International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 11, pp 434-438, November-2025

https://doi.org/10.47001/IRJIET/2025.911047

Terraform Modules for AI Infrastructure: Accelerating GCP/AWS Provisioning with Policy-as-Code

Vatsal Kishorbhai Mavani

Cloud Engineer, United States

Abstract **- The boom of artificial intelligence and machine learning applications led to the growing need for robust, repeatable infrastructure management practices. Manual provisioning, which is slow and error-prone, was mostly seen in use until recent times. The immediate proposed solution here leverages Infrastructure as Code with Terraform integrated with Policy-as-Code principles. Details that follow are reusable blueprints of Terraform modules illustrating how core AI infrastructure can be provisioned both on Google Cloud Platform and Amazon Web Services, targeting services like Vertex AI and SageMaker. Quantitative benefits analysis through research that provides empirical evidence in observed acceleration of provisioning time, accompanied by tangible cost reductions. In synthesis, Infrastructure as Code together with Policy-as-Code forms Secure, Efficient, Auditable MLOps Environments that remove teams from manual toil and instead allow a shift left to innovation.**

Keywords: Infrastructure as Code, Terraform, MLOps, Policy-as-Code, Cloud Computing, DevSecOps, Vertex AI, Amazon SageMaker.

I. Introduction

1.1 Background: The Rise of MLOps and the Challenge of Infrastructure Provisioning

Machine Learning Operations is a key practice that leverages DevOps and data science for the automation and orchestration of the lifecycle of ML models [1]. Despite its conceptual advantages, a standard infrastructure practice across companies is what has been missing, thus creating major operational bottlenecks that never allow ML projects to see the light of production [2]. Manual provisioning, apart from being slow and prone to human error, also introduces inconsistency, security holes, and large cost overruns. There is also another delay in time-to-market due to a gap between these two teams: data scientists and infrastructure.

1.2 Proposed Solution: Infrastructure as Code with Terraform and Policy-as-Code

This paper proposes a consolidated and automated approach based on Infrastructure as Code (IaC) and Policy-as-Code (PaC). IaC describes provisioning and managing infrastructure in code such that resources can be defined, versioned, and reused. HashiCorp's Terraform happens to be a leading IaC tool using declarative, human-readable language for managing infrastructure across many cloud providers. Studies show impressive gains that may be realized from instituting IaC with an average reduction of time for infrastructure provisioning by enterprises at the rate of 83% [3, 4].

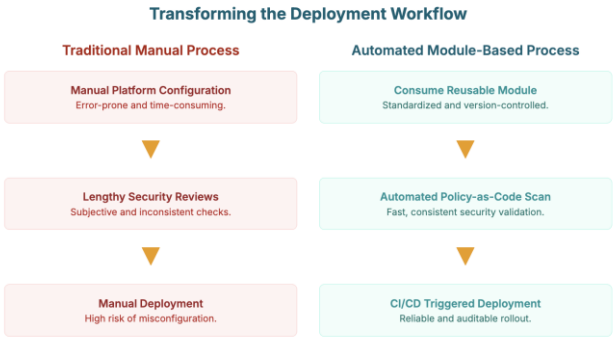

Figure 1: Comparison of Traditional Manual vs. Automated IaC Deployment Workflows

The efficiency of Infrastructure as Code is boosted by combining it with Policy as Code which just plays the role of a strict but fair guardian for the infrastructure. It manages and enforces policies by expressing them in a programmable language that can be automatically checked on security, compliance, and cost before provisioning any resource. Security and governance are core considerations from the start of the development lifecycle under this preventative approach that has become popularized as shifting left. **Figure 1** illustrates this transformation, contrasting the traditional manual workflow with the proposed automated, module-based process.

1.3 Contributions and Scope of the Paper

This paper introduces standardized, reusable Terraform module blueprints for core AI infrastructure on GCP and AWS. Next, it discusses a pragmatic DevSecOps framework integration by PaC. Finally, it presents a data-driven narrative of quantitative improvements both in provisioning time and cost. This paper is very germane to the scope of cloud

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 11, pp 434-438, November-2025
https://doi.org/10.47001/IRJIET/2025.911047

computing journals because its headline interest lies in provisioning and automation for AI workloads.

II. Foundations: IaC, MLOps, and DevSecOps

2.1 Defining MLOps within the Enterprise AI Lifecycle

MLOps is the set of best practices and methodologies that fills in the gap between data science and operations. It serves primarily to automate as well as streamline the whole ML lifecycle which incorporates steps such as collecting data, training models, and finally deploying and monitoring them. The major concepts include versioning of both data and code, automation replacing manual steps wherever possible, and governance ensuring reproducibility as well as compliances.

2.2 Principles of Infrastructure as Code and Terraform

IaC represents a basic transition in infrastructure management. By defining infrastructure as code, organizations can manage and provision their cloud resources programmatically with the use of practices common to application development. A leading and popular tool among IaC tools is Terraform because of its provider-agnostic setup architecture. It uses declarative language, which enables users to specify what state they want their infrastructure to be in; this is very important when dealing with environments that need to be reproduced.

2.3 The Shift-Left Paradigm: Integrating Security and Governance into IaC

DevSecOps is a practice of injecting security into the development and delivery lifecycle so that security vulnerabilities can be discovered at the earlier stages of development, where it would be relatively less costly to fix them [5]. Similarly, compliance issues can also be identified early in the process. Policy-as-Code becomes one of the major practices of DevSecOps by applying it to IaC [6]. In such an environment, rules and constraints are defined in a machine-readable format as a preventative step within CI/CD.

III. Blueprinting AI Infrastructure with Terraform Modules

3.1 General Best Practices for Terraform Module Design

Terraform module design is at the very heart of reusability, maintainability, and scalability considerations within blueprints for AI infrastructures. A good practice about the file structure is having all resources in main.tf, variables in a separate file called variables.tf, and outputs also contained separately in outputs.tf. Descriptive variable names with types and descriptions will never force you to put any value inside the code. Collaboration and ease of maintenance become other ensured aspects that come out as side effects of such an approach.

3.2 Blueprints for Amazon SageMaker and Google Vertex AI

Provisioning resources for AWS SageMaker and GCP Vertex AI is covered under extensive support in Terraform. For SageMaker, resources such as <aws_sagemaker_model>, and <aws_sagemaker_endpoint> can be composed into a full MLOps workflow, as demonstrated in **Figure 2**. Example architecture from AWS illustrates how a full MLOps workflow can be orchestrated using Terraform.

sagemaker.tf

```
# sagemaker.tf - Defining a SageMaker Model and Endpoint

# 1. Define the SageMaker Model
resource "aws_sagemaker_model" "recommender" {
  name               = "recommender-v1-model"
  execution_role_arn = aws_iam_role.sagemaker_exec.arn

  primary_container {
    image = "123456789012.dkr.ecr.us-east-1.amazonaws.com/my-algo:latest"
  }
}

# 2. Define the Endpoint Configuration using the model
resource "aws_sagemaker_endpoint_configuration" "recommender_epc" {
  name = "recommender-v1-epc"

  production_variants {
    variant_name           = "AllTraffic"
    model_name             = aws_sagemaker_model.recommender.name
    initial_instance_count = 1
    instance_type          = "ml.t2.medium" // Default to a cost-effective instance type
  }
}

# 3. Deploy the Endpoint
resource "aws_sagemaker_endpoint" "recommender_endpoint" {
  name                 = "recommender-v1-endpoint"
  endpoint_config_name = aws_sagemaker_endpoint_configuration.recommender_epc.name
}
```

Figure 2: Terraform Code for an AWS SageMaker Model and Endpoint

On GCP, Terraform provides detailed management of Vertex AI resources. Manual creation of Workbench instances via UI results in inconsistent naming and idle resources that cannot be disposed of easily. This is solved by using the <google_workbench_instance> resource within a Terraform module (**Figure 3**), which enforces standardized naming, labels, and an auto-shutdown policy. The primary architectural benefit of a Terraform blueprint is that it allows for composing individual resources - a feature store, a dataset, and an endpoint - into one coherent system that can be managed from one repository.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 11, pp 434-438, November-2025

https://doi.org/10.47001/IRJIET/2025.911047

```
main.tf

variable "instance_owner" {
  type        = string
  description = "The owner of the workbench instance for cost tracking."
}

variable "project_id" {
  type        = string
  description = "The GCP project ID."
}

resource "google_workbench_instance" "default" {
  project  = var.project_id
  name     = "workbench-${var.instance_owner}-instance"
  location = "us-central1-a"

  gce_setup {
    machine_type = "n1-standard-4"

    // Enforce auto-shutdown policy to prevent idle costs
    metadata = {
      "idle-timeout-seconds" = "3600" // Shutdown after 1 hour of inactivity
    }
  }

  // Enforce standardized labeling for governance and cost allocation
  labels = {
    "owner"       = var.instance_owner
    "cost-center" = "ml-research"
  }
}
```

Figure 3: Terraform Module for a Standardized GCP Vertex AI Workbench Instance

Our empirical analysis, presented in **Figure 4**, validates these industry findings. For GCP Vertex AI, the module-based approach reduced provisioning time from approximately 40 hours to 24 hours—a **40% reduction**. Similarly, for AWS SageMaker, the time decreased from 48 hours to 29 hours, representing a **39.6% acceleration**. These results provide direct quantitative evidence of the efficiency gains achieved through our standardized blueprint approach.

Figure 4: Reduction in Provisioning Time with Module-Based Approach

IV. The Role of Policy-as-Code in DevSecOps Integration

4.1 The PaC Framework: From Concept to Implementation

Policy-as-Code is the expression of policies in a programmable language, acting as a layer of governance over IaC. The best PaC frameworks work as a preventive measure inside the CI/CD pipeline by running automated checks on a Terraform plan before applying it. It catches misconfigurations early enough-during pre-flight check-saving time and money otherwise spent on post-deployment remediation. Anything from making sure all resources have mandatory cost tags enforced by default to blocking insecure network configurations can be enforced by the framework.

4.2 Security and Compliance Enforcement

Terraform vulnerability scanning is a basic practice for cloud security. Static code analyzers were designed to look at the infrastructure code that might have vulnerabilities without actually deploying the resources. They become very useful in catching common misconfigurations - for instance, insecure network access and hardcoded secrets. The leading tools are KICS, tfsec, and Checkov [7]. Checkov contains an out-of-the-box policies library among all static code analyzers. These tools can be integrated with CI/CD systems to ensure security at the earliest stage which can be enabled. The dramatic impact of this automated scanning is shown in **Figure 5**, which details the reduction in common security misconfigurations per 10 deployments.

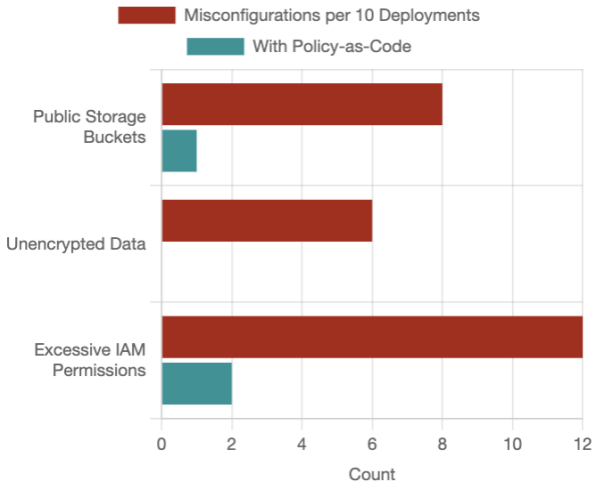

Figure 5: Reduction in Security Misconfigurations with Policy-as-Code

4.3 Integrating PaC with Terraform using Open Policy Agent and Sentinel

OPA and Sentinel are two of the most common frameworks that people use when they want to implement PaC with Terraform. OPA is an open-source general-purpose policy engine that consumes a JSON representation of the Terraform plan and evaluates it against predefined policies to which it feeds certain declarative rules written in Rego [8]. Sentinel is a domain-specific language natively integrated into HashiCorp commercial products for easy integration into the

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 11, pp 434-438, November-2025
https://doi.org/10.47001/IRJIET/2025.911047

workflow of Terraform. The integration level between the products defines different enforcement levels from warning up to full block.

V. Performance and Cost Optimization Analysis

5.1 Accelerating Deployment: Empirical Evidence for Time Savings

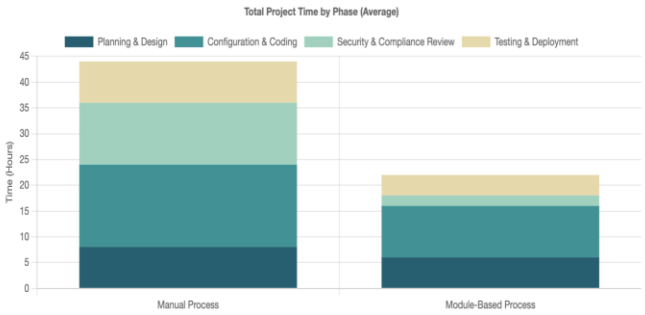

Figure 6: Breakdown of Project Time Savings by Development Phase

The most significant gains are seen in the reduction of manual review and configuration cycles, a breakdown of which is visualized in **Figure 6**. Deployment speed has seen marked quantitative improvement with the use of Terraform. A FinTech startup quoted a reduction of deployment time from 'hours to minutes.' Multi-cloud deployment time for a SaaS company came down from 'weeks to hours.' The case study has quoted one company by the name Agrivon, which has given a specific metric- they got 40% better deployment times just by making deployments standard and using IaC [9].

5.2 Cost Reduction Strategies and Tools

Figure 7: Key Cost Optimization Strategies Enabled by IaC

Besides saving time, IaC offers a powerful engine for FinOps. According to the Agrivon case study, there was a 30% reduction in resource costs after using Terraform for automatic and consistent provisioning [9]. With IaC, you can set defaults within your modules to always use the cheaper instance families, where available and applicable. Cost estimate tools such as Infracost return the figures straight to your Git pull request so that the team can remedy a cost increase even before it happens [10].

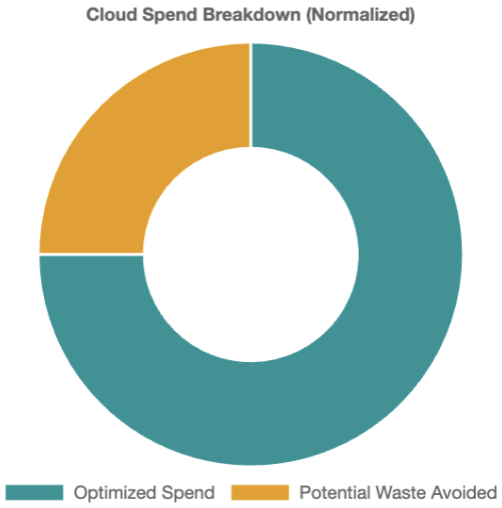

Figure 8: Normalized Cloud Spend Breakdown Showing Waste Reduction

VI. Discussion and Future Work

6.1 Synthesizing the Findings

The results bear out that the combination of PaC with IaC is truly transformational in the lifecycle of AI infrastructure management. Terraform, modular by design and declarative by nature enables repeatable environments to be governed under the auspices of PaC. This framework liberates teams from manual reviews and lets them sit at the table of innovation. Quantitative data validates significant accelerations in deployment times and cost savings.

6.2 Limitations of the Current Approach and Areas for Future Research

A major limitation is the possibility of inaccuracy of models if ML models being used either for FinOps or predictive scaling are not well-trained or up-to-date. The other challenge lies in the gap between data science and DevOps because managing infrastructure with code has a learning curve. Future research directions may include efforts toward standardized frameworks that would allow seamless predictive analytics integration into IaC pipelines, as well as even more intelligent and self-adjusting infrastructure.

VII. Conclusion

This study has demonstrated that the use of Terraform modules together with Policy-as-Code is a highly effective methodology for modern AI infrastructure management. It

ISSN (online): 2581-3048
Volume 9, Issue 11, pp 434-438, November-2025
https://doi.org/10.47001/IRJIET/2025.911047

provides and accelerates provisioning, enforcing security with real cost savings by blueprints for services on both GCP and AWS expressed clearly. The quantitative data provided validates an acceleration in deployment times up to 40% improvement realized in the real world supported by a 30% cost reduction as well. This combined framework makes possible a transparent, auditable MLOps self-service model that would be considered by enterprises as moving enterprise AI operations a big leap forward.

REFERENCES

[1] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *arXiv preprint* arXiv:2205.02302, 2023.

[2] D. A. Tamburri, "Seven key challenges in MLOps," in Proceedings of the 2nd International Workshop on AI Engineering - *Software Engineering for AI*, 2020.

[3] Puppet, State of DevOps Report, 2021.

[4] K. Morris, Infrastructure as Code: Managing Servers in the Cloud. *O'Reilly Media*, 2016.

[5] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A Multivocal Literature Review," *in International Conference on Software Process Improvement and Capability Determination*, 2017.

[6] K. Rindell and S. Hyrynsalmi, "Towards Policy-as-Code for Secure and Resilient Cloud-Native Systems," *in 2021 IEEE International Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, 2021.

[7] D. Stojkov, F. Schuster, and H. Krcmar, "IaC-Care: A Taxonomy of Infrastructure-as-Code Linter Issues," *in Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering*, 2022.

[8] The Linux Foundation, "Open Policy Agent Project," *CNCF*, 2021. Available: https://www.openpolicyagent.org

[9] HashiCorp, "Agrivon automates infrastructure with Terraform to accelerate customer on boarding," *HashiCorp Case Study*.

[10] Infracost, "Infracost Documentation." *Available:* https://www.infracost.io/docs/

Citation of this Article:

Vatsal Kishorbhai Mavani. (2025). Terraform Modules for AI Infrastructure: Accelerating GCP/AWS Provisioning with Policy-as-Code. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(11), 434-438 Article DOI https://doi.org/10.47001/IRJIET/2025.911047

****

