2026 Volume: 02 Issue: 02 www.ijrpa.com ISSN 2456-9995 Research Article

International Journal Research Publication Analysis

Page: 01-09

DETECTION AND ANALYSIS OF CONFIGURATION DRIFT IN INFRASTRUCTURE AS CODE BASED CLOUD SYSTEMS

***Saladi.H.L.S.S.Ganesh, K.Jaswanth, K.Sai Manikanta, G.Raju**

Students, Head of the Department, Computer Science and Engineering Department,
Ideal Institute of Technology, Vidyut Nagar, Kakinada-533005, Andhra Pradesh, India.

Article Received: 08 January 2026
Article Revised: 28 January 2026
Published on: 16 February 2026

***Corresponding Author: Saladi.H.L.S.S.Ganesh**
Students, Head of the Department, Computer Science and Engineering Department, Ideal Institute of Technology, Vidyut Nagar, Kakinada-533005, Andhra Pradesh, India.

DOI: https://doi-doi.org/101555/ijrpa.4310

ABSTRACT:

Infrastructure-as-Code (IaC) has emerged as a foundational practice in modern cloud computing by enabling automated, consistent, and version-controlled infrastructure provisioning. However, dynamic cloud operations and manual interventions frequently introduced configuration drift, causing the deployed infrastructure to diverge from the intended state defined in IaC specifications. Such drift can lead to security vulnerabilities, compliance violations, service instability, and increased operational complexity. While prior research has primarily focused on detecting configuration drift, there remains a lack of comprehensive approaches that address continuous monitoring, impact analysis, and practical integration with DevOps workflows. This paper proposes an automated and systematic approach for the detection and analysis of configuration drift in IaC-based cloud systems. The proposed system establishes a baseline infrastructure state form IaC repositories and continuously compares it with the real-time deployed environment using state comparison and rule-based verification techniques. Detected deviations are analysed and logged to support informed decision-making by administrators. A prototype implementation using widely adopted IaC and cloud technologies demonstrates the feasibility and effectiveness of the approach. Experimental results indicate that the proposed system improves infrastructure consistency, enhances operational reliability, and supports security and compliance requirements in dynamic cloud environments.

KEYWORDS: Infrastructure-as-Code, Configuration Drift, Cloud Computing, DevOps, Cloud Automation, Drift Detection.

I. INTRODUCTION

Cloud computing has changed how companies plan, set up, and handle their computer systems. It provides easy scalability, flexibility, and cost savings. As cloud systems become more complicated, managing them with old, manual methods has led to more mistakes and less efficiency. To solve these problems, Infrastructure-as-Code (IaC) has become a major approach. It lets you create and manage computer systems using simple text files that machines can read. This makes it easier to track changes, automate processes, and ensure consistent setups every time.

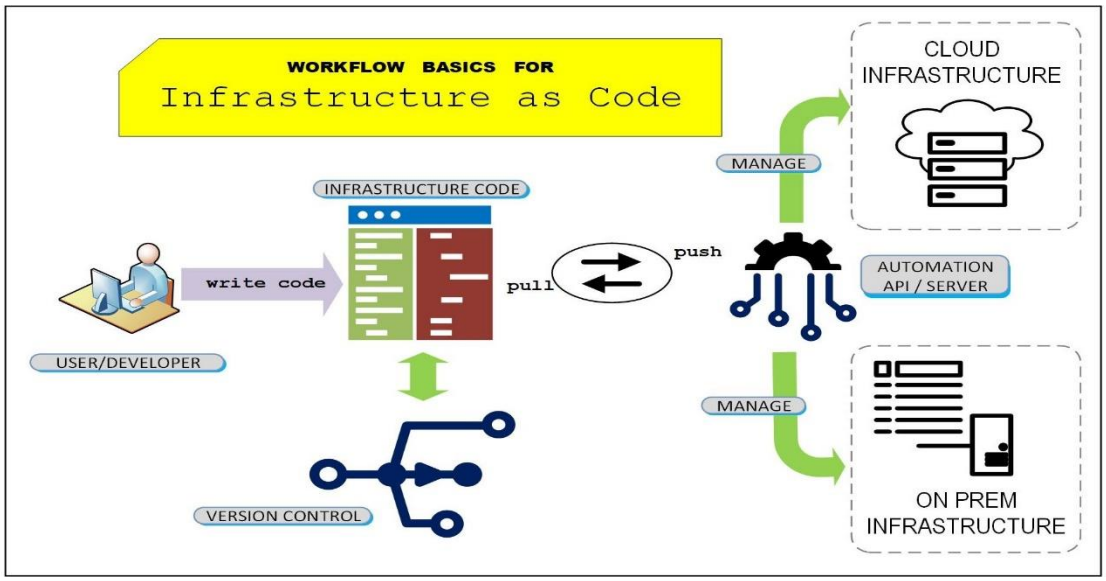
It is hard to track changes in the setup of cloud environments that use infrastructure as code because these environments are large, have many different parts, and keep changing all the time. Most current methods check for changes only at certain times or use tools that don't give up-to-date information or a full picture. Also, many solutions only find when something has changed but don't explain why or how it affects the system. This makes it harder to make smart decisions and manage the infrastructure well in advance.

In this situation, there's a bigger need for automatic and ongoing ways to detect changes in infrastructure that don't match the original plans. This helps with modern DevOps methods. The paper tackles this by offering an automatic way to find and check configuration drift in cloud systems built with infrastructure-as-code. The system creates a standard version of the infrastructure from the IaC files and keeps checking it against the actual cloud setup using comparisons and rule checks. The study shows through tests how this method improves consistency, reliability, and compliance in changing cloud environments.

II. INFRASTRUCTURE-AS-CODE

Infrastructure-As-Code (Iac): Infrastructure-as-Code (IaC) has become a popular way to automate and standardize the setup of cloud infrastructure. Morris (2021) highlighted that IaC is a key practice for managing cloud resources using code that is controlled through versioning. This helps ensure that deployments are repeatable and consistent. However, the study mainly looked at best practices and tools, with little attention given to configuration drift after deployment and how to spot it.

IaC makes operations more efficient by reducing mistakes that happen when people manually set up configurations.It also ensures that different environments, like development, testing, and production, are kept consistent. By treating infrastructure setups like code, organizations can use standard software development techniques such as version control, peer reviews, and automated testing. Tools like Terraform, Ansible, and AWS CloudFormation help define infrastructure in either declarative or imperative ways, making cloud resource management easier and more scalable.



But even with these benefits, IaC setups can face problems if changes are made directly to the actual infrastructure without updating the code that defines it.This mismatch, called configuration drift, can reduce the reliability and predictability of the system, leading to security issues and operational difficulties. Because of this, it's important to have strong monitoring and management in place for IaC-based infrastructure to fully benefit from automation and consistency in cloud environments.

Configuration Drift: Configuration drift happens when the actual setup of the deployed systems doesn't match the intended setup as described by the company's rules or the code used to define the infrastructure. In cloud setups, this often happens because people make changes by hand using the cloud platform's tools, fix problems quickly without following proper steps, apply updates automatically, or don't follow the same setup rules consistently. As time goes on, these changes create differences between what the system was supposed to be and how it's actually working.

Cloud Computing : Cloud computing is a way that lets people access shared computing tools like servers, storage, networks, and apps over the internet whenever they need them. It takes away the need for companies to own and manage their own physical equipment by offering resources that can grow or shrink depending on how much work is being done. This model charges you only for what you use, which makes it a good and efficient choice for both small and big systems.

DevOps: DevOps is a method used in software development and operations that focuses on teamwork, using automated tools, and constantly combining the work of developers and operations staff. Its main aim is to make the process of creating software faster while still delivering dependable and good-quality software. By bringing together the steps of building, testing, releasing, and managing the systems that support software, DevOps helps companies adapt quickly to new needs and changes in the market.

[1] TechRadar, "Infrastructure-As-Code Workflow," 2023.
[Online]. Available: https://cdn.mos.cms.futurecdn.net/EFgXKNVuN9gZ2wZThqdeYC.jpg.

III. PROBLEM STATEMENT AND MOTIVATION

Managing IaC-Based Cloud Infrastructure: Managing cloud infrastructure using Infrastructure-as-Code (IaC) means setting up and controlling cloud resources like servers, networks, storage, and security settings through code. This approach allows for automatic and consistent setup of infrastructure, which makes deploying systems easier and helps reduce mistakes made by people. But as cloud environments get bigger and more complex, managing IaC setups becomes harder because there are more changes, resources scale up or down often, and different services depend on each other. Keeping the actual setup in the cloud matching what is defined in the code needs ongoing checks and validation. If there are no good ways to manage these setups, changes can happen without control, leading to configuration drift. This can hurt the reliability, security, and compliance of cloud systems. So, it's really important to manage IaC-based cloud infrastructure well to keep things stable and to get the most out of cloud automation.

Impact of Configuration Drift: Configuration drift can cause big problems for the stability and security of cloud systems that use Infrastructure-as-Code. If the actual setup of the system doesn't match what was planned, it can create security issues like incorrect access settings, services that are left open, or old security rules. These issues can make it easier for unauthorized people to get into the system and steal data.

Need for an Automated Drift Detection Mechanism: In today's cloud setups, infrastructure settings often change because of scaling, updates, and other operational actions. In systems that use Infrastructure-as-Code, it's not efficient or reliable to track these changes manually and check if the configuration is consistent. As cloud systems get bigger and more complex, manually checking for configuration changes becomes hard and can't keep up with the speed of changes. Having an automated way to detect configuration drift is important because it helps keep track of the real state of cloud resources and compare it with the intended state from the Infrastructure-as-Code files. Automation helps catch configuration changes early, so admins can fix issues before they cause security problems, system breakdowns, or compliance violations. It also helps with DevOps by keeping environments consistent and reducing the work needed to manage them.

IV. PROPOSED SYSTEM FOR CONFIGURATION DRIFT DETECTION

IaC Repository and Baseline State Definition: The Infrastructure-as-Code (IaC) repository acts as the main place where cloud infrastructure setups are defined and managed. It holds files that can be read by machines, which describe how cloud resources should be set up. These resources include computers, networks, storage, and security rules. These files are kept in a version-controlled system, which helps track changes, understand the history, and go back to previous setups if needed.

Creating a baseline state means setting up a standard configuration that is used to check the actual setup against.After using IaC tools to set up the infrastructure, this setup becomes the baseline. This baseline shows what the infrastructure is meant to look like and what has been approved. It includes details about each resource, how they are connected, and other settings based on the IaC files.

Actual Infrastructure State Collection: Real infrastructure state collection means getting up-to-date information about the actual setup of cloud resources that are already in use. This helps understand how they are currently working. In the system we’re talking about, this data is gathered directly from the cloud using tools that manage infrastructure through code and APIs provided by cloud services. The data collected includes things like what type of resources are used, their settings, network arrangements, security permissions, and how they depend on each other. This data is collected either regularly or whenever needed to make sure any changes made outside of the infrastructure-as-code process are picked up.

By keeping track of the actual state of the infrastructure continuously, the system always has the latest information about the resources in place. This allows for accurate comparisons with

the standard setup stored in the infrastructure-as-code repository, which is key for spotting any differences or drift in configurations. Automating this collection process saves time and makes it easier to keep a close eye on cloud environments that change often.

Rule-Based Drift Identification Logic: Rule-based drift identification logic is used to check and find differences between the expected infrastructure setup and the actual setup that's been deployed. This method uses set rules to compare important settings like resource details, security options, and how different parts of the system depend on each other. When there's a difference between what was planned in the infrastructure as code (IaC) and what is actually in place, it's considered configuration drift.

The rule-based system helps find these differences in a clear and organized way, without needing complicated machine learning tools. Once the differences are found, they are grouped by how serious they are and what kind of issue they are, so administrators can focus on the most important ones first. This method ensures that configuration drift is found accurately, while keeping things simple, easy to understand, and efficient in cloud systems that use IaC.

V. RESULT

To solve the issues with current Infrastructure-as-Code drift management methods, this research suggests an automated, ongoing, and tool-free framework for finding and understanding configuration drift in cloud systems. The new approach makes sure that the actual cloud setup stays the same as the intended setup described in the IaC files.

The system starts by keeping track of the original infrastructure setup from the IaC files stored in a version-controlled system. This original setup is the target configuration for cloud resources and acts as a reference for checking drift. A monitoring part checks the real setup in the cloud regularly using APIs from the cloud provider, giving up-to-date information on the actual resources.

VI. CHALLENGES

- **Dynamic Nature of Cloud Environments**: Cloud environments are always changing, with frequent scaling, updates, and adjustments to resources. Because of these continuous changes, it's hard to keep an accurate and current baseline for detecting configuration drift.

- **Manual and Untracked Changes**:

Configuration drift often happens because people make changes manually through cloud management tools.These changes are usually not recorded, which makes it harder to spot and understand the drift.

- **Complex Resource Dependencies**: Cloud resources are connected in complex ways across networks, security, and computing layers.A small change in one part can affect other parts in unexpected ways, making it harder to detect drift.

- **Scalability Issues**: As the number of cloud resources grows, checking and comparing their states becomes slower and more resource-heavy, especially in big setups.

- **Tool and Platform Heterogeneity**: Different cloud providers and infrastructure-as-code tools use different ways to represent configurations.

This makes it difficult to consistently detect drift across multiple clouds or mixed environments.

- **Lack of Real-Time Monitoring**: Many current tools check for configuration drift at set intervals, not in real time.

This delay can let drift go unnoticed for a while, leading to potential problems in the system.

VII. CONCLUSION

In conclusion, this paper introduces a method for finding and understanding configuration drift in cloud systems that use Infrastructure-as-Code (IaC). Configuration drift is a major problem in today's cloud environments because frequent changes and manual actions can cause the actual infrastructure to move away from the intended setups defined in IaC scripts. These changes can harm the security, reliability, and compliance of the system. The system proposed in this paper starts by setting up a baseline of the infrastructure from the IaC repository and then keeps checking it against the real infrastructure. Using automatic data gathering and rules to find drift, the system can spot configuration differences and give useful information to administrators. This method lowers the need for manual work, keeps the infrastructure consistent, and helps DevOps practices by keeping the actual setup in line with the desired one.

In general, the drift detection method proposed helps make cloud infrastructure more reliable and easier to manage. By catching configuration drift early, organizations can stop possible problems and security threats, making cloud operations more stable and predictable.

VIII. ACKNOWLEDGEMENT

We would like to express our thanks to everyone who helped in making this research on "Detection and Analysis of Configuration Drift in Infrastructure-as-Code Based Cloud Systems" possible. First, we want to say how much we appreciate our research team. Their teamwork and smart ideas were very important throughout the whole project. Their dedication and knowledge helped us solve many problems related to managing cloud infrastructure and finding configuration drift. We also want to thank our institution for supporting us. They gave us the tools, resources, and environment needed to carry out this research.

We are grateful to the departments and labs that allowed us to use cloud platforms, tools, and infrastructure that were key for our experiments. We also want to thank our mentors and colleagues for their helpful feedback and guidance, which greatly improved the quality and depth of our work.

Finally, we acknowledge the work of the wider research community and the developers of the tools and technologies we used.Their work and ideas continue to help improve cloud automation, DevOps practices, and Infrastructure-as-Code systems.

IX. REFERENCES

1. Kief Morris, *Infrastructure as Code: Managing Servers in the Cloud*, 2nd Edition, O'Reilly Media, 2021.
2. HashiCorp, "Terraform Documentation," https://www.terraform.io/docs/, Accessed: Dec. 2025.
3. Michael J. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*, Wiley, 2014.
4. P. Modi, D. Patel, and S. Patel, "Detection of Configuration Drift in Cloud Infrastructure: A Survey," *International Journal of Cloud Computing*, vol. 9, no. 2, pp. 45-58, 2023.
5. B. R. Kandukuri, R. Paturi, and V. Rakshit, "Cloud Security Issues," *IEEE International Conference on Services Computing*, pp. 517-520, 2009.
6. M. Fowler, *Continuous Integration: Improving Software Quality and Reducing Risk*, Addison-Wesley, 2006.

7. R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," *10th IEEE International Conference on High Performance Computing and Communications*, 2008.
8. A. Humble and J. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Addison-Wesley, 2010.