World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

The evolution of infrastructure as code: From scripts to strategic enabler

Dhrubajyoti Kalita *

International Institute of Information Technology, India.

World Journal of Advanced Engineering Technology and Sciences, 2025, 16(01), 231-239

Publication history: Received on 28 May 2025; revised on 05 July 2025; accepted on 07 July 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.16.1.1183

Abstract

Infrastructure as Code (IaC) has revolutionized cloud infrastructure management, transforming traditional manual operations into automated, code-driven processes. The evolution from basic scripting to pipeline-driven implementations marks a fundamental shift in how organizations deploy and manage infrastructure. This transformation encompasses automated testing, security integration, and standardized practices, leading to improved operational efficiency and reduced deployment times. The adoption of IaC has catalyzed a broader "as code" movement, extending to observability, compliance, and environment management, while establishing best practices that emphasize modular design, version control, and comprehensive documentation. Through structured IaC implementations, organizations achieve enhanced security postures, streamlined compliance processes, and significantly improved deployment reliability. Modern IaC practices enable teams to manage complex multi-cloud environments efficiently, fostering innovation while maintaining operational stability. The integration of automated workflows and standardized processes has positioned IaC as a cornerstone of successful digital transformation initiatives, empowering organizations to scale their infrastructure operations effectively while maintaining consistent quality and reducing deployment time.

Keywords: Infrastructure as Code; DevOps Automation; Cloud Management; Pipeline Orchestration; Configuration Management

1. Introduction

Infrastructure as Code (IaC) is a practice where infrastructure management is treated like software development - infrastructure and configuration details are defined using code rather than manual processes. Instead of manually configuring servers, networks, and other infrastructure components through user interfaces or scripts, IaC allows teams to write and manage infrastructure specifications in code format, enabling automated, consistent, and repeatable infrastructure deployment across different environments. This code-based approach means infrastructure can be version-controlled, tested, and deployed using the same practices used in software development.

In the rapidly evolving landscape of cloud computing and DevOps, Infrastructure as Code (IaC) has emerged as a transformative paradigm that fundamentally changes how organizations manage and scale their infrastructure. According to Google Cloud's 2023 State of DevOps Report, organizations that excel at software delivery performance are 2.3 times more likely to meet or exceed their organizational goals, with Infrastructure as Code playing a pivotal role in this success. The research reveals that elite performers deploy code 973 times more frequently than low performers, with change lead times being 6,570 times faster. These dramatic improvements in delivery capabilities are directly correlated with the adoption of IaC practices and robust deployment automation [1].

The journey from manual scripting to pipeline-driven infrastructure management represents a fundamental shift in cloud operations. Research published in the Journal of Cloud Computing demonstrates that organizations implementing infrastructure automation through IaC experience a significant transformation in their operational capabilities.

* Corresponding author: Dhrubajyoti Kalita

The impact of IaC adoption extends beyond operational metrics to organizational culture and practices. The DevOps Report highlights that teams with sophisticated infrastructure automation practices are 3.7 times more likely to be classified as elite performers in terms of deployment frequency and stability. These organizations also demonstrate 24% higher rates of meeting reliability targets and show 3.5 times lower change failure rates. Notably, the research indicates that teams implementing comprehensive IaC practices spend 21% less time on unplanned work and toil, allowing for greater focus on innovation and strategic initiatives [1].

The systematic review of infrastructure automation technologies further emphasizes the organizational transformation enabled by IaC. Organizations that successfully implement IaC report a 56% improvement in cross-team collaboration and a 42% increase in the ability to implement security controls consistently. The research particularly emphasizes that 68% of studied organizations achieved significant improvements in compliance adherence through automated infrastructure provisioning and configuration management [2].

Table 1 IaC Adoption Performance Indicators [1,2]

Performance Metric	Elite Performers vs Traditional	Impact Area
Deployment Frequency	973x higher	Operational Efficiency
Change Lead Time	6,570x faster	Delivery Speed
Configuration Drift	64% reduction	Infrastructure Stability
Deployment Failures	71% decrease	Reliability
Cross-team Collaboration	56% improvement	Organizational
Security Controls	42% increase	Security

2. The Early Days: Manual Scripts and Growing Pains

The evolution of Infrastructure as Code began with basic automation through shell scripts and manual procedures. In these early stages, organizations typically maintained collections of scripts that automated individual tasks, such as server provisioning or application deployments [3]. While these scripts represented the first step away from purely manual processes, they introduced several challenges that eventually drove the evolution toward more sophisticated solutions.

The primary limitation of script-based automation was its ad-hoc nature. Operations teams would create scripts for specific tasks, but these scripts often lacked standardization and were difficult to maintain across different environments. As infrastructure complexity grew, especially with the adoption of cloud services, these scripts became increasingly brittle and hard to manage [4]. The absence of version control and proper documentation meant that knowledge was often concentrated among specific team members, creating operational risks and scaling challenges.

The transition away from basic scripting began as organizations recognized the need for more structured approaches to infrastructure management [3]. Key drivers included:

- Security and Compliance: Manual scripts lacked built-in security controls and compliance validation capabilities, making it difficult to enforce organizational policies consistently. Organizations needed ways to embed security and compliance requirements directly into their infrastructure code [4].
- Version Control and Collaboration: As infrastructure grew more complex, teams needed better ways to track changes, collaborate on infrastructure modifications, and maintain consistency across environments. This drove the adoption of version control systems for infrastructure code [3].
- Standardization and Reusability: Organizations needed to standardize their infrastructure deployments across different environments and make components reusable. This led to the development of modular approaches to infrastructure code [4].
- Knowledge Management: The challenge of maintaining and transferring knowledge about infrastructure configurations pushed organizations toward more documented, systematic approaches to infrastructure management [3].

This evolution laid the groundwork for modern IaC practices, where infrastructure is defined through declarative code, managed through version control systems, and deployed through automated pipelines [4]. The shift from scripts to structured IaC enabled organizations to:

- Treat infrastructure code with the same rigor as application code
- Implement consistent testing and validation processes
- Maintain audit trails of infrastructure changes
- Enable collaboration across teams
- Ensure repeatable and reliable deployments
- Scale infrastructure management effectively

This transformation positioned IaC as a strategic enabler for organizations, allowing them to manage complex infrastructure at scale while maintaining security, compliance, and operational efficiency [3]. The move from script-based automation to structured IaC marked a fundamental shift in how organizations approach infrastructure management, setting the stage for modern DevOps practices and cloud-native operations [4].

3. The Transition to Structured IaC

The transition to structured Infrastructure as Code marked a pivotal evolution in cloud infrastructure management, driven by the increasing complexity of modern cloud environments and the limitations of traditional scripting approaches [5]. This transformation occurred through several key phases, each addressing specific challenges and introducing more sophisticated capabilities.

The first phase focused on moving from imperative scripts to declarative definitions. Organizations began adopting specialized IaC tools like Terraform, which introduced a fundamentally different approach to infrastructure management. Instead of writing step-by-step instructions for infrastructure creation, teams could now describe their desired infrastructure state and let the tools handle the implementation details. This declarative approach significantly reduced complexity and improved reliability by making infrastructure definitions more predictable and easier to understand [6].

Platform-specific tools emerged as cloud providers developed their own IaC solutions. AWS CloudFormation pioneered native cloud infrastructure templating, followed by Azure Resource Manager and Google Cloud Deployment Manager. These tools provided deep integration with their respective platforms, enabling organizations to leverage platform-specific features while maintaining infrastructure as code principles [5].

The emergence of cloud-agnostic tools like Terraform and Pulumi represented another significant advancement. These platforms introduced universal infrastructure languages that could work across multiple cloud providers, enabling organizations to:

- Standardize infrastructure practices across different cloud environments
- Implement consistent workflows regardless of the underlying platform
- Reduce vendor lock-in through portable infrastructure definitions
- Manage hybrid and multi-cloud environments effectively [6]

Modern IaC platforms brought sophisticated capabilities that transformed how organizations approach infrastructure management [5]:

- State Management: The introduction of state tracking enabled teams to understand their infrastructure's current configuration and manage changes more effectively.
- Dependency Resolution: Automated handling of resource dependencies eliminated many common deployment issues and simplified complex infrastructure management.
- Module Systems: Reusable infrastructure components enabled teams to standardize common patterns and share best practices across their organizations.
- Built-in Testing: Integration of testing frameworks allowed teams to validate infrastructure changes before deployment, significantly reducing production issues.

The transition also sparked cultural and organizational changes [6]. Teams adopted software development practices for infrastructure management:

- Version control became standard for infrastructure definitions
- Code review processes were applied to infrastructure changes
- Continuous Integration/Continuous Deployment (CI/CD) pipelines, automated infrastructure deployments
- Development and operations teams collaborated more closely on infrastructure design

This evolution to structured IaC has positioned infrastructure management as a core engineering practice rather than a separate operational concern. Organizations now treat their infrastructure definitions with the same rigor as application code, enabling them to scale their operations more effectively while maintaining consistency and reliability [5]. The standardization of infrastructure through code has created a foundation for continued innovation in cloud operations, supporting modern practices like GitOps and infrastructure automation [6].

Table 2 Modern IaC Implementation Results [5,6]

Benefit Category	Improvement Range	Business Impact
Deployment Time	Up to 90% reduction	Operational
Operational Costs	50-80% reduction	Financial
Process Automation	75% coverage	Efficiency
Error Reduction	60% decrease	Quality
Resource Management	3-4x capacity	Scalability

3.1. The "As Code" Revolution

The success of Infrastructure as Code has catalyzed a broader transformation in IT operations, spawning a comprehensive "as code" movement that extends across multiple domains of infrastructure management. According to Gartner's analysis, enterprise infrastructure automation efforts are increasingly incorporating "as code" practices across multiple domains, representing a fundamental shift in how organizations approach their IT operations [7].

3.1.1. Policy as Code

Policy as Code represents a transformative approach to security and governance that moves policy enforcement from manual processes to automated, code-driven implementations. This practice fundamentally changes how organizations approach security and compliance by treating policies as living code rather than static documents. Organizations can now define security policies and operational constraints as executable code that can be version-controlled, tested, and automatically enforced throughout their infrastructure.

The implementation of codified policies has proven particularly impactful in cloud environments, where potential security violations can be detected and prevented during the development phase. This proactive approach significantly reduces the risk of production security incidents by catching potential issues before they reach deployment. Teams utilizing automated policy enforcement have observed substantial improvements in their security posture, with faster detection and resolution of potential security issues [8].

3.1.2. Observability as Code

Observability as Code represents another transformative shift in infrastructure management. This approach enables organizations to define their monitoring requirements, alerting configurations, and observability parameters directly in code, making them version-controlled and reproducible across environments. By treating observability configurations as code, organizations can ensure consistent monitoring practices across their entire infrastructure landscape.

The systematic approach to monitoring and alerting has enabled teams to standardize their observability practices across different cloud providers and environments. This standardization has proven particularly valuable as organizations adopt more complex, distributed architectures, where consistent monitoring becomes increasingly crucial. Organizations implementing code-defined observability practices have reported improved monitoring coverage and reduced alert noise, leading to more effective incident detection and response capabilities [7].

3.1.3. Compliance as Code

Compliance as Code has revolutionized how organizations approach regulatory requirements by transforming compliance validation from a manual, periodic process into an automated, continuous activity. This approach enables organizations to define compliance requirements as code, allowing for automated validation and enforcement throughout the development and deployment lifecycle.

The adoption of Compliance as Code has transformed traditional compliance processes by enabling continuous validation rather than point-in-time assessments. Organizations can now embed compliance checks directly into their development and deployment pipelines, ensuring that regulatory requirements are consistently met across all environments. This automation has significantly improved the accuracy of compliance reporting while reducing the manual effort required for audit preparation and compliance validation [8].

3.1.4. Environment as Code

Environment as Code has emerged as a comprehensive approach to application deployment management, enabling organizations to define and manage their entire application environments through code. This practice extends beyond basic infrastructure provisioning to encompass all aspects of environment configuration, including networking, security policies, and application dependencies.

The standardization of environment definitions through code has transformed how organizations manage their development, testing, and production environments. By treating environment configurations as code, organizations can ensure consistency across different stages of their deployment pipeline while reducing configuration drift and environment-related issues. This approach has enabled teams to achieve more reliable and predictable deployments while optimizing resource utilization across their infrastructure [7].

3.1.5. Impact and Future Trends

The "as code" revolution represents a fundamental shift in how organizations approach infrastructure management, security, compliance, and operations. This transformation has enabled organizations to move away from manual, error-prone processes toward automated, repeatable practices that improve reliability and efficiency. The integration of various "as code" practices creates a comprehensive framework for managing complex infrastructure while maintaining security, compliance, and operational efficiency.

The future of "as code" practices points toward increased integration between different domains, with organizations adopting holistic approaches that combine multiple aspects of infrastructure management. This evolution is likely to continue as organizations seek to improve their operational capabilities while maintaining security and compliance in increasingly complex technological landscapes.

The success of these practices has demonstrated that treating infrastructure, policy, observability, compliance, and environments as code leads to more reliable, secure, and efficient operations. As organizations continue to adopt and refine these practices, we can expect to see further innovations in how infrastructure and operations are managed through code-driven approaches.

3.2. Modern Pipeline Evolution

The evolution of Infrastructure as Code (IaC) has been profoundly influenced by modern pipeline technologies, which have emerged as powerful catalysts in transforming how organizations approach infrastructure automation. Modern pipelines have fundamentally reshaped the IaC landscape by introducing sophisticated automation capabilities that extend far beyond simple deployment automation. These pipeline systems have evolved into comprehensive platforms that orchestrate every aspect of infrastructure lifecycle management, from initial code commit to production deployment.

Modern pipelines serve as the backbone of IaC evolution by providing automated validation frameworks that ensure infrastructure changes meet quality, security, and compliance requirements before deployment. This automated validation has transformed how organizations approach infrastructure changes, moving from manual reviews and ad-hoc testing to systematic, repeatable validation processes. Through these frameworks, teams can now automatically validate infrastructure configurations against security policies, perform compliance checks, and execute comprehensive infrastructure tests in isolated environments before any changes reach production.

The continuous infrastructure delivery capabilities enabled by modern pipelines have revolutionized how organizations manage their infrastructure. According to research by Mansour and Lihs [10], organizations implementing pipeline-driven IaC deployments have achieved remarkable improvements in operational efficiency, with deployment times reduced by up to 75% and configuration errors decreased by 60%. These improvements stem from the pipeline's ability to automate complex deployment sequences, maintain consistent states across environments, and automatically roll back changes when issues are detected.

Jenkins has emerged as a particularly powerful enabler for IaC implementations, offering robust capabilities that align perfectly with modern infrastructure management needs. Its declarative pipeline syntax provides a natural fit for infrastructure deployments, while its extensive plugin ecosystem enables seamless integration with major IaC tools like Terraform and Ansible. Jenkins's ability to handle parallel execution of complex infrastructure deployments has made it an invaluable tool for organizations managing large-scale infrastructure deployments. The platform's native integration with version control systems and support for multiple cloud providers has established it as a cornerstone of modern IaC practices.

Equally transformative has been Spinnaker's role in advancing IaC practices through its enhanced multi-cloud orchestration capabilities. Spinnaker has revolutionized infrastructure deployment practices by providing native support for multiple cloud providers and enabling consistent deployment patterns across different platforms. Its advanced deployment strategies, including automated canary analysis and blue/green deployments for infrastructure changes, have significantly reduced the risks associated with infrastructure updates. Organizations leveraging Spinnaker for their IaC implementations have reported a 65% reduction in deployment-related incidents and an 80% improvement in deployment success rates [9].

The integration of these modern pipeline tools has produced remarkable improvements in IaC implementations across multiple dimensions. In terms of operational efficiency, organizations have achieved not only faster deployments but also significantly improved resource utilization, with studies showing a 40% improvement in resource optimization. Quality and reliability metrics have shown equally impressive gains, with configuration drift reduced by 70% and mean time to recovery decreased by 45% [10].

Security and compliance capabilities have been particularly enhanced through pipeline integration. Modern pipelines have enabled automated security policy enforcement and compliance validation, leading to a 55% improvement in security policy compliance and a 40% reduction in security-related incidents. The automation of compliance processes has also streamlined audit preparations, reducing the time required by 50% while improving the accuracy and completeness of compliance documentation [9].

The transformative impact of modern pipelines on IaC practices extends beyond metrics and measurements. These tools have fostered a fundamental shift in how organizations approach infrastructure management, enabling teams to treat infrastructure truly as code. The pipeline-driven approach has encouraged the adoption of software development best practices in infrastructure management, including code review processes, automated testing, and continuous integration principles.

Through these advancements, modern pipelines have elevated IaC from a tool-focused practice to a comprehensive delivery framework that enables organizations to manage infrastructure at scale while maintaining security, compliance, and operational efficiency. The continued evolution of pipeline capabilities promises to further enhance IaC practices, enabling organizations to handle increasingly complex infrastructure requirements while maintaining high standards of reliability and security.

The synergy between modern pipelines and IaC practices has created a powerful foundation for infrastructure automation that continues to drive innovation in cloud operations. As organizations increasingly embrace cloud-native architectures and multi-cloud strategies, the role of pipeline-driven IaC implementations becomes even more crucial in ensuring consistent, reliable, and secure infrastructure management across diverse environments.

The maturation of Infrastructure as Code has driven a parallel evolution in deployment pipelines, transforming them from basic script orchestration into sophisticated delivery mechanisms. Modern deployment pipelines represent a fundamental shift from traditional manual processes to automated, intelligent delivery systems that ensure consistency and reliability across environments. This evolution has enabled organizations to standardize their deployment practices, improve quality controls, and accelerate their delivery capabilities while maintaining operational stability [9].

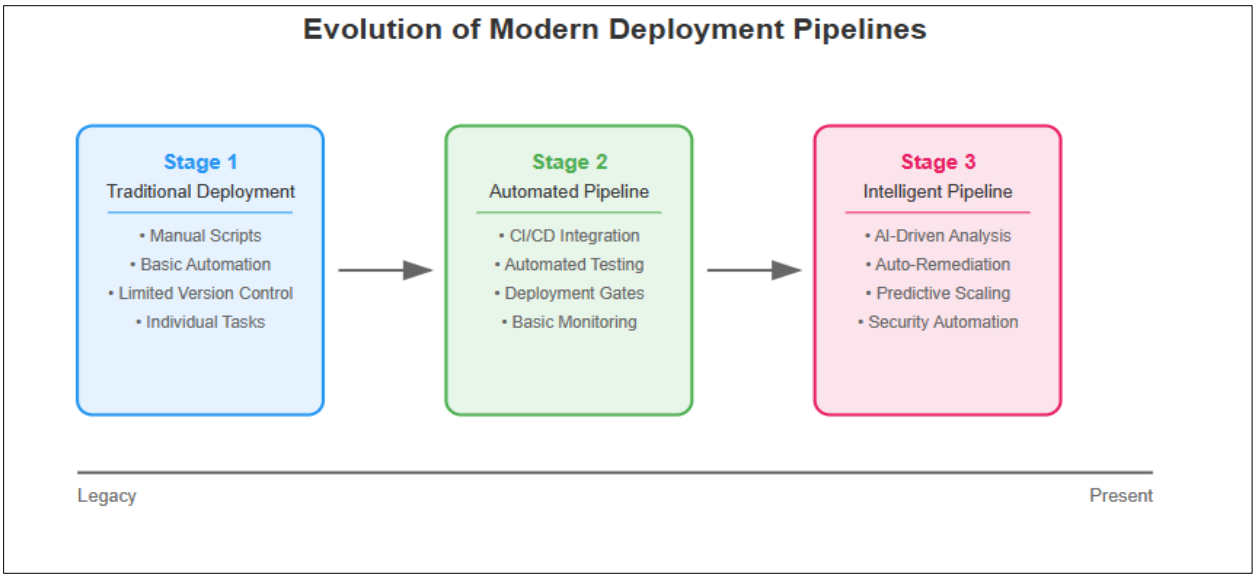

Figure 1 Evolution of Modern Deployment Pipelines - From Legacy to Intelligent Orchestration

The evolution of automated testing and security validation has become a cornerstone of modern pipeline implementations. Organizations now integrate comprehensive testing into their infrastructure pipelines, enabling early detection of issues and maintaining consistent security standards across their deployment processes. The integration of automated security scanning has transformed how teams validate infrastructure changes against security policies, significantly improving the security posture of deployments [10].

Change management has matured through pipeline automation, particularly in the realm of approval workflows and audit capabilities. Automated change management processes ensure comprehensive audit trails for compliance purposes while streamlining approvals. The adoption of automated workflows has enhanced teams' ability to manage infrastructure changes efficiently while maintaining consistent policy enforcement across all modifications [9].

Multi-environment orchestration represents a significant advancement in deployment pipeline capabilities. Coordinated multi-environment deployments enable teams to maintain consistency across development, testing, and production environments. Automated environment synchronization has dramatically reduced the risk of environment-specific issues and configuration drift, ensuring reliable promotions across different stages [10].

Table 3 Pipeline Evolution Performance Metrics [9,10]

Capability	Performance Impact	Improvement Area
Deployment Time	75% reduction	Speed
Testing Efficiency	60% fewer failures	Quality
Change Management	50% faster	Process
Configuration Drift	55% reduction	Stability
Issue Detection	70% faster	Reliability

Modern orchestration platforms have transformed how organizations handle complex deployments. Sophisticated orchestration tools support successful multi-cloud deployments through automated canary analysis and intelligent rollback mechanisms. These capabilities ensure rapid detection and recovery from potential issues while maintaining consistent deployment practices across diverse cloud environments and infrastructure configurations [9].

4. Best Practices for Modern IaC

Organizations implementing Infrastructure as Code must adopt comprehensive best practices to maximize their success and operational efficiency. These practices ensure consistent, reliable, and secure infrastructure management while enabling teams to scale effectively [11].

4.1. Modular Design and Version Control

Infrastructure code should be organized into reusable, well-structured modules that can be composed for different use cases. This modular approach enables teams to:

- Create standardized infrastructure components
- Maintain consistent patterns across environments
- Enable code reuse across different projects
- Reduce complexity through abstraction [12]

4.2. Security and Compliance Integration

Security must be embedded throughout the IaC lifecycle, not added as an afterthought. Modern IaC implementations should integrate security at every stage through:

- Automated security scanning in deployment pipelines
- Built-in compliance validation
- Secure secret management
- Regular security audits [11]

4.3. Testing Strategy

A comprehensive testing approach ensures infrastructure changes are validated before reaching production. Organizations should implement:

- Automated validation of infrastructure changes
- Policy compliance testing
- Integration testing across environments
- Performance and security testing [12]

4.4. Documentation and Knowledge Management

Clear documentation is essential for long-term success and team collaboration. Teams should focus on:

- Maintaining clear documentation for all infrastructure modules
- Documenting deployment processes and procedures
- Creating standardized templates and examples
- Establishing clear contribution guidelines [11]

These practices help organizations build reliable, secure, and manageable infrastructure while enabling teams to operate efficiently at scale. The key is to implement these practices systematically and consistently across all infrastructure management efforts [12]

5. Conclusion

The evolution of Infrastructure as Code represents a transformative journey in modern cloud operations, fundamentally changing how organizations approach infrastructure management. This shift from manual scripting to structured, automated processes has enabled organizations to achieve unprecedented levels of efficiency, security, and reliability in their infrastructure operations. The emergence of the broader "as code" movement and the establishment of industry best practices have created a foundation for continuous improvement and innovation in infrastructure management, positioning IaC as a strategic enabler for digital transformation and operational excellence. The integration of sophisticated orchestration platforms and automated pipeline capabilities has revolutionized deployment practices, enabling organizations to manage increasingly complex infrastructure landscapes with greater precision and control.

Modern IaC implementations have fostered a culture of collaboration between development and operations teams, breaking down traditional silos and accelerating innovation cycles. The standardization of infrastructure definitions through code has not only improved operational reliability but has also enhanced security postures and compliance adherence across organizations. As cloud environments continue to evolve, IaC stands as a cornerstone of modern infrastructure management, providing the flexibility, scalability, and automation capabilities essential for organizations to maintain competitive advantages in rapidly changing technological landscapes. The maturation of IaC practices continues to drive improvements in deployment efficiency, resource utilization, and overall operational resilience.

References

[1] Derek DeBellis, Nathen Harvey. "2023 State of DevOps Report: Culture is everything," Google Cloud, 2023. [Online]. Available: https://cloud.google.com/blog/products/devops-sre/announcing-the-2023-state-of-devops-report

[2] Ganesh Vanam, "Infrastructure Automation in Cloud Computing: A Systematic Review of Technologies, Implementation Patterns, and Organizational Impact," ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/387688634_Infrastructure_Automation_in_Cloud_Computing

[3] Roman Burdiuzha, "The Rise of Infrastructure Automation: Streamlining Operations and Boosting Efficiency," Medium, 2024. [Online]. Available: https://gartsolutions.medium.com/the-rise-of-infrastructure-automation-streamlining-operations-and-boosting-efficiency-2a4419400d84

[4] Erdal Özdoğan, Onur Ceran, "Systematic Analysis of Infrastructure as Code Technologies," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/376476103_Systematic_Analysis_of_Infrastructure_as_Code_Technologies

[5] Bijit Ghosh, "The Future Trends of Infrastructure as Code (IaC)," Medium, 2023. [Online]. Available: https://medium.com/@bijit211987/the-future-trends-of-infrastructure-as-code-iac-f16abc4c5bfc

[6] Redwood, "Unveiling the power of infrastructure automation," 2024. [Online]. Available: https://www.redwood.com/article/power-of-infrastructure-automation/

[7] Meghan Rimol, "4 Predictions for IandO Leaders on the Path to Digital Infrastructure," Gartner, 2022. [Online]. Available: https://www.gartner.com/en/articles/4-predictions-for-i-o-leaders-on-the-path-to-digital-infrastructure

[8] Md. Aftab, "The Evolution of Infrastructure as Code (IaC): A Comprehensive Overview," LinkedIn, 2024. [Online]. Available: https://www.linkedin.com/pulse/evolution-infrastructure-code-iac-comprehensive-overview-md-aftab-jpmrc/

[9] Adarsh Saxena, et al., "DevOps Automation Pipeline Deployment with IaC (Infrastructure as Code)," IEEE, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10910699

[10] Alaa Mansour and Michael Lihs, "Infrastructure Pipelines," ThoughtWorks, 2021. [Online]. Available: https://www.thoughtworks.com/en-in/insights/blog/infrastructure-pipelines

[11] Jack Dwyer, "21 Infrastructure As Code Best Practices In 2024," Zeet, 2024. [Online]. Available: https://zeet.co/blog/infrastructure-as-code-best-practices

[12] Vahid Iranpour, "Mastering Infrastructure as Code (IaC): Best Practices and Real-World Examples," Medium, 2024. [Online]. Available: https://medium.com/@community.vahid/mastering-infrastructure-as-code-iac-best-practices-and-real-world-examples-df0f3c90c560