International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 06 | June - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Empowering DevOps with Infrastructure as Code :Trends, Tools and Techniques

Shashi Ranjan
Department of ISE
R. V. College of Engineering®
Bengaluru, India

Dr. Mamatha G S
Department of ISE
R. V. College of Engineering®
Bengaluru, India

***Abstract*—DevOps has arisen as a pillar of modern software engineering, emphasising the integration of development and operations to ensure effective product delivery. Infrastructure as Code (IaC) is an important DevOps technique that involves defining and managing infrastructure requirements using code, enabling for automated provisioning and maintenance. This technique enhances traceability, reuse, and consistency across development and production environments. The introduction of microservices architectures has increased project teams' infrastructure responsibilities, making IaC essential for delivering reliable and efficient deployments. IaC enables developers to describe infrastructure in code, simplifying the deployment process. In both large and small businesses, IaC is essential for supporting efficient DevOps processes.The most recent breakthroughs, tools, and techniques in IaC demonstrate a revolutionary impact on software development and deployment workflows. As more businesses adopt cloud-native designs and containerisation technologies, the requirement for automated infrastructure provisioning grows, leading in the growth of IaC tools and methodologies. Organisations that combine IaC with continuous integration and delivery (CI/CD) pipelines can reduce time-to-market and improve operational efficiency. IaC not only automates infrastructure management, but it also includes software engineering principles like version control and testing into infrastructure provisioning, which improves consistency and reliability. This democratisation of infrastructure management encourages increased collaboration across cross-functional teams, hence improving accountability and innovation. Implementing IaC is therefore crucial for achieving agility, scalability, and resilience in the digital age.**

***Index Terms*—Iac, DevOps, CI/CD, Automation, Cloud-native, Infrastructure Provisioning ,Terraform, Ansible, AWS CloudFormation**

I. Introduction

DevOps integration has transformed software development and deployment, resulting in shorter delivery cycles and more communication between development and operations teams. The foundation of this change is Infrastructure as Code (IaC), which automates and controls infrastructure using machine-readable specification files. IaC improves the scalability, agility, and reliability of the software delivery pipeline by allowing DevOps teams to supply and configure infrastructure resources programmatically.

The introduction of cloud-native architectures and containerisation technologies like as Docker and Kubernetes has greatly accelerated DevOps and IaC trends. Businesses are increasingly using IaC to manage complex infrastructures across many clouds and hybrid environments, allowing for seamless application deployment and scalability. The rise of serverless computing has increased the demand for automated infrastructure provisioning, resulting in the development of IaC technologies and processes. Infrastructure automation is facilitated by tools like as Ansible, Chef, Terraform, and AWS CloudFormation, and integrating IaC with continuous integration and delivery (CI/CD) pipelines improves time-to-market and operational efficiency.

IaC represents a major shift in infrastructure management by treating infrastructure like code. This method enables the application of software engineering principles such as version control, code review, and testing to infrastructure configurations, ensuring consistency, repeatability, and reliability. IaC also democratises infrastructure management by encouraging cross-functional collaboration and increasing responsibility and innovation. As more firms implement DevOps and IaC approaches, the border between development and operations blurs, resulting in more dependable and efficient software delivery pipelines.

II. Literature Review

Some research and review papers have been published in this domain. The paper by Rishabh Sethia et al.,[1] focuses on developing a CI/CD pipeline for microservices applications using Cloud DevOps, emphasizing automation and the advantages of DevOps in terms of time and quality. It provides a thorough literature review covering the history and meaning of DevOps, its comparison with traditional IT infrastructure, and the tools used in the DevOps process such as Docker, Kubernetes, Jenkins, and Ansible. The proposal discusses the advantages of DevOps in financial companies, bug detection, and continuous testing, deployment, and release. It also addresses challenges in DevOps adoption and application development. The project aims to digitalize startup planning, create a user-friendly environment, capture staff efforts efficiently, and meet user requirements.

Another paper by Karthikeya Vaitla et al. [2] explores the integration of notifications and manual approval workflows within AWS CDK Pipelines to advance DevOps automation. It underscores AWS CDK's benefits, including abstraction, productivity, scalability, and consistency in cloud asset management, while emphasizing the importance of notifications

and manual approvals in CI/CD pipelines. A step-by-step methodology is provided for integrating these elements into AWS CDK Pipelines, supported by real-world case studies illustrating improvements in deployment speed, consistency, and client satisfaction. The findings suggest that this integration enhances visibility, balances automation with human intervention, and ensures compliance with security and regulatory requirements. It offers a systematic approach to enhance DevOps automation, addressing modern software delivery challenges while fostering both speed and control in the development process.

The research paper by Mohammed et al. explores the utilization of Infrastructure as Code (IaC) for setting up web applications via cloud infrastructure, particularly focusing on AWS's CloudFormation service. It underscores the significance of IaC in establishing a fully functional network infrastructure, including websites, databases, and other features, emphasizing benefits such as cost reduction, speed, risk mitigation, consistency, and enhanced security strategies. The study assumes developers' access to IaC resources and comprehensive knowledge to utilize them effectively. It outlines a detailed methodology covering planning, implementation, and testing phases, demonstrating successful infrastructure creation, update, and deletion. The introduction contextualizes the cloud's evolution, emphasizing its importance in data storage, security, and service flexibility. It positions IaC as pivotal in cloud infrastructure management, enabling automation, compliance management, and security. The paper concludes by discussing future work, focusing on streamlining IaC processes and potential advancements in efficiency and user-friendliness [3].

The paper by Matej Artac et al. [4] explores DevOps, IaC, and the TOSCA standard. It highlights DevOps as tactics for accelerating software design changes, emphasizing agility through the integration of software engineering and IT operations. IaC's significance in DevOps is underscored, advocating for infrastructure designs expressed as "source code" to streamline operations. TOSCA is presented as the industrial standard for IaC, automating deployment with technology-independent and multi-cloud compliant applications. Core DevOps concepts address organizational and technical distances, promoting socio-technical practices for collaboration. IaC's role in infrastructure design and TOSCA's specification capabilities are detailed, illustrated with the "DICER" example for big data cloud application design. The document concludes by highlighting IaC and TOSCA's potential to accelerate DevOps lifecycles, calling for further research on applying software development tactics to infrastructure design.

In a research paper by Michele Chiari et al., they provide a thorough analysis of Infrastructure as Code (IaC), focusing on its benefits in DevOps and the challenges it poses in terms of security and reliability during deployment operations. Through a literature review, it identifies techniques for static analysis of IaC scripts, categorizing them into syntactic-based methods and those utilizing automated verification techniques. The research questions addressed include available tools, techniques, and properties checked by these tools, alongside a systematic summary of code smells found in IaC. The methodology involved searching scientific literature databases for relevant entries on IaC static analysis techniques. The document discusses various tools and techniques for static verification and validation of IaC, targeting platforms such as Puppet, Ansible, TOSCA, and CloudFormation, to detect anti-patterns, deployment plan correctness, code smells, and security vulnerabilities. It concludes by highlighting areas for future research to enhance accuracy and automation in defect detection and deployment verification [5].

The research paper by Michele Guerriero et al. [6] delves into the intricate ecosystem of Infrastructure-as-Code (IaC) through a meticulous analysis of 44 semi-structured interviews across diverse industry sectors. With a systematic approach, the paper navigates through the nuances of IaC adoption, emphasizing the pivotal role of software engineering practices and tooling support. Highlighting both the current landscape and impending challenges, it underscores the necessity for continued research to bridge the gap between practitioner needs and tool capabilities. By shedding light on best practices and prevalent pitfalls, the document serves as a compass for both industry professionals and academic researchers, offering actionable insights to enhance IaC development, testing, and maintenance practices. Ultimately, the paper lays the groundwork for future endeavors aimed at refining IaC methodologies and fortifying its position in modern infrastructure management paradigms.

In the paper by Prashant Agrawal et al. [7], they explore the intersection of DevOps and cloud development/testing. It emphasizes the importance of automating DevOps processes using cloud technologies. DevOps practices enhance software development and operational agility. The article discusses migrating DevOps to the cloud and expanding automation to public/private clouds. It focuses on conceptual insights and best practices for cloud development and testing. The "results" lie in the understanding of how DevOps and the cloud collaborate.

The research conducted by Shashipraba Perera introduces the process of setting up a CI/CD pipeline for React apps, targeting beginners with no prior experience in CI/CD. Continuous Integration (CI) involves regular code pushes to the source code repository, followed by activities like static code analysis, building the application, and running tests. Continuous Deployment (CD) automates deploying builds from the CI process directly to the production environment. The architecture includes Git for source code management, GitHub for code hosting, CircleCI as the CI/CD tool, AWS S3 buckets for storing React app builds, and AWS CloudFront (a CDN) for content delivery. It guides readers through the process of setting up a CI/CD pipeline for React apps using specific tools and services. The outcome lies in the reader's ability to implement CI/CD practices effectively [8].

In the thesis done by D. Vladusic et al. [9], they address the challenge of setting up infrastructure for application deployment using the Infrastructure as Code (IaC) approach. IaC enables the creation of software-defined infrastructure

capable of running applications. The paper focuses on the complexities arising from heterogeneous infrastructures (combining Cloud and High-Performance Computing - HPC). Cloud infrastructure emphasizes servers, events, and functions, while HPC infrastructure executes applications directly on bare metal. The dynamics and longevity of applications deployed on these systems differ significantly. The paper provides an AI-supported Integrated Development Environment (IDE) for modeling system components and applications using a straightforward language. It orchestrates application execution within a software-defined environment, optimizing source code for the targeted infrastructure. SODALITE monitors application execution using machine learning and control theory approaches.

In the paper by Daniel Sokolowski [10], he addresses the challenge of setting up infrastructure for application deployment using the Infrastructure as Code (IaC) approach. IaC enables the creation of software-defined infrastructure capable of running applications. The paper focuses on the complexities arising from heterogeneous infrastructures (combining Cloud and High-Performance Computing - HPC). Cloud infrastructure emphasizes servers, events, and functions, while HPC infrastructure executes applications directly on bare metal. The dynamics and longevity of applications deployed on these systems differ significantly. The paper introduces the SODALITE project, which aims to simplify application deployment complexity while maintaining or improving performance on heterogeneous HPC and cloud systems. SODALITE abstracts application deployment through modeling, uses container technologies, and employs machine learning for runtime performance improvement.

III. Role of IaC in DevOps Transformation

Infrastructure management used to be a laborious, tedious, and prone to error manual procedure that frequently resulted in irregularities and scale issues. By enabling infrastructure to be specified and maintained through code, which is versionable, reviewed, and tested similarly to application code, Infrastructure as a Code (IaC) revolutionises this method. DevOps workflows benefit greatly from this transition to code-based infrastructure management, which increases efficiency, consistency, and repeatability.

IaC makes it possible to successfully reproduce environments at all development stages, from testing to production, guaranteeing configuration consistency and mitigating the ”it works on my machine” issue. Maintaining dependable and predictable application behaviour across many deployment contexts depends on this consistency. Version control is another feature of IaC that helps teams monitor modifications, work together more efficiently, and revert to earlier settings when needed. Because it takes much less time to set up and configure environments, the automation offered byIaC also improves the speed and agility of DevOps teams. In continuous integration and deployment (CI/CD) pipelines, where quick and frequent deployments are crucial, this acceleration is crucial. Teams can swiftly spin up test environments, roll out updates, and dynamically scale resources thanks to Infrastructure as a Service (IaC), which supports the agility and flexibility required in today’s software development. IaC, taken as a whole, is a cornerstone of the DevOps revolution, promoting increased productivity, dependability, and creativity in the administration and implementation of applications.

Fig. 1 shows how infrastructure as a Code Works and manages application infrastructure in the cloud and the premises.

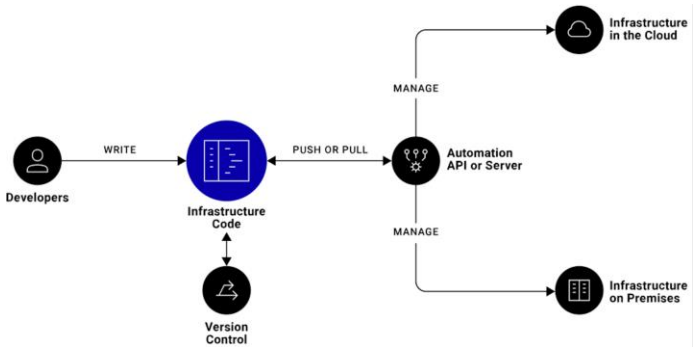

Fig. 1. Working of Infrastructure as a code in Devops

IV. Trends in IaC Adoption

The broad use of Infrastructure as Code (IaC) has resulted in a substantial paradigm shift in the way organisations approach infrastructure management in recent years. With a growing focus on considering infrastructure configurations as programmable code artefacts, this trend represents a fundamental shift in the provisioning, configuring, and management of infrastructure.

Automating repetitive operations and streamlining complicated infrastructure deployment procedures is a major trend in the adoption of Infrastructure as a Service (IaC) space. To do this, organisations are using IaC technologies and frameworks. Automation increases uniformity, lowers the possibility of human error in infrastructure management duties, and boosts operational efficiency.

The use of IaC has grown even further with the introduction of multi-cloud and hybrid cloud systems, which let companies manage infrastructure uniformly across numerous cloud platforms. Declarative configuration management systems are also becoming more and more common; they provide better control and predictability over infrastructure modifications by declaratively defining the infrastructure’s state.

This approach supports the ideas of DevOps by enabling automated, version-controlled infrastructure provisioning, which promotes cooperation between development and operations teams. In addition, the idea of immutable infrastructure—in which infrastructure elements are replaced rather than updated and considered as disposable—is becoming more and more popular, which is why IaC procedures that facilitate reliable and consistent deployments are necessary.

Fig.2 shows the trends of increasing market size of Infrastructure as a Code which is expected to be 2.8 Billion Dollars

in 2028 from 485.6 Million Dollars in 2018 as published by Global NewsWire in 2022.

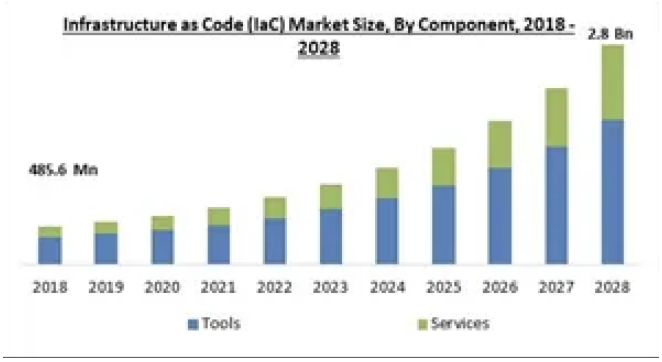

Fig. 2. Infrastructure as a Code Market size

V. Tools for Infrastructure as Code (IaC)

There are many tools for Infrastructure as Code (IaC) encompass a diverse range of solutions tailored to different infrastructure management needs. Among the most widely adopted are:

A. Terraform

Infrastructure as Code (IaC) has seen the rise of HashiCorp's flagship product Terraform, which is revolutionising the way infrastructure provisioning and management are approached by enterprises. Because it allows users to specify infrastructure requirements in a human-readable manner, Terraform's declarative configuration language is a key component of its attractiveness. With the help of this language, users may concentrate on expressing the intended state of their infrastructure rather than the technical procedures required to get there. It abstracts the complexity of cloud provider APIs and configuration syntax. By promoting consistency and reproducibility across settings, this abstraction not only makes IaC more accessible to practitioners.

Key Terraform selling point is its provider-based architecture shown in Fig.3, allowing easy integration with many cloud providers, including AWS, Azure, and the Google Cloud Platform. It enables organizations to pursue multi-cloud approaches without jeopardizing their operational capacity or becoming anxious about vendor lock-in. Terraform can be used to provision on-premises infrastructure as well as a plethora of third-party services, all of which can improve use case versatility and enable a variety of hybrid cloud layouts and deployment models. Finally, Terraform boosts infrastructure management initiative's compatibility and standardisation efforts by transforming cloud-specific APIs and resource definitions into provider plugins.

Terraform state management is an essential filer-level in controlling the integrity and the idempotent property of infrastructure deployment and redeployment. The information within the state file assembles the source of truth regarding the current state of the deployed resources, making it possible

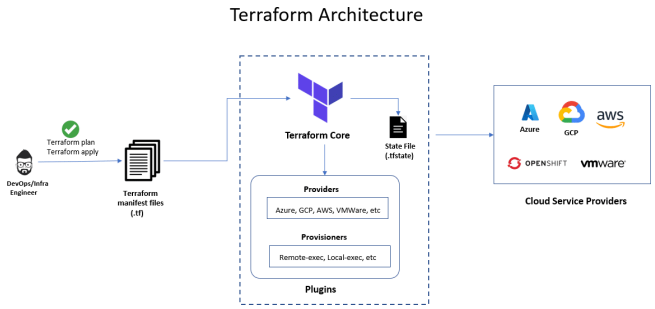

Fig. 3. Terraform Architecture

and discrete to update. This file enables the Terraform to manage the complexity of resource-dependent and infrastructures redeployment and reduce configuration drift; it is a mental state where the discrepancies between the configuration and state are concerned. Terraform enables a "desired state" reality where the infrastructure is consistently redeployed into the modulated configuration state and provides a flexible and clear interface within the dynamic disposition.

B. AWS CloudFormation

A key element of the Infrastructure as Code (IaC) environment is Amazon CloudFormation, which enables users to automate AWS resource provisioning and administration using templated configurations. With the help of this service from Amazon Web Services (AWS), it is possible to define and deploy infrastructure resources in a way that is repetitive and predictable.

YAML or JSON files that declaratively define the intended state of AWS resources are the foundation of AWS CloudFormation. Together with their interdependencies and characteristics, these templates include the configuration information for a variety of resources, including S3 buckets, RDS databases, and EC2 instances.UThe code infrastructure can be defined in templates to allow users to have AWS setups which are legible and controlled by versions. This promotes collaboration and maintains consistency during deployment.

AWS CloudFormation is favored because of its ease with which it allows for other services as well as capabilities offered by Amazon Web Services to be integrated seamlessly. CloudFormation templates can define flexible and dynamic infrastructure configurations by using parameters, mappings, conditions and outputs among others. Additionally, CloudFormation supports nested stacks allowing re-usability and modularity of various template components across different projects and contexts. This improves scalability, simplifies maintenance and enhances code reuse in large deployments.

Another key point of AWS CloudFormation is that it facilitates change management and rollback capabilities. AWS retains a record of any changes made to CloudFormation stacks, then lets users look back at them, verify them and undo the ones that are not needed. These built-in tools for managing alterations provide a way of mitigating infrastructure change

risks that can affect the integrity and stability of Amazon Web Services' systems.

To add on to this, AWS CloudFormation seamlessly incorporates into AWS Identity and Access Management (IAM) for more granular control over permissions as well as access policies for stack resources. IAM roles, policies, and permissions can be defined within the CloudFormation templates by users in order to ensure secure and compliant infrastructure deployments.

In addition, since many companies now utilize cloud-native architectures as well as DevOps principles; it has been discovered that; AWS CloudFormation is a fundamental element in automating provisioning together with management of infrastructures in AWS environments.

Table.1 shows differences between two most widely used Infrastructure as a Code Service Terraform and AWS CloudFormation

Terraform	AWS CloudFormation
Supports multiple cloud providers and on-premises environments.	Exclusively focuses on managing AWS resources.
Uses HashiCorp Configuration Language (HCL), with clear syntax for configuration language	Supports JSON and YAML formats for templates.
Follows "plan-apply" model for creating, updating, and deleting resources	Adopts "create-update-delete" model for stack management.
Utilizes state files for tracking infrastructure state and changes.	Manages stack state internally without exposing it.
Features a wide range of third-party providers and modules	Integrates deeply with AWS services and resources.
Offers a rich set of interpolation functions for dynamic configurations.	Provides intrinsic functions for dynamic template values and references.

TABLE I
DIFFERENCES BETWEEN TERRAFORM AND AWS CLOUDFORMATION

C. Ansible

Ansible is an open-source automation software that has been hailed as a game changer in infrastructure management due to its simplicity and flexibility. Ansible, created by Red Hat, is centred on the idea of simplicity and is user-friendly for both operations and development teams. It provides infrastructure configuration using a YAML (YAML Ain't Markup Language)-based syntax. Ansible's agentless architecture, in contrast to that of many other automation tools, allows it to administer distant systems over SSH without the need to install extra software on the target hosts. This feature also extends to its "playbook" idea, which enables users to streamline automation workflows by defining actions and configurations in a playbook file.

Ansible's extensive library of "modules," each of which encodes certain actions or tasks to be carried out on controlled hosts, is what gives the programme its fundamental capability. With the help of these modules, users can automate a variety of infrastructure-related operations, resulting in more flexible operating environments. These tasks range from package installation and service management to file manipulation and cloud resource provisioning. Moreover, Ansible's idempotent feature guarantees that executing the same playbook more than once will provide consistent outcomes, reducing the possibility of unforeseen modifications while maintaining the intended state of infrastructure.

When it comes to automating various parts of infrastructure management, such as orchestration, provisioning, application deployment, and configuration management, Ansible shines. It is an excellent choice for managing both big, distributed infrastructures and small-scale systems due to its light footprint and low dependency requirements. In addition, Ansible's "roles" feature facilitates the encapsulation and sharing of reusable task sets, which encourages code reuse and streamlines the administration of intricate infrastructure setups.

Ansible encourages cooperation and integration throughout DevOps workflows in addition to its basic automation features. Because of its interaction with Git and other version control systems, infrastructure configurations may be versioned, stored, and worked on collaboratively alongside application code. Furthermore, Ansible enables automated testing, deployment, and rollback of infrastructure changes in tandem with application code changes by smoothly integrating with continuous integration and deployment (CI/CD) pipelines.

Fig.4 shows basic architecture of Ansible, an open source provided of IaC.

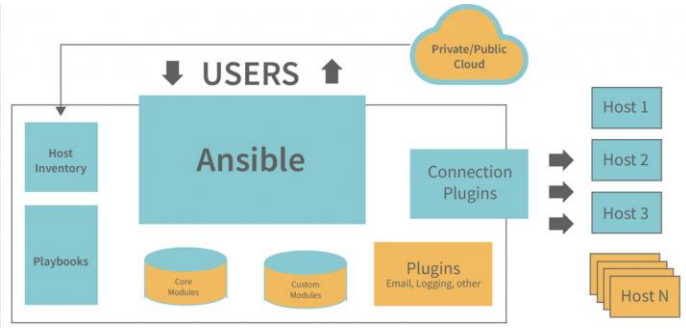

Fig. 4. Ansible Architecture

VI. TECHNIQUES FOR IAC DEVELOPMENT

Providing effective, dependable, and maintainable infrastructure provisioning and management requires implementing a number of strategies as part of the Developing Infrastructure as Code (IaC) process. The following are some essential techniques for IaC development:

- **Modularization:** Divide infrastructure setups into reusable modules, each of which stands for a logical part or pattern. By enabling teams to assemble complicated infrastructure from smaller, more manageable components, modularization fosters code reuse, streamlines maintenance, and improves scalability.
- **Template Inheritance:** Use template composition or inheritance to reduce code duplication and encourage uniformity between infrastructure settings. Specify foundation templates that contain common configurations that can be extended or incorporated into particular use cases

or environments. This method permits customisation as needed while guaranteeing uniformity.

- **Version Control:**Utilising Git or other similar tools, apply version control principles to infrastructure code. Version control makes it easier to collaborate, streamlines the code review procedure, and offers a history of modifications for auditing and rollback needs. Additionally, it encourages the use of branching and tagging as best practices for handling releases and test modifications.
- **Continuous Integration/Continuous Deployment (CI/CD):**Automate the testing, validation, and deployment of infrastructure changes by integrating IaC development into CI/CD pipelines. Create automated processes that enable fast iteration and smooth update delivery for developing, testing, and deploying infrastructure code in various contexts. Collaboration is improved, manual error rates are decreased, and infrastructure modification time-to-market is sped up with the use of CI/CD techniques.
- **Immutable Infrastructure:**Adopt the idea of immutable infrastructure, in which infrastructure parts are swapped out and replaced rather than updated on-site. By reducing configuration drift and lowering the possibility of configuration errors, immutable infrastructure fosters consistency, reproducibility, and resilience.
- **Monitoring and Observability:**Utilise monitoring and observability technologies in infrastructure settings to measure performance indicators, identify anomalies, and undertake proactive troubleshooting. Real-time visibility into the health and performance of infrastructure is made possible by monitoring, which makes it easier to respond quickly to events.

VII. Conclusion

In Conclusion, the increasing utilisation of Infrastructure as Code (IaC) is transforming DevOps methodologies by providing hitherto unseen degrees of flexibility, expandability, and dependability for IT operations. This study has explored important patterns, widely used tools, and crucial methodologies to emphasise the transformative impact of IaC. Organisations are adopting Infrastructure as a Service (IaC) to automate infrastructure provisioning, configuration, and administration across heterogeneous environments. This includes a trend towards automation and collaboration as well as the use of technologies like Terraform, AWS CloudFormation, and Ansible.

Furthermore, methods like continuous integration and delivery (CI/CD), parameterization, and modularization highlight how crucial reliability and maintainability are to IaC development workflows. In today's quickly changing digital economy, enterprises that embrace Infrastructure as a Service (IaC) build a culture of automation and collaboration, dismantling silos between development and operations teams, accelerating innovation, and fostering business growth.

References

[1] M. Staron, S. Abrahão, B. Penzenstadler and L. Hochstein, "Recent Research Into Infrastructure as Code," in IEEE Software, vol. 40, no. 1, pp. 86-88, Jan.-Feb. 2023, doi: 10.1109/MS.2022.3212035.
[2] Chaudhary, Ashutosh Gabriel, Mary Sethia, Rishabh Kant, Shubham Chhabra, Sonia. (2021). Cloud DevOps CI -CD Pipeline.
[3] Eleraky, Mohammed Anis Aziz, Wagdy Soliman, John. (2023). Using Cloud Infrastructure as a Code IaC to Set Up Web Applications. International Journal of Simulation: Systems, Science Technology. 24.
[4] Artač, Matej Borovssak, Tadej Di Nitto, Elisabetta Guerriero, Michele Tamburri, Damian. (2017). DevOps: Introducing Infrastructure-as-Code. 497-498. 10.1109/ICSE-C.2017.162.
[5] M. Chiari, M. De Pascalis and M. Pradella, "Static Analysis of Infrastructure as Code: a Survey," 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), Honolulu, HI, USA, 2022, pp. 218-225, doi: 10.1109/ICSA-C54293.2022.00049.
[6] M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092.
[7] P. Agrawal and N. Rawat, "Devops, A New Approach To Cloud Development Testing," 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 2019, pp. 1-4, doi: 10.1109/ICICT46931.2019.8977662.
[8] Perera, W. Sanduni Shashiprabha. (2022). CI/CD PIPELINE FOR A REACT APP WITH AWS DEVOPS.
[9] D. Vladusic and D. Radolovic, "Infrastructure as Code for Heterogeneous Computing," 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 2020, pp. 1-2, doi: 10.1109/SYNASC51798.2020.00011.
[10] Daniel Sokolowski. 2022. Infrastructure as code for dynamic deployments. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022). Association for Computing Machinery, New York, NY, USA, 1775–1779. https://doi.org/10.1145/3540250.3558912
[11] Howard, Michael. (2022). Terraform – Automating Infrastructure as a Service. 10.48550/arXiv.2205.10676.
[12] Makani, Sai Teja. (2021). Deep Dive into Terraform for Efficient Management of AWS Cloud Infrastructure and Serverless Deployment. 8. 6.
[13] Zadka, M. (2022). Terraform. In: DevOps in Python. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7996-015
[14] Masek, Pavel Štůsek, Martin Krejčí, Jan Zeman, Krystof Pokorny, Jiri Kudlacek, Marek. (2018). Unleashing Full Potential of Ansible Framework: University Labs Administration. Proceedings of the XXth Conference of Open Innovations Association FRUCT. 426. 10.23919/FRUCT.2018.8468270.
[15] Senapathi, Mali et al. "DevOps Capabilities, Practices, and Challenges: Insights from a Case Study." Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (2018): n. pag.