سومین کنفرانس بین المللی
مهندسی برق، کامپیوتر، مکانیک
و تکنولوژی های نوین مرتبط با هوش مصنوعی
The 3th international conference of electrical engineering, computer mechanics and new technologies related to artificial intelligence

An Implementation of Infrastructure as Code for Automating Cloud Resource Provisioning

Maryam Mayabi Joghal
Ferdowsi University of Mashhad
mayabi@um.ac.ir

Zeinab Khadem
Ferdowsi University of Mashhad
z.khadem@um.ac.ir

Abstract

Managing IT infrastructure presents significant challenges, including high costs and risks of changes, slow issue resolution, misconfigurations, complex disaster recovery, and reliance on infrastructure administrators. This article explores an Infrastructure as Code (IaC) approach to overcome these issues, drawing on software development practices like version control and automated testing.

The proposed IaC implementation ensures rapid, secure provisioning of cloud resources aligned with organizational needs, following key principles of reproducibility, idempotency, composability, and evolvability (RICE). It includes three layers: Decision, Deployment, and Documentation. The Decision Layer evaluates user requests against technical and managerial policies, rules, and constraints to optimize infrastructure selection. The Deployment Layer implements configurations based on predefined templates, ensuring consistency and reliability. The Documentation Layer aids process improvement, informed decision-making, disaster recovery, and security analysis.

The implementation utilizes YAML, Python, and Ansible. The results demonstrate an 83% reduction in provisioning time, significantly enhancing efficiency and accuracy while reducing failures. This automation shifts the focus of infrastructure specialists from maintenance to code improvement, optimizing resource management. In conclusion, the presented IaC implementation closely adheres to the RICE principles, providing a highly reproducible, idempotent, and evolvable process for the rapid and secure delivery of compute resources tailored to the organization's specific requirements.

Keywords: Infrastructure as Code, Cloud, Automation, Infrastructure Provisioning, Virtualization

Introduction

The main challenges in managing IT infrastructure include the high costs and risks of making changes, the difficulty in quickly resolving issues and restoring service after failures, the complexity of maintaining secure configurations across the environment, the resource-intensive nature of disaster recovery planning, and the reliance on infrastructure administrators (McKendrick, 2023). These issues underscore the need for a more automated, resilient approach to infrastructure management that reduces manual effort and improves overall reliability and responsiveness (Wang, 2022).
Infrastructure as Code represents an approach to infrastructure automation that draws upon practices from software development (Morris, 2020). By making changes to code and then leveraging automation to test and apply those changes across systems, this paradigm enables infrastructure to be version controlled, tracked, and managed in the same way as application code. In many cases, organizations can leverage the same tooling, processes, and procedures for their infrastructure that they employ for their application development workflows (McKendrick, 2023).
In implementing infrastructure as code, it is necessary to pay attention to multiple aspects. The key principles of IaC (RICE) include reproducibility, idempotency, composability, and evolvability (Wang, 2022). In a well-designed structure, the testing and delivery process should be easily executed (Morris, 2020). As a result, the development, retrieval, and troubleshooting process is accelerated. Additionally, given that each organization has different requirements, regulations, and infrastructure, these factors must be considered in the implementation. With the focus on automation, attention to security considerations also becomes particularly important (Kumara et al., 2021).
This article aims to explain an infrastructure as code implementation for the rapid and secure delivery of a compute resource (virtual machine) that is also aligned with the organization's requirements. In this implementation, the proposed solution follows the RICE principles. In addition to providing the infrastructure, all initial configurations are performed based on the security and organizational requirements.

Related works

The realm of Infrastructure as Code (IaC) has witnessed significant advancements and innovations over recent years, with research and development efforts focusing on various aspects of automating and optimizing IT infrastructure management. These endeavors can be broadly categorized into three groups.
The first category focuses on modeling cloud infrastructure using various solutions, which are then translated into code by IaC tools. These research endeavors aim to simplify provisioning and deployment processes across various cloud platforms. For example, Sandobalin et al. (2017) and Shvetcova et al. (2019) introduced models using Domain Specific Language (DSL) to abstract and represent cloud infrastructure, facilitating the automatic generation of provisioning scripts and simplifying the deployment process across different cloud platforms like AWS, Azure, and OpenStack. Additionally, Tankov et al. (2021) and Dalvi (2022) aimed to empower developers with self-service templates, ensuring governance and role-based access control while streamlining the provisioning process.
The second category focuses on addressing specific challenges with IaC. Lavriv et al. (2018) proposed methods for automating disaster recovery processes, while Chandra Patni et al. (2020) deployed infrastructure for web applications on the Azure platform. These works emphasize speed, cost reduction, and complexity transparency. Additionally, Kodolov et al. (2020) managed three network laboratories by automatically configuring physical and virtual equipment, facilitating multiple scenarios, and preventing configuration errors through the use of Ansible and GitHub. Furthermore, Sabil et al. (2022) proposed an application built using IaC for Data Warehouse Infrastructure, enabling immediate use for Business Intelligence purposes.
The third category represents the latest trend in leveraging advanced technologies like chatbots and machine learning for infrastructure provisioning. Sinha et al. (2023) and Popoviciu et al. (2023) introduced chatbots that utilize natural language processing to understand user input and automate infrastructure provisioning tasks. These advancements aim to simplify interaction with IT systems and make infrastructure provisioning more accessible to non-technical users, marking a significant leap forward in the evolution of IaC.

Implementation Overview

This implementation consists of three layers: Decision, Deployment, and Documentation. Each layer plays a specific role in the process, contributing to improved efficiency and reduced errors. The Figure1 diagram illustrates the relationship between these layers. Below, we detail the responsibilities of each layer:

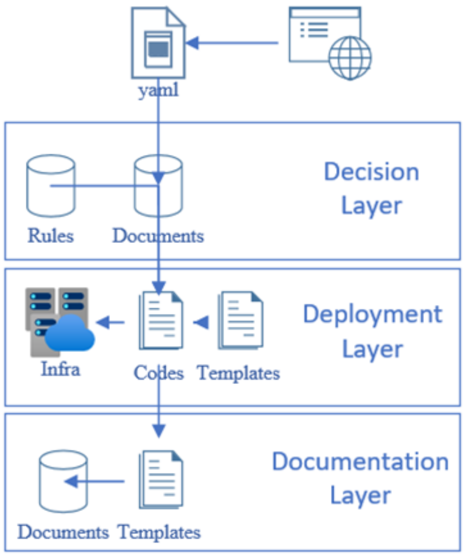


Figure 1 The relationship between layers

1. Decision Layer

    The Decision Layer receives and evaluates the YAML file created based on the user's request. This evaluation is conducted according to existing policies and rules, which can be categorized into technical and managerial.

    Managerial rules include access, permissions, costs, and organizational policies. Technical rules cover thresholds, conditions, requirements for each requested infrastructure, and infrastructure limitations.

    After evaluating the request, additional requirements are added based on the rules. This layer can leverage algorithms such as artificial intelligence and machine learning to select the optimal infrastructure. The output of this layer is a YAML file that includes the request, requirements, and decision-making results.

2. Deployment Layer

    This layer is responsible for executing commands and configuring infrastructure based on the output from the Decision Layer. Depending on the type of requested infrastructure, relevant codes are invoked and executed. This deployment is based on pre-prepared templates.

    The final result of this layer is the complete implementation of the requested infrastructure. The success or failure of this stage is then sent to the Documentation Layer.

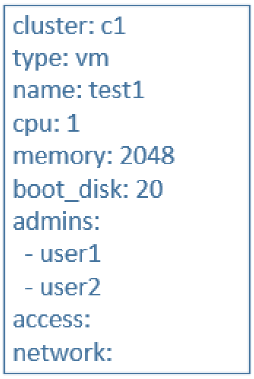
3. Documentation Layer

    Based on the previous stage's result, if successful, the changes made and the system's status are automatically recorded and documented. These documents include request details such as the requester and requested items, details of the provided infrastructure, and related code information, such as their versions.

    In case of failure, the relevant logs are recorded and reported to the responsible party. The documentation prepared by this layer aids in improving future processes and better decision-making. This layer plays a crucial role in ensuring the stability and transparency of infrastructure systems.

This layer automatically logs and documents changes made and the system's current status. Documentation is essential not only for troubleshooting but also for enabling teams to optimize and prevent repeating errors. Moreover, accurate and up-to-date documentation facilitates future decision-making processes, allowing tracking changes and analyzing past trends. Ultimately, this layer ensures continuous process improvement, enhancing the overall efficiency and reliability of the system.

The implementation of the mentioned layers uses Python and Ansible. Rules are configured based on cloud cluster infrastructure limitations, infrastructure optimization, and access restrictions. Additionally, rules such as the uniqueness of the virtual machine name in each cluster are included to improve the efficiency of the proposed architecture. These rules are converted into Python and Ansible codes and evaluate the user's YAML request file. The Figure 2 shows the user's YAML request file.



```yaml
cluster: c1
type: vm
name: test1
cpu: 1
memory: 2048
boot_disk: 20
admins:
  - user1
  - user2
access:
network:
```

Figure 2 User's YAML request

Conclusion

The implementation of the proposed system has yielded significant results compared to traditional methods. Traditionally, the process of creating and configuring virtual machines typically took over an hour and required the involvement of multiple administrators and infrastructure specialists, with documentation often being neglected. With the proposed automation system, the deployment time has been reduced to less than 10 minutes, indicating an improvement of over 83% in execution time. Additionally, the need for specialized expertise has decreased, and documentation is performed automatically and accurately. These improvements have not only increased the efficiency and accuracy of processes but also optimized the management of infrastructure resources. In effect, the workload of infrastructure specialists has shifted from maintaining and physically developing the infrastructure to improving and developing related codes.
This infrastructure as code implementation is designed to closely adhere to the key RICE principles. The three-layer architecture, consisting of Decision, Deployment, and Documentation, ensures a highly reproducible and idempotent process. The Decision Layer evaluates the user's YAML request against predefined rules, enabling consistent and reliable decision-making, while the Deployment Layer utilizes pre-prepared templates to ensure idempotent infrastructure provisioning. The comprehensive Documentation Layer tracks changes and system status, facilitating continuous process improvement and making the implementation evolvable over time. By aligning with the RICE principles, this infrastructure as code solution delivers compute resources in a rapid, secure, and tailored manner, optimized for the organization's specific requirements.

References

Dalvi, A. (2022, November). Cloud infrastructure self service delivery system using infrastructure as code. In *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* (pp. 1-6). IEEE.
Kodolov, S. D., Klimova, A. S., Aksyonov, K. A., & Filimonov, A. Y. (2020, May). Using the IaC Approach for Building a Distributed Laboratory Complex for Modern Communication Infrastructures Investigation. In *2020*

Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT) (pp. 0440-0443). IEEE.

Kumara, I., Garriga, M., Romeu, A. U., Di Nucci, D., Palomba, F., Tamburri, D. A., & van den Heuvel, W. J. (2021). The do’s and don’ts of infrastructure code: A systematic gray literature review. *Information and Software Technology*, *137*, 106593.

Lavriv, O., Klymash, M., Grynkevych, G., Tkachenko, O., & Vasylenko, V. (2018, February). Method of cloud system disaster recovery based on" Infrastructure as a code" concept. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 1139-1142). IEEE.

McKendrick, R. (2023). *Infrastructure as Code for Beginners*. Packt Publishing Ltd.

Morris, K. (2020). Infrastructure as code. O'Reilly Media.

Patni, J. C., Banerjee, S., & Tiwari, D. (2020, July). Infrastructure as a code (IaC) to software defined infrastructure using Azure Resource Manager (ARM). In *2020 International Conference on Computational Performance Evaluation (ComPE)* (pp. 575-578). IEEE.

Popoviciu, C., Sobieski, J., Jabbari, B., Sawyer, C., & Zhang, L. (2023, November). A No Code Approach to Infrastructure Provisioning in Support of Science. In *2023 IEEE Future Networks World Forum (FNWF)* (pp. 1-2). IEEE.

Sabil, A. A. S., Martono, H. Y., & Permatasari, D. I. (2022, August). Effective Building Data Warehouse Infrastructure by Code in Cloud Platform. In *2022 International Electronics Symposium (IES)* (pp. 418-424). IEEE.

Sandobalin, J., Insfran, E., & Abrahao, S. (2017, June). An infrastructure modelling tool for cloud provisioning. In *2017 IEEE international conference on services computing (SCC)* (pp. 354-361). IEEE.

Shvetcova, V., Borisenko, O., & Polischuk, M. (2019, September). Domain-specific language for infrastructure as code. In *2019 Ivannikov Memorial Workshop (IVMEM)* (pp. 39-45). IEEE.

Sinha, G., Chapagain, R., Budhathoki, A., Sarkar, K., Mandal, A. K., & Manorishik, O. (2023, April). Infrastructure as a Code Chatbot using Natural Language Processing. In *2023 International Conference on Inventive Computation Technologies (ICICT)* (pp. 567-571). IEEE.

Tankov, V., Valchuk, D., Golubev, Y., & Bryksin, T. (2021, November). Infrastructure in code: towards developer-friendly cloud applications. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 1166-1170). IEEE.

Wang, R. (2022). *Infrastructure as Code, Patterns and Practices: With Examples in Python and Terraform*. Simon and Schuster.