World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:18, No:12, 2024

# Enhancing Network Management through Continuous Integration and Continuous Delivery Pipelines and Infrastructure as Code Practices

Tharunika Sridhar



***Abstract*—**This research explores the benefits and methods of integrating Continuous Integration and Continuous Delivery (CI/CD) methodologies into network automation, focusing on Infrastructure as Code (IaC) applications. The primary goal is to enable IT organizations to achieve scalable network resources with enhanced security and compliance while streamlining deployment processes. The study highlights some key advantages of CI/CD, including improved version control, reduced manual errors and enhanced operational efficiency. Despite these benefits, using cloud-based systems and open-source tools introduces security challenges that organizations must address to optimize network configurations effectively.

***Keywords*—**Infrastructure as Code, Continuous Integration, Continuous Delivery, pipeline, azure, terraform.

## I. Introduction

MODERN enterprises face the challenges of rapidly evolving competitive environments, increasing security mandates, and the need for scalable performance. Over the past decade, it has become essential for organizations to balance operational stability with swift feature development. CI/CD is the key answer to this problem, a methodology that facilitates swift software modifications while ensuring system stability and security [1].

Organizations leveraging software development as a competitive edge continually face the challenge of accelerating software delivery while maintaining quality and stability. A set of technical practices has emerged to meet this demand, collectively known as Continuous Delivery (CD). The primary objective of CI/CD methods is to deliver products to clients regularly by automating various stages of development, thereby enhancing process efficiency and minimizing the risk of human error [2].

As CI/CD has revolutionized software development, network teams seek to incorporate the CI/CD methodologies into their network automation and orchestration efforts to enhance speed and optimize delivery [3]. Network engineers are increasingly utilizing repositories, pipelines, and various technologies to expedite the creation, testing, and implementation of network automation and orchestration workflows [4].

This paper analyzes the feasibility and practical implications of CI/CD-based automation for secure network configuration. Specifically, it examines the application of open-source IaC in CI/CD to improve network setup functionality. To our knowledge, this study serves the following purposes [5]:

a) IaC in CI/CD enables resource scaling without impacting network performance, allowing adaptation to workload changes.
b) CI/CD pipelines incorporate IaC support security and compliance checks, standardize network settings and reduce vulnerabilities.
c) IaC facilitates DevOps practices, meeting operational demands through accelerated testing, iteration, and deployment of network changes.
d) Version control, IaC documentation, and easy rollback mechanisms reduce downtime and manual errors.
e) Network provisioning automation minimizes manual labor and costs and optimizes resource usage.
f) IaC enforces compliance and accountability by auditing network setups to document modifications.

## II. Related Works

Several studies have explored the integration of CI/CD and IaC in network automation. For instance, [6] discussed the challenges of creating manual Enterprise Architecture (EA) models and proposed automating data collection to enhance model reliability. This study presented network scanning for autonomous data collecting and leverages ArchiMate to generate EA models based on company IT infrastructure. While promising, the approach required additional effort to ensure practical applicability.

Reference [7] highlighted the benefits of CI/CD, such as version control, automated testing, and rapid validation of software changes. The research justified that developers could merge code changes into a repository and execute automated builds and tests in continuous integration (CI). CI speeds up software validation and release, improves software quality and adds to bug detection and fixes. The study emphasized the efficiency gains in software development, but there is still a need to explore the implications for secure and automated network architecture.

As 5G standards evolved, as analyzed in [8], they enabled significant advancements in network capabilities. This research confirmed that mobile carriers must adapt to an expanding range of industries and complex use cases. Engaging in vertical 5G markets involves various approaches, particularly for

Tharunika Sridhar is with Microsoft, WA 98052 USA (phone: 571-297-5205; e-mail: tsridhar@micrsoft.com).

onboarding vertical Network Applications. This study introduces a streamlined, standards-based methodology for vertical application onboarding. To ensure compatibility and reproducibility, only industry-standard data models are utilized. In practice, the methods outlined in this paper are applied within a large-scale distributed 5G infrastructure.

Reference [9] examined the adoption of CI/CD in deploying the Academic Information System at Paramadina University, where application development relies on manual processes. CI/CD is a software development methodology that enables teams to automate code integration, run tests, and deliver applications consistently and continuously. The researchers aimed to implement CI/CD in the deployment process to improve development efficiency, enhance application quality, and increase adaptability.

Reference [10] examined IaC technologies, focusing on their management and coding techniques. The study stressed that network device configuration, resource allocation, and application deployment are automated by "IaC" technologies. Machine-readable codes allow IaC tools to perform time-consuming tasks dynamically. The findings indicated the potential of IaC to enhance automation but noted limitations in network security optimization.

Similarly, the study in [11] analyzed the attack surfaces and security implications of CI/CD pipelines on a large scale. An analytical tool was developed to examine over 320,000 CI/CD-configured GitHub repositories and quantify security-critical practices. The findings revealed that late script updates and script runtimes could conceal malicious code and exploit vulnerabilities. The research confirmed that even valid scripts can contain significant flaws. Furthermore, the current CI/CD ecosystem relies on numerous simple scripts, which could create single points of failure.

The publications examined show CI/CD and IaC network infrastructure automation, standardization, and security gaps. According to [6], manual EA model generation with automation but unintegrated CI/CD and IaC have drawbacks. References [7] and [9] address the deployment efficiency benefits of CI/CD, but their potential for secure, automated network architecture is less well-known.

The reviewed literature underscores the benefits of CI/CD and IaC in automating network processes, but gaps in standardization and security remain, necessitating further exploration in this study.

## III. Problem Statement

Despite the growing adoption of CI/CD and IaC, challenges with standardization, security, and reliability persist. This is leading to a shift to automated network infrastructure via CI/CD and IaC, as it claims for robust and extensive validation on standardization, security, and reliability issues because these models are particularly being developed as improvements over manual testing processes. Manual processes, as observed, make configurations inconsistent, error-prone, and more vulnerable to security breaches. For standard, secure, and dependable CI/CD configurations, tight automated checks and compliance standards with little human interaction are needed. It proposes a CI/CD-based IaC framework for network management that improves operational consistency, security, and dependability.

## IV. Findings and Results

This research employed exploratory and deductive methods to demonstrate the practical benefits of CI/CD and IaC in network automation. The proposed analysis uses reputable academic research and publications, case studies, and logic to justify the research aims.

Here, we have analyzed a cloud configuration model developed by incorporating Terraform's open-source IaC tool into Azure infrastructure configuration platform and validated its benefits over manual configuration processes and associated risks [12].

The study focuses on providing insights into the scope of deploying a scalable and secure Azure infrastructure. Based on the research in [12], an Azure infrastructure model of cloud configuration was analyzed using Terraform in a CI/CD setup. The process involves creating reusable modules for easier scaling and modification, configuring the Terraform backend to store state files in Azure Blob Storage, and integrating Terraform with Azure DevOps for pipeline automation to manage deployments. This includes linting, validation using terraform plan, and applying configurations through terraform apply. Azure policies are implemented to enforce security and compliance rules, such as mandating a minimum VM SKU, while Azure Monitor provides a centralized platform for logging and monitoring infrastructure. The architecture was tested and proven to deliver satisfying scalability, automate deployment processes, enhance security, and minimize human error effectively.

TABLE I
Configuration of Model

| Component | Specification |
|---|---|
| Azure Resource Group | Purpose: Logical organization, lifecycle management, access control, cost tracking. Resource: azurerm_resource_group with name and location variables. |
| App Service Plan | Purpose: Compute resources and scalability. Resource: azurerm_app_service_plan, set for Linux, with defined SKU for pricing and capacity. |
| Linux Web App | Purpose: Managed platform for Java apps. Resource: azurerm_linux_web_app, with configurations for app service, runtime, and connection strings. |
| MySQL Server | Purpose: Managed relational database. Resource: azurerm_mysql_server with admin login, SKU, and security settings, including SSL enforcement. |
| MySQL Database | Purpose: Logical container for data. Resource: azurerm_mysql_database, configurable performance, and supports point-in-time restore. |
| Firewall Rule | Purpose: Network security. Resource: azurerm_mysql_firewall_rule to allow access, with a rule applying to all IP addresses. |
| Terraform Configuration | Provider: azurerm, with state management in Azure storage. Variables: Defined in variables.tf for flexibility. Main Configuration: Uses locals. |

This is best explained as a stepwise approach in the context of an Azure framework: To justify, we provide a feasibility analysis of the three aspects of the model that are considered the key areas of improvement of the CI/CD models over manual network configuration processes [13].

### A. Scalability

In the studied model, the Azure-based modular Terraform infrastructure demonstrated significant scalability. To enhance infrastructure efficiency across development, testing, and production environments, computing resources and infrastructures can dynamically scale up or down based on predefined policies when workload thresholds are reached, and scale back down when demand decreases. Terraform's state is stored in Azure Blob Storage, ensuring 24/7 availability without impacting scaling or deployment performance.

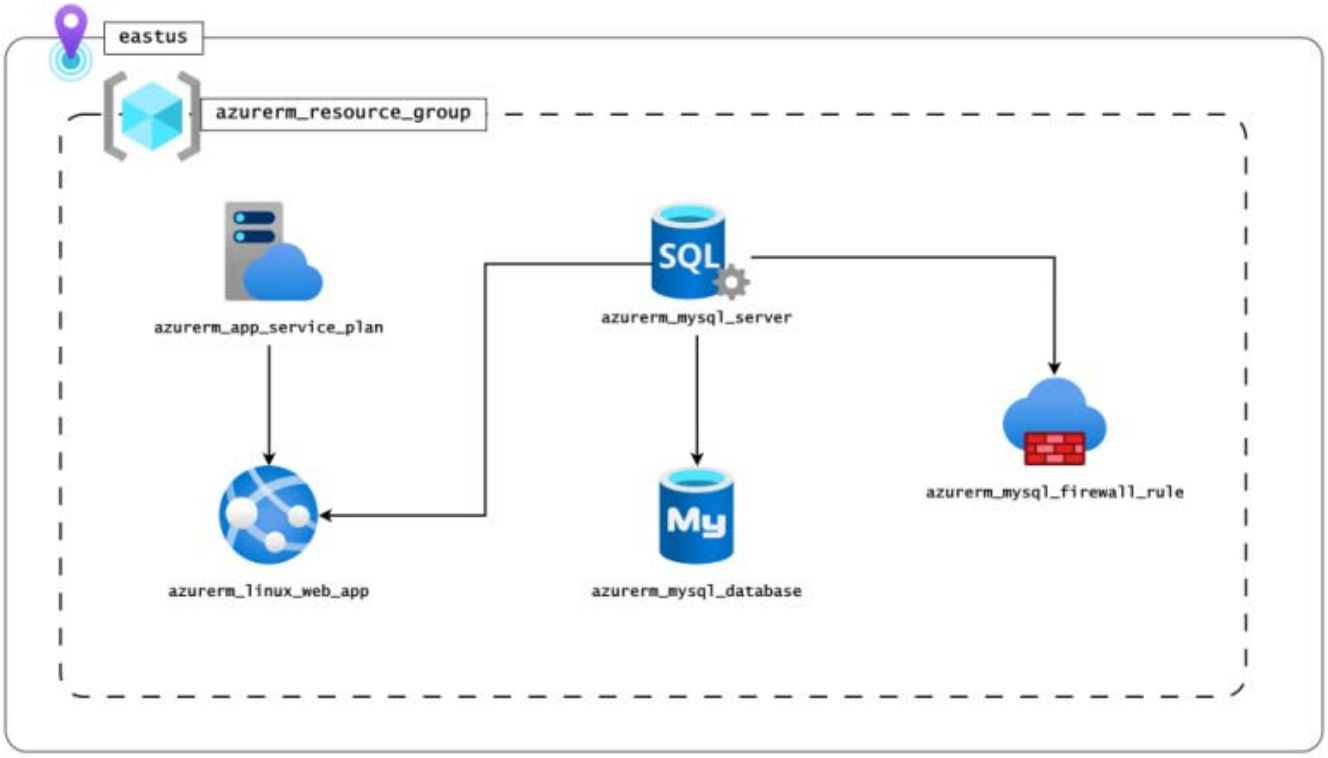


Fig 1. Example IOT Wireless Sensor network architecture benefiting from efficient-routing protocols

### B. Automation

Automation is achievable with Azure IaC using Terraform. By coding infrastructure, deployment and updates are automated, minimizing human involvement. This enables teams to execute Terraform plans and deploy Azure DevOps code changes seamlessly through CI/CD pipelines, enhancing efficiency and consistency. Furthermore, Terraform configurations can integrate with Azure Functions or Logic Apps to enable event-driven or scheduled deployments, such as scaling workloads during business hours. Additionally, Azure Policy and Terraform can automatically enforce compliance standards during deployment, eliminating the need for manual monitoring and ensuring that resources meet corporate requirements.

### C. Security and Risk Management

This is due to its critical role in security, particularly its integration with Azure Active Directory for Azure role-based access control. This integration allows organizations to establish clear access controls, prevent unauthorized resource access, and restrict critical resource permissions [14]. Additionally, Azure Key Vault is used to secure secrets, keys, and certificates, avoiding the inclusion of hard-coded data in Terraform scripts. Automated validation with terraform plan helps detect application misconfigurations during deployment, protecting the state file from corruption caused by concurrent operations, which is especially vital in multi-user environments. Terraform also resolves inconsistencies across environments by standardizing IaC practices, reducing errors from manual configuration, and enhancing operational security.

## V. Conclusion

In conclusion, implementing CI/CD for network configuration using IaC with Terraform in cloud environments offers significant benefits. Role-Based Access Control (RBAC) and Azure Key Vault enhance security by preventing sensitive data exposure and managing access controls. Automated validation with terraform plan safeguards against accidental changes, while Terraform's state locking ensures consistency and prevents state corruption across environments. However, challenges persist, particularly with cloud-based access and the inherent vulnerabilities of open-source tools, which pose security risks. Organizations must proactively address these vulnerabilities to fully leverage the efficiency and reliability of Terraform for deploying and managing their cloud infrastructure.

## REFERENCES

[1] Rismanda Kusumadewi and R. Adrian, "Performance Analysis of Devops Practice Implementation Of CI/CD Using Jenkins," Matics Jurnal Ilmu Komputer dan Teknologi Informasi (Journal of Computer Science and Information Technology), vol. 15, no. 2, pp. 90–95, Oct. 2023, doi: https://doi.org/10.18860/mat.v15i2.17091

[2] J. Fairbanks, A. Tharigonda, and N. U. Eisty, "Analyzing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab," arXiv (Cornell University), Mar. 2023, doi: https://doi.org/10.48550/arxiv.2303.16393

[3] D. Gustavo Cruz, João Rafael Almeida, and José Luís Oliveira, "Open Source Solutions for Vulnerability Assessment: A Comparative Analysis," IEEE Access, vol. 11, pp. 100234–100255, Jan. 2023, doi: https://doi.org/10.1109/access.2023.3315595

[4] Intential, "CI/CD Pipelines for Network Automation & Orchestration with Itential," Itential, Sep. 12, 2024. Available: https://www.itential.com/solutions/network-infrastructure-as-code/. [Accessed: Oct. 30, 2024]

[5] S. Chinamanagonda, "Automating Infrastructure with Infrastructure as Code (IaC)," International Journal of Science and Research (IJSR), vol. 8, no. 11, pp. 123-128, Nov. 2019. Online. Available: www.ijsr.net.

[6] H. Holm, M. Buschle, R. Lagerström, and M. Ekstedt, "Automatic data collection for enterprise architecture models," Software & Systems Modeling, vol. 13, no. 2, pp. 825–841, Jun. 2012, doi: https://doi.org/10.1007/s10270-012-0252-1

[7] Y. Ska and J. P., "A study and analysis of continuous delivery, continuous integration in software development environment," JETIR, vol. 6, no. 9, pp. 1-5, Sep. 2019. Online. Available: www.jetir.org

[8] K. Trantzas et al., "An automated CI/CD process for testing and deployment of Network Applications over 5G infrastructure," IEEE Xplore, Sep. 01, 2021. doi: https://doi.org/10.1109/MeditCom49071.2021.9647628. Available: https://ieeexplore.ieee.org/abstract/document/9647628. Accessed: Mar. 09, 2023.

[9] Rendy Indriyanto and Diki Gita Purnama, "CI/CD Implementation Application Deployment Process Academic Information System (Case Study Of Paramadina University)," Jurnal Indonesia Sosial Teknologi, vol. 4, no. 9, pp. 1503–1516, Sep. 2023, doi: https://doi.org/10.59141/jist.v4i9.729

[10] Erdal Özdoğan, O. Ceran, and Mutlu Tahsin ÜSTÜNDAĞ, "Systematic Analysis of Infrastructure as Code Technologies," Gazi university journal of science part a: engineering and innovation, vol. 10, no. 4, pp. 452–471, Dec. 2023, doi: https://doi.org/10.54287/gujsa.1373305

[11] Z. Pan, W. Shen, X. Wang, Y. Yang, R. Chang, Y. Liu, C. Liu, Y. Liu, and K. Ren, "Ambush from all sides: Understanding security threats in open-source software CI/CD pipelines," arXiv, vol. 2401.17606v1 cs.CR, 31 Jan. 2024.

[12] Sanket Dhole, "Terraform for Azure Cloud: Simplifying Infrastructure as Code (IaC) - Canarys," Canarys, Dec. 27, 2023. Available: https://ecanarys.com/terraform-for-azure-cloud-simplifying-infrastructure-as-code-iac/. Accessed: Oct. 30, 2024.

[13] Always learning, "Azure Terraform Pipeline — DevOps - Always learning - Medium," Medium, Feb. 17, 2024. Available: https://ibrahims.medium.com/azure-terraform-pipeline-devops-b57005a37936. Accessed: Oct. 30, 2024.

[14] Terraform, "Network infrastructure automation," Terraform by HashiCorp, 2024. Available: https://www.terraform.io/use-cases/manage-network-infrastructure. Accessed: Oct. 30, 2024.