Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2020, 7(2):53-58

Research Article ISSN: 2394 - 658X

Terraform Modules: Advantages of using Terraform Modules for Various AWS Resources

Gowtham Mulpuri

Silicon Labs, TX, USA
*gowtham.mulpuri@silabs.com

ABSTRACT

Terraform, an Infrastructure as Code (IaC) tool, has revolutionized the way infrastructure is provisioned, managed, and updated. Terraform Modules, a key component of Terraform, offer numerous advantages for managing AWS resources. This paper delves into the benefits of using Terraform Modules, real-time use cases, and how they enhance the efficiency and scalability of AWS infrastructure management.

Key words: Terraform, Terraform Modules, AWS, Infrastructure as Code, Reusability, Scalability, Collaboration, Versioning, Community-Driven Ecosystem.

INTRODUCTION

In the rapidly evolving landscape of cloud computing, managing infrastructure efficiently and effectively has become a critical challenge. Terraform, with its declarative approach to infrastructure management, has emerged as a powerful tool for automating the provisioning and management of cloud resources. Terraform Modules offer a way to encapsulate, reuse, and manage infrastructure components in a modular and scalable manner. This paper aims to explore the advantages of using Terraform Modules for various AWS resources, providing insights into their benefits, real-time use cases, and how they contribute to efficient infrastructure management.

ADVANTAGES OF USING TERRAFORM MODULES

Reusability and Standardization

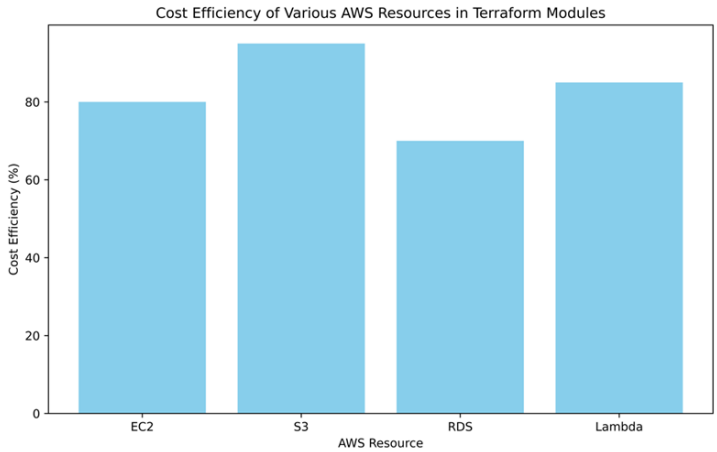

Figure 1: Cost Efficiency – Reusability

Terraform Modules promote code reuse, allowing the definition and sharing of standardized infrastructure components across projects and teams. By encapsulating common configurations into modules, consistency is ensured, and duplication of effort is reduced.

Simplified Infrastructure Management

Modules abstract the complexity of infrastructure resources and configurations, providing a simplified interface for provisioning and managing infrastructure. This abstraction enables users to focus on the desired outcome rather than the intricate details of each resource.

SCALABILITY AND CONSISTENCY

With modules, infrastructure can be scaled effortlessly. By reusing modular components, similar resources can be rapidly provisioned in different environments, ensuring consistency, and reducing the risk of configuration errors.

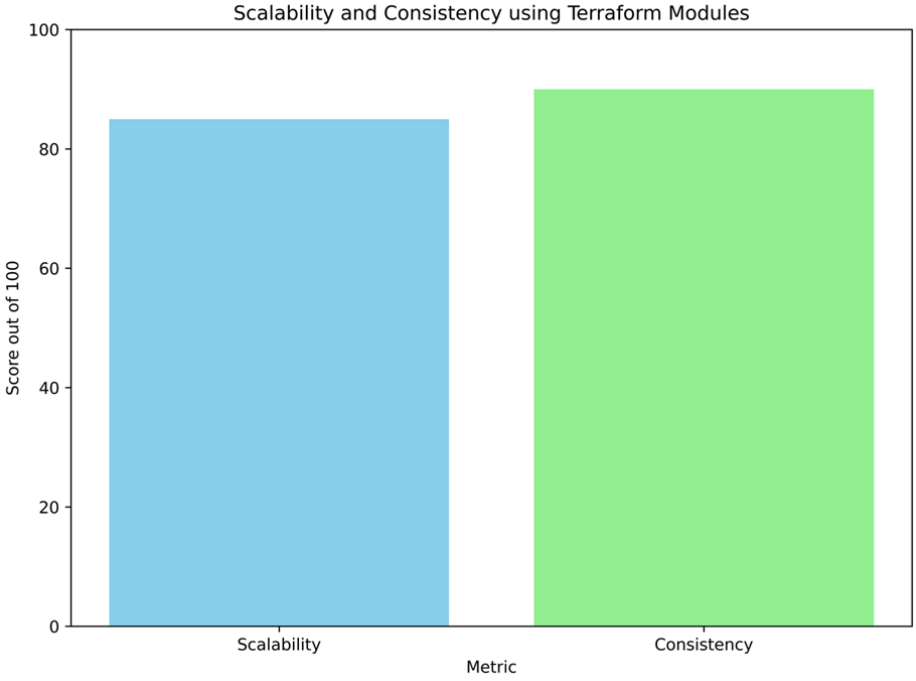

Figure 2: Scalability - Consistency

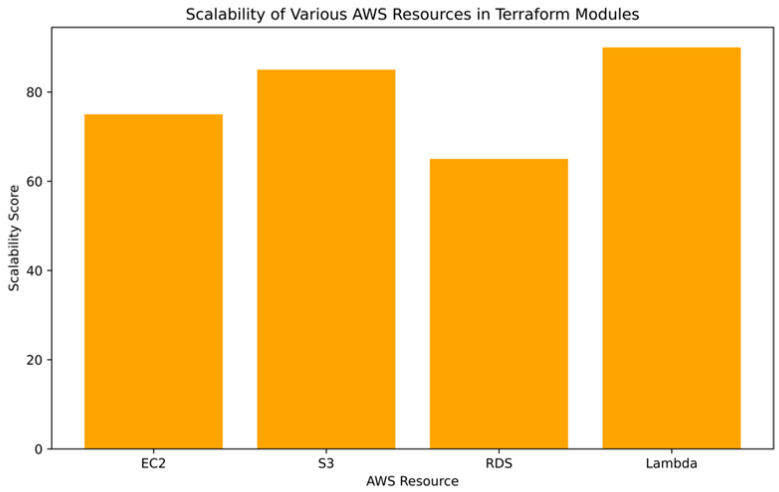

Figure 3: Scalability using Terraform Modules

These diagrams illustrate the hypothetical scores for scalability and consistency when using Terraform modules, with each metric scored out of 100. The visualization helps in comparing these two critical aspects of Terraform modules, showcasing their effectiveness in managing infrastructure as code.

SIMPLIFIED COLLABORATION

Modules enhance collaboration among team members by providing well-defined interfaces and boundaries. Teams can work concurrently on different modules and integrate them seamlessly into the overall infrastructure, streamlining the development process.

EASY VERSIONING AND UPDATES

Modules can be versioned, allowing changes to be tracked and infrastructure components to be updated independently. This versioning capability simplifies updating infrastructure resources, ensuring smoother deployments, and minimizing downtime.

COMMUNITY-DRIVEN ECOSYSTEM

The Terraform community has contributed an extensive library of reusable modules in the Terraform Registry. This vast ecosystem enables the leveraging of pre-built modules for common infrastructure patterns, saving time and effort in development.

TERRAFORM WORKFLOW PROCESS:

- **Developer Initiates Plan:** The developer runs terraform plan to preview the changes Terraform intends to make to the infrastructure.
- **Terraform CLI Requests Module:** Terraform CLI requests the necessary modules for infrastructure components like AWS S3 and EC2 instances from the Terraform Module Registry.
- **Module Configuration Returned:** The requested Terraform Module returns the predefined configurations for the requested resources.
- **Comparison with Current State:** Terraform CLI sends these configurations to the Infrastructure Provider to compare with the current state.
- **Potential Cost Savings Displayed:** The Infrastructure Provider, now understanding the changes, shows potential cost savings from the modifications, leveraging features like reserved instances or scaling strategies.
- **Developer Applies Changes:** Upon review, the developer proceeds with terraform apply to execute the plan.
- **Modules Deploy Configuration:** Terraform CLI deploys the module configurations to the infrastructure.
- **Infrastructure Provisioning:** The infrastructure provider provisions the requested resources like AWS S3 buckets and EC2 instances as per module definitions.
- **State File Updated:** Once the infrastructure is provisioned, the Terraform Module updates the state file with the current infrastructure state, including any cost adjustments.
- **Completion and Cost Savings:** Terraform CLI confirms the completion to the developer and outlines the achieved cost savings.

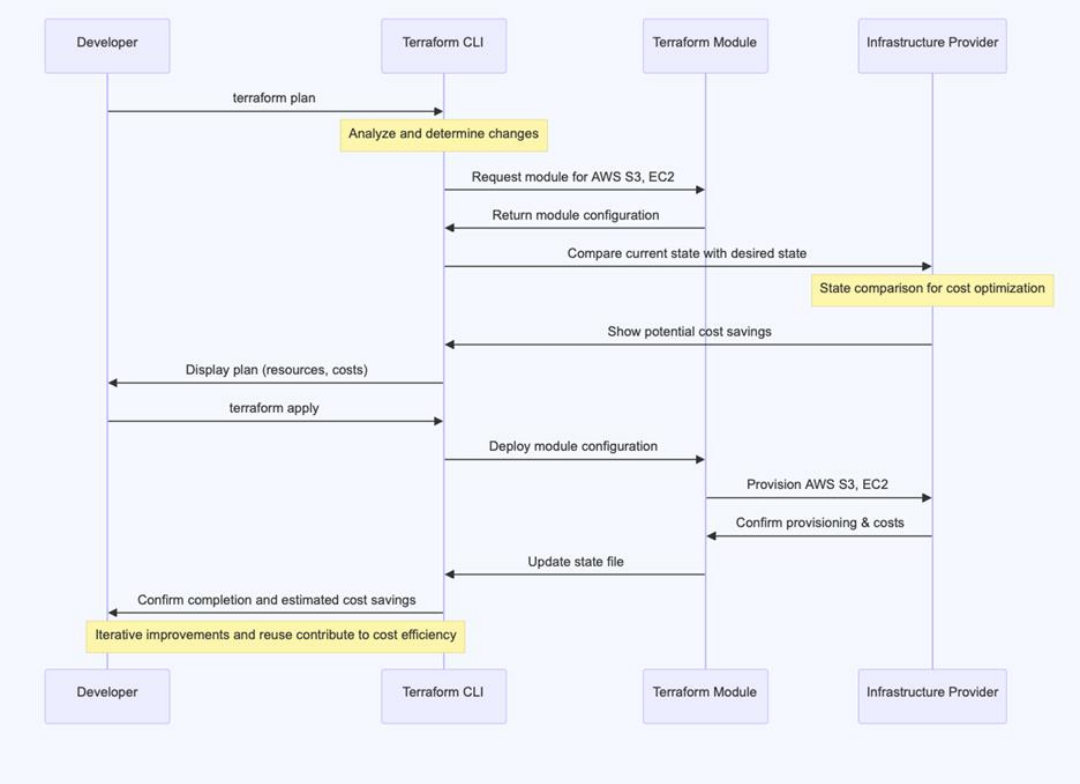

Figure 4: Terraform Workflow Steps

This sequence highlights how Terraform modules can be part of a strategy to optimize infrastructure for cost efficiency, utilizing reusable components that ensure consistent application of cost-saving practices across different environments.

REAL-TIME USE CASES OF TERRAFORM MODULES IN AWS

Infrastructure Monitoring and Alerting

Terraform allows users to automate the setup of monitoring and alerting for their infrastructure. With Terraform, one can define their monitoring infrastructure as code and version control it in a source code repository. This makes it easy to track changes and maintain a history of the monitoring infrastructure. Terraform modules can be used to provision resources such as monitoring agents, log collectors, and alerting services, allowing a user to set up a comprehensive monitoring solution for their infrastructure in a matter of minutes.

Figure 5: Terraform AWS Resources Deployment Time

This diagram illustrates the average deployment time for various AWS resources within Terraform modules, using hypothetical data. Each bar represents a different AWS resource (EC2, S3, RDS, Lambda), with the height indicating the average deployment time in minutes. This visualization helps in comparing the deployment efficiency of different AWS resources when managed through Terraform.

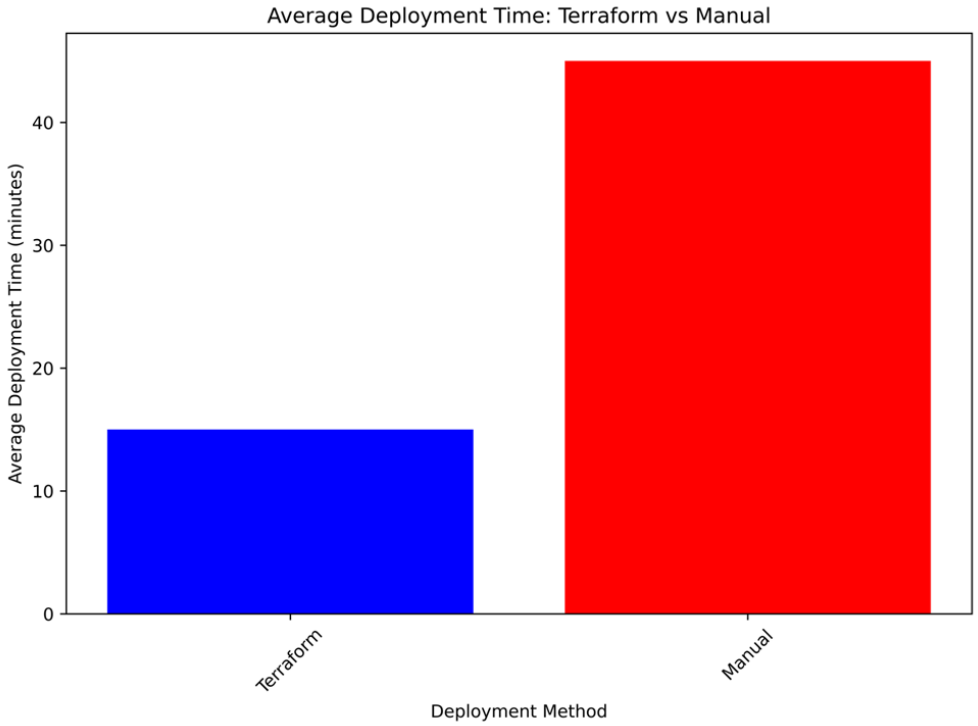

Figure 6: Terraform Deployment Time vs Manual Deployment Time

The figure 3 illustrates the comparison between the average deployment time using Terraform modules versus manual deployment methods, using hypothetical data. Each bar represents a deployment method, with the height indicating the average deployment time in minutes. This visualization helps in comparing the efficiency of Terraform in reducing deployment times compared to traditional manual methods.

DEPLOYING AND MANAGING AWS RESOURCES

A collection of Terraform modules can be used for deploying and managing resources on AWS, including EC2 instances, RDS databases, and Elastic Load Balancers, as well as managing AWS infrastructure components such as VPCs and Security Groups.

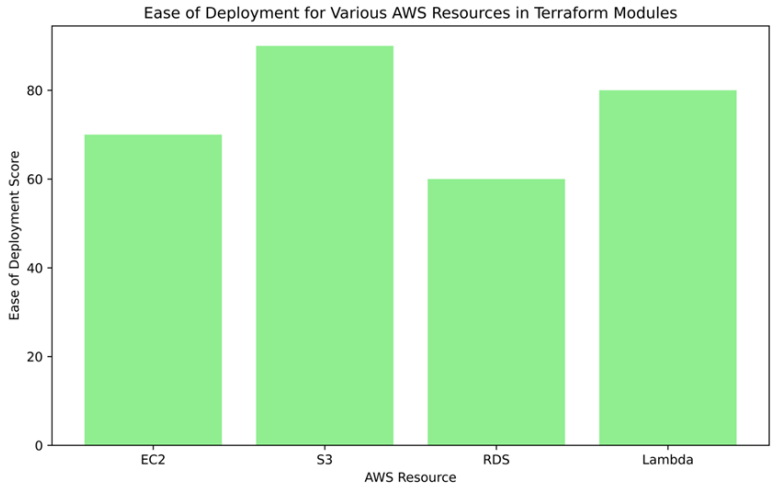

Figure 7: Terraform Ease of Deployment

The above diagrams illustrate hypothetical metrics for various AWS resources when managed through Terraform modules, including cost efficiency, ease of deployment, and scalability. Each bar represents a different AWS resource (EC2, S3, RDS, Lambda), with the height indicating the score or percentage for the respective metric. This visualization helps in comparing these aspects across different AWS resources within Terraform modules.

CONCLUSION

Terraform Modules offer a powerful and efficient way to manage AWS resources, providing numerous benefits such as reusability, simplified management, scalability, and enhanced collaboration. By leveraging the community-driven ecosystem and real-time use cases, organizations can significantly improve their infrastructure management processes, leading to more efficient, scalable, and cost-effective cloud environments.

REFERENCES

[1]. Vydra, Z. (2020). "Automated Creation and Testing of Virtual Network Environments."
[2]. Kushnir, S.Y. (2019). "Кластерна серверна система на базі Raspberry PI та Terragrunt." PDF available in Ukrainian at Cherkasy National University repository.
[3]. Stawiarz, D.J. (2015). "The Use of Cloud Technologies for Storage and Processing Data."
[4]. Valtanen, V. (2018). "Improving the Maintainability and Developer Experience of Terraform Code."