JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Infrastructure-as-Code (IaC) Drift Security with GitOps-Driven Auto-Remediation in AWS

[1]Liktha Yogesh, [2]Nidhi N, [3]Spoorthi Rai, [4]Shreyas Reddy B, [5]Dr. Deepthi VS

[1]Student, [2]Student, [3]Student, [4]Student, [5]Professor,
[1 2 3 4 5]Computer Science Engineering (Cyber Security),
[1 2 3 4 5]Dayananda Sagar College of Engineering, Bangalore, India

Abstract: With the fast growth of cloud computing and DevOps, Infrastructure-as-Code (IaC) has become the key-stone of managing scalable and reproducible cloud environments as yet, configuration drift-the difference between declared infrastructure computer code and deployed state-represents fundamental threats to cloud security system, compliance, and operational reliability. Nonetheless, this work presents an Automated IaC Drift Security Framework using Terraform, AWS Config, Open Policy Agent (OPA), and GitOps processes for continuous drift detection, validation, and remediation. The approach path uses GitOps pipeline to automatically roll back to the prior secure state after endlessly detecting difference between AWS resources and IaC templates and classifying them into security severity level. Additionally, the envisioned modeling be intended to create a self-healing, security-oriented, and auditable cloud infrastructure that addresses research and industry demands for robust cloud management.

IndexTerms **- Infrastructure-as-Code, Drift Detection, GitOps, Terraform, AWS, DevSecOps, Cloud Security.**

I. INTRODUCTION

With cloud environments becoming increasingly dynamic in nature, ensuring their configuration integrity is critical. Additionally, Tools like Terraform and AWS CloudFormation for Infrastructure-as-Code (IaC) allow provisioning and versioned deployment in an automated manner, which takes human error out of the manual configuration. Even so, infrastructure drift-wherein the live state of the cloud differs from its specified IaC template-is still an ongoing issue.

II. LITERATURE REVIEW

The increasing complexity of cloud infrastructures governed by Infrastructure-as-Code (IaC) has introduced novel challenges in ensuring configuration consistency and security.
This section discusses recent industry and research works addressing IaC drift, fix, and DevSecOps automation.

A. Drift and Reconciliation Defects

Hassan et al. [1] conducted a large empirical study examining 5,110 reconciliation defects in IaC projects and discovered new defect classes like inventory and semantic mismatches. Although fundamental, their effort was merely diagnostic in nature and did not provide any automatic remediation. Additionally, Gera and Gabrani [9] introduced an AI-based anomaly detection model that automatically detects cloud configuration drift patterns in realtime. Consequently, while promising, it doesn't support runtime rollback or risk prioritization. These results highlight the necessity for realtime, automated drift management embedded within operational pipelines.

B. Automated IaC Repair Systems

Saavedra et al. [2] proposed InfraFix, a tech-agnostic repair system based on Intermediate Representation (IR) and Satisfiability Modulo Theory (SMT) reasoning to fix erroneous IaC scripts with a high rate of more than 95 Weiss et al. [6] created Tortoise, which balances imperative repairs with declarative IaC definitions. Its ideas influenced the reconciliation logic taken in this work, but it was only applicable to small-scale system configurations.

C. Security and Policy-as-Code Integration

Verdet et al. [4] performed a systematic review of security practice in IaC and found sparse usage of tools like Checkov and Tfsec within enterprise workflows. Nevertheless, Velu et al. [10] presented Security as Code with Open Policy Agent (OPA) and Rego to write compliance rules directly into IaC pipelines, which was effective in security validation. Okhonmina and Trodd [15] benchmarked a number of IaC security scanners, including Checkov, Tfsec, Terrascan, and Tflint, and found it to have fragmented coverage and endorsing the necessity of converged policy enforcement, an area that this work bridges.

D. GitOps and Continuous Delivery

Farcic et al. [8] introduced the model of GitOps for Continuous Delivery, where Git is the single source of truth and reconciliation is defined via pull requests. The Secure GitOps research [7] emphasized trust boundaries and hazards in Git-

based automation. Consequently, both increase the operational reliability but do not include security-conscious rollback, which our model incorporates by introducing GitOps and policy- driven remediation together.

E. Industry Tools and Practical Implementations

AWS documentation [11][12] and HashiCorp guidelines [13] address built-in drift detection (CloudFormation Drift API, AWS Config) and the Terraform refresh-only plan. These are operational baselines but detection-only. Vendor -neutral guides such as Code Ocean [18] and Firefly [19] address drift detection best practices, while Spacelift/env0 [20] describes edge cases in plan behavior. Additionally, these industry references show active development but with minimal automation and cross-tool integration. Additionally,

F. IaC Quality, Maintainability, and Smells

Bessghaier et al. [5] etected IaC "smells"- bad patterns that are associated with drift-prone misconfigurations. Furthermore, Mkaouer et al. [17] asked practitioners to rate these smells by severity, providing a glimpse of risk prioritization. Consequently, Bolhuis [14] expanded static analysis by identifying non-security drifts like cost inefficiencies, demonstrating the necessity for complete checks that this project undertakes. Nevertheless,

G. Emerging Directions

Isazadeh et al. [16] studied change drift in microservices with a comparison to configuration drift in infrastructure. Zeng [21] suggested automated serverless system configuration management, concurrent with our solution for resource drift. Hence, The SANER 2024 profile by Bessghaier et al. Therefore, [22] confirms previous results, showing the prevalence of IaC quality problems at scale. Collectively, these papers form a solid foundation but identify the lack of an automated drift remediation framework that is security-focused-the gap which the suggested research fills directly.

H. Secure Network and Additional Resources

Deepthi et al. [23]-[26] explored intelligent network models for crowd prediction and malicious node detection in MANETs, emphasizing adaptive monitoring and trust-based evaluation. Adnan et al. [25] proposed an AI-driven ransomware detection framework integrating optimization with machine learning for improved threat classification. Nevertheless, Tajuddin and Nandini [27]-[30] developed biometric-based cryptographic key generation and multi-trust agent systems to enhance authentication and security. Collectively, these studies advance automation and secure computing-principles aligned with the proposed IaC drift remediation framework.

TABLE I: Comprehensive Literature Review on Infrastructure-as-Code (IaC)

No	Paper Title	Authors	Problem Addressed	Methodology	Key Findings	Limitations	Relevance to Present Work
1	State Reconciliation Defects in Infrastructure as Code	Hassan et al.	State defects in IaC deployments	Empirical analysis of 5,110 defects	Identified new defect classes	Lacks remediation automation	Motivates automated reconciliation
2	InfraFix: Technology-Agnostic Repair of IaC	Saavedra et al.	Repairing faulty IaC scripts	IR + SMT reasoning	95% repair success rate	Handles static issues only	Supports generatio automatic fix n
3	Automated Drift Detection and Remediation in IaC Deployments	Solanki A.M.	Drift detection and correction	Terraform + AWS APIs	Validated CFN drift	Lacks GitOps support	Basis for drift automation
4	Exploring Security Practices in IaC	Verdet et al.	Security practices in IaC	Empirical study	Found major security adoption gaps	No practical solution	Justifies OPA/Checkov Layer
5	On the Prevalence of IaC Smells	Bessghaier et al.	IaC quality and code smells	Smell taxonomy	Linked to misconfigurations	No mitigation	Supports risk detection
6	Tortoise: Interactive Configuration Repair	Weiss et al.	System config correction	Synthesis-based repair	Successful in controlled settings	Not scalable	Informs rollback logic
7	Secure GitOps: Analysis and Solutions	ACM Queue	Secure GitOps practices	Security review	Identified GitOps threats	Conceptual only	Supports secure rollout
8	GitOps for Continuous Delivery	Farcic et al.	Continuous delivery via GitOps	Case study	Git as source of truth	No runtime enforcement	Basis for version control
9	AI-Driven Configuration Drift Detection	Gera & Gabrani	ML-based drift analysis	Anomaly detection	Detected runtime drift	No remediation pipeline	Guides adaptive thresholds
10	Security as Code with OPA/Rego	Velu et al.	IaC policy enforcement	Rule-based validation	Strong compliance assurance	Prototype coverage	Used for policy module
11	Detect Drift on CloudFormation Stacks	AWS Docs	Drift identification gap	Drift API usage	Effective detection	Detection- only	Integrated into design
12	AWS Config Conformance Packs	AWS Docs	Automated compliance	Pre-built rules	Simplifies validation	Vendor lock- in	Base for generic solution
13	Terraform Plan (Refresh Mode)	HashiCorp Docs	Drift via refresh	Comparison engine	Identifies infra drift	Partial detection	Core detection method
14	Catching Cost Issues in IaC	Bolhuis	Cost-related drift	Static linting	Found cost inefficiencies	Security ignored	Supports cost optimization
15	Fortifying Cloud DevSecOps with Terraform	Okhonmina & Trodd	IaC security automation	Tool evaluation	Compared tools	No automation layer	Aids tool choice
16	Change-Drift in Microservices	Isazadeh et al.	Drift in microservices	Comparative study	Found dependency Drifts	Not IaC- specific	Contextual insight
17	Do Experts Agree on IaC Smells?	Mkaouer et al.	IaC smell severity	Expert survey	Ranked critical smells	Subjective limits	Guides prioritization
18	Detecting Drift in CloudFormation	Code Ocean	CLI drift check automation	Practical workflow	Effective demo	Non- research	Process reference
19	Firefly Drift Detection Guide	Firefly Docs	Multi-cloud drift visibility	Tool documentation	Unified drift view	No remediation	Operational reference
20	Terraform Refresh & Plan Behavior	Spacelift/env0	Drift edge cases	Technical comparison	Improved detection	Informational only	Refines drift logic
21	Automating Serverless Configurations	Zeng	Serverless IaC automation	IaC automation flow	Reduced manual config errors	Limited to serverless	Technique reference
22	IaC Smells Profile (SANER 2024)	Bessghaier et al.	Empirical validation of smells	Large-scale study	Confirmed prior findings	No mitigation	Supports dataset reliability

III. Research Gaps

Literature points out several important gaps in current IaC management and security studies:

- **No automated remediation:** The majority of frameworks only detect drift and need to be corrected by hand. Hence,
- **Security prioritization is limited:** Drifts are not differentiated based on risk or compliance effect. Additionaly,
- **No GitOps integration:** Not many systems return configuration consistency through version control rollbacks. Thus, consequently,
- **Seller lock-in:** Solutions such as AWS Config are confined to individual ecosystems. Hence, Nevertheless,
- **Fragmented validation tools:** There is no one pipeline that integrates drift detection, security validation, and remediation.

The system proposed here fills these gaps by creating an end- to-end AWS-centric drift security framework.

IV.PROPOSED METHODOLOGY

The intended study seeks to develop and automate an Infrastructure-as-Code (IaC) Drift Security Framework to identify, inspect, and remediate AWS environment configuration drifts. Nonetheless, The Terraform use by framework, AWS Config, Open Policy Agent (OPA), and GitOps automation to keep the deployed cloud infrastructure in sync with its stated IaC definitions continuously and uphold compliance and security posture.

Consequently, the approach is separated into the following main phases Figure 1 illustrates the architecture of the proposed IaC Drift Security Framework. It shows how Terraform templates from the Git repository be endlessly monitor by the Drift Detection Engine, validated through the Security Policy Engine, and remediated via a GitOps CI/CD pipeline. The Monitoring and Alert System collects drift and violation logs, sending alerts and reports to the user. This workflow ensures automated drift detection, policy-based validation, and secure rollback within the AWS cloud environment.

Fig. 1. Architecture Diagram

A. Drift Detection

Utilize Terraform plan (refresh-only) and AWS Config APIs to detect differences between current and desired states of infrastructure. Detect configuration drifts for compute, storage, and network resources.

B. Validation & Policy Enforcement

Invoke Open Policy Agent (OPA) and Checkov to check the detected drifts against compliance and security policies (e.g., public S3 buckets, open ports). Categorize drifts as Low, Medium, or High severity. Hence,

C. Automated remediation

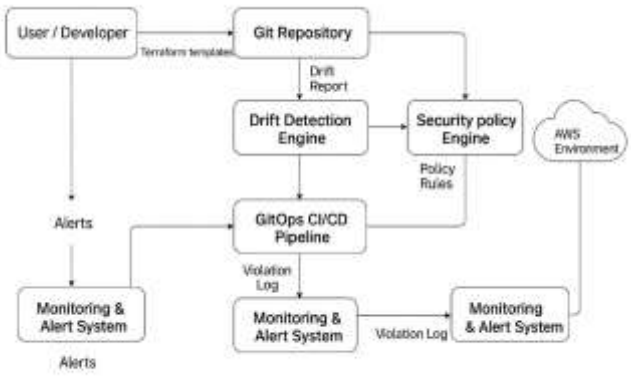
Use GitOps pipelines (GitHub Actions/ArgoCD) to roll back to the last secure commit. Keep complete audit logs and change traceability. Nevertheless,

D. Monitoring and Reporting

Utilize AWS CloudWatch and Grafana to monitor drift metrics, rollback activity, and compliance status. Nevertheless, Offer security dashboards and alerting systems. Moreover, this process guarantees continuous compliance and security resilience for all AWS deployments.

V. Significance of the Proposed System

The IaC Drift Security Framework proposed is of both academic and practical importance:



A. Security Assurance

Avoids misconfigurations that potentially leak cloud assets or break compliance regulations. Moreover,

B. Operational Resilience

Facilitates self-healing infrastructure with minimum human input.

C. Auditability

keeps immutable logs and rollback history for governance purposes. Hence,

D. Sustainability

is in alignment with SDG 9 (Industry, Innovation, and Infrastructure) and SDG 16 (Peace, Justice, and Strong Institutions) by promoting secure and transparent digital infrastructure.

E. Industry Relevance

Complements seamlessly with current AWS and DevOps toolchains, enabling it to be deployed in actual production environments.

VI. RESEARCH OPPORTUNITIES AND FUTURE WORK

Nonwithstanding the immense advancements outlined in this model, there are still some research directions and unresolved issues around Infrastructure-as-Code (IaC) security and drift control. Furthermore, the following research directions are listed for further investigation:

A. **Multi-Cloud Drift Reconciliation**

The majority of current drift detection methodologies, including the present framework, are platform-specific (e.g., AWS). Future work may involve the creation of cloud -agnostic drift reconciliation mechanisms that concurrently support Terraform, Azure ARM, and Google Deployment Manager. This would allow organizations to have consistent security posture across hybrid and multi-cloud environments.

B. **AI-Driven Drift Prediction and Risk Scoring**

While existing systems identify information drift and remediate it reactively, subsequent efforts can concentrate on predictive models based on machine learning to predict likely drift events prior to occurrence. A model trained on data can scan commit history, user activity, and cloud change information to provide risk scores to deployments at runtime and prevent misconfigurations ahead of time.

C. **Continuous Compliance Assurance**

Combining compliance-as-code models for standards like CIS Benchmarks, ISO 27001, PCI-DSS, and GDPR is a primary research area. Future development can look into automatic mapping between OPA/Rego regulations and world -wide regulatory needs, with real-time continuous compliance verification.

D. **Autonomous Remediation Intelligence**

Intelligence Though the system in question has GitOps - based rollback support, future releases might include context -aware remediation engines that can choose the best fix (rollback, patch, or quarantine) based on impact analysis and policy severity. Nevertheless, these systems may employ reinforcement learning to get better over time by learning from past drift and remediation experience. Moreover,

E. **Distributed and Serverless Drift Correlation**

New architectural styles like Kubernetes clusters and server-less functions create new challenges for drift tracking. Furthermore, Research can be pushed to cross-layer drift correlation, correlating application-level configuration changes (e.g., Helm charts) with underlying IaC drift in compute or network infrastructure.

F. **Blockchain-Based Provenance and Auditability**

Incorporating blockchain to enable immutable logging of drift events, policy evaluations, and rollbacks can improve transparency and accountability. This kind of strategy can offer tamper-proof audit trails for regulatory and forensic use in DevSecOps processes.

VII. CONCLUSION

The paper introduces a security-focused methodology for controlling infrastructure drift in IaC- based AWS environments. Through the integration of Terraform's declarative provisioning, OPA's policy enforcement, and GitOps automated rollback, the presented framework accomplishes continuous security verification, remediation automation, and compliance visibility. The study makes a contribution to DevSecOps automation research as it shows the way policy-based drift management may improve cloud system integrity. Future developments will apply this framework to multi-clouds and investigate AI-based drift prediction as the basis for proactive remediation.

VIII. ACKNOWLEDGMENT

The authors wish to offer their sincere thanks to Dr. Deepthi V.S., Department of Computer Science and Engineering (Cybersecurity), Dayananda Sagar College of Engineering, for her precious guidance and mentorship during this research.

REFERENCES

[1] A. Hassan *et al.*, "State Reconciliation Defects in Infrastructure as Code," in *Proc. ACM FSE*, 2024.
[2] M. Saavedra *et al.*, "InfraFix: Technology-Agnostic Repair of Infrastructure as Code," *arXiv/FSE*, 2025.
[3] A. M. Solanki, "Automated Drift Detection and Remediation in IaC Deployments," MSc Thesis, National College of Ireland, 2024.
[4] A. Verdet et al., "Exploring Security Practices in Infrastructure as Code," arXiv, 2023.
[5] N. Bessghaier et al., "On the Prevalence, Co-occurrence, and Impact of IaC Smells," in Proc. IEEE SANER, 2024.
[6] A. Weiss et al., "Tortoise: Interactive System Configuration Repair," arXiv, 2017.
[7] ACM Queue, "Secure GitOps: Analysis and Solutions," ACM Queue Magazine, 2021.
[8] V. Farcic et al., "GitOps for Continuous Delivery," Research Compendium, 2022.
[9] A. Gera and A. Gabrani, "Configuration Drift in Cloud Systems: An AI -Driven Approach," Int. J. of Communication Networks and Information Security (IJCNIS), 2023.
[10] V. Velu et al., "Security as Code with OPA and Rego," in Proc. CEUR Workshop, 2023.
[11] AWS, "Detect Drift on CloudFormation Stacks," AWS Documentation, 2024.
[12] AWS, "AWS Config Conformance Packs," AWS Documentation, 2024.
[13] HashiCorp, "Terraform Plan (Refresh-Only)," HashiCorp Documentation, 2023.
[14] J. Bolhuis, "Catching Cost Issues in IaC Artifacts using Linters," MSc Thesis, 2024.
[15] T. Okhonmina and J. Trodd, "Fortifying Cloud DevSecOps using Terraform," MSc Project, University of West London, 2025.
[16] R. Isazadeh et al., "Change Drift in Microservices," arXiv, 2025.
[17] M. Mkaouer et al., "Do Experts Agree About Smelly Infrastructure?," Preprint, 2025.
[18] Code Ocean, "Detecting Drift in CloudFormation," Technical Guide, 2025.
[19] Firefly, "Guide: Detect Drift in CloudFormation," Firefly Documentation, 2024.
[20] Spacelift/env0, "Terraform Refresh & Plan Behavior," Blogs and Technical Docs, 2024.
[21] J. Zeng, "Automating Serverless Resource & Security Config," MSc Thesis, University of British Columbia (UBC), 2024.
[22] N. Bessghaier et al., "IaC Smells (Profile)," in Proc. IEEE SANER, 2024.
[23] V. S. Deepthi, N. Venkat Chavan, S. Shanbhag, and S. S. Dandappala, "Crowd density estimation and location prediction in public transport system," Int. J. Eng. Res. Technol. (IJERT), vol. 11, no. 7, pp. 1–4, 2022.
[24] V. S. D. Vagdevi, "Behaviour Analysis and Detection of Blackhole Attacker Node under Reactive Routing Protocol in MANETs," in Proc. Int. Conf. on Networking, Embedded and Wireless Systems, 2018.
[25] M. M. Adnan, S. V. Devi, and N. Yamsani, "Sine Cosine Reptile Search Algorithm with Grid Search Support Vector Machine Based Ransomware Detection and Classification," in Proc. Int. Conf. on Integrated Circuits and Communication, 2024.
[26] V. S. Deepthi and S. Vagdevi, "Multiphase Detection and Evaluation of AODV for Malicious Behaviour of a Node in MANETs," in Proc. Int. Conf. on Electrical, Electronics, Communication & Computing, 2018.
[27] M. Tajuddin and C. Nandini, "Secured Crypto Biometric System Using Retina," Int. Adv. Res. J. Sci. Eng. Technol., vol. 2, no. 5, pp. 38–42, 2015.
[28] M. Tajuddin and C. Nandini, "Performance Measurement of Cryptographic Key Using Biometric Images," Int. J. Electrical Sciences, vol. 1, no. 2, pp. 15–20, 2015.
[29] M. Tajuddin and C. Nandini, "More Secured Cryptographic Key Generation through Retinal Biometric Using EBI Algorithm," Int. J. Eng. Innovations and Research, vol. 3, no. 5, pp. 616–620, 2014.
[30] M. Tajuddin and C. Nandini, "An Agent-Based Negotiating System with Multiple Trust Parameter Evaluation Across Networks," in Proc. 48th Annu. Conv. Computer Society of India (CSI), Springer, 2014, pp. 225 -233.