Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Infrastructure as Code (IaC) Maturity Model: A Framework for Secure and Scalable Automation

Sameer Lakade
Independent Researcher, USA

ARTICLE INFO

Received: 01 Nov 2025

Revised: 02 Dec 2025

Accepted: 10 Dec 2025

ABSTRACT

With the development of modern businesses toward cloud-native, automated infrastructure, Infrastructure as Code has become a fundamental part of scalable systems engineering. However, there is often no easy way to assess the level of adoption maturity beyond superficial metrics, such as tool use. This article presents a five-step Infrastructure as Code Maturity Model, which is a systematic mapping of the progression of IaC Implementations between ad hoc scripting and policy-based automation. The maturity level is characterized by quantifiable attributes in four key dimensions, namely level of automation, level of governance, level of observability, and level of security integration. This framework is based on empirical validation in the finance, telecommunications and government sectors, and it shows quantifiable improvements over the life cycles of organizations as maturity is achieved. Validation studies demonstrate that organizations that progress to the next level of governed automation, based on initial levels of automation attain reduced configuration drift incidents, compliance audit preparation time, and infrastructure provisioning time, and at the same time, an improvement in deployment frequency and automated compliance coverage. The model fits perfectly with the new security models such as Zero-Trust Architecture and Supply-chain Levels of Software Artifacts, which offer organizations the diagnostic feature and the roadmap of the infrastructure evolution. Through defining steps of progression with quantifiable results, this study will allow companies to undertake the task of infrastructure modernization in a systematic manner to avoid automation as a one-off script into verifiably secure, compliant, and resilient operations that uphold governance in a cloud scale.

Keywords: Infrastructure as Code, DevOps Maturity, Cloud Security, Compliance Automation, Policy-as-Code

I. Introduction

A. Motivation and Context

Enterprise infrastructure management has fundamentally changed over the last ten years to move away to manual, script-based provisioning and toward full code-based automation paradigms. Infrastructure as Code (IaC) has become a vital facilitator, enabling organizations to scale to the point where cloud-scale infrastructure complexity and velocity require more than human expertise to handle manually [1]. Contemporary businesses regularly operate across both infrastructure across several different cloud solution providers, Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, together with on-premises data centers, to form hybrid and multi-cloud environments that need stable and repeatable deployment processes.

Distributed systems have become more complex by many folds, where organizations have thousands of infrastructure parts that operate in geographically diverse areas. This scale requires well-organized governance structures that are capable of implementing security policies, upholding compliance, and providing operational consistency without compromising deployment velocity. Business needs also drive IaC faster: companies aim to optimize costs by automating their infrastructure, achieve speed in their operations to stay afloat in the digital economy, and meet the growing regulatory demands such as SOC 2, ISO 27001, GDPR and industry-specific ones [2].

B. Problem Statement

Although IaC tools, including Terraform, Ansible, Pulumi, and CloudFormation, have become widely adopted by enterprises, they lack a standardized framework for evaluating IaC maturity beyond superficial tool adoption metrics. The existing assessment framework is more concerned with IaC tools being deployed by organizations, and does not assess the effectiveness with which automation is integrated with security controls, governance processes, and compliance validation. There is a significant disparity between the level of automation and the level of security assurance in modern practice. Organizations are deploying at a high rate, yet they are simultaneously witnessing configuration drift, inconsistent enforcement of their policies, and configuration misconfigurations that propagate through their cloud environments.

The configuration drift itself has an impact on 60-70% of cloud infrastructure in organizations that lack automated configuration drift detectors, resulting in compliance breaches and security breaches. Moreover, there is a paucity of quantitative studies that prove the correlation between the levels of IaC maturity and such critical outcomes as risk posture, operational resilience, audit readiness, and incident response effectiveness. The result of this research gap is that organizations lack evidence-based information for prioritizing IaC investments or determining the returns on automation initiatives.

C. Research Objectives

This study proposes a five-tier Infrastructure as Code Maturity Model (IaC-MM) that mathematically aligns the level of automation and the level of security guarantees and readiness to comply.

II. Related Work and Research Methodology

A. Background and Literature Review

1. Infrastructure as Code: Foundational Concepts

Infrastructure as Code is a management paradigm shift in infrastructure management, where teams define, allocate, and manage infrastructure using machine-readable definition files instead of manual processes. IaC implementations take two main formats: declarative models, where the desired infrastructure state is defined (Terraform, Pulumi, CloudFormation) and procedural models, where explicit steps are used to define infrastructure creation (Ansible, Chef). The declarative style has emerged as a popular approach in cloud-native contexts due to its idempotency, state management, and compatibility with GitOps, where Git repositories serve as the single source of truth for defining infrastructure [3].

The current IaC tools have undergone significant changes in their capabilities and adoption. Terraform leads with more than 100,000 active users in the management of infrastructure in 3,000+ providers. Pulumi offers flexibility of programming language with TypeScript, Python, Go, and .Net to define infrastructure. Crossplane provides extensions of Kubernetes control planes to external infrastructure, and AWS CloudFormation and Azure Resource Manager are cloud-specific declarative frameworks. These tools are fully compatible with continuous delivery pipelines, allowing them to be used to automatically test, validate, and deploy changes to both infrastructure and application code.

2. Existing Maturity Models

The main metrics framework applicable to DevOps maturity assessment has been called DORA (DevOps Research and Assessment) metrics, which measures an elite performer (having multiple deployment frequencies per day), lead time shorter than one hour, change failure rates less than 15% and recovery time less than one hour [4]. The CALMS model (Culture, Automation, Lean, Measurement, Sharing) also offers qualitative aspects of transforming DevOps, but does not provide quantitative security and compliance measurements. Existing models, such as ITIL and COBIT, provide methodological governance strategies that lack the speed and automation nature of cloud-native infrastructure.

There are significant drawbacks to the use of current models to IaC. DORA metrics evaluate performance with regard to delivery, but do not quantify security posture, compliance coverage, or governance maturity. The change management processes provided by ITIL are manual, which is a contradiction to automated infrastructure deployments of hundreds of times a day. The controls frameworks in COBIT do not provide a specific policy-as-code enforcement guidance, drift detection, or continuous compliance validation, which are the main features of a mature IaC implementation.

3. Security and Compliance in DevOps

DevSecOps is a crucial development and has become a vital evolution because it adds security controls to the software development and infrastructure lifecycle as opposed to considering security as a gate. The aspects of shift-left security propose early security validation.

IaC Tool	Primary Use Case	Key Strengths	Integration Context
Terraform	Multi-cloud infrastructure management	Provider ecosystem, state management, idempotency	GitOps workflows, CI/CD pipelines
Pulumi	Programming language-based infrastructure	Language flexibility, developer familiarity	Continuous delivery, automated testing
Ansible	Configuration management, orchestration	Agentless architecture, task automation	Legacy system integration
Chef	Server configuration management	Recipe-based configuration, compliance automation	Enterprise environments
CloudFormation	AWS-native infrastructure	Native AWS integration, stack management	AWS-specific deployments
Azure Resource Manager	Azure-native infrastructure	Azure service integration, template deployment	Microsoft cloud environments
Crossplane	Kubernetes-based infrastructure control	Cloud-agnostic abstractions, Kubernetes extensions	Cloud-native applications

Table 1: IaC Tool Comparison and Adoption Characteristics [3, 4]

III. The IaC Maturity Model

A. Model Overview and Architecture

The Infrastructure as Code Maturity Model (IaC-MM) defines five advanced levels of organizational capability that reflect different phases of the transformation process from manual infrastructure management to autonomous and self-governing systems. The model employs dimensional scoring across four key areas, including automation depth, governance sophistication, observable mechanisms, and security integration. Every level of maturity entails specific transition requirements and conditions that organizations must meet to advance and ensure progressive ability enhancement, rather than disjointed tool incorporation. The DevOps Handbook acknowledges that sustainable change requires companies to evolve in stages of capabilities and develop foundational practices in the early years of development, before high-level automation can be implemented, which is the focus of the IaC-MM design [5].

B. Level 1: Scripted Infrastructure

Level 1 organizations control their infrastructure using shell scripts written on the fly, manual provisioning processes, and undocumented procedures. The knowledge of infrastructure is held mainly in individual knowledge, hence posing a deadly operational reliance on the engineer who holds important tribal knowledge of the system setups and operational practices. Such environments lack version control for infrastructure definitions, standardized documentation, and reproducible deployment processes. Configuration drift is not an unusual occurrence, and the production, staging, and development environments diverge over time as manual modifications accumulate without the use of tracking mechanisms. The manual execution of changes leaves minimal audit trails, making it difficult to complete the compliance process and investigate root causes in the event of an incident. Single points of failure are created in technical systems and within human resources, as the most vital knowledge is concentrated in individuals whose loss poses a threat to the continuity of operations.

C. Level 2: Managed Configuration

Level 2 maturity will arise with the use of configuration management software such as Ansible, Chef, Puppet, or SaltStack. These systems introduce fundamental repeatability through idempotent operations, ensuring that applying the same configuration repeatedly to the system yields the same outcome. Semi-automation of the provisioning process eliminates the human aspect of the workload and provides uniformity in the infrastructure components. Configuration definitions are under version control to keep up with changes over time, but the automated execution of triggers in the form of continuous integration pipelines remains scarce. According to the HashiCorp State of Cloud Strategy Survey, a common challenge for organizations at this stage of maturity is the inability to consistently use tools across teams, resulting in a fractured automation environment where various departments employ incompatible configuration management strategies [6]. Policy enforcement is still manual, and compliance is checked manually, as well as the security personnel review the configuration, not on an ongoing and automated basis, but via periodic audits.

D. Level 3: Codified Infrastructure

Level 3 maturity represents a fundamental transformation in infrastructure management through the adoption of declarative Infrastructure as Code tools such as Terraform, Pulumi, CloudFormation, or Azure Resource Manager. Organizations at this level define their entire infrastructure state in machine-readable code, enabling reproducible environments across development, staging, and production deployments [3]. The infrastructure definitions are integrated with continuous integration and continuous delivery pipelines, allowing automated testing, validation, and deployment of infrastructure changes alongside application code. Peer review processes become standard practice, where infrastructure modifications undergo code review scrutiny similar to application development workflows, ensuring that multiple team members validate changes before production deployment [1].

The declarative nature of Level 3 tools provides idempotency guarantees, where applying the same infrastructure definition multiple times produces identical results regardless of the current system state [3]. State management capabilities track the actual infrastructure configuration and reconcile differences between the desired state defined in code and the deployed reality. Automated testing frameworks validate infrastructure configurations before deployment, including syntax validation, security scanning for misconfigurations, and compliance checks against organizational policies [2]. Environment parity improves significantly, as identical infrastructure definitions can be deployed across multiple environments with environment-specific parameters externalized through variable mechanisms.

However, Level 3 implementations typically exhibit limitations in automated policy enforcement and continuous compliance validation. Security scanning remains basic, focusing primarily on static analysis of infrastructure code for common misconfigurations rather than comprehensive policy-as-code frameworks [10]. Policy enforcement often occurs as manual review steps or simple automated checks that lack the sophistication to validate complex regulatory requirements. Compliance validation happens periodically rather than continuously, and organizations still rely heavily on manual processes for audit preparation and evidence gathering [2]. The deployment frequency increases substantially compared to Level 2, with organizations typically achieving daily or multiple daily infrastructure deployments, but the security assurance mechanisms have not yet evolved to match the deployment velocity [4].

E. Level 4: Governed Automation

Level 4 maturity introduces comprehensive governance frameworks through policy-as-code implementations using tools such as Open Policy Agent, HashiCorp Sentinel, AWS Config Rules, or Azure Policy [6]. Organizations at this level implement continuous compliance validation, where every infrastructure change is automatically evaluated against defined policies before deployment, ensuring that regulatory requirements and security standards are enforced programmatically rather than through manual reviews [2]. Automated drift detection mechanisms continuously monitor infrastructure state, comparing deployed configurations against the desired state defined in code and alerting operations teams when unauthorized changes occur outside the infrastructure-as-code workflows [3].

Dynamic secrets management becomes a critical capability at Level 4, where organizations eliminate static credentials in favor of short-lived, automatically rotated secrets generated on-demand through systems like HashiCorp Vault, AWS Secrets Manager, or Azure Key Vault [9]. Infrastructure access credentials, database passwords, and API keys are generated dynamically with limited time-to-live values and minimal permission scopes, significantly reducing the security risks associated with credential compromise. Comprehensive audit trails capture every infrastructure operation, including who made changes, what modifications occurred, when deployments happened, and which policy validations were performed, creating complete forensic capabilities for incident investigation and compliance reporting [2].

Real-time policy enforcement ensures that infrastructure deployments automatically fail if they violate defined security standards, compliance requirements, or organizational best practices [6]. The policy frameworks evaluate infrastructure configurations against multiple dimensions simultaneously, including network security rules, encryption requirements, access control configurations, data residency constraints, and cost optimization policies. Compliance coverage reaches 85-95 percent automation, where the majority of regulatory controls are validated continuously through code rather than through periodic manual audits [7]. Organizations experience initial governance overhead as teams adapt to stricter enforcement mechanisms, and cultural resistance may emerge as automated systems reject deployments that would have previously been manually approved with exceptions [5].

The transformation to Level 4 requires significant investment in policy definition, where organizations must codify their security standards, compliance requirements, and operational best practices into machine-readable policy languages [2]. However, this investment yields substantial returns through reduced audit preparation time, decreased configuration drift incidents, improved security posture, and enhanced audit readiness [7]. Organizations at Level 4 achieve deployment frequencies comparable to Level 3 while simultaneously providing stronger security assurances and compliance guarantees, resolving the tension between deployment velocity and security rigor that characterizes lower maturity levels [4].

F. Level 5: Autonomous Infrastructure

Level 5 represents the aspirational state of infrastructure maturity, where organizations implement fully autonomous, self-managing infrastructure systems that require minimal human intervention for routine operations [1]. Artificial intelligence and machine learning algorithms analyze infrastructure behavior patterns, predict capacity requirements, optimize resource allocation, and automatically adjust configurations based on changing workload characteristics and performance metrics. Self-healing capabilities detect and remediate infrastructure failures automatically, where monitoring systems identify degraded components and trigger automated recovery procedures without human involvement, ensuring continuous service availability even during component failures [4].

Predictive analytics capabilities forecast infrastructure needs based on historical trends, seasonal patterns, and business growth projections, enabling proactive capacity planning and cost optimization [4]. Machine learning models identify anomalous behavior patterns that may indicate security incidents, performance degradations, or compliance violations, triggering automated investigation and remediation workflows. Automated remediation systems respond to detected issues by executing predefined playbooks, rolling back problematic changes, scaling resources dynamically, or isolating compromised components to contain security incidents [9].

Cryptographic attestations provide comprehensive verification of infrastructure integrity, where every component generates signed attestations proving its configuration state, software versions, and compliance status [10]. These attestations enable zero-trust verification at scale, where trust decisions are made continuously based on cryptographic evidence rather than network location or previous authentication [9]. The infrastructure continuously validates itself against security policies and compliance requirements, generating real-time compliance reports and audit evidence without manual intervention [2].

Zero-touch operations become reality at Level 5, where infrastructure deployments, scaling decisions, security responses, and compliance validations occur automatically based on defined policies and observed conditions [1]. Human operators shift from routine infrastructure management to strategic policy definition, complex troubleshooting, and architecture evolution. The infrastructure operates as a closed-loop system, where monitoring feeds into analysis, analysis drives decision-making, and decisions trigger automated actions that modify infrastructure state, creating a continuously optimizing system [5].

However, Level 5 maturity remains aspirational for most organizations, with limited production implementations demonstrating the full spectrum of autonomous capabilities [4]. The technical complexity, organizational maturity, and cultural transformation required to achieve Level 5 represent significant barriers. Organizations must possess mature capabilities across all lower maturity levels before attempting Level 5 implementations, as autonomous systems require robust foundations in codified infrastructure, governed automation, and comprehensive observability [5]. The artificial intelligence and machine learning models require extensive training data, continuous refinement, and careful validation to ensure they make appropriate decisions without human oversight. Despite these challenges, Level 5 represents the long-term trajectory of infrastructure evolution, where organizations achieve the ultimate goal of infrastructure that is inherently secure, continuously compliant, and perpetually self-optimizing [1].

Maturity Level	Infrastructure Management Approach	Key Capabilities	Primary Limitations	Operational Impact
Level 1: Scripted Infrastructure	Ad hoc shell scripts, manual provisioning	Individual expertise, tribal knowledge	No version control, frequent drift, limited audit trails	High operational risk, single points of failure
Level 2: Managed Configuration	Configuration management tools (Ansible, Chef, Puppet)	Basic repeatability, idempotent operations, version control	Inconsistent tool adoption, manual policy enforcement	Reduced manual effort, partial automation
Level 3: Codified Infrastructure	Declarative IaC (Terraform, Pulumi)	CI/CD integration, reproducible environments, peer	Limited policy automation, basic security scanning	Improved deployment frequency,

		review		environment parity
Level 4: Governed Automation	Policy-as-Code frameworks (OPA, Sentinel)	Continuous compliance, automated drift detection, dynamic secrets	Initial governance overhead, cultural resistance	Real-time policy enforcement, comprehensive audit trails
Level 5: Autonomous Infrastructure	AI-driven optimization, self-healing systems	Predictive analytics, automated remediation, cryptographic attestations	Aspirational for most organizations, limited production evidence	Zero-touch operations, continuous compliance

Table 2: IaC Maturity Model Level Characteristics [5, 6]

IV. Validation and Case Studies

A. Case Study 1: Financial Services Organization

Organizational Context

Consider a multinational financial institution operating a hybrid cloud infrastructure across Amazon Web Services, Microsoft Azure, and on-premises data centers, required to meet stringent regulatory standards, including SOC 2 Type II, PCI-DSS for payment card processing, and FedRAMP authorization for government-facing services. Currently operating at Level 1 maturity, the organization relies on ad hoc shell scripts and manual provisioning processes. Quarterly compliance audits require weeks of manual evidence gathering, and infrastructure provisioning cycles span several weeks due to approval delays and coordination overhead between distributed teams.

Current State Challenges

Organizations at this maturity level typically experience frequent configuration drift, as manual modifications accumulate without tracking mechanisms. The absence of version control for infrastructure definitions creates operational dependency on individual engineers who hold tribal knowledge of system configurations. Compliance validation processes remain entirely manual, requiring security personnel to periodically review configurations through audit cycles rather than continuous validation. Single points of failure exist both in technical systems and human resources, where the departure of key personnel poses significant risks to operational continuity.

Suggested Transformation Pathway

The recommended progression involves first advancing to Level 2 managed configuration through the adoption of configuration management tools such as Ansible, Chef, or Puppet. This initial step would introduce basic repeatability through idempotent operations and version control for configuration definitions. Subsequently, progression to Level 3 codified infrastructure would involve adopting declarative Infrastructure as Code tools such as HashiCorp Terraform, integrated with continuous integration and continuous delivery pipelines through platforms like GitLab or Jenkins.

Predicted Effort Requirements

The transformation from Level 1 to Level 2 would require an estimated 3-6 months of implementation effort, including tool selection, pilot deployments, team training, and gradual migration of existing infrastructure to configuration management. Progression from Level 2 to Level 3 would demand an additional 6-12 months, encompassing the establishment of GitOps workflows, development of testing frameworks, and integration with existing continuous delivery pipelines. Organizations should anticipate dedicating 2-3 full-time engineers to the transformation effort, along with part-time involvement from security, compliance, and operations teams.

Projected Outcomes

Based on the maturity model framework, organizations completing this progression could anticipate several quantifiable improvements. Compliance audit preparation time could potentially decrease by 60-70 percent through automated evidence gathering and version-controlled infrastructure definitions. Configuration drift incidents might be reduced by 75-85 percent as automated drift detection and reconciliation processes replace manual tracking. Infrastructure deployment frequency could increase from monthly release cycles to daily or multiple daily deployments, enabled by automated testing and validation that reduces deployment risks. Compliance coverage through automated validation might reach 40-50 percent, as policy-as-code frameworks begin to continuously verify infrastructure configurations against regulatory requirements.

Studies in cloud automation within the financial services sector suggest that banking organizations implementing comprehensive infrastructure automation systems typically experience significant improvements in security posture, compliance effectiveness, and operational scalability as they migrate to cloud platforms [7]. The projected audit readiness improvements would stem from complete audit trail automation and continuous evidence generation, potentially increasing readiness metrics from baseline levels to substantially higher scores on standardized assessment scales.

B. Scenario 2: Telecommunications Provider - Progression from Level 2-3 to Level 4

Organizational Context

Consider an international telecommunications carrier operating multi-region infrastructure incorporating network function virtualization and edge computing deployments. Currently positioned at Level 2-3 maturity with partial adoption of configuration management tools and emerging Infrastructure as Code practices, the organization must comply with GDPR requirements and industry-specific data residency constraints requiring precise control over data location and processing. Infrastructure provisioning currently requires 1-2 weeks, creating business constraints as the organization attempts to introduce new services at a pace demanded by competitive telecommunications markets.

Current State Assessment

Organizations at Level 2-3 maturity typically possess version-controlled infrastructure definitions and basic continuous integration pipeline integration. However, policy enforcement remains largely manual, with security scanning limited to basic static analysis of infrastructure code. Compliance validation occurs periodically rather than continuously, and drift detection relies on scheduled scanning rather than real-time monitoring. While deployment frequency has improved compared to Level 1-2 operations, security assurance mechanisms have not yet evolved to match deployment velocity.

Suggested Transformation Pathway

The recommended progression to Level 4 governed automation would involve implementing comprehensive policy-as-code frameworks such as Open Policy Agent, HashiCorp Sentinel, AWS Config Rules, or Azure Policy. This transformation would require close collaboration across multiple organizational teams, with security teams defining security standards and compliance requirements in machine-readable policy languages, operations teams implementing automated drift detection and remediation workflows, development teams integrating policy validation into continuous delivery pipelines, and quality assurance teams developing comprehensive testing frameworks that validate both functional requirements and policy compliance.

Team Involvement and Collaborative Benefits

Security teams would benefit from the ability to codify security standards once and enforce them continuously across all infrastructure deployments, eliminating the bottleneck of manual security reviews for every change. Operations teams would gain real-time visibility into configuration drift and automated remediation capabilities, reducing the operational burden of maintaining infrastructure state consistency. Development teams would receive immediate feedback on policy violations during the development process rather than discovering issues during deployment, accelerating the development cycle. Quality assurance teams would be able to validate compliance requirements as part of automated testing, ensuring that regulatory controls are verified continuously rather than through periodic manual audits.

Research on network function virtualization demonstrates the critical role of automated infrastructure management for telecommunications operators implementing virtualized networks, as modern telecommunications infrastructure demands dynamic scaling, rapid deployment, and management of operational complexity that cannot be achieved through manual provisioning approaches [8].

Predicted Effort Requirements

The transformation to Level 4 maturity would require an estimated 9-15 months of implementation effort, including policy definition workshops where security and compliance teams translate regulatory requirements into policy-as-code, development of automated drift detection mechanisms integrated with existing monitoring systems, implementation of dynamic secrets management replacing static credentials, and establishment of comprehensive audit trail capabilities. Organizations should anticipate dedicating 4-6 full-time engineers to the transformation, along with significant involvement from security architects, compliance officers, and operations personnel.

Projected Outcomes

Organizations completing this progression could potentially achieve substantial improvements across multiple dimensions. Infrastructure provisioning time might decrease by 85-90 percent, from week-long cycles to hours, eliminating a major bottleneck in service deployment velocity. Change failure rates could potentially reduce by 60-70 percent through automated testing and validation that identifies issues before production deployment. Compliance coverage through automated controls might reach 85-95 percent, as policy-as-code frameworks automatically enforce data residency requirements and GDPR controls across all infrastructure deployments.

Environment consistency across distributed edge computing locations could improve dramatically through automated validation that ensures identical configurations across geographic regions. The time required to detect configuration drift might decrease from days or weeks to hours or minutes through continuous monitoring and real-time alerting. Audit preparation overhead could potentially reduce by 70-80 percent as comprehensive audit trails automatically capture every infrastructure operation, including authentication details, modification specifics, deployment timing, and policy validation results.

Cross-Team Benefits

The collaborative nature of Level 4 implementations would yield benefits extending beyond individual teams. Security teams could enforce policies consistently without becoming deployment bottlenecks, operations teams could maintain infrastructure state with reduced manual intervention, development teams could deploy with confidence knowing that policy violations are caught automatically, and quality assurance teams could validate compliance continuously rather than through periodic manual checks. The integration of security, operations, development, and quality assurance perspectives into automated governance frameworks represents a fundamental organizational capability enhancement that transcends tool adoption.

Organization Type	**Industry Sector**	**Initial Maturity**	**Target Maturity**	**Primary Challenges**	**Suggested Transformation Pathway**	**Projected Outcomes**
Multinational Financial Institution	Financial Services	Level 1 (Shell scripts)	Level 2-3 (Codified Infrastructure)	Manual compliance validation, approval delays, and audit preparation overhead	Ansible/Chef adoption (Level 2), then Terraform with GitLab CI/CD integration (Level 3)	Potential 60-70% reduction in audit preparation time, 75-85% decrease in drift incidents, and daily deployment capability
International Telecommunications Carrier	Telecommunications	Level 2-3 (Partial IaC)	Level 4 (Governed Automation)	Manual provisioning delays, GDPR compliance, service deployment constraints	Policy-as-Code implementation with cross-team collaboration, security scanning integration	Projected 85-90% reduction in provisioning time, 85-95% compliance coverage, improved environment consistency
Federal Government Agency	Government	Level 2 (Configuration Management)	Level 4 (Automated Compliance)	FedRAMP requirements, audit overhead, and documentation gaps	Phased Terraform migration with automated drift detection and policy enforcement	Expected increase in security automation coverage, reduced audit preparation burden, and comprehensive documentation

Table 3: Scenario-Based Maturity Progression Analysis and Projected Outcomes [7, 8]

V. Integration with Security and Compliance Frameworks

A. Zero-Trust Architecture Alignment

Principle Mapping

The Zero-Trust Architecture framework is compatible with advanced IaC implementations based on three principles. Constant authentication and authorization during IaC operations uses the principle of verify explicitly, which states that all operations on infrastructure are authenticated and authorized by identity checking and permission checking without regard to network location or prior authentication. Role-based access control and dynamic credentials reflect the principle of using the least privilege access, providing users with the minimum permissions through time-limited credentials that automatically expire. The assumption breach principle is put into practice by immutable infrastructure and automated forensics features, where infrastructure is assumed to be compromised and a full audit trail of every incident is retained.

Implementation Patterns

No long-lived static credentials in the identity-based infrastructure access system removes security risks due to long-lived static credentials, but rather produces temporary credentials, which are valid only to certain operations, and have limited time periods. Code-based Network segmentation and micro-segmentation define the boundaries of trust programmatically between different components of the infrastructure. The infrastructure definitions determine what components can communicate with each other and under what conditions. Trust boundaries must be continually monitored and validated to ensure security policies are implemented throughout the infrastructure lifecycle, any violation is detected and automated responses are taken.

IaC Maturity Correlation

The level of IaC maturity is directly related to the level of sophistication of zero-trust implementation. Level 3 organizations are those that have simple identity integration and network segregation through infrastructure-based code, which defines access limitations and network edge cases. Level 4 maturity enables all-purpose role-based access control, dynamic credential creation, and persistent verification processes that ensure the security posture is continuously validated, as opposed to being validated periodically. Level 5 autonomous infrastructure enforces AI-managed threat response along with automated and dynamically adjusted threat boundary control. The Zero-Trust Networks architecture is based on the principle that organizations require no implicit trust in their network architecture, but instead identify and authenticate all access requests irrespective of origin. Infrastructure automation supports this principle at scale via constant verification and policy enforcement [9].

B. SLSA Framework Integration

SLSA Level Correlation

The Supply Chain Levels Software Artifacts framework is systematically aligned with IaC maturity progression. SLSA Level 1 is also a documented build process, which can be implemented at IaC Maturity Levels 2-3, where infrastructure definitions are maintained in version control and are created with minimal documentation. SLSA Level 2 requires tamper-resistant build services, which is consistent with IaC Maturity Levels 3-4, where pipelines that have integrity controls are automated and used to deploy infrastructure. At Level 3: hardened builds with provenance, Level 3 is equivalent to IaC Maturity Level 4: full audit trails and cryptographic signing are used to determine the source of artifacts. Level 4 of SLSA requires hermetic constructions with reviews by two parties, which is only possible at the IaC Maturity Level 5, with a fully automated evaluation and attestation procedure.

Supply Chain Security Practices

The generation of the software Bill of Materials of infrastructure components presents the inventories of all dependencies in a detailed way, allowing tracking vulnerabilities and checking license compliance. Tamper detection is provided by cryptographically signing infrastructure artifacts, and the digital signatures confirm that the artifacts received are in the correct form and originate from authorized sources. Module dependency provenance. The provenance of module dependencies encompasses the entire provenance of a module throughout its deployment, recording each transformation and approval. Many of the attestations can be reproduced to provide independent verification that the infrastructure artifacts define precisely what they are intended to define. According to research investigating infrastructure as code security, infrastructure automation attacks are highly risky vectors since the damaged modules of infrastructure can spread vulnerabilities throughout the entire cloud environment, so complete verification and provenance tracking are critical security controls [10].

Zero-Trust Principle	Security Implementation	Level 3: Codified Infrastructure	Level 4: Governed Automation	Level 5: Autonomous Infrastructure
Verify Explicitly	Continuous authentication and authorization	Basic identity integration, manual verification	Comprehensive RBAC, automated credential validation	AI-driven identity verification, adaptive authentication
Least Privilege Access	Role-based access control, dynamic credentials	Static RBAC definitions, basic permission controls	Dynamic credential generation, time-limited access	Intelligent permission adjustment, context-aware access
Assume Breach	Immutable infrastructure, automated forensics	Version-controlled infrastructure, basic audit trails	Continuous monitoring, automated drift detection	Predictive threat detection, autonomous remediation
Network Segmentation	Micro-segmentation as code	Basic network boundary definitions	Policy-enforced segmentation, automated validation	Dynamic boundary adjustment, threat-responsive isolation
Trust Boundary Validation	Continuous monitoring and enforcement	Periodic validation, manual intervention	Real-time policy enforcement, automated alerting	Continuous verification, cryptographic attestation

Table 4: Zero-Trust Architecture Integration with IaC Maturity Levels [9]

Conclusion

The article presents the Infrastructure as Code Maturity Model, a comprehensive framework that addresses the critical gap in evaluating IaC implementations beyond tool acceptance by establishing quantifiable progression metrics across automation depth, governance sophistication, observability mechanisms, and security integration dimensions. The scenario-based analysis in Section IV demonstrates practical applicability across diverse organizational contexts, illustrating how financial services institutions and telecommunications providers progressing through maturity levels could achieve significant improvements in compliance audit preparation, infrastructure provisioning efficiency, and cross-functional collaboration through systematic automation adoption and policy-as-code implementation. The practical implications extend to multiple stakeholders, offering DevOps and platform engineering teams systematic roadmaps for progressive capability development, security and compliance officers structured frameworks for risk evaluation and automated regulatory enforcement as demonstrated in the financial services scenario, leadership teams data-driven insights for prioritizing automation investments based on deployment speed and audit readiness improvements, and operations and quality assurance teams enhanced capabilities through automated drift detection, remediation workflows, and continuous compliance validation rather than periodic manual audits. Integration with contemporary security frameworks, including Zero-Trust Architecture, Supply-chain Levels of Software Artifacts, and regulatory standards such as SOC 2, ISO 27001, FedRAMP, and GDPR, establishes complementary objectives, with scenario analyses demonstrating practical pathways where policy-as-code enforcement satisfies data residency constraints and automated monitoring supports continuous compliance requirements. Although the study acknowledges limitations, including the scenario-based nature of analyses rather than longitudinal empirical studies, reliance on projected outcomes based on theoretical modeling, and aspirational characteristics of autonomous infrastructure capabilities in early adoption phases, the established capabilities at intermediate maturity stages provide robust foundations for advanced automation practices, with projections deriving from established patterns in cloud automation research and DevOps transformation literature. Future research should focus on longitudinal validation studies, standardized assessment instruments, cultural and organizational success factors, and artificial intelligence applications for autonomous infrastructure, while industry collaboration through standardization efforts, benchmarking platforms, tooling enhancements, and educational resources would accelerate framework adoption. As organizations face increasing pressures from multi-cloud complexity, evolving security threats, and stringent regulatory requirements, Infrastructure as Code maturity emerges as a strategic imperative for competitive advantage, providing both diagnostic assessment capabilities and prescriptive roadmaps for systematic progression toward infrastructure that is inherently secure, continuously compliant, and operationally resilient by design.

References

[1] Nicole Forsgren, Jez Humble, and Gene Kim, "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations," IT Revolution Press. [Online]. Available: https://itrevolution.com/product/accelerate/

[2] Ramaswamy Chandramouli et al., "NIST Special Publication 800-204D: Strategies for Integration of Software Supply Chain Security in DevSecOps CI/CD pipelines," National Institute of Standards and Technology, 2024. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/204/d/final

[3] Kief Morris, "Infrastructure as Code: Dynamic Systems for the Cloud Age," O'Reilly Media, 2020. [Online]. Available: https://www.oreilly.com/library/view/infrastructure-as-code/9781098114664/

[4] Google Cloud and DORA, "Accelerate: State of DevOps Report 2023," 2025. [Online]. Available: https://cloud.google.com/devops/state-of-devops

[5] Gene Kim et al., "The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations, Second Edition," IT Revolution Press. [Online]. Available: https://itrevolution.com/product/the-devops-handbook-second-edition/

[6] Fredric Paul, "HashiCorp State of Cloud Strategy Survey 2023: Maturity Drives Operational Efficiency," HashiCorp Blog, 2023. [Online]. Available: https://www.hashicorp.com/en/blog/hashicorp-state-of-cloud-strategy-survey-2023-maturity-drives-operational-efficiency

[7] Sreeja Reddy Challa, "Cloud Automation in Financial Services: Securing and Scaling Banking Infrastructure in AWS," ResearchGate Publication, 2025. [Online]. Available: https://www.researchgate.net/publication/395983075_Cloud_Automation_in_Financial_Services_Securing_and_Scaling_Banking_Infrastructure_in_AWS

[8] Rashid Mijumbi et al., "Network Function Virtualization: State-of-the-Art and Research Challenges," ResearchGate, 2015. [Online]. Available: https://www.researchgate.net/publication/281524200_Network_Function_Virtualization_State-of-the-Art_and_Research_Challenges

[9] Evan Gilman and Doug Barth, "Zero Trust Networks," O'Reilly Media, 2017. [Online]. Available: https://www.oreilly.com/library/view/zero-trust-networks/9781491962183/

[10] Ruben Opdebeeck et al., "Analysing Software Supply Chains of Infrastructure as Code: Extraction of Ansible Plugin Dependencies,". [Online]. Available: https://soft.vub.ac.be/Publications/2024/vub-tr-soft-24-07.pdf