

# Infrastructure as Code Security: A Review of Automation, Compliance, and Risk Management

Mr. Swapnil Joshi <sup>1</sup>

<sup>1</sup> Assistant Professor, Department of Computer Sciences and Applications

**ABSTRACT:** Infrastructure as Code (IaC) has become an important approach for automating the provisioning, configuration, validation, and management of cloud and software infrastructure. This review covers five aspects of IaC: DevOps integration, security, automation, compliance, validation, and risk management. By version controlling, testing, validating, and automating the deployment of infrastructure configurations, IaC makes infrastructure changes and deployments more efficient and automated, which enables continuous integration and continuous deployment. Security considerations include misconfigurations, hardcoded secrets, inadequate access controls, vulnerable third-party modules, supply-chain attacks, and configuration drift. Infrastructure governance is reinforced by automation and compliance mechanisms that validate the infrastructure in the interest of a policy, check the compliance at runtime, and enforce the policy automatically. Automated validation embeds shift-left principles, static analysis, unit testing and policy-as-code to catch errors and violations before they reach the end-users. The other elements of risk management are identification of risk, configuration drift, vulnerability management, risk assessment, threat detection and continuous risk monitoring. In conclusion, the review underscores the importance of implementing secure, scalable, and reliable IaC practices that incorporates automation, compliance, validation, and risk management in the infrastructure lifecycle. These practices help reduce human error, ensure deployment uniformity, adhere to regulatory requirements, and enhance organizational resilience through sustainable infrastructure security, auditability, flexibility and suitability for changing operational needs.

**KEYWORDS:** Infrastructure as Code, DevOps, Automation, Compliance, Security, Automated Validation, Risk Management, Threat Detection, Configuration Drift, Terraform.

## 1.1 Introduction

The increasing adoption of automated infrastructure management has created a need for Infrastructure as Code Security to ensure secure, reliable, and controlled provisioning, configuration, and deployment of IT resources. Infrastructure as Code (IaC) is one of the most important technology components of DevOps, which allows to automate and code manage infrastructure deployment and configuration. With IaC, organizations can automate a wide variety of tasks, including installation, configuration, provisioning, and deployment of a variety of resources on the cloud or on-premises, such as servers, operating systems, virtual machines, containers, and applications [1]. Today, IaC has become a key part of the software development and deployment process, thanks to the success of the DevOps practices [2]. Consequently, the security and integrity of the underlying infrastructure and systems has become of the utmost priority. However, IaC is not

necessarily secure and has many security issues, such as misconfigurations, unintended secrets disclosure, weak access control, resource leaks, and more. It is critical that this security challenges is detected, understood and mitigated to ensure the assets of deployed systems are protected.

Compliance as Code (CaC) extends IaC into the realm of compliance policies and regulatory requirements. This allows organizations to continually test their environments on cloud to ensure that they comply with regulation without the need for manual compliance processes [3][4]. CaC sets policies and stores them as code to enable automatic compliance monitoring and notification of compliance breaches immediately after they occur. CaC offers several benefits, such as improved accuracy and consistency, making it a useful tool to help maintain compliance in large and dispersed cloud environments [5]. To overcome these challenges, intelligent automation can be used to develop self-healing systems, predictive analytics and autonomous

decision making [6] through the integration of AI and ML with IaC. Automation with AI optimizes infrastructure change on the fly, reducing risk for human intervention. Intelligent automation can improve DevOps teams agility, scalability, and security [7].

At every point in the infrastructure life cycle, security, configuration, compliance, and operational risks are identified, evaluated, and remedied as part of the risk management process in infrastructure as a service (IaC) [8]. IaC allows to enforce consistent and traceable settings which eliminates configuration drift, deployment mistakes, and security issues [9]. Automated validation, policy enforcement, vulnerability detection, and continuous monitoring can also bolster risk management through identifying potential issues prior to deployment and while infrastructure is in operation. Business, engineering, healthcare, transportation, energy, and climate change are just a few of the many fields that employ risk management strategies [10][11].[12] Risk assessment, perception and assessment, risk policy, and risk management are the components that are considered in these domains while thinking about risk control procedures [13]. The goal of the manager responsible for integrating the systems should be risk management, as

there are many potential difficulties and concerns that might affect the company's sustainability, including those pertaining to quality, the environment, and workplace health and safety.

### A. Structure of the paper

The structure of the paper: Section II presents the fundamentals of IaC and its integration with DevOps. Section III discusses automation, compliance, and automated validation. Section IV examines risk management, vulnerability assessment, and threat monitoring. Section V reviews recent IaC security studies, while Section VI concludes the paper and presents future research directions.

## II. FUNDAMENTALS OF INFRASTRUCTURE AS CODE

Infrastructure as a service focuses on programmatic methods for the supply and management of infrastructure resources [14]. Complete automation of the software lifecycle within a DevOps setting relies on this last action, making it a crucial contributor. The IaC solution integrates DevOps code into system deployment and management processes including setup, deployment, monitoring, analysis, and self-healing.

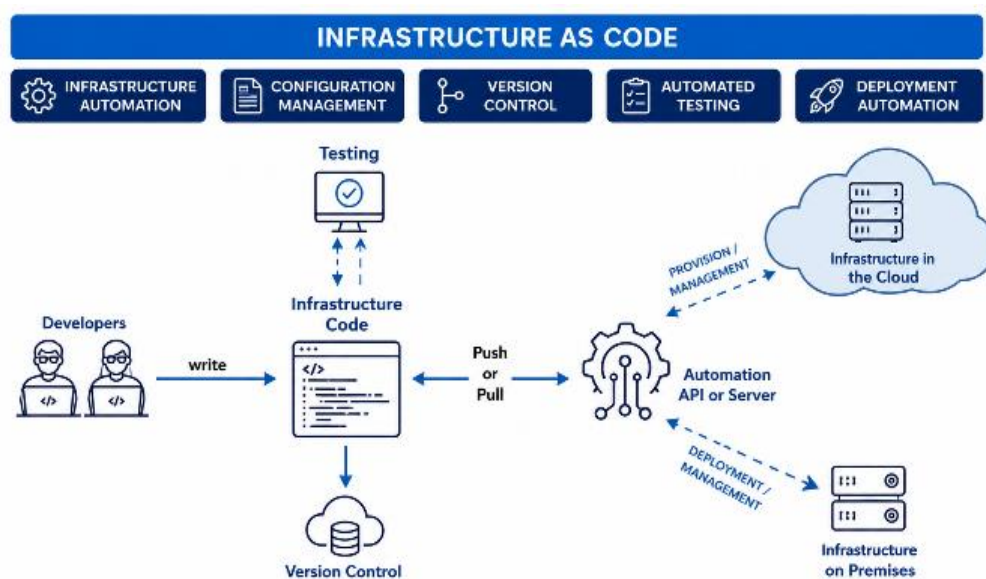


Fig. 1. Infrastructure as Code (IaC)

Figure 1 illustrates the IaC workflow, integrating infrastructure automation, configuration management, version control, automated testing, and deployment across cloud and on-premises environments through code-driven provisioning and management.

### A. Infrastructure-as-Code and DevOps,

A review can be structured around this idea.

Infrastructure as code, or IaC, is a way of thinking about and doing things with server and network infrastructure. IaC is a solution for software operations; it outlines the last steps of the DevOps process [15]. The goal of the DevOps methodology, as shown in Figure 2, is to improve communication and collaboration between the IT operations and software development teams so that high-quality software may be regularly and quickly produced.

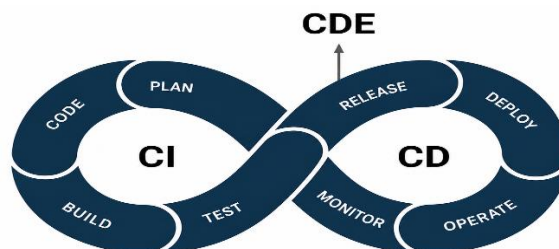


Fig. 2. DevOps automation and CI/CD pipelines

### 1) Continuous Integration (CI)

Several times during the day, it symbolizes the automatic process that integrates code changes made by numerous contributors into one codebase [16]. In order to avoid more significant disagreements later on in the development process, the approach aims to detect integration issues early on and provides developers with immediate feedback. An advanced form of automation, CD manages the whole deployment process, enabling teams to submit modifications to production many times per day. Key CI practices of CD include:

- Integration of code into the shared repository automatically.
- Executing automated tests, including unit tests, on every integration.
- Verification of code quality with static analysis.
- The ability to quickly update developers on the progress of their commits.

### 2) Continuous Deployment (CD)

CI takes it a notch further by automatically releasing updates to production after successful integration and

testing [17]. This allows for the continuous and expedited distribution of new features and issue fixes to end customers. CD is composed of two primary components:

- Continuous Delivery: The capability to automatically deploy changes to production or staging environments guarantees that code is release-ready at all times.
- Continuous Deployment: An automated deployment procedure that bypasses human interaction and directly pushes all changes to production after passing automated testing.

### 3) Integrating Terraform With CI/CD Pipelines

Integrating with CD and CI pipelines reveals the best way to automate infrastructure. When making changes to their software, many companies utilize Terraform as a component of their continuous integration and continuous delivery processes to update their infrastructure [18]. Figure 3 shows the typical process for storing Terraform configurations in Git and other version control repositories. Through systems such as Jenkins, GitLab CI, GitHub Actions, etc., the CI system runs the terraform fmt format check and the plan execution for suggested modifications on pull requests.

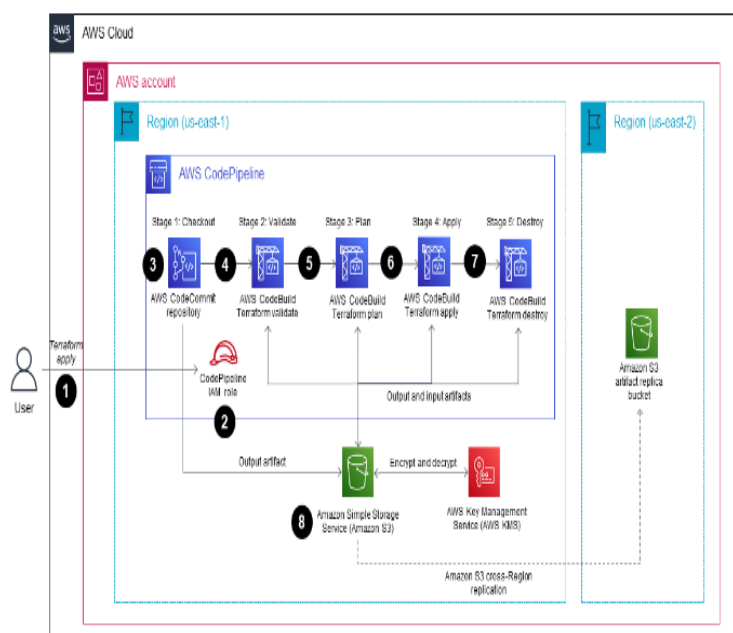


Fig. 3. Integrating Terraform with CI/CD Pipelines

- The terraform apply command in the AWS command line interface initiates the planned operations inside Terraform plans.
- In order to collaborate with Code, Code Build, AWS KMS, and Amazon S3, AWS Code Pipeline assumes leadership of a service role with suitable access policies.
- Since the "checkout" pipeline stage initiates this process, the Code Pipeline system receives the Terraform configuration from an AWS Code Commit repository.
- The "validate" phase of a Code Build project checks the Terraform configuration with the use of IaC validation tools and commands specific to Terraform.
- Code Pipeline's "plan" step generates a plan that adheres to the Terraform settings by utilizing the Code Build project. Prior to the test environment receiving the requested modifications, the AWS user has an opportunity to review the plan.

### B. Infrastructure as code security

IaC security focuses on protecting infrastructure configurations, deployment processes, and cloud resources throughout the IaC lifecycle. It addresses misconfigurations, vulnerabilities, credential exposure, compliance violations, and configuration drift through automated security testing, policy enforcement, vulnerability scanning, and continuous monitoring.

#### 1) Security Risks in IaC and DevSecOps

Although IaC offers many benefits, including automation, efficiency, and scalability, it also introduces new security complexities [19]. The main concerns with IaC are:

- **Misconfigurations:** Configuration drift—the most common risk in IaC is infrastructure configuration is different across environments (dev, test, prod). Vulnerabilities like data exposure or unauthorized access can occur if the security groups, permissions, and access controls are misconfigured.
- **Hardcoded Secrets:** There are instances where IaC files contain information that is required to be kept private, like API keys, database passwords, and encryption keys. This practice may reveal sensitive data and expand the attack surface of the infrastructure.
- **Lack of Access Controls:** If access controls are not managed correctly, it is possible for unauthorized users to access IaC files or templates, potentially causing changes to the cloud infrastructure.
- **Vulnerabilities in Third-Party Libraries:** IaC often

involves the use of third-party modules and libraries (e.g., Terraform providers, CloudFormation templates). These libraries may have weaknesses that could be inserted into the infrastructure.

- **Supply Chain Attacks:** External modules used in IaC may create potential supply chain attacks, where attackers compromise an external source to introduce malicious code into the infrastructure.
- **Dynamic Security Testing:** Tools like OWASP ZAP can be used to test IaC in real-time by simulating attacks on the deployed infrastructure.

### III. AUTOMATION AND COMPLIANCE IN IAC

Automation and compliance in IaC enable consistent, secure, and policy-driven infrastructure management. Automated validation, policy-as-code, continuous integration, and compliance checks identify misconfigurations before deployment. These practices improve security, reduce human errors, maintain regulatory requirements, and ensure infrastructure remains aligned with organizational standards throughout its lifecycle.

#### A. Runtime Iac-Based Compliance Management Method

Introduce the RICMa method in this Section. It enables runtime modelling, checking, and enforcement of compliance rules while protecting the integrity of application instances. The Compliance Rule Modelling Process and the Compliance Checking and Enforcement Process are the two main components of the technique, as shown in Figure 3 [20]. An application instance is a unique deployment of an application; this is due to the fact that the same IaC code may be utilized to instantiate an application numerous times.

##### 1) Compliance Rule Modeling Process

This procedure transfers all relevant compliance requirements arising from enterprise-external legislation and enterprise-internal policies into technically sound compliance rules that IaCMF can validate automatically: After reviewing the relevant compliance rule documentation, a Compliance Rule Expert employ an appropriate Technical Compliance Rule Modeling Tool to translate the rules into a machine-readable format.

##### 2) Compliance Checking and Enforcement Process

The completion of compliance jobs is the main goal of this procedure. An application instance's compliance rules can be represented by a user-defined object called a compliance job. This job specifies the information needed to remedy any possible violations and then verify the application instance. Each compliance work is treated independently, as seen in Figure 4, and the procedure is

described in additional detail below. The specifications for RICMa, the program that puts this procedure into action, are part of the given data.

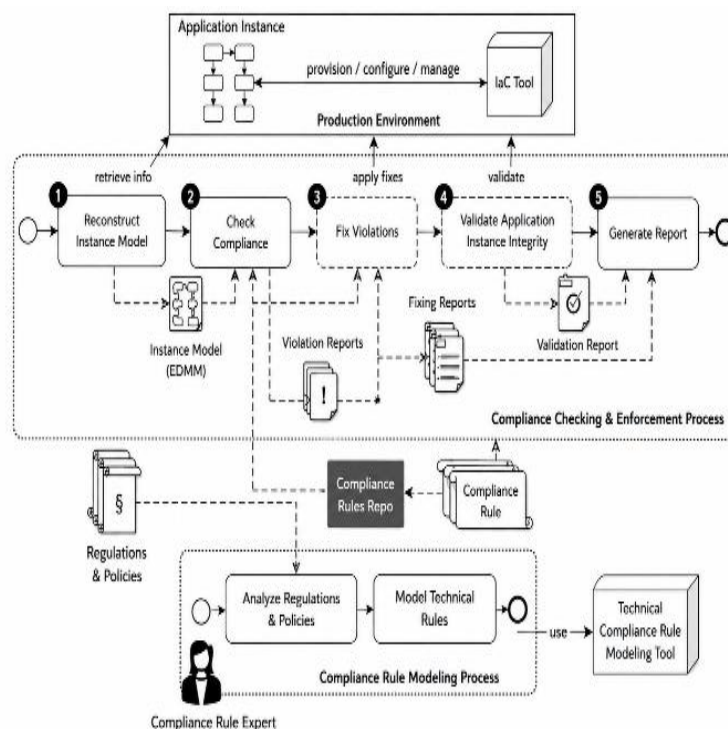


Fig. 4. The RICMa method

## B. Automated Validation

Automated validation in IaC continuously checks infrastructure configurations before and during deployment. It detects and notifies syntax errors, security vulnerabilities, violations of policies, and misconfigurations automatically. The inclusion of validation in CI/CD pipelines can help organizations avoid deployments that are faulty, ensure compliance, lessen manual work, and enhance the reliability and security of their infrastructure.

### 1) Shift-Left Testing in IaC

Validation should take place before changes are made in production environments to ensure deployments are error-free. This principle is called "shift-left testing," and it means putting checks into the earliest phases of the development pipeline. Usually pre-deployment validation consists of:

- **Syntax Checks:** Make sure configuration files are correct and are parsable.
- **Static Code Analysis:** Tools such as tflint, Checkov and cfn-lint can be used to spot insecure configurations, outdated modules or policy violations.
- **Unit Tests for IaC:** Tools such as Terratest, KitchenCI, or AWS CloudFormation Guard ensure that infrastructure is deployed as intended.
- **Policy-as-Code:** OPA (Open Policy Agent) or

Sentinel is a tool that can enforce compliance rules on cloud resource deployments prior to deployment, such as "no public S3 buckets."

### 1) Best Practices for IaC and Automated Validation

Version control of infrastructure, standardized components, pre-deployment configuration testing, automated security scanning, configuration compliance policies, and automated testing are all good practices for IaC and automated validation. IaC should be broken down into small, modular components that represent a specific configuration or set of resources [21]. The modules can be shared amongst teams, versioned and tested in isolation. This approach.

- Improves code reuse and maintainability.
- Makes it easier to test and validate discrete components.
- Allows abstraction of cloud vendor specifics for cross-platform flexibility.

## IV. RISK MANAGEMENT IN INFRASTRUCTURE AS CODE (IAC)

Risk management in IaC focuses on identifying, assessing, and controlling risks associated with infrastructure provisioning and configuration. It addresses security vulnerabilities, misconfigurations, credential exposure, configuration drift, compliance violations, and deployment failures. Automated scanning, risk assessment, policy enforcement,

monitoring, and recovery mechanisms help minimize operational and security risks.

### A. Risk Management

There is a tight relationship between policy analysis and risk management [22]. Everyone from people to governments to private sector companies and groups may benefit from having a well-thought-out policy to follow when making choices and working toward goals.

#### 1) Risk Identification

The key to maximizing AI's benefits while minimizing or avoiding its hazards is to identify them early on in the development and deployment process [23]. stress that credibility underpins the three pillars of society, economics, and long-term prosperity. People, businesses, and society at large never able to tap into AI's full potential unless they have faith in its creation, deployment, and utilization.

#### 2) Configuration Drift Management

The term configuration drift is used to describe systems that slowly change from their expected or intended configuration state. This may occur as a result of manual changes not authorized by the vendor, emergency patching, local script changes or mismatches in system updates, or changes in policies that aren't documented. When environments grow and evolve rapidly, these uncontrolled changes start to add up [24]. They can cause inconsistencies between identical systems, unpredictability in system behaviour, and make it hard to troubleshoot problems or apply patches reliably and securely. Drift undermines trust in automation pipelines and exposes organizations to operational and compliance risks, particularly in industries governed by strict regulations.

#### 3) Security Risks Management for Cyber-Physical System

As previously mentioned, a Risk Breakdown Structure (RBS) method was suggested for controlling the risks associated with CPS [25]. The risk matrix approach was used to develop and categorize countermeasures. An ISMS was implemented with risk values, and a quantitative analysis was carried out for a thorough risk assessment. The suggested remedies might somewhat lower risk, according to the quantitative assessment. Future research on the cost-effectiveness of the suggested countermeasures is crucial.

#### 4) Vulnerability Management

Vulnerability is a social science-derived term with many practical applications in fields such as economics, politics, project management, information systems, organizational management, and more. Addressing the

project's (or system's) vulnerabilities is the primary goal of vulnerability management in order to forestall the occurrence of potentially disastrous occurrences in the future. What makes a project vulnerable to disruptive occurrences is its vulnerability [26]. Thus, disruption events are not necessary for vulnerabilities to exist. One example of a project vulnerability is the lack of appropriately skilled personnel to do a particular task.

#### 5) Risk Assessment

Risk managers must identify and prioritize a potentially large list of dangers in order to determine which ones need to be addressed and resources allocated. This rests on the foundation of risk assessment, which determines the likelihood and consequences of each risk [27]. Some of the assessment categories described by ISO include expert intuition, expert audit, weighted scores, traditional financial analysis (without probabilities), a calculus of preferences (like multicriteria decision making), and probabilistic methods (based on subjective input, historical data, and empirical data)[28]. These methods include basic stratification techniques such as red-yellow-green "scoring" methods.

### B. Threat Detection and Monitoring

Threat detection and monitoring in IaC continuously identifies suspicious activities, vulnerabilities, misconfigurations, and policy violations across infrastructure environments. Automated monitoring tools analyze infrastructure changes, deployment activities, and security events to detect potential threats [29]. Real-time alerts and continuous assessment enable organizations to respond quickly and minimize security risks.

- **Real-Time Risk Assessment:** The system is always looking for security holes, comparing threat intelligence with preexisting regulations. It can identify any suspicious activity, data anomalies, and potential vectors of attack before they impact a cloud service.

- **Threat Risk Distribution:** During Threat Risk Distribution, the system simulated 50,000 log entries. A risk level (INFO, LOW, MEDIUM, HIGH, CRITICAL) was assigned to each event based on pre-defined rules (e.g., failed logins, unauthorized access attempts).

- **Anomaly Detection Accuracy:** The standard model was the Isolation Forest model, and the accuracy was evaluated by the ability to detect anomalies that simple signature-based rules are not detect. Synthetic anomalies (abnormal packet size and deviation in packet latency) were injected.

- **Machine Learning in Threat Detection:** One of the most widely used techniques in ML is supervised learning, which uses labelled datasets to train models, and detect known threat types in new sets of data [30].

Supervised learning is commonly employed in cybersecurity applications, like intrusion detection, spam filtering, and malware detection [31].

- **Deep Learning in Threat:** One great advantage of the threat detection of cybersecurity is Deep Learning in Threat, which uses multi-layer neural networks to extract and learn features from complex data automatically.

## V. LITERATURE REVIEW

The literature review covers some of the current IaC security strategies including governance, secure provisioning, multi-cloud automation, intelligent analysis, and configuration security. A concise overview of their method, results, limitations, and suggestions for further study is provided in Table I.

D. Boinpally (2026) introduced the GenIaC-GF, which incorporates generative AI, DevSecOps concepts, and governance-conscious validation into the IaC pipeline. GenIaC-GF transforms high-level natural language infrastructure requirements into validated and explainable IaC templates and imposes organizational security policies and compliance rules. The model automatically checks the security compliance, cost-effectiveness, and configuration availability prior to deployment[32].

H. Arora, et al. (2026) presented a novel security enhanced modular framework for provisioning cloud infrastructure using IaC which integrates security directly into the modular components. Such integration creates a unified, plugin-based solution with enhanced security provisions. The proposed modular design simplifies resource management by ensuring consistency across environments and reducing deployment errors. Security is embedded at the module level through NSGs, secure credential handling, utilizing public and private IPs, push and pull mechanism with validation and access controls at various levels. The procedure is fully automated by utilizing a Jenkins-based CI/CD pipeline and its service principles. This pipeline manages the infrastructure deployment lifecycle. [33].

A. Lubis, et al. (2025) explores how to use IaC with Terraform and Aviatrix on Azure and AWS to automate the deployment of multi-cloud networks. First, the infrastructure has to be provisioned. Then, the configuration needs to be enhanced. Finally, secure automation using CI/CD pipelines should be included in the implementation paradigm. Collaborative responsibilities of the NetOps, DevOps, and SecOps teams execute each phase utilizing a standardized IaC process. Functional validation proves the successful integration of many cloud environments and the efficient implementation of egress filtering with FQDN-

based constraints [34].

V. T. D. Jakkaraju (2025) Graph Neural Networks (GNNs) are revolutionizing IaC security in inventive ways, particularly when it comes to automating cloud security, identifying errors, and protecting privacy. These GNNs outperform the majority of DL models when examining Design, TAC-San, and IPAG-HAGNN in depth. They employ graph principles and code semantics to find complex security flaws in a wide variety of code, including code written in smart contract designs or in more than one language. In addition, the article examines GNN applications in dynamic cloud settings, demonstrating that GNNs securely chain services, communicate across microservices, and manage different cloud compliance with success rates over 90% [35].

S. I. Abbas and A. Garg (2024) The ideas of IaC have greatly improved deployment agility, cutting the implementation time of infrastructure changes to minutes. With the use of automation, dynamic scaling was made possible, which greatly improved resource usage and reduced costs. IaC ensured consistent and reliable configurations by codifying them, reducing configuration drift and security risks. This transition greatly simplified the operating process and enabled IT teams to focus on strategic efforts. Strict adherence to regulatory norms was achieved through the use of uniform and auditable settings, which enhanced security [36].

R. Opdebeeck, et al. (2023) introduce GASEL, an innovative Ansible IaC language security smell detector. The application can identify seven security issues by doing graph queries on dependency graphs. By taking syntax, control flow, and data flow into account, method is able to greatly enhance recall and accuracy, as demonstrated by assessment on an oracle of 243 real-world security smells and comparison with two top-tier security scent detectors. sometimes wonder if it's really worth it to put in the extra work to create and manage this strategy. Specifically, look at over 15,000 Ansible scripts to see how often security smells include indirection through data flow and control [37].

P. R. Reddy Konala, et al. (2023) gives the results of a survey that looked at how tools and methodologies are being utilized for IaC static configuration analysis. The research shows that methods like code and security smell detection are becoming more important for guaranteeing high-quality IaC scripts. research shows that regular expressions are popular, but they might not be the best long-term or entirely automated way to detect odors. On top of that, research shows that configuration management and image building aren't the primary areas for tool and technique development, but rather infrastructure provisioning [38].

There are still a number of unanswered questions

about IaC, despite the fact that recent studies have shown that generative AI, modular security, multi-cloud automation, GNNs, and static analysis processes are successful. Most existing studies concentrate on individual areas such as IaC governance, secure provisioning, vulnerability detection, multi-cloud deployment, or security-smell analysis instead of developing an integrated IaC security framework. Moreover, many of the proposed solutions are assessed

in the context of a certain cloud platform, IaC language, or just one specific environment, limiting their applicability when deployed in a heterogeneous infrastructure. Real-time security validation, continuous compliance, configuration management, scalability and interoperability are still under-explored in the area of issues. The combination of generative AI and GNN based analysis, automated governance and continuous monitoring of security is limited in a single IaC lifecycle.

**TABLE I. SUMMARY OF KEY STUDIES ON INFRASTRUCTURE AS CODE SECURITY**

| Reference                        | Study Focus                                | Methodology Technology                                                  | Key Findings                                                                      | Challenges Limitations                          | Future Direction                                               |
|----------------------------------|--------------------------------------------|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------|----------------------------------------------------------------|
| D. Boinpally (2026)              | Generative AI-based Governance             | Generative AI, DevSecOps, GenIaC-GF, policy validation                  | Enables explainable IaC generation with security, compliance, and cost validation | Governance and validation complexity            | Improve automated governance and policy enforcement            |
| H. Arora et al. (2026)           | Security-Enhanced Modular IaC Provisioning | Modular IaC, NSGs, secure credentials, IP controls, Jenkins CI/CD       | Improves security, consistency, and deployment automation                         | Integration and lifecycle management complexity | Enhance scalability and automated security controls            |
| A. Lubis et al. (2025)           | Multi-Cloud Network Deployment             | Terraform, Aviatrix, AWS, Azure, CI/CD, NetOps-DevOps-SecOps            | Enables secure multi-cloud connectivity and egress control                        | Multi-cloud configuration complexity            | Extend automation across additional cloud platforms            |
| V. T. D. Jakkaraju (2025)        | GNN-Based IaC Security                     | Graph Neural Networks, Design, TAC-San, IPAG-HAGNN                      | Improves security analysis and compliance detection                               | Model complexity and broader applicability      | Develop scalable cross-platform GNN-based analysis             |
| S. I. Abbas & A. Garg (2024)     | IaC Automation and Operational Efficiency  | Infrastructure as Code, automated scaling, standardized configurations  | Improves agility, consistency, scalability, and compliance                        | Operational and implementation challenges       | Strengthen automated security and compliance                   |
| R. Opdebeeck et al. (2023)       | Security Smell Detection in Ansible IaC    | GASEL, graph queries, program dependence graphs                         | Improves detection precision and recall for security smells                       | Computational and implementation effort         | Develop efficient and scalable smell detection                 |
| P. R. Reddy Konala et al. (2023) | Static IaC Configuration Analysis          | Static analysis, regular expressions, security and code-smell detection | Identifies trends and limitations in existing IaC analysis tools                  | Limited automation and focus on provisioning    | Expand analysis to configuration management and image building |

## VI. CONCLUSION AND FUTURE WORK

IaC is repeatable and programmable approach to infrastructure management. The studies reviewed show that IaC can enhance deployment consistency, operational efficiency, scalability and infrastructure governance when used in conjunction with DevOps practices. However, such changes in security risk, compliance, validation and infrastructure will need to be addressed if they are to be effective. Key issues persist with exposed credentials, vulnerable third-party components, configuration drift, and inadequate access controls and misconfigurations. Automated validation and policy-driven compliance can minimize deployment mistakes and uncover compliance issues before they impact production environments. Risk management also supports IaC by systematically identifying, assessing, prioritizing and mitigating risks to infrastructure. Threat detection and continuous monitoring provide additional capabilities for identifying suspicious activities and vulnerabilities during infrastructure operation. Overall, secure IaC requires coordinated use of automation, validation, compliance management, monitoring, and risk-control practices. Such integration can improve infrastructure reliability, maintainability, security, and organizational control across changing cloud and DevOps

environments.

Future research should develop intelligent IaC security frameworks that combine automated validation, compliance enforcement, ML-based threat detection, continuous risk assessment, and automated remediation. Greater attention is also needed toward multi-cloud environments, evolving security policies, configuration drift, supply-chain risks, and scalable real-time monitoring for secure infrastructure management.

## References

1. S. Girke, R. Klöforn, and M. Ohlberger, "Efficient Parallel Simulation of Atherosclerotic Plaque Formation Using Higher Order Discontinuous Galerkin Schemes," *Scenario*, vol. 9, no. 1, pp. 688–696, Mar. 2014.
2. R. Azmeera, "The Effects of Increased Software Errors on Software Product Functionality," University of the Cumberland, 2025.
3. D. Antiya and O. Corporation, "Compliance as Code: Automating Compliance in Cloud Systems," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 8, no. February, 2025.
4. K. Jangiti, "Design and Validation of a Machine

- Identity Governance Framework for AI Agents in Multi-Cloud Environments," in SoutheastCon 2026, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/SoutheastCon63549.2026.11476363.
5. R. K. Kanneganti, "Security and Compliance Issues in Cloud-Based Deployments of Content and Workflow Management Systems," Eastasouth J. Inf. Syst. Comput. Sci., vol. 2, no. 01, pp. 131–138, Aug. 2024, doi: 10.58812/esiscs.v2i01.1122.
6. S. Rodrigues, "Intelligent Automation for Infrastructure as Code ( IaC ) in DevOps," 2024.
7. S. Remella, J. Ligam, U. K. R. Gangula, V. Sannamuri, R. Ganta, and Q. T. Sadat, "End-to-End Cloud Procurement Lifecycle Automation," in 2025 1st International Conference on Advancement in Futuristic Technologies (ICAFT), Belagavi, India: IEEE, 2025, pp. 1–8. doi: 10.1109/ICAFT66710.2025.11452870.
8. E. Damayanti, "Risk Management: In an Overview of Literature Review," Formosa J. Sci. Technol., vol. 2, pp. 1115–1122, 2023, doi: 10.55927/fjst.v2i4.3837.
9. R. Azmeera and J. Hyatt, "Software Errors, Product Functionality, and Organizational Cascades: Evidence from Developer-Lived Experience," Mar. 2026. doi: 10.2139/ssrn.6924439.
10. S. Sreevathsa, "Bridging Technical Telemetry and Business Decision-Making through Data Product Management," Univers. Libr. Innov. Res. Stud., vol. 3, no. 3, pp. 32–36, Aug. 2026, doi: 10.70315/uloap.ulirs.2026.0303006.
11. A. V. S. R. Dantuluri, "Patient-Centric Markov-Chain Framework for Predicting Medication Adherence Using De-Identified Data," 2026. doi: 10.64898/2026.02.08.26345856.
12. K. K. Mohammed, "A Survey on Digital Health Care Data Analysis Techniques for Developing Machine Learning Models," Int. J. Sci. Eng. Technol., vol. 13, no. 5, October, pp. 1–6, 2025, doi: 10.5281/zenodo.17339813.
13. S. C. L. Koh, S. M. Saad, A. Ahmed, B. Kayis, and S. Amornsawadwatana, "A review of techniques for risk management in projects," Benchmarking An Int. J., vol. 14, no. 1, pp. 22–36, 2007, doi: 10.1108/14635770710730919.
14. H. El Bhilat and M. Baaddi, "Artificial intelligence for emotion-aware customer journeys: A systematic literature review," Multidiscip. Rev., vol. 9, no. 9, 2026, doi: 10.31893/multirev.2026417.
15. A. War, A. Habib, A. Diallo, J. Klein, and T. F. Bissyandé, "Security Vulnerabilities in Infrastructure as Code: What, How Many, and Who?," Nov. 2023. doi: 10.21203/rs.3.rs-3600645/v1.
16. N. Thallapally, "Implementing continuous integration and continuous deployment (CI/CD) pipelines," Int. J. Sci. Res. Arch., vol. 3, no. 2, pp. 248–253, Oct. 2021, doi: 10.30574/ijrsra.2021.3.2.0073.
17. L. Chen, "Microservices: Architecting for Continuous Delivery and DevOps," in 2018 IEEE International Conference on Software Architecture (ICSA), IEEE, Apr. 2018, pp. 39–397. doi: 10.1109/ICSA.2018.00013.
18. R. T. Natarajan, "Journal of Artificial Intelligence , Machine Learning and Data Science Pharmaceutical Manufacturing," pp. 4–7, 2023.
19. H. Castro, "Infrastructure as Code (IaC) Security in AWS with DevSecOps," ResearchGate., 2024.
20. G. Falazi et al., "Compliance Management of IaC-Based Cloud Deployments During Runtime," in Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, New York, NY, USA: ACM, Dec. 2023, pp. 1–11. doi: 10.1145/3603166.3632135.
21. B. Teslim, "Error-Free Cloud Deployments: Automated Validation and IaC Best Practices," 2024.
22. T. Aven, "Risk assessment and risk management: Review of recent advances on their foundation," Eur. J. Oper. Res., vol. 253, no. 1, pp. 1–13, Aug. 2016, doi: 10.1016/j.ejor.2015.12.023.
23. S. Thiebes, S. Lins, and A. Sunyaev, "Trustworthy artificial intelligence," Electron. Mark., vol. 31, no. 2, pp. 447–464, 2021, doi: 10.1007/s12525-020-00441-4.
24. J. Alam, R. Haque, T. Akter, and F. Nishi, "Multi-Os Configuration Drift Detection Using Ansible," no. December, 2022.
25. S. Yoneda, S. Tanimoto, T. Konosu, H. Sato, and A. Kanai, "Risk Assessment in Cyber-Physical System in Office Environment," in 2015 18th International Conference on Network-Based Information Systems, IEEE, Sep. 2015, pp. 412–417. doi: 10.1109/NBiS.2015.63.
26. C. Bilir and E. Yafez, "Project success/failure rates in Turkey," Int. J. Inf. Syst. Proj. Manag., vol. 9, no. 4, pp. 24–40, 2021, doi: 10.12821/ijispm090402.
27. B. Stosic, M. Mihic, R. Milutinovic, and S. Isljamovic, "Risk identification in product innovation projects: new perspectives and lessons learned," Technol. Anal. & Strateg. Manag., vol. 29, no. 2, pp. 133–148, 2017, doi: 10.1080/09537325.2016.1210121.

28. S. Irfan, "Enhancing Email Security Through Accurate Phishing Detection Using Deep Transformer Models," in 2026 World Conference on Computational Science and Technology (WcCST), 2026, pp. 239–244. doi: 10.1109/WcCST67302.2026.11495864.
29. A. A. I. Yanguema and C. Yin, "Real-Time Cyber Monitoring and Threat Detection System with Hybrid AI Analysis," OALib, vol. 13, no. 01, pp. 1–16, 2026, doi: 10.4236/oalib.1114742.
30. Z. Wang, "Artificial Intelligence in Cybersecurity Threat Detection," Int. J. Comput. Sci. Inf. Technol., vol. 4, no. 1, pp. 203–209, 2024, doi: 10.62051/ijcsit.v4n1.24.
31. S. Chatterjee, "Advanced Malware Detection in Operational Technology: Signature-Based Vs. Behaviour-Based Approaches," ESP J. Eng. Technol. Adv., vol. 1, 2021, doi: 10.56472/25832646/JETA-V1I2P128.
32. D. Boinpally, "Generative AI-Powered Infrastructure-as-Code: Automating Cloud Setup with Explainability and Governance," in 2026 IEEE International Conference on Emerging Computing and Intelligent Technologies (ICoECIT), IEEE, Jan. 2026, pp. 1–7. doi: 10.1109/ICoECIT68303.2026.11497432.
33. H. Arora, A. Khudlain, N. Mahalwal, and N. Kandhoul, "Security Integrated Modularized Framework for Automated Infrastructure Provisioning in Cloud Environments," in 2026 IEEE 18th International Conference on Computational Intelligence and Communication Networks (CICN), IEEE, Jun. 2026, pp. 802–807. doi: 10.1109/CICN70047.2026.11594141.
34. A. Lubis, E. M. Zamzami, A. Candra, and H. Mawengkang, "Application of Infrastructure as Code (IaC) in Multi-Cloud Network Infrastructure Management Using Terraform and Aviatrix on AWS and Azure," in 2025 13th International Conference on Cyber and IT Service Management (CITSM), IEEE, Sep. 2025, pp. 1–4. doi: 10.1109/CITSM67730.2025.11291297.
35. V. T. D. Jakkaraju, "AI-Powered Infrastructure as Code (IaC) Security using Graph Neural Networks," in 2025 3rd International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), IEEE, Aug. 2025, pp. 87–93. doi: 10.1109/ICSCDS65426.2025.11167786.
36. S. I. Abbas and A. Garg, "Integrating Emerging Technologies with Infrastructure as Code in Distributed Environments," in 2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), IEEE, Jun. 2024, pp. 1138–1144. doi: 10.1109/ICAAIC60222.2024.10575600.
37. R. Opdebeeck, A. Zerouali, and C. De Roover, "Control and Data Flow in Security Smell Detection for Infrastructure as Code: Is It Worth the Effort?," in 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), IEEE, May 2023, pp. 534–545. doi: 10.1109/MSR59073.2023.00079.
38. P. R. Reddy Konala, V. Kumar, and D. Bainbridge, "SoK: Static Configuration Analysis in Infrastructure as Code Scripts," in 2023 IEEE International Conference on Cyber Security and Resilience (CSR), IEEE, Jul. 2023, pp. 281–288. doi: 10.1109/CSR57506.2023.10224925.