

Designing fully-automated AWS multi-account infrastructure using Terraform with governance, policy-as-code, and centralized IAM boundaries

Sneha Palvai, DevOps/AWS Engineer,
Comcast, Philadelphia, USA
snehapalvai09@gmail.com

Abstract—Cloud adoption has significantly increased the demand for scalable, secure, and well-governed infrastructure environments. However, managing multiple cloud accounts, teams, and compliance requirements often leads to configuration drift, inconsistent security policies, and operational complexity when infrastructure is provisioned manually. Infrastructure as Code (IaC) provides a systematic approach to automate infrastructure provisioning [1] and enforce standardized configurations. This paper presents a framework for designing a fully automated AWS multi-account infrastructure using Terraform integrated with governance mechanisms such as policy-as-code and centralized Identity and Access Management (IAM) permission boundaries. The proposed architecture leverages AWS Organizations to establish isolated accounts for workloads, security, and shared services, while governance is enforced through Service Control Policies (SCPs) [4], [6], helping organizations maintain better control over large-scale cloud infrastructures. Terraform modules are used to enable repeatable, scalable, and auditable infrastructure deployment. Additionally, a centralized IAM boundary model ensures least-privilege access while maintaining operational flexibility across teams. The proposed framework improves cloud security posture, enforces governance consistency, and reduces operational overhead, enabling enterprises to deploy and manage AWS infrastructure in a secure, scalable, and automated manner.

Keywords — *Infrastructure as Code (IaC), Amazon Web Services (AWS), Terraform, Multi-Account Architecture, Policy-as-Code, Identity and Access Management (IAM), Cloud Governance, AWS Organizations.*

I. INTRODUCTION

Cloud platforms have reshaped enterprise IT operations by enabling organizations to provision, scale, and manage infrastructure resources dynamically. Public cloud platforms such as Amazon Web Services (AWS) provide scalable and flexible resources that enable enterprises to build and operate modern applications efficiently [3], [10]. However, as organizations expand cloud adoption across multiple teams and environments, managing resources becomes increasingly complex. Without proper governance and automation, cloud environments may face security misconfigurations, inconsistent policies, excessive permissions, and operational inefficiencies

A widely adopted solution to address these challenges is the **multi-account architecture model in AWS** [9]. In this approach, workloads, environments, and organizational units

are separated into individual accounts, improving security isolation, billing management, and governance control. AWS Organizations enables centralized management of these accounts [4] and allows enterprises to enforce governance policies through mechanisms such as **Service Control Policies (SCPs)** [4], [6], helping organizations maintain better control over large-scale cloud infrastructures.



Fig. 1. Infrastructure provisioning workflow using Infrastructure as a Code (IaC) with Terraform.

Despite the benefits of a multi-account architecture, manually provisioning and managing infrastructure across multiple cloud accounts introduces operational overhead and increases the risk of configuration drift. Manual configurations can lead to inconsistencies across environments and reduce deployment reliability. To address these challenges, Infrastructure as Code (IaC) tools such as Terraform enable automated infrastructure provisioning [2] using declarative configuration files, ensuring consistent deployments, version-controlled changes, and automated lifecycle management of cloud resources.

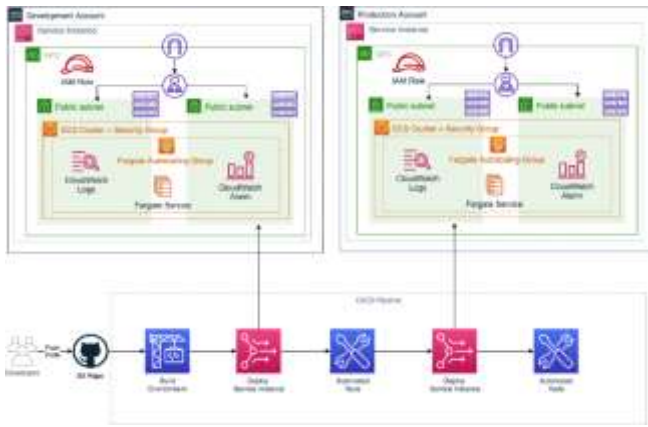


Fig. 2. AWS multi-account deployment architecture integrating CI/CD pipeline automation and containerized services

Beyond infrastructure automation, ensuring consistent security governance across complex cloud environments remains a significant challenge for many organizations. Modern cloud platforms increasingly rely on policy-driven governance approaches, often described as *policy-as-code*, where compliance rules and security policies are defined programmatically and enforced automatically during infrastructure deployment. By integrating policy-as-code frameworks with automated provisioning systems, organizations can ensure that cloud resources are created in accordance with established security policies, regulatory obligations, and internal operational standards.

Another critical aspect of secure cloud infrastructure management is identity and access management. In multi-account cloud environments involving multiple development and operations teams, misconfigured permissions may lead to excessive privileges and potential security vulnerabilities. Implementing centralized Identity and Access Management (IAM) permission boundaries provides a governance model that enforces least-privilege access while still allowing teams to operate independently within predefined security constraints.

This paper proposes a fully automated AWS multi-account infrastructure framework built using Terraform, integrating governance mechanisms such as policy-as-code enforcement and centralized IAM permission boundaries. The proposed architecture aims to improve infrastructure consistency, strengthen security governance, and reduce operational complexity in large-scale cloud environments. By combining infrastructure automation with centralized governance controls, the framework demonstrates how enterprises can design scalable, secure, and compliant AWS infrastructure environments.

The main contributions of this paper are summarized as follows:

- Design of a fully automated AWS multi-account infrastructure framework using Terraform.

- Integration of policy-as-code mechanisms to enforce governance and compliance policies across cloud environments.
- Implementation of centralized IAM permission boundaries to enforce least-privilege access across multiple teams and accounts.
- Demonstration of a scalable architecture that combines automation, governance, and security best practices for enterprise cloud environments.

II. BACKGROUND AND RELATED WORK

Recent research in cloud infrastructure management has focused on **Infrastructure as Code (IaC)**, **cloud governance**, and **policy-driven automation** to improve scalability and security in cloud environments.

Several studies have investigated the use of **Terraform** and **other IaC tools** [1], [2] to automate infrastructure provisioning. These approaches demonstrate that infrastructure automation reduces configuration errors, improves deployment consistency, and enables repeatable cloud deployments. However, most of these studies focus primarily on infrastructure automation and provide limited discussion on enterprise governance mechanisms.

Another research direction explores **policy-as-code frameworks**, where compliance and security policies are enforced automatically during infrastructure deployment. Tools such as Open Policy Agent (OPA) and Terraform Sentinel [7], [8] have been proposed to enforce governance policies in CI/CD pipelines. While these frameworks improve compliance enforcement, they often lack integration with broader multi-account cloud architectures.

Research Area	Primary Focus	Technologies Used	Limitations
Infrastructure as Code (IaC)	Automated provisioning of cloud infrastructure	Terraform, AWS CloudFormation, declarative configurations	Limited governance and security policy enforcement
Policy-as-Code	Security & compliance enforcement	OPA, Terraform Sentinel	Not integrated with architecture
Cloud Governance	Centralized resource control	Access control, auditing	Limited automation
Multi-Account Architecture	Workload isolation	AWS Organizations	Needs Automation & Policy Integration

Table 1. Summary of related work in cloud Infrastructure Governance

In large-scale enterprise environments, **multi-account cloud architectures** are commonly used in enterprise environments [9] to isolate workloads, enforce security boundaries, and manage organizational resources more effectively. Existing governance models rely on centralized policy controls and access management systems to enforce security policies across multiple accounts. However, most existing solutions

are either platform-specific or require significant manual configuration.

Although these approaches address individual aspects of cloud infrastructure management, **few studies provide a unified framework combining automated infrastructure provisioning, policy-driven governance, and centralized identity access controls in multi-account cloud environments [16]**. This paper proposes an integrated architecture that addresses these gaps by combining Terraform automation with policy-as-code governance and centralized IAM boundaries for secure and scalable cloud infrastructure management.

III. ARCHITECTURE

The proposed architecture introduces an automated framework for managing multi-account cloud infrastructure on Amazon Web Services (AWS) [3], [9]. The framework integrates Infrastructure as Code (IaC), policy-driven governance, and centralized identity management to enable scalable, secure, and consistent infrastructure deployment across multiple AWS accounts. By leveraging AWS Organizations, the architecture provides centralized management of accounts while maintaining workload isolation and enforcing governance policies.

The framework is structured into three primary layers: the **Account Structure Layer**, the **Automation and Governance Layer**, and the **Security and Compliance Layer**. These layers collectively enable automated infrastructure deployment while maintaining strong security and operational governance.

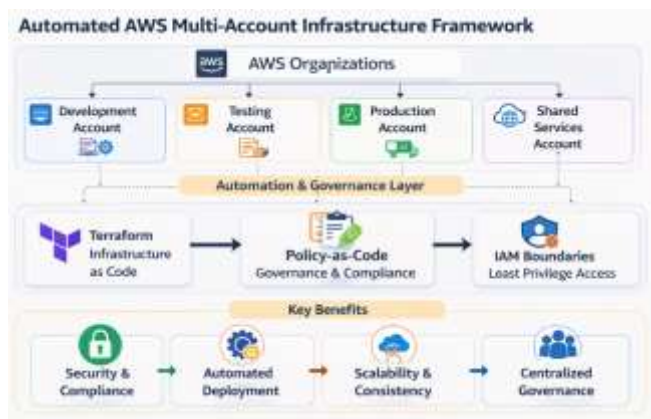


Fig. 3. Automated AWS multi-account governance architecture using Terraform automation and IAM permission boundaries.

A. Account Structure Layer

The account structure layer organizes cloud resources across multiple AWS accounts to provide logical separation of workloads and operational responsibilities. Each account serves a specific purpose within the cloud environment, enabling improved security isolation and management efficiency.

The architecture includes development, testing, production, and shared services accounts. The development account allows development teams to build and test applications and infrastructure configurations in an isolated environment. The testing account provides a controlled environment where infrastructure changes and application deployments can be validated prior to production release. The production account hosts mission-critical workloads and operational services that support live applications and end users.

In addition, the architecture includes a shared services account that provides centralized services such as logging, monitoring, networking components, and security tools. These shared resources can be accessed by other accounts while maintaining centralized governance. This multi-account structure improves security boundaries and enables organizations to manage infrastructure more effectively.

B. Automation and Governance Layer

The automation and governance layer ensures consistent infrastructure deployment and policy enforcement across the cloud environment. Infrastructure provisioning is implemented using Terraform [2], which enables Infrastructure as Code (IaC) through declarative configuration files. This approach allows repeatable deployments, version-controlled infrastructure configurations, and automated resource management across multiple AWS accounts.

To enforce compliance, the framework integrates a policy-as-code model that automatically validates infrastructure configurations before deployment. These policies ensure adherence to organizational security and operational standards, including encryption requirements, restricted public access, standardized resource tagging, and controlled service usage [15]. By integrating policy validation within the deployment pipeline, the architecture maintains governance compliance while supporting automated infrastructure deployment.

C. Security and Compliance Layer

The security and compliance layer provides centralized monitoring and access control across the multi-account cloud environment. This layer ensures consistent enforcement of security policies and continuous monitoring of infrastructure activities to detect potential threats or configuration issues.

Monitoring mechanisms track infrastructure operations across AWS accounts and generate alerts for security events or policy violations. In addition, automated policy enforcement validates infrastructure configurations during deployment to ensure compliance with predefined security standards.

The framework also integrates centralized identity management using AWS Identity and Access Management (IAM) permission boundaries [5]. These permission boundaries enforce least-privilege access policies by

restricting the maximum permissions that users or roles can obtain.

By combining monitoring, governance, and identity management controls, the proposed architecture improves the overall security posture and ensures that the cloud infrastructure remains secure, compliant, and scalable across enterprise environments.

IV. IMPLEMENTATION

The proposed framework was implemented using Terraform-based Infrastructure as Code (IaC) combined with AWS multi-account management. Terraform was used to automate the provisioning of cloud infrastructure resources across multiple AWS accounts while maintaining consistent configurations and governance policies. By defining infrastructure resources using declarative configuration files, Terraform enables repeatable and version-controlled deployments across different cloud environments.

Infrastructure resources such as networking components, IAM roles, logging systems, and monitoring services were defined using reusable Terraform modules. These modules standardize infrastructure deployment [19] across development, testing, and production environments while reducing configuration errors and improving deployment consistency.

To support automated infrastructure delivery, the implementation integrates Terraform with a Continuous Integration and Continuous Deployment (CI/CD) pipeline. Infrastructure code is stored in a version-controlled repository, enabling teams to track changes and maintain infrastructure version history. When changes are committed, the CI/CD pipeline automatically triggers infrastructure validation and deployment workflows.

During pipeline execution, infrastructure definitions undergo validation stages. Terraform plan validation verifies resource configuration changes before deployment, while governance policies are evaluated using policy-as-code mechanisms to ensure compliance with predefined security and compliance rules. These validation steps help prevent insecure or misconfigured resources from being deployed into production environments.

Governance policies enforce organizational security standards such as encryption requirements, restricted public access to services, standardized resource tagging, and controlled service usage. Policy validation within the CI/CD workflow ensures that governance controls are consistently applied during infrastructure provisioning.

Furthermore, AWS Identity and Access Management (IAM) permission boundaries enforce least-privilege access across the infrastructure environment. These boundaries define the maximum permissions that users or roles can obtain, preventing unauthorized privilege escalation while enabling

secure infrastructure management across multiple AWS accounts.



Fig. 4. Implementation workflow of terraform (IaC) with Policy-as-code and IAM boundaries

The overall infrastructure deployment workflow is illustrated in **Figure. 4**, which demonstrates how Terraform integrates with the CI/CD pipeline [12], [17] to automate infrastructure provisioning and governance enforcement.

V. SECURITY MODEL

The proposed framework incorporates a security model designed to protect infrastructure resources, enforce governance policies, and maintain secure operations across the automated AWS multi-account environment. The security model focuses on three core aspects: **identity protection, policy enforcement, and infrastructure monitoring**. These mechanisms operate together to ensure that infrastructure deployments remain compliant with organizational security requirements while supporting automated provisioning.

To secure infrastructure access across multiple accounts, the framework enforces a centralized identity governance model. Access to cloud resources is controlled through predefined roles [5] and permission boundaries that restrict the maximum privileges available to users or services. This model ensures that infrastructure operations follow the **principle of least privilege**, reducing the risk of unauthorized resource access or privilege escalation.

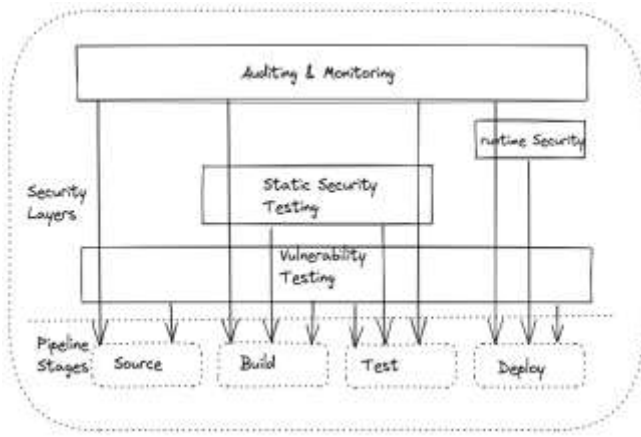


Fig. 5. Integrated security testing and monitoring workflow across infrastructure deployment stages

In addition to identity controls, security policies are applied to protect infrastructure configurations during the deployment lifecycle. Security validation mechanisms are integrated throughout the infrastructure lifecycle, including source validation, build verification, testing, and deployment stages, as illustrated in **Figure 5**. Infrastructure definitions are validated before deployment to ensure compliance with predefined security rules. Policy validation and automated IAM access analyzer [18] are performed within the CI/CD pipeline to detect security violations before infrastructure deployment, as illustrated in **Figure 6**. These controls prevent insecure configurations such as publicly exposed resources, missing encryption settings, or unauthorized service usage from being deployed into the production environment. By validating configurations before provisioning, the framework reduces configuration drift and ensures that deployed infrastructure adheres to security and governance standards.

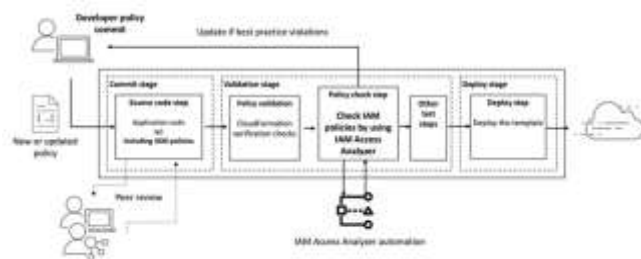


Fig. 6. Implementation Policy-as-code validation workflow with IAM Access Analyzer integrated into the CI/CD deployment pipeline

The security model also integrates continuous monitoring and auditing mechanisms that track infrastructure activity across accounts. Infrastructure changes, policy violations, and configuration updates are recorded to maintain operational visibility. Monitoring systems generate alerts using AWS monitoring services [14] when abnormal behavior or policy violations are detected, allowing security teams to respond quickly to potential risks.

VI. EVALUATION AND DISCUSSION

To evaluate the effectiveness of the proposed framework, the automated AWS multi-account infrastructure model was analyzed in terms of **deployment consistency, governance enforcement, security control, and operational efficiency**. The evaluation focuses on how integrating Terraform-based Infrastructure as Code (IaC), policy-as-code governance, and centralized IAM boundaries improves cloud infrastructure management compared to traditional manual provisioning approaches.

One key improvement observed in the proposed framework is the **reduction of configuration inconsistencies** across multiple cloud environments. Traditional infrastructure deployment methods often rely on manual configurations, which increase the risk of configuration drift and inconsistent security policies. By using Terraform modules and version-controlled infrastructure definitions, the framework ensures consistent resource provisioning across development, testing, and production accounts.

Another important aspect evaluated is **governance enforcement during infrastructure deployment**. The integration of policy-as-code validation within the CI/CD pipeline [7] enables automatic verification of infrastructure configurations before deployment. This mechanism ensures that security policies such as encryption requirements, restricted public access, and resource tagging standards are consistently enforced across all environments. As a result, infrastructure deployments remain compliant with organizational governance policies without requiring manual security reviews.

The framework also improves **identity and access governance** through centralized IAM permission boundaries. By defining maximum permission limits for roles and users, the system reduces the risk of excessive privileges and prevents unauthorized access to critical infrastructure resources. This approach supports secure collaboration across multiple teams while maintaining strong security controls within the cloud environment.

From an operational perspective, the use of automated infrastructure provisioning significantly reduces deployment complexity and manual effort. Infrastructure changes can be tracked through version control systems and automatically validated through CI/CD pipelines, enabling faster and more reliable infrastructure updates. Additionally, automated monitoring and auditing mechanisms improve visibility into infrastructure activities and allow rapid detection of policy violations or security anomalies.

Overall, the evaluation demonstrates that combining **Terraform automation, policy-driven governance, and centralized IAM controls** provides a scalable and secure approach for managing enterprise-level AWS infrastructure. The proposed framework enhances governance consistency, improves security enforcement, and reduces operational overhead in complex cloud environments.

VII. LIMITATIONS

Although the proposed framework demonstrates an effective approach for automating AWS multi-account infrastructure using Terraform and governance mechanisms, certain limitations remain. The implementation primarily focuses on AWS-based environments and may require adaptation to support heterogeneous multi-cloud deployments [13]. Additionally, the framework emphasizes infrastructure automation and governance enforcement but does not fully incorporate advanced automated threat detection or intelligent anomaly monitoring capabilities. The evaluation is also based on architectural analysis rather than large-scale production benchmarking. Addressing these limitations could further strengthen the applicability of the framework for broader enterprise cloud environments.

VIII. FUTURE WORK

While the proposed framework demonstrates the effectiveness of combining Terraform-based Infrastructure as Code (IaC), policy-as-code governance, and centralized IAM permission boundaries for automated AWS multi-account environments, several areas can be explored to further enhance the framework.

Future research can focus on extending the framework with advanced security automation and intelligent monitoring mechanisms. Integrating machine learning-based anomaly detection systems could help identify unusual infrastructure behavior, configuration anomalies, or potential security threats in real time. Such capabilities would enhance the ability of the framework to proactively detect and mitigate security risks in complex cloud environments.

Another potential direction is the integration of advanced policy engines and automated compliance validation tools. While the current framework uses policy-as-code mechanisms for infrastructure validation, future implementations could incorporate dynamic compliance verification for regulatory standards such as SOC 2, HIPAA, or ISO 27001 [20]. This would allow organizations to continuously evaluate infrastructure compliance with industry-specific security requirements.

Additionally, the framework can be extended to support multi-cloud infrastructure environments. Many modern enterprises deploy workloads across multiple cloud platforms such as AWS, Microsoft Azure, and Google Cloud Platform. Extending the framework to manage multi-cloud infrastructure using a unified governance model would improve infrastructure portability and reduce vendor dependency.

Finally, future work may explore the integration of automated cost optimization and resource management mechanisms. Intelligent cost monitoring tools and predictive analytics could help organizations optimize cloud resource usage and reduce operational expenses while maintaining governance and security controls. The potential extensions of the

proposed framework, including intelligent monitoring, advanced policy automation, multi-cloud governance, and automated cost optimization, are conceptually illustrated in Figure 7.

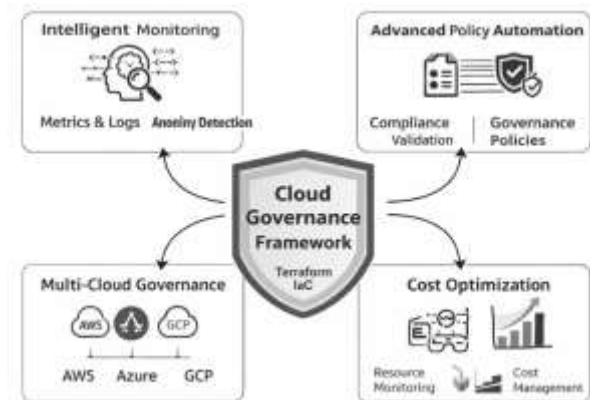


Fig. 7. Future cloud governance framework with monitoring, policy automation and multi-cloud support

In addition, future implementations may incorporate automated infrastructure self-healing capabilities that detect and remediate configuration issues without manual intervention. Integrating advanced observability platforms could further enhance visibility into infrastructure performance, security posture, and operational metrics across multiple cloud environments. These enhancements would strengthen the framework's ability to support large-scale enterprise deployments while maintaining high levels of reliability, security, and operational efficiency.

IX. CONCLUSION

This paper presented a framework for designing a fully automated AWS multi-account infrastructure architecture using Terraform, integrated with governance mechanisms such as policy-as-code enforcement and centralized Identity and Access Management (IAM) permission boundaries. The proposed framework addresses key challenges associated with large-scale cloud environments, including configuration inconsistencies, governance enforcement difficulties, and security risks caused by excessive access permissions.

By leveraging Infrastructure as Code (IaC), the framework enables automated and repeatable infrastructure deployments across multiple AWS accounts. The integration of policy-as-code validation within CI/CD pipelines ensures that infrastructure configurations comply with predefined security and governance standards before deployment. In addition, centralized IAM permission boundaries strengthen access control mechanisms and enforce least-privilege policies across development and operations teams.

The evaluation demonstrates that combining automation, governance, and centralized identity management improves

infrastructure consistency, enhances security enforcement, and reduces operational complexity in enterprise cloud environments. Overall, the proposed framework provides organizations with a scalable and secure infrastructure management model aligned with modern DevOps and cloud governance practices.

REFERENCES

- [1] M. Fowler, "Infrastructure as Code: Managing Servers in the Cloud," O'Reilly Media, 2016.
- [2] HashiCorp, "Terraform Documentation," HashiCorp Inc., 2024. [Online]. Available: <https://developer.hashicorp.com/terraform>.
- [3] Amazon Web Services, "AWS Well-Architected Framework," AWS Whitepaper, 2023.
- [4] Amazon Web Services, "AWS Organizations User Guide," AWS Documentation, 2024.
- [5] Amazon Web Services, "AWS Identity and Access Management (IAM) Best Practices," AWS Security Documentation, 2023.
- [6] Amazon Web Services, "Service Control Policies in AWS Organizations," AWS Documentation, 2023.
- [7] Open Policy Agent (OPA), "Policy-as-Code for Cloud Infrastructure," CNCF Documentation, 2023.
- [8] HashiCorp, "Terraform Sentinel Policy as Code Framework," HashiCorp Documentation, 2023.
- [9] Amazon Web Services, "AWS Multi-Account Strategy Best Practices," AWS Whitepaper, 2022.
- [10] S. Newman, "Building Microservices: Designing Fine-Grained Systems," O'Reilly Media, 2015.
- [11] Amazon Web Services, "AWS Cloud Security Best Practices," AWS Security Whitepaper, 2022.
- [12] J. Humble and D. Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation," Addison-Wesley, 2010.
- [13] Google Cloud, "Site Reliability Engineering: How Google Runs Production Systems," O'Reilly Media, 2016.
- [14] Amazon Web Services, "AWS CloudTrail and CloudWatch Monitoring," AWS Documentation, 2023.
- [15] NIST, "Security and Privacy Controls for Information Systems," NIST Special Publication 800-53, 2020.
- [16] M. Rahman et al., "Policy-based Cloud Governance Framework for Enterprise Cloud Systems," IEEE Cloud Computing, 2021.
- [17] B. Fitzgerald and K. Stol, "Continuous Software Engineering: A Roadmap and Agenda," Journal of Systems and Software, 2017.
- [18] Amazon Web Services, "IAM Access Analyzer Documentation," AWS Security Documentation, 2023.
- [19] HashiCorp, "Terraform Modules Best Practices," HashiCorp Developer Guide, 2023.
- [20] ISO/IEC 27001, "Information Security Management Systems Requirements," International Organization for Standardization, 2013.