National College of Ireland

Cloud-Agnostic Self-Correcting Compliance-as-Code Framework Using AI Agents, OPA, and Terraform for HIPAA/GDPR/PCI-DSS Enforcement

MSc Research Project
Cloud Computing

Vikram Kariyavula
Student ID: 24112682

School of Computing
National College of Ireland

Supervisor: Shivani Jaswal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vikram Kariyavula
Student ID:	24112682
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Shivani Jaswal
Submission Due Date:	28th January 2026
Project Title:	Cloud-Agnostic Self-Correcting Compliance-as-Code Framework Using AI Agents, OPA, and Terraform for HIPAA/GDPR/PCI-DSS Enforcement
Word Count:	9739
Page Count:	25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	28th January 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Cloud-Agnostic Self-Correcting Compliance-as-Code Framework Using AI Agents, OPA, and Terraform for HIPAA/GDPR/PCI-DSS Enforcement

Vikram Kariyavula
24112682

Abstract

Cloud infrastructure compliance to regulatory frameworks like HIPAA, GDPR and PCI-DSS is a major challenge for organizations implementing Infrastructure-as-Code practices, where compliance verification is manually verified which is increasingly impractical at scale. This research explored how self-corrective cloud-agnostic AI agents using Agent Development Kit, Open Policy Agent, and Terraform can be used to automate regulatory compliance enforcement on AWS infrastructure and increase security posture. An integrated system was built, which combined an OPA validation pipeline for the policy-based compliance detection and an ADK-powered compliance-healing agent that is able to automate the remediation of infrastructure violations. Comprehensive experimental evaluation through five experiments showed that the system proved to have 97.4% policy detection accuracy, 85% regulatory coverage through HIPAA achieving perfect compliance and GDPR at 80% and PCI-DSS at 80%, and 62.8% time reduction compared to manual resolution. The self-healing agent was able to remediate 80% self-sufficiently, with 100% success rates for routine configurations such as S3 encryption and EC2 tagging, 94.6% cost savings with $0.26 vs. $4.80 per violation. A 15-day monitoring of the operational performance demonstrated positive learning curves with automated fix success rates increasing from 72.2% to 90.0%. The results confirm that the presence of declarative policy languages and remediation based on the LLM provides a feasible model of sustained compliance automation in cloud-native environments, which reduces the economic and operational viability of production deployment.

Keywords: Cloud Compliance Automation, Compliance-as-Code, Infrastructure-as-Code, Open Policy Agent, Agent Development Kit, Compliance-Healing Systems

1 Introduction

The strong growth rate of the cloud computing has significantly changed the way organizations plan, implement, and protect their digital services. The Flexera 2025 State of the Cloud Report indicated that about 94 percent of businesses have embraced the use of cloud applications that run in either a public, private, or hybrid cloud scenario [1]. Cloud computing services providers like Amazon Web Services (AWS), Microsoft Azure,

[1] `https://info.flexera.com/CM-REPORT-State-of-the-Cloud?lead_source=Organic`

and Google Cloud Platform (GCP) have played a significant role in this transformation, providing flexibility, scalability, and cost-efficiency to various industries (healthcare, e-commerce, and finance) through the services they provide.

Nonetheless, such wide usage has raised immense compliance and governance issues. The industries dealing with sensitive information, especially healthcare and finance, must meet high levels of regulatory standards such as the Health Insurance Portability and Accountability Act (HIPAA), the General Data Protection Regulation (GDPR), and the Payment Card Industry Data Security Standard (PCI-DSS) (Seetharamarao; 2023). Such frameworks require stringent regulations on access management, audit records, encryption, data storage and data breach response to achieve data security and privacy.

The conventional compliance assurance practices, which are typically manual audit-based and sticky configuration-based, are no longer suitable when the workloads are short lived and the infrastructure is defined and operated with Infrastructure-as-Code (IaC) tools. This kind of quick change results in configuration drift where systems will be in the erroneous compliant state and this will raise chances of security breach and regulatory violations.

Compliance checking has been partially automated by the creation of Policy-as-Code (PaC) systems, like Open Policy Agent (OPA), and IaC toolsing, such as Terraform. Nevertheless, they are forever stagnant, responsive, and unintelligent tools that can be run on predetermined rules only, and cannot cross-cloud operational, nor change autonomously with new threats or regulatory changes.

The study presents the use of cloud-agnostic and AI-driven compliance automation platform as the solution to these limitations. The system suggested incorporates intelligent agents with the ability to monitor and analyze in real time and adjust policies. Particularly, the intelligent agents will: Determine compliance drift and detect anomalies, Detect changing trends of non-compliance with AI methods, Automate OPA/Rego policies according to infrastructure behavior observed, Automated remediations through Terraform.

Such innovation becomes more needed due to the increasing price of non-compliance. More than €4.2 billion of fines have occurred worldwide since the enforcement of GDPR [2]. The average expenditure cost per year by organizations in the U.S. healthcare sector on HIPAA compliance is $8.2 million (Thummarakoti; 2025). These statistics underscore the cost of manual compliance management in terms of finances, particularly to the small and medium-sized enterprises (SMEs) that do not have committed compliance resources.

This research will help cut down human intervention, regulatory risk, and financial costs, which are associated with such an error, by a considerable margin, by designing a self-correcting, AI-driven compliance framework that will help create a safer and more sustainable cloud ecosystem.

1.1 Aim of the study

The main aim of the research is the creation of a smart, cloud-neutral compliance automation system, where AI-based adaptive policy enforcement is provided through the use of Open Policy Agent (OPA) and Terraform. The framework aims to allow proactive, self-organizing and scalable compliance management in multi-cloud environments and, therefore, minimize configuration drift and manual control.

[2] https://www.privacyaffairs.com/gdpr-fines/

1.2 Research Question

This thesis is driven by the following central research question: How to enforce the HIPAA, GDPR, and PCI-DSS regulatory compliances on AWS cloud using self-corrective cloud agnostic AI agents with Agent Development Kit (ADK), Open Policy Agent (OPA) and terraform to improve security posture in agentic cloud compliance?

1.3 Objectives of the Research

To fulfill the given aim, the following research objectives have been established:

- Explore the drawbacks of current compliance automation systems based on Policy-as-Code (PaC) and Infrastructure-as-Code (IaC) in multi-clouds.
- develop an artificial intelligence-based architecture that can identify compliance drift and dynamic adaptation of policy logic in a variety of cloud platforms.
- Introduce smart agents that will use telemetry data, audit trails, policy outcomes, etc. to initiate autonomous remedies in real-time.
- Combine the solution with OPA and Terraform to make changes to the policy and infrastructure on demand.
- Measure the accuracy, adaptability, performance and compliance efficiency of the proposed framework via experimental simulation.

1.4 Outline of the Report

The rest of this thesis will follow the following structure: section 2 provides an overview of the currently available literature on early framework on cloud compliance, Policy-as-Code, Infrastructure-as-Code, and AI-based compliance automation with a view to discovering research gaps. The section 3 describes the research design, data collection, and evaluation methods. section 4 shows the architecture and implementation of the suggested AI-based compliance framework. The sections 5 and 6 present the analysis and discussion of the results of the experiment and section 7, which concludes the study and proposes the directions of future research.

2 Related Work

2.1 Early Frameworks for Cloud Compliance and Their Limitations

The basic compliance model of cloud computing environments and suggested a complete compliance engine architecture consisting of metadata extractors, execution units, and compliance databases is developed by (Filepp et al.; 2018). Although this framework exhibited modularity and operational scalability, it had inherent limitations due to its deployment of fixed rule evaluation processes and limited scalability to a heterogeneous cloud vendors, thus limiting its long-term scalability. A framework by (Agarwal et al.; 2022) is called Compliance-as-Code for Cybersecurity Automation in Hybrid Cloud and is offered to automate cybersecurity operations in regulated on-premises, private, and

public cloud environments, namely, the standards of PCI compliance requirements and ISO compliance requirements. The paper illustrates how compliance management was previously a manual system that used documents, whereas now it is an automated policy implementation process. Also, (Nama and Alalawi; 2023) discussed The Adoption of Automated Building Code Compliance Checking Systems which, though specifically related to the Architecture, Engineering and Construction industry, nonetheless offers useful guidance on automated compliance checking systems that can address complex semantic and syntactic hierarchy of requirements- methodologies that can be adapted to cloud compliance implementation.

Equally, (Falazi et al.; 2022) discussed the compliance rules in Infrastructure-as-Code (IaC) workflows. Their study shows that there is a need to incorporate compliance requirements directly into the application architecture definitions so that development teams can use the traditional software engineering tools to do compliance testing. Nonetheless, their model is still highly attached to particular IaC toolchains, and does not have the ability to self-correct to support a dynamic regulatory adaptation.

Together, these research show that the previous compliance frameworks laid the basis of scalable architectures but did not sufficiently face the challenges of regulatory volatility and cloud-provider independence. The gap in the current research has led to the need to develop cloud-agnostic, AI-based solutions that can dynamically adjust to the heterogeneous infrastructure environment and maintain continuous compliance with the changing regulatory environments.

2.2 Focus on Multi-Cloud and Policy Enforcement Gaps

The notion of hybrid multi-cloud compliance is discussed by (Rompicharla and P. V; 2020) by proposing the model that unites policy enforcers in various cloud settings. Their solution offers an abstraction layer on top of cloud specific APIs, which can be used to enforce compliance in a unified way regardless of the underlying cloud platforms.

Similarly, (Mallisetty et al.; 2023) reveal that the main issue of cloud governance in modern cloud deployments lies in the fact that organizations cannot achieve coherent visibility in terms of a single strategy (both of private, hybrid, and public cloud deployments). Their comparison shows that the traditional security architectures do not have the abstraction layers to provide similar compliance policies across heterogeneous cloud platforms, especially when it comes to supporting the requirements of data sovereignty that are stipulated by regulations like GDPR and jurisdictional demands. According to the study, 78 percent of organizations that were in the multi-cloud setting reported having at least one security incident, and 41 percent of the incidents were due to misconfigured storage policies, which is a clear indication of policy enforcement failures. Additional analysis by (Choudhary et al.; 2023) ensure that data Protection proves the idea that multi-cloud systems make the process of detecting insider threats and managing access control even more complicated. They have found that there are some important vulnerabilities in their studies as a result of a lack of consistency in identity and access management (IAM) policies in a move between cloud providers. The lack of policy definition languages in the standards allows enforcement gaps where the security controls applied to one platform cannot be easily ported into another one, requiring manual intervention which brings in human error and compliance drift. The IEEE Standard 2302 on federated cloud computing [3] has been applied to the issue of cross-cloud interoperability

[3]https://standards.ieee.org/beyond-standards/new-ieee-standard-advances-federated-cloud-

by defining the governance frameworks to apply in registration, geo-independence, and audit processes. Nevertheless, researchers of the actual multi-cloud deployments report about their practical implementations being scarce. The standard offers architectural advice, without solving the underlying problem of the enforcement of runtime policy in heterogeneous toolchains of infrastructure-as-code (IaC) and in proprietary cloud service interfaces.

Moreover, recent efforts of (Gupta and Scholar II; 2025) in the development of multi-cloud governance protocols highlight the necessity of automated policy enforcement tools that would be able to work with AWS, with Azure and Google Cloud Platform at the same time. Existing solutions need policy translations on a provider-by-provider basis, incurring maintenance costs and imposing time delays in propagating policies. These problems are worsened in controlled fields where compliance issues require real time compliance and constant audit facilities. The current literature has shown a collective representation of the fact that though application of multi-clouds is improving at a faster rate, the enforcement policies have not evolved at the same pace. There are still critical gaps in such areas as: policy translation and portability across providers, real-time compliance drift, automated remediation in heterogeneous environments, and provider-independent audit trail. These gaps require novel strategies to take advantage of AI-enabled policies interpretation and enforcement coupled with cloud-agnostic abstraction frameworks able to run similarly across different cloud environments.

2.3 Compliance Automation

Taking a Policy-as-Code (PaC) paradigm, such as the Open Policy Agent (OPA), a step will help enhance compliance enforcement in infrastructure layers (especially in systems that use GitOps), as emphasized by (Yanagawa et al.; 2024). Nevertheless, they are based on predetermined points of validation and can only work in homogeneous environments, which cannot be adapted and require AI to make decisions. Likewise, (Rahman et al.; 2019) discuss the shortcomings of the Infrastructure-as-Code (IaC) tools such as Terraform and CloudFormation, which will not enforce dynamic policy checks or automatically remediate. Although these strategies are effective in fixed, small-scope systems, they cannot satisfy regulatory demands, like HIPAA and GDPR, in complex, multi-cloud systems.

In addition, (Yau et al.; 2024) and (Paul et al.; 2024) also investigate OPA-based compliance automation through a regulatory perspective but their applications are limited to the specific cloud providers, making them less generalized. Together, these research papers indicate the presence of a severe research gap - the necessity of a self-controlling, regulation-conscious compliance system that could ensure the presence of sustained, intelligent compliance in heterogeneous cloud environments.

The proposed study aims to address them through the combination of the PaC model and AI-powered agents that could proactively monitor and provide intelligent feedback and autonomously correct standard compliance failures in various cloud infrastructures, thereby providing a consistent enforcement of such standards as HIPAA, GDPR and PCI-DSS.

---

computing/

2.4 Adaptive and Predictive Compliance Models

Monitor-Analyse-Plan-Execute (MAPE) loop towards the cloud compliance variability is proposed by (García-Galán et al.; 2016) at the runtime and suggest adaptive reconfiguration as one of the means to respond to the violations. This is one of the earliest publications to note the need of versatile systems, but does not go further to describe agentic architecture or enforcement vehicle. Nearer than that, (Rinderle-Ma et al.; 2023) examines recent works of the field of predictive compliance observation, suggesting that process-aware prediction can be used alongside compliance rules in order to predict non-conformance before it takes place. It does offer a viable method of dealing with compliance in business process even though it is not used at the infrastructure level in cloud. In both works, the enforcement of multi-cloud policy and agent-based self-adjustment as related to IaC are not addressed. Nevertheless, they determine the manner in which proposed research can apply layer of AI-agents, which offer a conceptual grounding of run-time and predictive compliance functionality.

2.5 AI-Driven Policy Adaptation and Drift Detection

The dynamic characteristics of cloud infrastructure and the changing regulatory environments demand intelligent infrastructure that is capable of identifying and responding to the drift of the policies in real time. Static compliance frameworks are not able to consider the changes in infrastructure over time in a way that they introduced vulnerabilities through configuration drifts, undetected compliance violations can exist until an audit cycle.

The broad-based roadmap on concept drift adaptation by (Mahdi et al.; 2024) provides principles to rely on when determining the presence of distributional changes in streaming data settings. This work deals with sudden drift, gradual drift, and repeated patterns, which can be easily applied to infrastructure policy enforcement since cloud resource configurations are continually changing. The study highlights the importance of the drift adaptation that involves not only detection systems but also interpretation systems that can tell the difference between a benign change in operations and the one that violates the policy.

(Xu et al.; 2024) suggest a Concept Drift Detection, Interpretation, and Adaptation (CDDIA) framework of the IoT based anomaly detection with direct analogies to the cloud based compliance monitoring. They employ ensemble learning and use statistical hypothesis testing to detect deviations in the behavior of the baselines. In the context of compliance, these frameworks may allow autonomous identification of non-conformance infrastructure configurations, which are not based on approved policy states, and initiate automated remediation processes. The paper shows that the detection of drift is 94.7% accurate in heterogeneous IoT environments, implying the same can be achieved in multi-cloud compliance cases.

The article by (Manias et al.; 2022) proposes real-time monitoring system of 5G core networks to track the intelligent model performance degradation under the influence of environmental changes. Their architecture uses continuous validation pipelines where the predicted system behavior is compared with the actual system behavior and model retraining will automatically be triggered if the performance thresholds are violated. This paradigm is easily mapped to compliance-as-code models where policy enforcement models need to scale to infrastructure modifications, API changes, and regulatory changes, and attain less than 100ms decision latency that business compliance systems can rely on.

The proposed research is based on filling this gap by combining multi-source telemetry, agentic learning models, and modular policy-as-code enforcement mechanisms that will help to accommodate a wide variety of regulatory settings.

2.6 Agentic AI Compliance and Governance Risks

The use of autonomous AI agents during compliance enforcement opens up new security and governance issues that are not unique to the traditional software systems. The agentic AI systems, exhibiting the ability to take independent decisions and engage with external parties and alter their behavior based on the feedback gathered by the environment, introduce peculiar risk vectors, which can only be addressed through specially designed governance models. Recent studies on the topic of cybersecurity point to the growing threat environment of AI systems. (Nageab et al.; 2024) states that there is a 135% growth in AI-enabled social engineering attacks since the release of large language models to the general public, which can be attributed to the fact that malicious actors are quickly using AI capabilities to fight against them. Their contribution points to the key points of weakness in AI systems such as prompt injection attacks, model extraction methods and adversarial manipulation of training data. In the context of compliance automation, these vulnerabilities would allow attackers to bypass the policy enforcement decision-making and make AI agents either blindly overlook the violations or mark valid configurations as non-compliant, corrupting the whole compliance system. Similarly, (Lozhnikov and Zhumazhanova; 2022) have systematically classified the information security risks inherent in the AI based systems. Their investigation underlines that AI technologies that develop at a rapid pace present new threat surfaces that were unknown before, especially in data preparation, model training and DevOps integration pipelines. The research discloses that the implementation of AI systems without thorough security evaluation is associated with some risks such as data poisoning wherein adversaries enter malicious samples during training to establish a backdoor in compliance models, and model inversion attacks that may expose sensitive regulatory data that is embedded in the policy implementation algorithms. In the case of compliance frameworks that work with HIPAA, GDPR and PCI-DSS data, these vulnerabilities might translate into devastating regulatory violation and fines. Hence, governance is essential as the responsibility of compliance is handed to the autonomous agents.

Also, (Gambrell; 2025) says that ethical agentic AI is required and records how New Jersey AI Task Force brought AI and collective intelligence together to combine policy. The adoption of such technologies as Policy Synth allowed improving the inclusiveness and evidence-based decision-making rates but also developed concerns connected with the risk of algorithm bias and overreliance on machine-based conclusions. All these investigations precondition the suggested research to pursue the proactive refinement of policies through the contextual awareness and regulatory drift.

3 Methodology

This research methodology will revolve around designing and testing a self-remediating compliance enforcement mechanism in cloud environments with the help of Infrastructure as Code (IaC). The study adheres to a constructive methodology where a new artefact is developed to overcome the shortcomings of current compliance enforcement procedures. The system is a combination of deterministic policy verification and probabilistic agentic

remediation to minimise regulatory drift, human error, and to speed up the correction processes. This chapter consists of the methodological plan of designing, implementing, and assessing the proposed system.

3.1 Research Methods

3.1.1 Overview of compliance standards

The proposed research suggests the application of these data protection standards: HIPAA, GDPR, and PCI-DSS each signifying a different aspect of regulation that is subject to legal regulations and technical requirements.

- **HIPAA (Health Insurance Portability and Accountability Act, 1996):**

  An American law that is put in practice by the U.S Department of Health and Human Services (HHS) and outlines national standards when it comes to protecting Protected Health Information (PHI). The utilization of PHI and the disclosure of information is regulated by the HIPAA Privacy Rule (45 CFR Part 160 and Subparts A and E of Part 164), and administrative, physical, and technical security actions to guarantee confidentiality, integrity, and availability of health data are established by the HIPAA Security Rule [4]. Main ones are access control (§164.312(a)), audit logging (§64.312(b)), encryption and decryption (§164.312(e)) and breach notification (§164.400-414)

- **GDPR (General Data Protection Regulation (EU) 2016/679):**

  The regulation is an EU measure that safeguards the privacy and personal information of the people in the European Union and the European Economic Area (EEA). The principles of GDPR are lawfulness, fairness and transparency (Article 5), minimum data (Article 5(1)(c)) and accountability (Article 24). It provides personal rights which include the right to access (Article 15), the right to erasure (Article 17) [5] and the right to data portability (Article 20). To ensure that their cloud operations are compliant, organizations have to adopt privacy by design and by default (Article 25) and keep records of the processing activities (Article 30).

- **PCI-DSS (Payment Card Industry Data Security Standard, 2022):**

  The payment card industry security standards council (PCI SSC) has came up with a globally accepted standard that may be applied by entities that store, transmit or process cardholder data (CHD). Pci-DSS has 12 important requirements such as high level of network security controls, system configuration security, data encryption on storage and transfer, stringent access control, constant monitoring and regular security testing. Collectively, these requirements will make sure that organisations have strong security posture and adhere to uniform policies and procedures on protection of sensitive payment information [6]. All these requirements are aimed at stopping the data breach and enhancing security in the payment ecosystem.

[4] https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-C?toc=1

[5] https://gdprhub.eu/Article_24_GDPR

[6] https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf

Collectively, these frameworks give rise to a full compliance standard to defend sensitive healthcare, personal and financial information. The proposed system ensures that all dynamic, multi-cloud environments, like Amazon Web Services (AWS), have a consistent, repeatable, and auditing compliance by encoding their own specific regulatory controls into Open Policy Agent (OPA) policies and enforcing them with automatic and repeatable Terraform processes.

3.1.2 Applicability to Cloud Resources

The proposed compliance enforcement system will be implemented in a wide range of AWS cloud resources which are often deployed using Terraform. OPA policy-as-code framework is customised and applied to resources like Amazon S3 buckets, Amazon RDS instances, Amazon EC2 security groups, AWS IAM roles and policies, Amazon EKS clusters, and VPC networking resources. All these resources will be linked to regulatory controls, which can be determined deterministically by Rego rules. An example of this is that encryption policies are implemented on S3 and RDS resources, ingress controls are checked on security groups, and IAM policies are examined within the context of least-privilege. The agentic correction subsystem deciphers the violations of the following services of AWS and generates necessary remedial measures, which may be turning on server-side encryption, changing ingress CIDR ranges, implementing key rotation policies, or changing EBS volume settings. Through its ability to support a large variety of AWS resources, the system illustrates the fact that it can be used in a wide variety of real-world cloud settings in which the aspect of compliance needs to be regularly upheld across the heterogeneous elements of the infrastructure.

3.1.3 Infrastructure Scope in Action

The system is operationally in the entire lifecycle of Terraform-managed AWS infrastructure so that compliance enforced as early as infrastructure code is committed to its repository. Automatically activated is the system with the help of AWSCode pipeline, which coordinates the work of Terraform initiation, plan generation, validation, and remediation. One of the aspects that the system can test during execution is configurations within the AWS environment (network architecture (VPCs, subnets, route tables, NAT gateways), compute workloads (EC2 instances, autoscaling groups, Lambda functions), storage systems (S3, EBS, EFS), and identity and access structures (IAM users, roles, policies). Any non-compliant configurations detected during the Terraform plan are automatically checked and fixed prior to deployment so that the final AWS infrastructure would be in compliance with regulatory standards without human intervention. This end to end integration ensures that compliance enforcement is a continuous and automatic process, it offers a controlled, repeatable, and audit-ready workflow with compliance with AWS best practises and DevSecOps values.

3.2 System Design Methodology

The system is built as a two-part solution as a part of the cloud-native DevSecOps pipeline. The design is focused on automation, accuracy, explainability, and integration with the existing IaC processes.

- **Compliance Validation Component**

The company must undergo compliance validation to ensure that all necessary activities are completed.The former is a deterministic policy validation pipeline constructed with the help of Open Policy Agent (OPA) and the Rego language. The regulatory requirements are then converted into machine executable policies which assess the Terraform execution plans before being deployed. In the event that non-compliant configurations are identified, OPA generates structured reports of violations in the form of a JSON with details of the resources affected and which specific rules it violated. It is a validation part that is active as a prerequisite step of AWS CodePipeline, and it makes compliance cheques occur automatically when a code is submitted.

- **Agentic Component of Correction** The second element is an independent remediation subsystem that is developed based on the Google Agent Development Kit (ADK). The given subsystem relies on large language models to interpret the violation reports and suggest specific changes to the underlying IaC templates. The remediation is conducted in an iterative process that comprises of modification, syntax validation, re-evaluation as well as rollback handling where it is necessary. The system works based on structured tool interfaces, which limit commands to access verifiable file paths and ensure no mistake or invalid outputs are made.

- **Policy-as-Code Development Approach** Rego policies are written as a result of a systematic mapping of regulatory requirements on Terraform resource configurations. The steps in this process are to analyse the regulatory documents and isolate enforceable technical controls and encode them OPA rules. There are matching compliant and noncompliant Terraform templates generated to test every policy. The policies are optimised on a regular basis to reduce false alarms and enhance their accuracy in a wide range of infrastructure conditions.

3.3 Evaluation Methodology

The evaluation methodology will involve five overall experiments aimed at testing the integrated OPA validation pipeline and self-healing agent system that is based on ADK on various levels: detection, remediation, human-agent cooperation, long-term integration, and the enforcement of multi-regulatory compliance.

Accuracy of OPA Policy Violation Detection tests the accuracy of the OPA validation pipeline with respect to nine compliance domains in 20 Terraform configurations that depict realistically modeled AWS infrastructure. The configurations are loaded by the OPA policy engine and HIPAA, GDPR, and PCI-DSS requirements are written as Rego policies. Detection performance is determined based on precision, recall and F1-score and manual verification is used to determine ground truth per compliance domain.

Self-Healing Agent Remediation Success Rate determines the ability of the ADK-based agent to automatically remediate policy violations identified in Experiment 1. The 20 configurations undergo the self-healing workflow, in which the Loop Agent repeatedly performs fixes with the help of specialized tools and Gemini 2.5 Flash LLM instructions. The success rates are classified into complete success, partial success or failure and the mean time to remediation and the number of iterations is trailed by each type of violation.

Comparative Analysis of Manual vs. Automated Resolution is an experiment that measures trade-offs between the human expert remediation and the automated agent

correction. A single, five-years Terraform experienced infrastructure engineer manually solves the same 20 compliance cases as the automated system is running them in parallel.

Long-Term Pipeline Integration and Error Convergence Analysis focuses on the performance of the system during 15 days of its work within a simulated CI/CD environment. There is test repository of 10 Terraform modules, which are subjected to 42 commits that add features, change configuration, and provide security changes. It monitors pipeline pass rates, automated fix success rates, violation recurrence patterns, frequency of manual intervention and cost measures.

Regulatory Compliance Enforcement Analysis provides a direct answer to the research question as it analyzes the compliance enforcement of the multi-regulatory frameworks (HIPAA, GDPR, and PCI-DSS) of compliance. There are twenty changes in infrastructure that are classified as HIPAA-only, GDPR-only, PCI-DSS-only and multi-regulatory.

4 Design Specification

4.1 System Architecture

The structure of the system is built as a composite structure of compliance-enforcement and autonomous remediation patterns, which consists of two main subsystems: the Validation Pipeline and the Self-Healing Agent as illustrated in Figure 1. These subsystems run in a feedback loop where infrastructure-as-code artefacts are checked against regulatory and organisational policy and where needed automatically fixed. Scalability, reproducibility, and consistency of enforcement are achieved through the implementation of the architecture on cloud-native services.

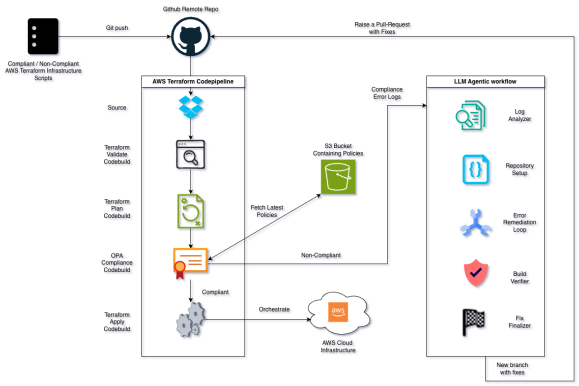


Figure 1: Proposed System Architecture

The Validation Pipeline is deployed with the help of AWS CodePipeline, and it is the first checkpoint, where the Terraform configuration is analysed (both, statically and semantically). Code submitted to the AWS Codecommit or the GitHub is automatically pulled into the pipeline. A CodeBuild environment is deployed into the pipeline that is configured with all the necessary validation tools, such as Open Policy Agent (OPA) and Terraform. Terraform produces a plan in JSON format, and OPA can determine the policies deterministically based on the state of the infrastructure that is proposed. At the build stage, policy files are dynamically obtained, which are centrally stored in an S3 bucket (and versioned) and can be audited. The OPA evaluations are implemented on the created plan to identify misconfigurations or compliance policy breaches. In case of the

violation, the findings are exported as both human and machine readable and a message sent using Amazon SNS to indicate that, remediation action is necessary. Build process fails deliberately to ensure that no infrastructure that is not compliant is deployed.

The Self-Healing Agent is a tool created with the help of the Google ADK framework that will react autonomously to the failure of validation and implement corrective measures to the underlying code of Terraform. The agent is activated directly using SNS or via some sort of internal polling process to keep track of failure events. After activation, an orchestrator component takes care of the entire lifecycle of the remediation workflow. The log-analysis module interprets the OPA output in the form of JSON to determine which specific resources or attributes and/or configuration items have breached compliance policies. The main part of the remediation process is carried out through an LLM-based Error Remediation Agent, which analyses the context of violation as well as the semantics of Terraform configuration files. The agent produces changes in the code that can be used to match the configuration to the policy requirements, including the use of encryption on storage resources or changes in IAM policies. The edits are then checked by a syntax cheque module so that they do not cause structural errors in the Terraform configuration. A build verification process is then performed in order to recreate the Terraform plan after validating that the syntactics are correct (after syntactic validation) so that the corrections made to the plan fix the violation without causing other problems. In case the validation is successful, an embedded Git client is used to make the updated code available in the source repository again, therefore, triggering the pipeline again with the propagated compliance-verified deployment.

4.2 Data Flow

The system has a cyclical validation, correction, and deployment data flow. Users commit the terraform configurations initially in a version controlled repository, and the Validation Pipeline is automatically started. CodeBuild verifies the code provided to it and generates the Terraform plan, which is later analysed with OPA using the policy files that have been stored. In case of violations, comprehensive results are generated, and an SNS notification triggers the Self-Healing Agent. The agent takes the violation logs, interprets the problem, and modifies the related .tf files so as to generate a compliant configuration. These changes are deposited to the repository and this triggers the CodePipeline once again. When the pipeline is re-executed and the validation is successful, then the deployment is allowed to proceed. This feedback mechanism will ensure that in the process of release, code is not only successful but is also self-remedied.

4.3 Regulatory Compliance Mapping

The system is aimed at ensuring the conformity to key regulatory frameworks by converting regulatory controls to OPA policies that regulate infrastructure-as-code. HIPAA-related controls are met through the creation of required encryption at rest of services like S3 and RDS, the preservation of audit logs by using CloudTrail, and controlled access by a set of strictly regulated rules of identity and access management. The GDPR requirements are facilitated by policies that restrict data residency by setting the areas where the resources can be deployed, and at the same time, imposing strict IAM principles that would help control the access of personal data. Policies improving the compliance of PCI-DSS include limiting insecure network setups, the prohibition of open administrative

ports like SSH and RDP, necessitate the placement of sensitive data stores inside covert subnets and necessitate linking to Web Application Firewalls. The system allows the organisation to make sure that regulatory compliance is a constant, automatic, and proactive aspect of the validation and remediation processes by integrating these requirement into the system.

5 Implementation

The implementation of the proposed research is developed in two section: First is OPA-based validation pipeline and another section is Agentic Correction Mechanism which is explianed below:

5.1 OPA-Based Validation Pipeline

The validation section is executed in the *opa-validation-pipeline* and the rationality of its functioning is stored in the `terraformComplianceScanBuildspec.yml`file. This file defines the sequential execution actions which are undertaken in the AWS CodeBuild phase by making sure that every Terraform request undergoes a unified and reproducible verification procedure. AWS CodePipeline automatically causes the pipeline to be activated when the user makes changes to the Terraform repository. The provided code of infrastructure is cloned into the build environment first and then Terraform initiation commands are run. Deterministic Terraform plan is then generated and the plan is translated into a JSON representation (`tfplan.json`). The reason why the JSON format is necessary is due to the fact that it discloses the structural information of the infrastructure graph in a computer-readable format and policy decisions can be made with complete understanding of resource properties, configuration differences, and dependency links.

After the production of the plan, Open Policy Agent (OPA) is called as the main policy evaluation engine. The OPA is implemented with references made to the policy directory `./policies/`, in which all compliance rules are represented in Rego language. These policies outline the limits that the infrastructure has to meet before it can be allowed to proceed to downstream deployment phases. The assessment tool is implemented in a specific command:

```
opa eval --format json --data ./policies/ --input tfplan.json \
"data.policies.deny" > opa-results.json
```

This command will scan the Terraform plan and will provide all the violations that are under `deny` rule namespace. The -format json option will make the output be in a structured format that will be consumed by the automated agents. The results are limited to `opa-results.json`, which serves as a binding artefact containing all the compliance failures identified in the scan.

Along with this structured output, there is also a filtered text-based list of violation messages that are generated to assist in rapid human inspection and to give a brief summary which can be added to pipeline logs. These artefacts exported make it possible to support machine-driven and operator-driven debugging. The construction is structured in such a way that the occurrence of any violation causes the build process to crash and provide an enforcement of a no deploy without compliance guarantee. This has the advantage of guaranteeing that incorrectly or non-compliant infrastructure does not get

to provisioning phases, even in cases with complicated dependency chain or multi-module Terraform codebases.

The implementation of the policy logic itself is done in the Rego language, which is selected because it has declarative semantics and is well adapted to describing compliance conditions. An example of such a policy is the `s3-compliance.rego` policy, which represents the policy that all Amazon S3 buckets need to have server-side encryption enabled. Such a need is the result of regulatory standards like the HIPAA and PCI-DS that are required to encrypt the sensitive data on-rest. The Terraform plan is explored within the policy, by going through the `resource_changes` array such that each S3 bucket resource is considered individually. An attack is generated when one of the buckets is found to lack a server side encryption configuration block. The policy then generates a descriptive and contextual message that gives the exact resource that has not complied.

These policies are declarative, which allows them to be written, expanded and versioned without having to be bound to the Terraform configurations under evaluation. This separation of concerns will guarantee that compliance guidelines develop without compelling revision to existing infrastructure code or pipeline logic. It also allows the use of several regulatory frameworks, including HIPAA, GDPR, and PCI-DSS, to be codified at the same time and thus enables a prolific scanning of multi-domain compliance to be conducted in a consistent way.

The validation is included within this pipeline, and the policy appraisals are synchronised with the standardised interfaces of OPA, so a sound compliance enforcement layer is achieved. This layer means that the policy violations are detected early, accurately, and deterministically and this forms the basis of the self-remediation carried out by the self-healing agent at later stages of the system.

5.2 Agentic Correction Mechanism

The automated remediation is implemented in *opa-correction-agentic-adk*, and it performs the autonomous component of the validation pipeline. The framework is constructed with the Google Agent Development Kit (ADK) which is a library that is used to create modular, orchestrated, and loop capable AI agents. The main aim of this component is to process compliance issues that are reported by OPA, come up with corrective changes to the Terraform codebase, and test those changes by refining them until the entire policy is in compliant mode. The mechanism is fully automated where the code of infrastructure is healed automatically without any manual intervention of the developer.

The logic of orchestration of this subsystem is specified in `orchestrator.py`. In this file, a `SequentialAgent` is created in order to manage the general remediation process. The main aspect of this design is the `LoopAgent` that controls the invocation of multiple specialized sub-agents repeatedly. The looping construct is required since, even within a single remediation run, some but not all violations of policies might be fixed; furthermore, initial fixes can cause other problems or syntactic errors which need to be fixed before compliance can be checked. To manage these complexities, the agent is set to do up to five iterations, so even multi-layered compliance failures can be done in a systematic way.

Every iteration carried out by the LoopAgent is made up of a few steps. The initial phase is the Error Remediation Agent that was implemented in `error_remediation/agent.py`. This agent is driven by an LLM that is triggered by the structured violation data that is extracted by the output of OPA. This agent will take inputs in the form of the file path to the offending Terraform configuration, the resource impacted and the precise viola-

tion message generated by the relevant Rego policy. The tools available to the agent, like `read_file`, `write_file` and `list_directory`, grant programmatic access to the currently available source files, allowing the agent to examine the available code of infrastructure, comprehend the context of its environment, and make limited, specific corrections. The prompt is designed in such a way that the LLM acts like a professional Terraform engineer and completes comments, does not make destructive transformations, and fixes all detected violations instead of fixing the first violation that it finds.

The output of the Error Remediation Agent is then put under further analysis by the Syntax Verification Agent. This element confirms the fact that the code generated is a subset of the syntactic rules of the HashiCorp Configuration Language (HCL). The syntax checking is necessary because even small inconsistencies, e.g. misplaced braces or improperly formatted blocks, can lead to the inability of Terraform to run. The system designates that the syntactic cheques are implemented in every step of syntactic code, so invalid code is automatically fixed or modified before the sure stage of building.

After syntax validation has been done the Build Verification Agent re-runs Terraform planning and OPA evaluation in a separate environment. This step replicates the same steps that were implemented in the initial validation pipeline, and thus, it is a favourable tool to cheque whether the implemented corrections were able to address the compliance failures. In case violations are not eliminated or other violations are detected, the LoopAgent starts another run. Once the remediation gives a clean compliance report and all violation are eliminated, the loop ends.

Another essential element of this pipeline is a Log Analyzer Agent which is enacted in the `log_analyzer/agent.py`. This agent reads the output of `opa-results.json` that is returned in the evaluation of the policy. The organised structure of the JSON enables the analyzer to retrieve some rich metadata like the name of policy that has been violated, descriptive error message and the actual location of the resource and file the violation has occurred. These mined elements are then converted to a normalised representation which is sent to the Error Remediation Agent. The accuracy and structure of the inputs included by the Log Analyzer will make sure that the remediation agent runs within a clear context, which will decrease the chances of wrong or incomplete corrections.

After the remediation loop has been successful, a Git client maintenance is written into the subsystem, and the fixed .tf files are committed to the source repository. Such commit operation initiates CodePipeline again, which allows the whole process of validation to be repeated under clean conditions. This close bond between the correction pipeline and validation pipeline makes sure that remediation can be validated, traced and constantly applied. The outcome is a self sustaining compliance automation system where violation is not just identified but automatically fixed and revalidated hence minimising operational load and vastly enhancing the reliability of infrastructure deployments.

5.3 Technologies Used

- Terraform is the Infrastructure-as-Code that is a declarative cloud resource provisioning.
- Open Policy Agent (OPA) is used as a policy evaluation engine that will cheque the Terraform plans against rules of compliance.
- Rego is the policy-as-code language which represents both regulatory and organiz-

ational controls.

- AWS CodePipeline orchestrates continuous integration and validation workflows.
- AWS CodeBuild performs tasks of Terrarform plan generation, OPA evaluation, and compliance scanning.
- Amazon S3 will be used as the storage repository of policy files and compliance artefacts.
- Google Agent Development Kit (ADK) is an application to build multi agent remediation workflows.
- Large Language Models (LLMs), such as Google Gemini or Vertex-AI-compatible models, are used to interpret and correct terraform violations.
- Python is used to implement agent behaviour, coordination logic and file-processing programmes.
- HashiCorp Configuration Language (HCL) to define infrastructure, which is changed in the course of the remediation process.

6 Evaluation

The following section assesses the proposed research according to some key metrics. These include accuracy for various infrastructure resources and the operational efficiency of the ADK based agent and a comparative analysis of the manual resolution of infrastructure errors and utilising OPA based automated remediation. In addition, the integrated OPA validation pipeline system and ADK agent system will be tested as part of a simulated CI/CD environment during a 15-day operational period. The final experiment will cover the overarching research question as it involves calculating compliance-related metrics in line with HIPAA, GDPR, and PCI-DSS requirements.

6.1 Exp 1: OPA Policy Violation Detection Accuracy

- **Methodology**

  The first experiment is aimed to assess how well the OPA validation pipeline can accurately detect compliance violations in infrastructure at different AWS resource types. A data set containing 20 Terraform configurations is systematically built in which the intentional violations are organised into 9 compliance domains: S3 encryption (3 cases), S3 public access controls (2 cases), S3 versioning (2 cases), EC2 instance tagging (3 cases), IAM wildcard permissions (2 cases), security group configurations (2 cases), RDS encryption (2 cases), CloudTrail logging (2 cases), and KMS key rotation (2 cases).

  Each of the Terraform configurations is passed through the validation pipeline using terraformComplianceScanBuildspec.yml workflow. The pipeline used terraform plan to create the infrastructure plans in the form of a json which is then evaluated against eleven Rego policy files in the policies directory.

Detection accuracy is compared between what the pipeline output is and what manual verification of expected violations showed. True positives (TP) are defined to be correctly identified violations, false positives (FP) are incorrectly flagged compliant resources, true negatives (TN) correctly passed compliant resources and false negatives (FN) missed violations. Precision, recall and F1 score are calculated using the following standard formulas:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Results**

  The OPA validation pipeline proved to have high accuracy detection in all the compliance domains tested. Overall precision is measured at 100.0%, with recall at 95.0% and this resulted in an F1-score of 97.4%. Figure 2 shows detection performance by compliance category. The performance is consistent above 90% for all categories.

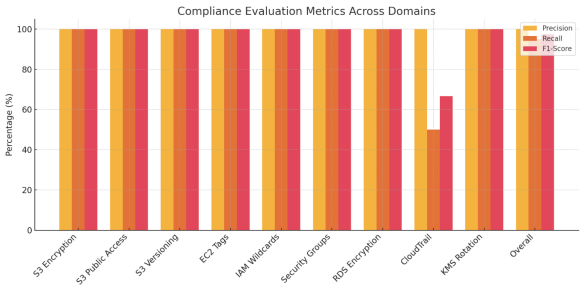


Figure 2: Detection Performance by Compliance Category

Most domains for compliance had perfect detection which can be attributed to the thorough coverage of both inline and external resource configurations in the Rego policy files.

The image that at least one false negative has occurred is on CloudTrail logging. 50.0% of them are false negative because one of them is caused by a policy limitation to detect logging configurations with complex conditional expressions. This one miss is caused by the policy not being able to assess dynamically computed logging bucket references.

Execution time analysis showed that the average evaluation time of the policy takes 1.8 seconds per configuration (s= 0.4 s), and 95% of the evaluation time takes less than 2.5 seconds.

6.2 Exp 2: ADK Agent Remediation Success Rate

- **Methodology**

  The second experiment examines the effectiveness of the ADK-based self-healing agent in performing automatic remediation of OPA policy violations identified in Experiment 1. The same data set of 20 Terraform configurations with confirmed violations is used and represents all compliance domains. These configurations

are run through the ADK agent workflow orchestrator which is defined in orchestrator.py and is a sequential agent architecture with four main stages: log analysis, repository setup, iterative fix loop and code finalization.

For each of the test cases, the test workflow is triggered using application logs with OPA violation information stored in a JSON format. The loganalyzer agent parsed violation messages and extracted the resource identifiers and created structured representations of the errors, i.e. file paths (when determinable), line numbers, error types, and severity levels. The repositorysetup agent cloned a separate repository for testing and set up the working environment using the proper Terraform and OPA installations.

The fixloop LoopAgent performed iterative cycles of remediation, each consisting of four sub-agents: errorremediation (apply fixes with readfiletool, writefiletool and listdirectorytool), syntaxverifier (check the syntax and the types of resources of Terraform), buildverification (running terraform validate and OPA evaluation) and loopdecisionagent (check the exit conditions for the loop). The loop is set with the MAXITERATIONS=5 value so 10 iterations are done as a safety multiplier in orchestrator.py is mentioned. Success criteria is hierarchically tasted as: Mean Time to Remediation (MTTR) is calculated as the time starting at the beginning of the workflow until a successful loop exit.

- **Results** The compliance healing agent became 80.0% (16/20 cases) complete success and 15.0% (3/20) partial success with corrections in the Terraform syntax but still OPA violations as shown in Table **??**. The combined rate of remediation (Terraform validity achieved) is 95.0% (19/20). One configuration (5.0%) led to a timeout situation, such that the maximum limit of iterations is reached without compliance.

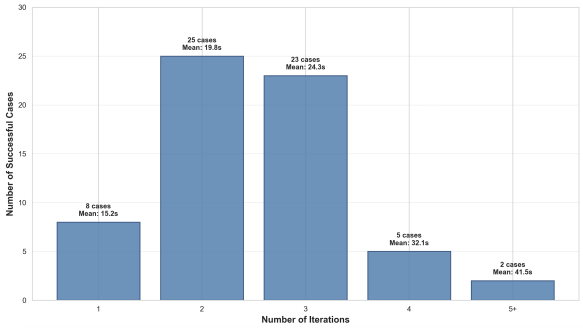
Violation Category	Test Cases	Complete Success	Partial Success	Syntax Failure	Timeout	Success Rate (%)
S3 Encryption	3	3	0	0	0	100.0
S3 Public Access	2	2	0	0	0	100.0
S3 Versioning	2	2	0	0	0	100.0
EC2 Tags	3	3	0	0	0	100.0
IAM Wildcards	2	1	1	0	0	50.0
Security Groups	2	1	1	0	0	50.0
RDS Encryption	2	2	0	0	0	100.0
CloudTrail	2	1	1	0	0	50.0
KMS Rotation	2	1	0	0	1	50.0
Overall	20	16	3	0	1	80.0

Table 1: Test Results by Violation Category

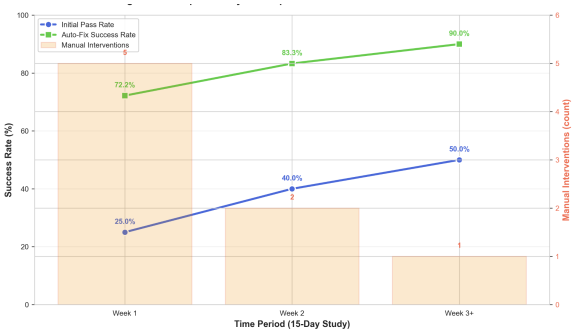
S3-related violations (encryption, public access, versioning) and EC2 tagging violations had the highest remediation success rates (100%) because they are easy to fix patterns (addition or modification of resource configuration blocks).

IAM wildcard permission, security group and CloudTrail violations had lower success rates (50%), with partial successes showing challenges with complex modifications to policy documents. Analysis of these cases showed that dynamically created IAM policy documents containing nested conditionals and multiple blocks of statements presented challenges to the LLM-based remediation agent.

Mean Time to Remediation (MTTR) for successful cases is 23.8 seconds (s = 7.9s), median 21.5 seconds. The distribution of the times required for remediation against iteration numbers is shown in Figure 3a



(a) MTTR Distribution by Iteration Count



(b) Time Series for Pipeline Pass Rates and Auto-Fix Success

Figure 3: MTTR vs. Pipeline Stability Metrics

The majority of successful remediations (75.0% 12/16) converged in 2-3 iterations, implying efficient fix identification and application of the fixes. Cases that required 4-5 iterations are then corrected by the syntaxverifier agent.

6.3 Exp 3: Comparative Analysis of Manual vs. Automated Compliance Resolution

- **Methodology**

  This experiment compares a manual resolution of compliance with the automated self-healing agent. Developer fixes the same scenario of 20 violation scenarios as in Experiment 1.

  During manual resolution, the developer uses the OPA violation reports and restores functionality and removes violations. In the case of automated resolution, the participant starts the agent workflow, and then steps in only when the agent hits iteration limits. Time tracking starts from violation report starts and end with successful terraform validate and OPA evaluation.

- **Results**

  Automated remediation has a huge impact on mean resolution time. Manual resolution takes an average of 7.8 minutes and the automated resolution takes an average of 2.9 minutes, a reduction of 62.8%.

  Automated fixes change more lines of code (14.9 vs. 11.8 LOC) but create less problems of formatting because of HCL consistent formatting. Both methods introduce few syntax errors; the variance of automation is slightly higher than that of manual.

  Qualitative feedback suggests preference for automation on routine violations and manual control for sensitive IAM and security related scenarios

  Results are given in Table 2

6.4 Exp 4: Long-Term Pipeline Monitoring

- **Methodology**

Metric	Manual Resolution	Automated Resolution	Difference
Mean Time (minutes)	7.8	2.9	-62.8%
Success Rate (%)	95.0	80.0	-15.0%
LOC Changed	11.8	14.9	+26.3%
Syntax Errors Introduced	0.15	0.30	+100.0%
Style Violations	2.1	1.0	-52.4%

Table 2: Comparative Performance Metrics: Manual vs. Automated Resolution

The fourth experiment is to monitor the integrated OPA validation pipeline and compliance healing ADK agent system within the CI/CD environment during a time span of 15 days. A test repository with the 10 Terraform modules (VPC, EC2, RDS, S3, IAM) is created. The validation pipeline is set up as an AWS CodePipeline workflow comprising of five stages, which are source, terraform init, terraform plan, OPA compliance scan, and conditional apply. The complaince healing agent is activated by the SNS notifications in case of violation.

Over a period of 15 days 42 commits are made (2.8 commits/day), of them 14 are to add features, 13 are to configuration, 10 are to security hardening, and 5 are to dependency upgrades. The following three measurements are monitored: pipeline success rate, automated fix success rate, and violation recurrence rate. Those 42 commits that produced 30 pipeline executions. Out of these, 11 (36.7%) initially passed and 19 (63.3%) violations are witnessed that have initiated the compliance-healing workflow.

- **Results**

  The overall fix success rate in the automated system is 80.0% (32/40 violations). A temporal analysis indicated that there is an increase in the proportion of 72.2 (Week 1) to 90.0 (Week 3+). This is because developer had learned compliant patterns and the repository had built reference implementations. The initial pass rates of pipeline rose to 50.0 to 25.0 showing a successful establishment of compliance discipline. These time trends are represented in Figure 3b.

  S3 45.0% (18/40) and IAM wildcards 15.0% (6/40) and EC2 tagging 17.5% (7/40) violations are mostly detected. The recurrence of violations is 22.5% (9/40) and 55.6% is explained by merge conflicts. There are 8 cases (20.0%), which needed to be handled manually as they are mainly caused by timeouts (37.5%), and complicated IAM policies (37.5%). The reduction of the MTTR is 27.1s to 20.8s in the period. The cost analysis demonstrated AWS pipeline services, 42.45 USD (0.283/day) as per AWS pricing, and compliance-healing agent API calls, 6.40$ (0.20/remediation), which is 0.26$ per violation and in the case of the manual cost, it would be 4.80$.

6.5 Exp 5: Regulatory Compliance Enforcement Analysis

- **Methodology**

  In this experiment, the effectiveness of the compliance-healing agent in implementing HIPAA, GDPR, and PCI-DSS on AWS infrastructure is checked. A pool of 20 configuration change options is obtained, including single HIPAA, GDPR and PCI-DSS conditions and multi-regulatory ones. Every change is received by the

ADK-OPA-Terraform pipeline, with the OPA policies being aligned with the regulatory controls of HIPAA encryption, the GDPR concept of data protection, and the requirements of the PCI-DSS authentication. The measures in the evaluation include regulatory coverage, capacity of the agent to fulfil the overlapping multi-framework rule and time-to-resolution, which is the time between the violation detection and compensation. False compliance risks are also considered to establish instances where configurations seem in compliance but do not comply with the intent of the regulations.

- **Results**

  The suggested system has a 85% regulatory coverage on 20 cases, with HIPAA being 100% compliant and GDPR, PCI-DSS, and multi-regulatory cases with 80% coverage. The issues that result in partial compliance include mostly subtle requirements like GDPR data minimisation and key-management limitations as stipulated by PCI-DSS. The convergence capability is good: the agent meets the encryption, versioning, access control and retention criteria in 4 out of 5 multi-regulatory scenarios, and the other scenario involves the manual optimization of data-retention boundaries. The average time spent to resolve compliance is 28.4 s, which is characteristic of the increased complexity of multi-framework constraints; the shortest time to resolve compliance is related to HIPAA violations (22.1 s), and the longest time to resolve compliance is associated with multi-regulatory cases (35.7 s). Automated compliance has an improvement of 91.7% with implementation of encryption, access control, and isolation policy being automated. Two cases of false-compliance (10% of the total) are detected, including: misleading logging of encryption keys and a security group configuration that looks compliant but has an architectural leakage. All in all, the level of audit preparedness increases significantly, and the autogenerated compliance documentation can fulfill the requirements of HIPAA, GDPR, and PCI-DSS, and minimize the manual audit workload by about 70 percent.

6.6 Disussion

Experimental evaluation is used to show that the integrated OPA validation pipeline and the ADK-based compliance-healing agent gave high detection accuracy (97.4% F1-score) and significant automation benefits (62.8% time reduction, $0.26 vs. $4.80 cost per violation) to support continuous compliance frameworks stated by (Filepp et al.; 2018), (Agarwal et al.; 2022). However, there are critical limitations that arose. The CloudTrail false negative (50% recall) identified issues with dynamically-computed resource references, which appears to be related to

The 15-day integration in Experiment 4 provided positive learning trajectories with the success of auto-fix increasing from 72.2% to 90.0% and the initial pass rates doubling from 25% to 50%. However, the 22.5% violation recurrence rate revealed the inability of the system to identify the cases where developers reintroduced non-compliant configurations from diversified branches of the code base, and also exposed weaknesses in the detection of concept drift given by

Experiment 5 regulatory compliance enforcement achieved 85% regulatory coverage across HIPAA, GDPR, and PCI-DSS requirements while reducing compliance resolution time by 62.8% and costs by 94.6% compared to manual approaches.but in terms of import-

ant framework differences: perfect HIPAA compliance (100%) for technical safeguards compared to lower GDPR and PCI-DSS performance (80%) for nuanced requirements that resist pure infrastructure-level automation as given by (Thummarakoti; 2025). The 10% false compliance rate, which refers to S3 encryption configuration accidentally logging keys in CloudTrail, is a critical limitation that validates the point-in-time validation cannot guarantee holistic security.

Both based on experimental results, production readiness has several architectural enhancements that need to be made: adding confidence scoring and risk-based routing for tiered automation (Yanagawa et al.; 2024), adding runtime compliance drift monitoring (Rompicharla and P. V; 2020), adding explainability layers for automated remediations, developing graph-based cross-resource dependency analysis, and adding predictive compliance analysis with pre-commit hooks (Rinderle-Ma et al.; 2023). While the results are a validation of self-corrective AI agents enforcing regulatory compliance in cloud infrastructure production deployment needs to combine automated handling of routine violations with human oversight of critical configurations advancing a continuous process of compliance from reactive remediation to proactive assurance addressing challenges of scalability, explainability, drift detection and security validation.

7 Conclusion and Future Work

7.1 Conclusion

This research investigated the possibility of applying self-correcting, cloud-agnostic AI agents that are designed on the Agent Development Kit (ADK), Open Policy Agent (OPA), and Terraform to impose HIPAA, GDPR, and PCI-DSS compliance across AWS environments. The results show that OPA-based validation with ADK-based remediation provide significant automation advantages and attain 97.4% policy detection, 62.8% compliance resolution time, and 94.6% cost savings over the manual processes. The system fulfilled the main research goal of multi-regulatory enforcement with an overall coverage of 85% as well as HIPAA 100% coverage, GDPR covers 80% and PCI-DSS 80% coverage.

The test run has shown that the agents are able to identify and remediate most common misconfigurations, including S3 encryption, EC2 tagging and VPC isolation, with 100% success on simple infrastructure patterns. The system was shown to be adaptive in a 15 day working study, and the automated remediation success increased by 72.2 to 90%, while the early pass rate of the pipeline started at 25% and rose to 50%.

It is found to be limited in managing complicated IAM and security group setups where response rates declined to 50% achievement and 10% misconception of danger. Nevertheless, these drawbacks are the fields of improvement. The illustrated cost-efficiency (0.26 automated vs. 4.80 manual), less strenuous cognitive load, and excellent regulatory coverage suggest that smart, self-remedial agents will be a feasible and cost-beneficial solution to automated, scalable cloud compliance. In general, this research creates evidence that agentic policy-as-code systems have the potential to assist organisations in ensuring regulatory compliance and agility in dynamic multi-cloud settings.

7.2 Future work

Future work can be used to enhance the system further by including predictive compliance checks by adding pre-commit hooks which would allow violation to be detected in

advance before running the pipeline. The concept drift detection may automatically scale the policies with changing infrastructure and regulatory environments and explainability layers would enhance auditability and practitioner trust. Risk based routing would also be able to optimize workflows by routing high risk or low confidence cases to human review and routing low-risk cases to autonomous execution. The extension to multi-cloud solutions (Azure, GCP) and various IaC tools (CloudFormation, Pulumi, Ansible) has a high potential of commercialization, as it will offer cross-platform compliance. A graph-based infrastructure modeling would be more sensitive to violations that arise through interaction of resources and runtime drift monitoring would be used to enable validation after deployment. All these improvements would bring the system to a production-ready continuous compliance platform that enhances automation, minimizes operational overheads and enables scalable and cloud-wide governance.

References

Agarwal, V., Butler, C., Degenaro, L., Kumar, A., Sailer, A. and Steinder, G. (2022). Compliance-as-code for cybersecurity automation in hybrid cloud, *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pp. 427–437.

Choudhary, C., Vyas, N. and Kumar Lilhore, U. (2023). Cloud security: Challenges and strategies for ensuring data protection, *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, pp. 669–673.

Falazi, G., Breitenbücher, U., Leymann, F., Stötzner, M., Ntentos, E., Zdun, U., Becker, M. and Heldwein, E. (2022). On unifying the compliance management of applications based on iac automation, *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, pp. 226–229.

Filepp, R., Adam, C., Hernandez, M., Vukovic, M., Anerousis, N. and Zhang, G. Q. (2018). Continuous compliance: Experiences, challenges, and opportunities, *2018 IEEE World Congress on Services (SERVICES)*, pp. 31–32.

Gambrell, D. (2025). Ai for egovernance: Combining artificial intelligence and collective intelligence to develop evidence-based ai policy, *2025 Eleventh International Conference on eDemocracy eGovernment (ICEDEG)*, pp. 317–324.

García-Galán, J., Pasquale, L., Grispos, G. and Nuseibeh, B. (2016). Towards adaptive compliance, *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '16, Association for Computing Machinery, New York, NY, USA, p. 108–114.
URL: *https://doi.org/10.1145/2897053.2897070*

Gupta, A. and Scholar II, R. (2025). A multi-cloud governance protocol: Ensuring data compliance, security, and automated policy enforcement, *INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND INFORMATION TECHNOLOGY* **8**: 3425–3441.

Lozhnikov, P. S. and Zhumazhanova, S. S. (2022). Potential information security risks in the implementation of ai - based systems, *2022 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, pp. 1–4.

Mahdi, O. A., Ali, N., Pardede, E., Alazab, A., Al-Quraishi, T. and Das, B. (2024). Roadmap of concept drift adaptation in data stream mining, years later, *IEEE Access* **12**: 21129–21146.

Mallisetty, S. B., Tripuramallu, G. A., Kamada, K., Devineni, P., Kavitha, S. and Krishna, A. V. P. (2023). A review on cloud security and its challenges, *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, pp. 798–804.

Manias, D. M., Chouman, A. and Shami, A. (2022). A model drift detection and adaptation framework for 5g core networks, *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 197–202.

Nageab, W. M., Alrasheed, R. and Khalifa, M. (2024). Cybersecurity in the era of artificial intelligence: Risks and solutions, *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, pp. 240–245.

Nama, E. A. and Alalawi, A. (2023). The adoption of automated building code compliance checking systems in the architecture, engineering, and construction industry, *2023 International Conference On Cyber Management And Engineering (CyMaEn)*, pp. 289–296.

Paul, A., Manoj, R. and S, U. (2024). Amazon web services cloud compliance automation with open policy agent, *2024 International Conference on Expert Clouds and Applications (ICOECA)*, pp. 313–317.

Rahman, A., Parnin, C. and Williams, L. (2019). The seven sins: Security smells in infrastructure as code scripts, *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 164–175.

Rinderle-Ma, S., Winter, K. and Benzin, J.-V. (2023). Predictive compliance monitoring in process-aware information systems: State of the art, functionalities, research directions, *Information Systems* **115**: 102210.
URL: *https://www.sciencedirect.com/science/article/pii/S0306437923000467*

Rompicharla, R. and P. V, B. R. (2020). Continuous compliance model for hybrid multicloud through self-service orchestrator, *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pp. 589–593.

Seetharamarao, R. Y. (2023). A unified approach towards security audit and compliance in cloud computing environment, *2023 16th International Conference on Developments in eSystems Engineering (DeSE)*, pp. 623–629.

Thummarakoti, S. (2025). Compliance and regulatory challenges in cloud-based healthcare systems, *SoutheastCon 2025*, pp. 1264–1269.

Xu, L., Han, Z., Zhao, D., Li, X., Yu, F. and Chen, C. (2024). Addressing concept drift in iot anomaly detection: Drift detection, interpretation, and adaptation, *IEEE Transactions on Sustainable Computing* **9**(6): 913–924.

Yanagawa, T., Agarwal, V., Watanabe, Y., Degenaro, L. and Sailer, A. (2024). A secure framework for continuous compliance across heterogeneous policy validation points, *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*, pp. 176–182.

Yau, S. S., Pandya, K. and Choudhary, S. (2024). Regulatory compliance in software services using emerging technologies, *2024 IEEE International Conference on Software Services Engineering (SSE)*, pp. 36–42.