AWS powered cloud research environment PaaS

Diego Frazatto Pedroso, Lucas Eduardo Gulka Pulcinelli, Wiliam Akihiro Alves Aisawa and Sarita Mazzini Bruschi

[1] *ICMC/USP - Institute of Computer Sciences and Mathematics*
São Carlos, Brazil

{diegopedroso, lucasegp, alveswill}@usp.br, sarita@icmc.usp.br

Abstract. *Cloud computing has transformed organizational IT resource management. Among the prominent cloud service models, Infrastructure as a Service and Platform as a Service provided by Amazon Web Services stand out as integral components of modern cloud computing ecosystems. We've employed open-source tools to create a configurable infrastructure pipeline using Kubernetes to achieve high availability and consistency for research projects. Our platform integrates modern open-source stacks in order to reduce delivery time and complexity and increase observability, leveraging tools such as IaC and load testing frameworks. These practices blend academic and industry principles for comprehensive deployment and management of cloud-based applications.*

1. Overview and Introduction

Cloud systems have revolutionized software development by offering services with different levels of abstraction. These services are commonly classified into three categories: Infrastructure as a Service (IaaS), granting users complete control over their systems; Software as a Service (SaaS), delivering managed individual functionalities to applications; and Platform as a Service (PaaS), offering a comprehensive environment facilitating rapid deployments with minimal configuration.

Another important concept in cloud computing is the adoption of microservices, contrasting with monolithic systems. Microservices consist of smaller components with specific responsibilities, thereby enhancing scalability and performance. Practically, their implementation involves decomposing a large system into multiple smaller units to facilitate simultaneous development. This approach yields a high number of components, releases, processes, and tools being utilized.

Due to the extensive utilization of microservices and cloud services, achieving seamless integration and automation across multiple stacks presents a challenge. This challenge arises from factors such as compatibility issues, cost constraints, and administrators' proficiency in these environments.

In this scenario, research [Beyer et al. 2018] suggests that 70% of system interruptions result from poorly executed implementations, excessive integrations, deficient processes, and inadequate monitoring. Consequently, a robust and comprehensive cloud framework becomes essential in the lifecycle of an application. Therefore, the development of such a comprehensive framework is imperative within the realm of academic research.

2. Encouraging experimentation by utilizing cloud-native Infrastructure and Platform as a Service

Cloud-native environments offer numerous fully managed components, marked by fragmentation, distribution, and multiple layers of abstraction, aimed at improving user accessibility (Picoreti, 2018). Given this context, it is crucial to replicate behaviors and environments that align with these characteristics. Consequently, we have created a service management and test generation platform leveraging AWS infrastructure to fulfill our objectives.

Figure 1 presents a comprehensive diagram that illustrates the operational framework and the components that make up the overall system.

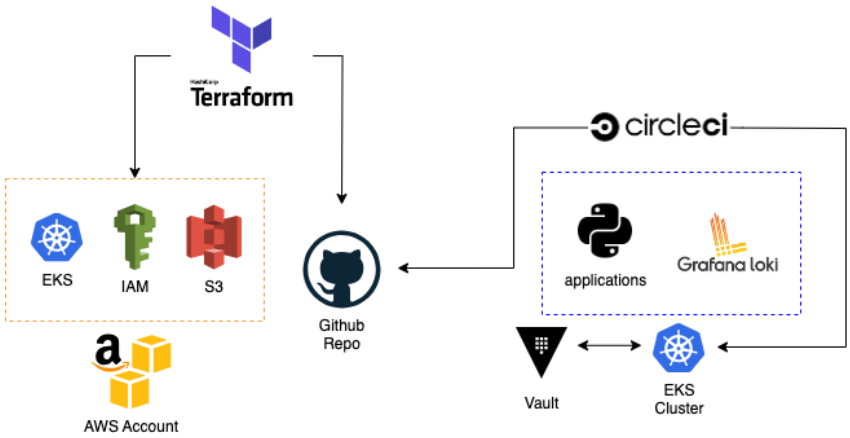

Figure 1. AWS Platform-as-a-Service Top-Down

Initially, the basic cloud infrastructure and services are defined using Infrastructure as Code (IaC) via Terraform and Helm, which allows most constructs to be created in a stateless manner as atomic pieces that can be rolled back as needed[1]. Such a model ensures maximum consistency not only inside the infrastructure and its components but also within the code of the applications operating on the framework, enabling easy replicability. Consequently, it streamlines large-scale experimentation for users, ensuring consistent execution, providing operational oversight, controlling costs, and implementing a robust governance model [Fowler 2016].

Once defined, the Continuous Delivery and Continuous Integration (CI/CD) pipelines act as key components that improve consistency and streamline execution without compromising stability. This is achieved through automated tests and a unified pipeline for all applications, abstracting low-level structures from users and facilitating seamless integration among applications, including with monitoring systems derived from the Grafana Labs Stack.

Within the observability context, monitoring, logging and traces are available by default for all defined applications in a centralized location using Grafana. This automatic configuration is facilitated partly by the Prometheus monitoring infrastructure and partly by the Istio service mesh, ensuring a meticulous monitoring environment with minimal development overhead.

[1]https://www.hashicorp.com/resources/what-is-mutable-vs-immutable-infrastructure

Finally, a unified load testing platform plays a critical role in experimentation. Due to different application requirements, we created a single interface for two load testing frameworks (namely, Locust and Gatling) allowing users to script tests using different languages based on their familiarity. Due to integrations, the results are stored for an extended period (typically months), allowing for dynamic analysis concurrent with ongoing tests [Awada 2018].

3. Discussion and Results

Through meticulous orchestration of these processes, we significantly reduce the likelihood of errors across various categories. Errors are detectable at multiple stages prior to final version approval.

The implementation of load-testing mechanisms facilitates a comprehensive examination of resource consumption patterns within the code base. These testing tools offer a detailed perspective on the operational dynamics of the system, providing comprehensive visibility and insight across all layers, from infrastructure to application.

As a result , developers can focus their efforts on precise improvements within the code base, focusing exclusively on segments requiring optimization. This approach mitigates the need for extensive refactoring or modification of code segments that have negligible discrepancies in metrics or observability points.

The future trajectory of monitoring practices within distributed systems suggests a move toward greater independence. Traditional monitoring tasks, such as creating analytic solutions, incident dashboards, alerts, and troubleshooting, have historically relied heavily on manual efforts of expert analysts. These manual and repetitive tasks necessitate an intuitive understanding of the monitoring landscape.

Figure 2. Socks-Shop Polyglot architecture

Figure 2 illustrates an experimental system depicting a virtual store with a diverse stack of programming languages, frameworks, databases, and more. This system is used together with the platform developed in a recent paper[Pulcinelli et al. 2023]. By utilizing the PaaS model provided by our platform, users can seamlessly modify and conduct experiments in a secure, parallel, and controlled manner. This facilitates agility, innovation, and risk-free testing, ultimately improving the development and optimization processes.

Operational processes are gradually shifting towards greater automation. Although many organizations in the distributed software domain currently handle day-to-day operations manually, there is a growing need to adopt an event-driven paradigm. Events, resulting from systemic interactions, provide developers with a comprehensive, top-down view of the current state of the system.

The effectiveness of system testing and operation depends on several factors, such as scale, contextual requisites, financial resources, and operational capacity. Achieving optimal system validation isn't always possible, highlighting the crucial importance of automated pipelines, especially in the PaaS context.

These automated processes provide essential alternatives to potentially cumbersome manual operations, streamlining tasks and ensuring unwavering consistency without the need for numerous manual interventions. We anticipate that the articulated approach, encompassing a distributed, resilient, observable, consistent, and secure platform model, could act as a catalyst for the development of innovative platforms to support the operation and foundation of complex systems.

4. Acknowledgment

We thank Amazon Web Services (AWS) for their generous sponsorship of our project, in collaboration with the Laboratory of Distributed Systems and Concurrent Programming at the Institute of Mathematical and Computer Sciences (ICMC). We also gratefully acknowledge the financial support provided by the São Paulo State Research Foundation (FAPESP) under grant #2019/26702-8.

References

Awada, U. (2018). Application-container orchestration tools and platform-as-a-service clouds: A survey. *International Journal of Advanced Computer Science and Applications*.

Beyer, B., Murphy, N. R., Rensin, D. K., Kawahara, K., and Thorne, S. (2018). *The site reliability workbook: practical ways to implement SRE*. " O'Reilly Media, Inc.".

Fowler, S. J. (2016). *Production-ready microservices: building standardized systems across an engineering organization*. " O'Reilly Media, Inc.".

Pulcinelli, L. E. G., Pedroso, D. F., and Bruschi, S. M. (2023). Conceptual and comparative analysis of application metrics in microservices. In *2023 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, pages 123–130.