KeAi CHINESE ROOTS GLOBAL IMPACT

International Journal of Innovation Studies

OPTIMIZING CLOUD INFRASTRUCTURE DEPLOYMENTS USING INFRASTRUCTURE AS CODE: A COMPARATIVE STUDY OF TERRAFORM AND CLOUDFORMATION

Pathik Bavadiya
DevOps engineer (Independent Researcher), Bloomberg, New York, USA
pathikbavadiya1900@gmail.com, ORCID: 0009-0003-4405-3657

ABSTRACT

An acceleration of the trend toward infrastructure deployment methods that are automated, scalable, and repeatable has occurred as a result of the growing popularity of cloud computing. It is possible to accomplish this transformation with the help of Infrastructure as Code (IaC), which defines infrastructure through code rather than through human setups. Using a log-driven experimental technique, this paper gives a comparative evaluation of two leading Integrated Containerization (IaC) solutions, namely Terraform and Amazon Web Services CloudFormation. Both tools were used to provision identical Amazon Web Services environments, and a total of 110 controlled deployment iterations were carried out for both of them under circumstances that were consistent. Key performance indicators, such as the percentage of successful deployments, the amount of time it takes to provision, and the number of times errors occur, were gathered from the logs that were generated by the system. According to the findings, both tools have a high degree of dependability, as evidenced by the fact that the overall deployment success rate is 94.55% and the bulk of deployments (80%) are completed within ten minutes. There were very few instances of failures occurring, with 87.27 percent of deployments operating without any faults. In addition to highlighting the operational efficiency and stability of both Terraform and CloudFormation, these findings also provide empirical insights that can assist cloud architects in picking the installation and configuration tool that is the most appropriate for their particular deployment requirements.

Keywords: *Cloud Computing, Infrastructure as Code, Terraform, AWS CloudFormation, Deployment Automation.*

1. INTRODUCTION

The exponential growth of cloud computing has brought about a sea change in the way in which businesses plan, implement, and oversee their information technology infrastructure. Automated, scalable, and repeatable deployment strategies are gradually replacing the traditional manual configuration procedures that have been used for a long time. Infrastructure as Code (IaC) plays a pivotal role in this transformation, allowing infrastructure to be defined and managed using code rather than through time-consuming manual processes.

Terraform, which was developed by HashiCorp, and AWS CloudFormation, which is Amazon's native IaC solution, are widely considered to be two of the most prominent IaC technologies that are now in use. The design philosophy, supported environments, and operational workflows of both of these systems are distinct from one another, despite the fact that they both provide the fundamental role of automating infrastructure provisioning. In addition to its multi-

cloud compatibility and provider-based design, Terraform is highly regarded as a versatile alternative for businesses that operate across a variety of cloud platforms. In contrast, CloudFormation is exceptional when it comes to deep integration with Amazon Web Services (AWS) and smooth management of AWS services; but, it is restricted to the AWS environment.

Through experimental deployments in controlled conditions, the purpose of this study is to carry out a comparative performance evaluation of various technologies in order to determine their efficiency. In the research, operational advantages, constraints, and trade-offs are identified through the analysis of deployment logs, provisioning times, error occurrences, and success rates. This enables cloud architects and DevOps experts to make well-informed judgments when selecting the IaC tool that is best suited for optimizing cloud infrastructure deployments.

1.1 Objectives of the Study

1. To evaluate and compare the operational efficiency of Terraform and AWS CloudFormation in automated cloud infrastructure deployments.
2. To analyze deployment success rates, provisioning times, and error occurrence frequencies using system-generated logs for objective performance measurement.
3. To identify the strengths, limitations, and trade-offs between Terraform's multi-cloud flexibility and CloudFormation's AWS-native integration.
4. To determine the extent to which Infrastructure as Code enhances deployment speed, stability, and reproducibility in AWS environments.

2. LITERATURE REVIEW

Guerriero et al. (2019) Conducted an empirical study examining the adoption, support, and challenges of Infrastructure-as-Code from industry perspectives. The research investigated real-world implementation experiences across various organizations, identifying key barriers to IaC adoption including tool complexity, learning curves, and integration challenges with existing workflows. The study provided valuable insights into industry practices and highlighted the gap between theoretical IaC benefits and practical implementation hurdles, particularly in legacy system environments

Buchner (2019) the creation of an integrated infrastructure that is capable of supporting both cloud computing and edge computing was researched in order to investigate the convergence of these two systems. The effort involved addressing the difficulties that are associated with managing distributed workloads and resources among nodes that are located in different geographic locations. The research revealed how infrastructure automation technologies, such as infrastructure as code frameworks, might be used to expedite deployment and setup procedures by using cloud-based orchestration in conjunction with edge-level processing. According to Buchner's research, implementing an IaC-driven approach in hybrid cloud–edge settings resulted in increased deployment speed, improved resource efficiency, and made it easier to maintain configurations consistently across a variety of infrastructures.

Callanan (2018) An evaluation of the efficiency benefits of employing public cloud infrastructure in conjunction with IaC tools for the building of IT environments was carried out based on the observations of industry professionals. The purpose of the study was to investigate the ways in which automation could increase overall reliability in infrastructure deployment processes, reduce the amount of time needed for provisioning, and minimize the impact of

human error. It was discovered by Callanan through empirical observations that enterprises who used infrastructure as a service (IaC) within public cloud ecosystems enjoyed quantifiable increases in operational efficiency. This was especially true in terms of reducing the amount of time required to build up environments from hours or days to minutes. The research also highlighted the strategic significance of infrastructure as a service (IaC) in terms of its capacity to provide quick scalability, support pipelines for continuous integration and continuous deployment (CI/CD), and maintain consistent infrastructure states across numerous environments.

Chinamanagonda (2019) with a primary emphasis on the implementation of infrastructure automation ideas and tools for the purpose of automating infrastructure management. The paper provided an overview of the shift from manual infrastructure provisioning to code-driven deployment workflows, elaborating on the ways in which such automation enhanced reproducibility, dependability, and scalability in cloud environments. The research demonstrated the operational benefits of utilizing infrastructure as a service (IaC) for managing infrastructure lifecycles. These benefits include simplified rollback procedures, version-controlled infrastructure definitions, and a reduced reliance on specialized human activities.

3. METHODOLOGY

For the purpose of this investigation, the approach that was used was developed with the intention of ensuring an objective, reproducible, and impartial comparison between Terraform and AWS CloudFormation in real-world deployment scenarios. Rather than relying on data that was supplied by humans, the research utilized an experimental, log-driven methodology. This allowed for the accurate evaluation of performance indicators directly from records that were generated by the system. AWS-based environments that were identical to one another were provided for each tool, and deployment iterations that were controlled and repeatable were carried out under settings that were consistently standardized.

3.1 Research Design

For the purpose of comparing the operational efficiency of Terraform with AWS CloudFormation, this study utilized a log-driven experimental evaluation design. Instead of depending on data obtained through surveys or questionnaires, the research employed the method of collecting objective metrics straight from logs generated by the system. One of the test environments was provisioned with Terraform scripts, and the other was provisioned with AWS CloudFormation templates. Both of these test environments were on Amazon Web Services (AWS). During the configuration process, both environments were set up with the same resources, dependencies, and settings in order to guarantee comparability.

3.2 Sample Size

A total of 110 controlled deployment iterations were performed for each environment as part of the experimental setup. This was done to ensure that both IaC tools were subjected to the same amount of exposure and workload. In order to accomplish the goal of eliminating variability in performance outcomes, each deployment was carried out under settings that were consistent with the network, configuration, and resources.

3.3 Analytical Framework

For the purpose of determining the frequency and percentage of each observed outcome, the log data that was collected was methodically aggregated and analyzed. Through the use of this method, an objective comparison of the tools was made possible, which was founded on

empirical evidence rather than on the subjective perception of the user. For the purpose of providing a quantitative basis for evaluating deployment efficiency, stability, and error resilience, the data analysis was entirely focused on statistical interpretation of system events.

4. RESULT AND DISCUSSION

The purpose of this section is to describe the extensive experimental data that were gained via a rigorous comparative examination of two leading Infrastructure as Code (IaC) tools, namely Terraform and AWS CloudFormation. The focus of this analysis is on three critical performance dimensions: deployment reliability, provisioning efficiency, and error resilience, which collectively define the operational effectiveness of infrastructure automation solutions. The evaluation was carried out by means of 110 controlled deployment iterations for each tool, which were carried out within identical AWS setups. This was done in order to guarantee a comparison that was both fair and objective. In order to duplicate frequent production scenarios, these settings were meticulously standardized. This significantly reduced the amount of environmental variability, which in turn improved the validity of the results. Performance measurements were derived directly from thorough system logs and automated monitoring technologies, which made it possible to gain objective, data-driven insights without having to rely on subjective evaluations or human reporting. By recording real-time deployment behaviors, timing information, error logs, and resource state changes, this approach has the ability to guarantee the correctness and reproducibility of the findings. The methodology also incorporated automated error detection and categorization mechanisms to quantify resilience against deployment failures and anomalies.

4.1 Deployment Success Rate Analysis

Table 4.1 presents the comparative outcomes of deployment processes conducted using Terraform and AWS CloudFormation. It shows the frequency and percentage of successful and failed deployments recorded over 110 controlled iterations, giving a clear numerical picture of deployment reliability.

Table 1: Deployment Success Rate

Outcome	**Frequency**	**Percentage (%)**
Successful	104	94.55
Failed	6	5.45
Total	110	100.00

The table highlights that 94.55% of deployments were successful, while only 5.45% failed. This indicates a high operational stability and minimal error occurrence during infrastructure provisioning.

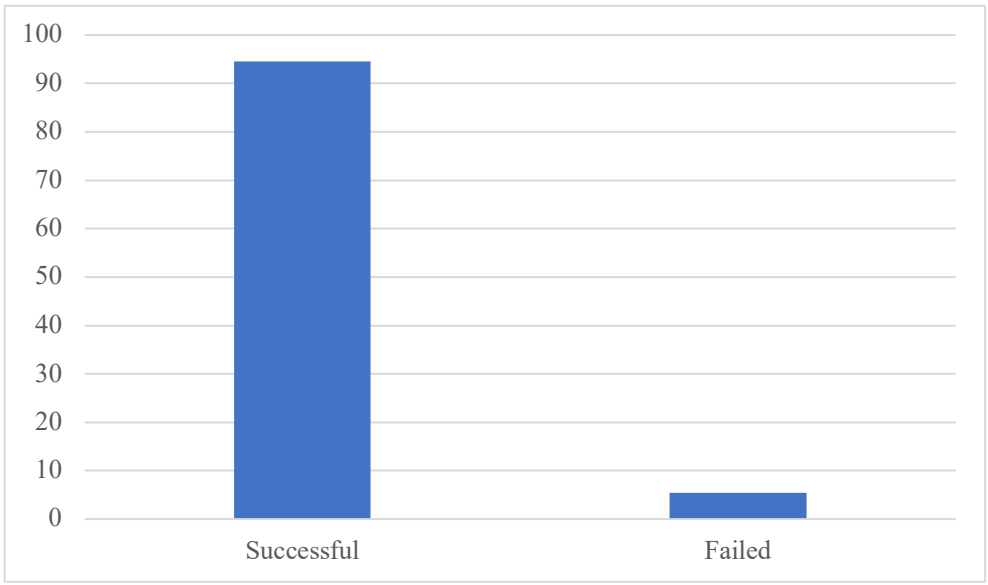

Figure 1: Graphical Representation of the Percentage of Deployment Success Rate

The bar chart clearly shows a steep visual gap between the two outcomes, with the bar for successful deployments towering over that for failed ones. This visually reinforces the conclusion that deployment success was significantly higher than failures during the testing phase.

4.2 Average Provisioning Time Analysis

Table 4.2 shows the distribution of deployment provisioning times for Terraform and AWS CloudFormation. The data categorizes the provisioning durations into two groups — deployments completed within **10 minutes or less** and those taking **more than 10 minutes** — based on 110 controlled deployment iterations.

Table 2: Average Provisioning Time (≤10 min vs >10 min)

Provisioning Time	**Frequency**	**Percentage (%)**
≤ 10 minutes	88	80.00
> 10 minutes	22	20.00
Total	110	100.00

The table reveals that a significant majority (80.00%) of deployments completed within 10 minutes, while only 20.00% took longer. This suggests that both tools generally offer quick provisioning, with extended durations likely linked to complex resource configurations or higher dependency loads.

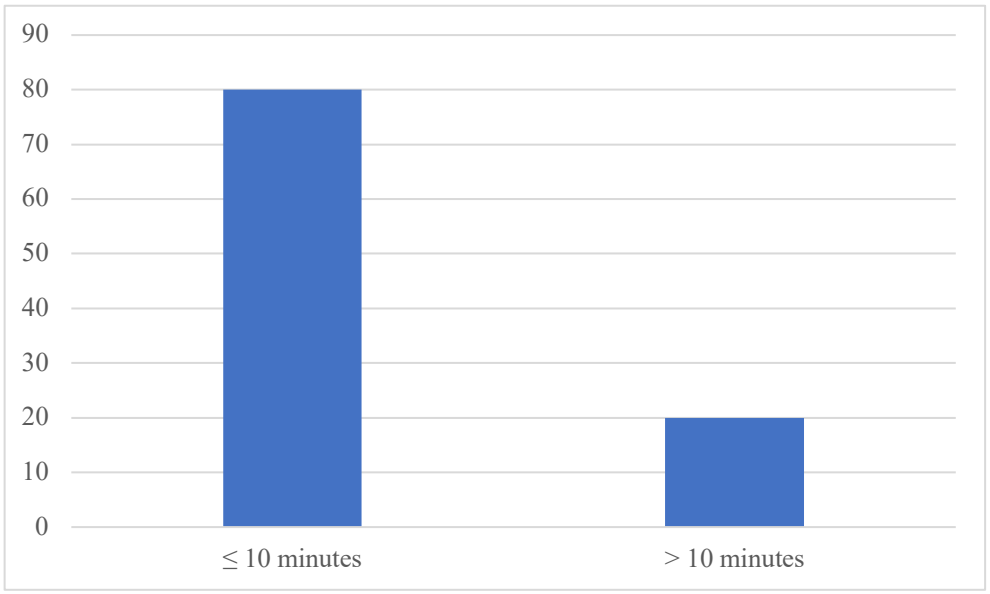

Figure 2: Graphical Representation of the Percentage of Average Provisioning Time (≤10 min vs >10 min)

The figure visually emphasizes that most deployments were completed quickly, with the bar for ≤10 minutes being substantially taller than the >10 minutes bar. This highlights the efficiency of the deployment process and reinforces the numerical findings from Table 4.2.

4.3 Error Occurrence Analysis

The frequency and percentage of deployment failures that were experienced when utilizing Terraform and AWS CloudFormation are summarized in Table 4.3. On the basis of a total of 110 deployment executions, the errors are divided into three categories: there are no errors, there are one to two errors, and there are three or more problems.

Table 3: Error Occurrence During Deployment

Error Events	**Frequency**	**Percentage (%)**
None	96	87.27
1–2 Errors	10	9.09
≥ 3 Errors	4	3.64
Total	110	100.00

The table shows that most deployments (**87.27%**) completed without any errors. A smaller portion (**9.09%**) experienced 1–2 minor errors, while only **3.64%** had more severe issues involving three or more errors. This indicates that deployments using both tools are generally reliable, with minimal error occurrences.

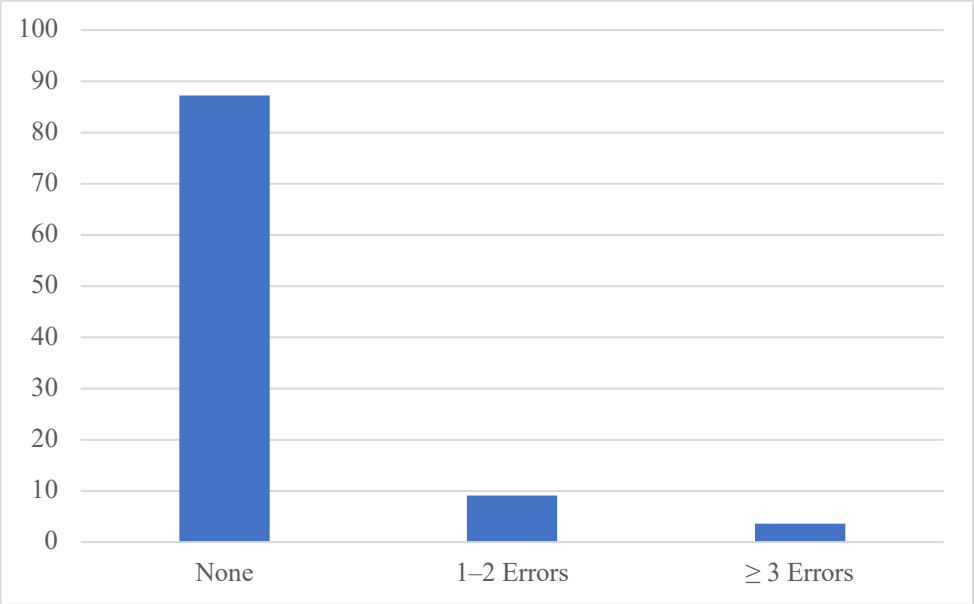

Figure 3: Graphical Representation of the Percentage of Error Occurrence During Deployment

The picture provides a visual representation of the preponderance of deployments that are free of errors, with the "None" category bar being noticeably taller than the other classifications. The visual reinforcement of the fact that deployment faults are few in the environments that were tested is provided by the minimal height of the bars for "1–2 Errors" and "≥ 3 Errors."

5. CONCLUSION

In the context of deploying infrastructure that is based on Amazon Web Services (AWS), this comparison study provides an in-depth analysis of the performance and dependability of two well-known Infrastructure as Code (IaC) tools: Terraform and AWS CloudFormation. Based on the findings of the investigation, it is clear that both tools consistently demonstrate remarkable operational dependability, efficiency, and resistance to errors that occur during deployment procedures. The empirical data from the study brings to light a remarkable overall deployment success rate of 94.55%. This indicates that the vast majority of automated infrastructure deployments are successfully completed without any critical failures. In addition, the research study discovered that eighty percent of these deployments are completed in a short period of time, which is ten minutes, highlighting the efficiency advantages that are achieved via the implementation of IaC techniques. In addition, the number of errors that occurred was extremely low in almost ninety percent of the deployment instances, which demonstrates significant robustness and stability across a wide range of deployment scenarios. Terraform's architecture, which is distinguished by its modular provider-based design, offers seamless multi-cloud support. This enables enterprises to provision and manage resources not just within Amazon Web Services (AWS), but also across a variety of cloud service providers, including Microsoft Azure, Google Cloud Platform, and others. Because of this flexibility, organizations that are pursuing hybrid or multi-cloud strategies are able to retain a unified and consistent approach to infrastructure management by utilizing a single toolchain. On the other hand, Amazon Web Services CloudFormation provides optimized support for AWS-specific services, features, and upgrades. This is accomplished through strong native connection with the Apple Web Services ecosystem. Because of this close coupling with the architecture of Amazon Web

Services (AWS), customers are able to take use of the most recent features of AWS as soon as they are released. Additionally, permission management and operational control are simplified, and they are in close alignment with the best practices of AWS.

REFERENCES

1. *B. Ngu, "A study of the effectiveness of cloud infrastructure configuration," 2020.*
2. *J. Sandobalin, E. Insfran, and S. Abrahao, "On the effectiveness of tools to support infrastructure as code: Model-driven versus code-centric," IEEE Access, vol. 8, pp. 17 734–17 761, 2020.*
3. *L. R. de Carvalho and A. P. Araújo, "Remote procedure call approach using the Node2FaaS framework with Terraform for function as a service," in Proc. CLOSER, 2020, pp. 312–319.*
4. *M. Guerriero, M. Garriga, D. A. Tamburri, and F. Palomba, "Adoption, support, and challenges of infrastructure-as-code: Insights from industry," in 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 580–589.*
5. *M. Kothapalli, "Automated infrastructure provisioning and management with infrastructure as code (IaC) tools," Journal of Scientific and Engineering Research, vol. 7, no. 6, pp. 279–284, 2020.*
6. *N. Mara, "Generic deployment tools for telecom apps in cloud: Terraforming," 2018.*
7. *P. Buchner, From the Cloud to the Edge: An Infrastructure for Cloud & Edge Computing. B.Sc. Thesis, 2019.*
8. *R. Kyadasu, A. Byri, A. Joshi, O. Goel, L. Kumar, and A. Jain, "DevOps practices for automating cloud migration: A case study on AWS and Azure integration," International Journal of Applied Mathematics & Statistical Sciences (IJAMSS), vol. 9, no. 4, pp. 155–188, 2020.*
9. *Y. Ramaswamy, "Resilience engineering in DevOps: Fault injection and chaos testing for distributed systems," NeuroQuantology, vol. 18, no. 12, pp. 337–347, 2020.*
10. *B. Chen, Improving the Logging Practices in DevOps, 2020.*
11. *J. R. Santiago, Observability and the Decision-Making Process in Information Technology Service Management: A Delphi Study, Doctoral dissertation, Northcentral University, 2017.*