framework generates policy-as-code artefacts (e.g., Open Policy Agent (OPA) rules, Sentinel policies, Kubernetes admission controls) directly from mapped requirements [21].

Policy-as-Code (PaC): The practice of expressing security and compliance rules as version-controlled code, allowing automated validation and enforcement within CI/CD pipelines [21].

Compliance Drift Detection

NLP-derived control definitions are continuously compared against actual infrastructure state. Detected deviations (e.g., unencrypted storage, misconfigured network policies) trigger remediation workflows or pipeline gates.

Reinforcement Learning Component

The RL subsystem provides adaptive decisions for deployment and incident response.

State Representation

Each state vector encodes:

- Severity and count of active security violations
- Recent anomaly scores from ML models
- Current system load and latency
- Estimated user impact and business criticality
- Compliance risk level and audit posture

Action Space

Available actions include:

- Allow deployment to proceed
- Block deployment
- Roll back to a previous version

- Isolate a workload or namespace
- Restart specific services
- Trigger auto-remediation playbooks
- Escalate to human responders

Reward Function

The reward function balances security and availability:

- *Successful mitigation of a real threat:* +10
- *Reduction in false positives:* +5
- *Missed anomaly leading to incident:* −10
- *SLA violation or unnecessary outage:* −6
- *Unnecessary rollback or excessive blocking:* −4

RL Algorithm: Deep Q-Network (DQN)

AID-DF employs a Deep Q-Network (DQN) that approximates the action-value function for each state [11–13].

Key mechanisms include:

- Experience replay buffers to break correlation between successive samples
- Target networks to stabilize training
- ε-greedy exploration policies to balance exploration and exploitation

RL Integration in DevSecOps

The RL agent integrates with CI/CD orchestrators and incident-response tooling to:

- Decide whether to allow or block deployments
- Select optimal remediation strategies for detected incidents
- Adjust anomaly thresholds based on past outcomes
- Prioritize alerts and escalation paths

Safety constraints ensure that high-risk actions (e.g., infrastructure-wide isolation) require human approval.

Unified ML–NLP–RL Workflow

The integrated AID-DF workflow operates as follows:

1. Developers push code; ML models evaluate code and dependency risk.
2. During build, logs and SBOMs are analyzed for anomalies and vulnerabilities.
3. At deployment, RL evaluates risk and decides whether to allow, block, or modify the rollout.
4. NLP continually checks configurations against regulatory baselines and updates policy-as-code rules.
5. Runtime telemetry is monitored by ML for behavioural anomalies.
6. When incidents occur, RL selects remediation actions, which are logged and fed back into training.
7. Compliance drift is detected by NLP-driven comparison and triggers corrective actions.
8. Feedback from all stages is used to retrain ML and RL models, improving performance over time.

This closed-loop architecture turns DevSecOps into a self-learning security ecosystem.

RESULTS AND EVALUATION

The AID-DF framework was evaluated in a production-like environment to measure its effectiveness in improving security outcomes, compliance coverage, and operational efficiency.

Experimental Setup

Microservice Environment

The testbed comprised:

- 100 microservices implemented in Go, Python, Node.js, and Java
- Kubernetes clusters on AWS EKS and Azure Kubernetes Service
- Service mesh-based traffic management
- Horizontal Pod Autoscaler (HPA) for autoscaling
- Multiple independent CI/CD pipelines (e.g., Azure DevOps, GitHub Actions)
- Infrastructure managed through IaC templates

Over the evaluation period, the environment produced:

- 9, 300 builds
- 12, 700 deployments
- million runtime log entries
- 2.7 million CI/CD log entries

Baseline Comparison Models

Three baseline configurations were defined:

1. *Traditional DevOps:* CI/CD without integrated security automation.
2. *Standard DevSecOps:* Inclusion of SAST, SCA, DAST, container scanning, and basic policy checks.
3. *Enhanced DevSecOps:* Standard DevSecOps plus policy-as-code enforcement and IaC scanning.

The proposed AID-DF framework was compared against these baselines.

Quantitative Evaluation

Six core metrics were measured: detection accuracy, compliance coverage, MTTD, MTTR, manual audit workload, and false-positive rate.

Detection Accuracy

Detection accuracy evaluates the ability to identify:

- Behavioural anomalies
- Policy violations
- Vulnerable dependencies
- Configuration drift events

Table 1 summarizes the results.

As shown in Table 1, the proposed AID-DF framework achieves the highest detection accuracy, primarily due to LSTM-based log modelling and autoencoder-driven anomaly detection [7, 8, 15].

Compliance Coverage

Compliance coverage measures the proportion of regulatory controls that are correctly mapped, validated, and continuously enforced (Table 2).

Table 1. Detection accuracy comparison between approaches.

Approach	Accuracy
Traditional DevOps	81.3%
Standard DevSecOps	86.4%
AID-DF (proposed)	92.7%

Table 2. Compliance coverage across approaches.

Approach	Coverage
Manual audit	52%
Standard DevSecOps	65%
AID-DF	88%

NLP-driven control extraction and semantic mapping substantially increase coverage compared with manual processes and baseline DevSecOps configurations [9, 10].

Mean-Time-to-Detect (MTTD)

MTTD captures the average time between the onset of an anomalous condition and detection.

- *Standard DevSecOps:* 4.2 hours
- *AID-DF:* 2.4 hours (42% reduction)

The use of ML-based behavioural models enables earlier recognition of deviations from normal operation.

Mean-Time-to-Respond (MTTR)

MTTR measures the average time from detection to containment or mitigation.

- *Standard DevSecOps:* 6.5 hours
- *AID-DF:* 3.1 hours (52% reduction)

RL-driven decision support enables more timely and effective remediation, improving overall resilience [11–13, 30].

Manual Audit Workload

Manual effort required for audit preparation and evidence collection decreased by approximately 50% when using AID-DF compared with baseline processes, due to automated evidence generation and continuous mapping of controls [7, 17].

False Positive Reduction

AID-DF reduced false positives by 36% relative to the baseline DevSecOps toolchain and led to a 29% decrease in ignored alerts. Developers reported greater trust in security signals when noise was reduced.

Qualitative Evaluation

Developer Experience

Feedback from development teams indicated:

- Fewer irrelevant alerts and clearer prioritization
- More actionable remediation recommendations embedded in CI/CD logs
- Reduced friction during deployment reviews and approvals

These improvements align with prior findings that ML-assisted filtering increases the usability of security tools [7, 19].

Security Team Efficiency

Security engineers observed:

- Prioritized alerts with contextual risk scores
- Faster root-cause analysis supported by correlated telemetry
- Automated generation of compliance evidence and audit artefacts

These factors collectively improved the team's ability to focus on high-impact issues.

Drift Reduction and Governance Consistency

NLP-based semantic compliance significantly reduced configuration drift. AID-DF automatically detected patterns such as non-encrypted storage buckets, permissive firewall rules, and missing network segmentation, triggering remediation or policy enforcement [9, 10].

Continuous Security Enforcement

By integrating checks at pre-commit, build, deployment, and runtime stages, AID-DF enforced continuous security rather than periodic scanning. This eliminated blind spots that often arise between scheduled scans or audits.

Ablation Study

An ablation study measured the contributions of each AI component:

ML only

- *Detection accuracy:* 87%
- *MTTD:* 3.3 hours

ML + NLP

- *Detection accuracy:* 89%
- *Compliance coverage:* 81%

ML + NLP + RL (full AID-DF)

- *Detection accuracy:* 92.7%
- *Compliance coverage:* 88%
- *MTTR:* 3.1 hours

The largest incremental gains in response times arise from RL-driven remediation, while NLP primarily enhances compliance coverage.

Enterprise Implications

Scalability in Multi-Cloud Systems

AID-DF is designed to operate across AWS, Azure, and GCP identity and security models, as well as Kubernetes RBAC. This multi-cloud support simplifies operations for organizations with heterogeneous environments [26–28].

Cost Reduction

AI-driven automation reduces manual review hours, mitigates the likelihood and impact of breaches, and optimizes pipeline throughput. These benefits collectively translate into significant cost savings over time [14, 19].

Improved Security Maturity

Organizations adopting AID-DF can progress towards higher DevSecOps maturity levels characterized by:

- Real-time threat detection
- Continuous and automated compliance alignment
- Autonomous governance actions guided by AI [18, 19, 29]

Better Audit Readiness

With automated control mapping and evidence generation, audit preparation becomes faster and less disruptive. Security teams can provide regulators and internal auditors with up-to-date evidence and clear mappings between controls and technical configurations [7, 17].

Enhanced Resilience

RL-driven remediation enables:

- Autonomous rollback of risky releases
- Isolation of compromised workloads
- Proactive blocking of high-risk deployments
- Reinforcement of zero-trust principles [13, 29, 30]

Threats to Validity

Several factors may influence the generalizability of results:

- *Dataset generalizability:* The training data reflects a specific microservice environment and may differ from other organisations' infrastructures.
- *RL safety and robustness:* Exploration strategies must be constrained to avoid unsafe actions; additional guardrails may be required for production use [12, 30].
- *Model interpretability:* Deep models can be difficult to interpret, which may affect acceptance in highly regulated sectors. Explainable AI techniques are needed [29].
- *Integration complexity:* Legacy systems and bespoke architectures may require additional engineering work to integrate with AID-DF.

CONCLUSION

The transition to cloud-native architectures, microservices, and IaC-driven CI/CD pipelines has exposed limitations in traditional security and compliance practices. Static scanners, fragmented tooling, and manual audits cannot keep pace with the scale and dynamism of modern distributed systems. This paper presented the AI-Driven DevSecOps Automation Framework (AID-DF), which integrates ML for behavioural anomaly detection, NLP for semantic regulatory interpretation, and RL for adaptive remediation and deployment governance.

Evaluation across 100 microservices in a hybrid AWS–Azure environment showed that AID-DF improves detection accuracy, compliance coverage, MTTD, and MTTR while reducing manual audit workload and false positives. The framework provides developers and security teams with context-aware insights and automated actions that enhance both security and delivery velocity. These findings indicate that AI-enabled DevSecOps can transform security from a reactive, rule-driven control to a proactive, self-optimizing governance capability aligned with cloud-native operations.

Future Work

Several opportunities remain to extend the proposed framework:

1. *Explainable AI (XAI) for governance transparency:* Integrating techniques such as SHAP and LIME would help stakeholders understand model decisions and support regulatory audits.
2. *Federated learning for privacy-preserving DevSecOps:* Training models across multiple organizations without sharing raw logs or sensitive data could improve performance while maintaining confidentiality.
3. *LLM-enhanced remediation generation:* Domain-specific large language models could generate remediation playbooks, policy-as-code templates, and compliance evidence summaries tailored to organizational standards.
4. *Multi-agent reinforcement learning (MARL):* Coordinated RL agents across clusters and regions could optimize policies and remediation strategies in globally distributed environments.
5. *Open benchmark datasets:* Public datasets for CI/CD logs, IaC misconfigurations, and compliance drift would facilitate reproducible research and objective comparison of DevSecOps AI approaches.

REFERENCES

1. Zhao X, Clear T, Lal R. Identifying the primary dimensions of DevSecOps: A multi-vocal literature review. Journal of Systems and Software. 2024 Aug 1;214:112063.

2. Rzig DE, Houerbi A, Chavan RG, Hassan F. Empirical Analysis on CI/CD Pipeline Evolution in Machine Learning Projects. arXiv preprint arXiv:2403.12199. 2024 Mar 18.
3. Belouaddane L, Ait Said M, Marzouk A, Benmakhlouf A. Microservice Architecture DevOps Integration Challenges: A Qualitative Study. InInternational Conference on Advanced Engineering, Technology and Applications 2024 May 24 (pp. 96–108). Cham: Springer Nature Switzerland.
4. Devarakonda RR. An Integrated Approach for Security and Compliance on a Cloud-Based DevOps Platform. Available at SSRN 5234673. 2021 Dec 1.
5. Deng S, Zhao H, Huang B, Zhang C, Chen F, Deng Y, Yin J, Dustdar S, Zomaya AY. Cloud-native computing: A survey from the perspective of services. Proceedings of the IEEE. 2024 Feb 12;112(1):12–46.
6. Filepp R, Adam C, Hernandez M, Vukovic M, Anerousis N, Zhang GQ. Continuous compliance: Experiences, challenges, and opportunities. In2018 IEEE World Congress on Services (SERVICES) 2018 Jul 2 (pp. 31, 32). IEEE.
7. Neto EC, Iqbal S, Buffett S, Sultana M, Taylor A. Deep learning for intrusion detection in emerging technologies: a comprehensive survey and new perspectives. Artificial Intelligence Review. 2025 Nov;58(11):1–63.
8. Islam MS, Rakha MS, Pourmajidi W, Sivaloganathan J, Steinbacher J, Miranskyy A. Anomaly detection in large-scale cloud systems: An industry case and dataset. In2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) 2025 Apr 27 (pp. 377–388). IEEE.
9. Gillioz A, Casas J, Mugellini E, Abou Khaled O. Overview of the Transformer-based Models for NLP Tasks. In2020 15th Conference on computer science and information systems (FedCSIS) 2020 Sep 6 (pp. 179–183). IEEE.
10. Gaddam N. AI-Based Cloud Governance for Multi-Cloud Compliance Management. Journal ID. 2024;2563:4512.
11. Lopez-Martin M, Carro B, Sanchez-Esguevillas A. Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Systems with Applications. 2020 Mar 1;141:112963.
12. Dowling S, Schukat M, Barrett E. Improving adaptive honeypot functionality with efficient reinforcement learning parameters for automated malware. Journal of Cyber Security Technology. 2018 Apr 3;2(2):75–91.
13. Ahmadi C, Chen JL. Survey on Reinforcement Learning Techniques for Enhancing Security and Efficiency in Zero Trust Networks. In2024 10th International Conference on Applied System Innovation (ICASI) 2024 Apr 17 (pp. 427–429). IEEE.
14. Algarni AM, Malaiya YK. A consolidated approach for estimation of data security breach costs. In2016 2nd International Conference on Information Management (ICIM) 2016 May 7 (pp. 26–39). IEEE.
15. Giorgio L, Nicola M, Fabio S, Andrea S. Continuous defect prediction in CI/CD pipelines: A machine learning-based framework. InInternational Conference of the Italian Association for Artificial Intelligence 2021 Dec 1 (pp. 591–606). Cham: Springer International Publishing.
16. Shin Y, Williams L. An empirical model to predict security vulnerabilities using code complexity metrics. InProceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement 2008 Oct 9 (pp. 315–317).
17. Fu M, Pasuksmit J, Tantithamthavorn C. Ai for devsecops: A landscape and future opportunities. ACM Transactions on Software Engineering and Methodology. 2025 Apr 28;34(4):1–61.
18. Pitkar H. Cloud Security Automation Through Symmetry: Threat Detection and Response. Symmetry. 2025 Jun 1;17(6):859.
19. Jia Z, Shen C, Yi X, Chen Y, Yu T, Guan X. Big-data analysis of multi-source logs for anomaly detection on network-based system. In2017 13th IEEE conference on automation science and engineering (CASE) 2017 Aug 20 (pp. 1136–1141). IEEE.

20. Gorle S, Muthusamy P, Inampudi RK. Consent-Driven Continuous Delivery with Open Policy Agent and Spinnaker. Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online). 2025 Jul 5;4(2):102-12.
21. Force JT. Security and privacy controls for information systems and organizations. National Institute of Standards and Technology; 2020 Mar 16.
22. Folorunso A, Mohammed V, Wada I, Samuel B. The impact of ISO security standards on enhancing cybersecurity posture in organizations. World Journal of Advanced Research and Reviews. 2024;24(1):2582-95.
23. Li J, Li H. Evolution of Application Security based on OWASP Top 10 and CWE/SANS Top 25 with Predictions for the 2025 OWASP Top 10. In2025 International Conference on Inventive Computation Technologies (ICICT) 2025 Apr 23 (pp. 1178–1183). IEEE.
24. Rodrigues BB. Google Cloud Digital Leader Certification Guide: A Comprehensive Study Guide to Google Cloud Concepts and Technologies. Packt Publishing Ltd; 2024 Mar 15.
25. Okeyode D, Kirui J. DevSecOps for Azure: End-to-end supply chain security for GitHub, Azure DevOps, and the Azure cloud. Packt Publishing Ltd; 2024 Aug 28.
26. Zhang R, El-Gohary N. Transformer-based approach for automated context-aware IFC-regulation semantic information alignment. Automation in Construction. 2023 Jan 1;145:104540.
27. Roger J, Alexander D. AI-Powered Risk Assessment Models for Enhancing Data Governance Compliance. URL: https://www. researchgate. net/publication/390941575. 2025 Jan 13.
28. Thiyagarajan G, Bist V, Nayak P. AI-Driven Configuration Drift Detection in Cloud Environments. Gogulakrishnan Thiyagarajan, Vinay Bist, Prabhudarshi Nayak.(2024). AI-Driven Configuration Drift Detection in Cloud Environments. International Journal of Communication Networks and Information Security (IJCNIS). 2024 Nov 10;16(5):721-43.
29. Zeydan E, De Alwis C, Khan R, Turk Y, Aydeger A, Gadekallu TR, Liyanage M. Quantum Technologies for Beyond 5G and 6G Networks: Applications, Opportunities, and Challenges. arXiv preprint arXiv:2504.17133. 2025 Apr 23.
30. Rodrigues BB. Google Cloud Digital Leader Certification Guide: A Comprehensive Study Guide to Google Cloud Concepts and Technologies. Packt Publishing Ltd; 2024 Mar 15.

