Agnostic Cloud Services with Kubernetes

João Bonacho[0009−0004−9758−1733], Carlos Gonçalves[0000−0001−9113−6269], and A. Luís Osório[0000−0001−5680−9114]

ISEL - Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, Portugal, a46014@alunos.isel.pt, carlos.goncalves@isel.pt, luis.osorio@isel.pt

Abstract. The vendor lock-in concept represents a customer's dependency on a particular supplier or vendor, eventually becoming unable to easily migrate to a different one. Cloud computing is frequently associated with vendor lock-in restrictions, motivated by the proprietary technological arrangements of each cloud provider. In this paper, an agnostic cloud provider model is proposed as a solution to address such challenges, focusing on the establishment of a model for deploying and managing computational services in cloud environments. Concretely, it aims to enable computer systems to be executed agnostically on multiple cloud platforms and infrastructures, thereby decoupling them from any cloud provider. Moreover, this model intends to automate service deployment by generating container images and defining the running configurations for the services. Within this context, container technology is deemed as an efficient and standard strategy for deploying computational services across cloud providers, promoting the migration of applications between vendors. Additionally, container orchestration platforms, which are becoming increasingly adopted by organizations, are essential to effectively manage the life-cycle of multi-container computer systems by monitoring their performance, and dynamically controlling their behavior. In particular, the Kubernetes platform, an emerging open standard for cloud services, is proving to be a valuable contribution on achieving service agnostic deployment, namely with its Cloud Controller Manager mechanism, helping abstracting specific cloud providers. As a validation for the proposed solution, it is intended to prove model's adaptability to different services and technologies supplied by heterogeneous organizations, through the deployment of containerized applications in alternative cloud providers, public or on-premises.

Keywords: Agnostic Cloud · Containers · Kubernetes · Cloud Computing · Software Deployment · Distributed Systems · Vendor Lock-In.

1 Introduction

Deploying informatics systems (applications) in multiple cloud providers can be very useful since organizations can leverage the unique features and capabilities of each provider, in order to optimize their operations and to benefit from the currently available offerings. In addition, by being able to distribute workloads

(i.e., a unit of processing) across multiple providers, organizations can also improve the performance and reliability of their systems. Nevertheless, deploying applications on different cloud platforms can be complex and demanding, as it can require significant technical expertise, and be very time-consuming and costly. The management and maintenance of services deployed across multiple providers is also an arduous task, especially when it involves coordinating heterogeneous platforms and ensuring their seamless integration. The vendor lock-in concept is closely associated with these challenges and concerns.

Cloud computing is a paradigmatic example of vendor lock-in restrictions, due to the proprietary technologies used by each cloud provider. Accordingly, [1] describes the vendor lock-in in cloud computing as the difficulty and complexity that customers face when attempting to switch between cloud service providers without facing technological barriers, and significant costs or legal limitations.

This paper presents and discusses the development of a model for deploying and managing computational services in cloud environments, which emerged as the result of a Master's Thesis. This work aims to reduce vendor lock-in in cloud computing, namely by decoupling informatics systems from any specific provider. The research question focuses on establishing a vendor-agnostic cloud provider model, streamlining the deployment and management of computational services, mainly in a cloud context. Specifically, the objective is to develop a system that enables systems to be executed on multiple cloud platforms without being tied to a particular provider, whether public or on-premises platforms.

The proposed solution to address the referred challenge considers the use of container technology, as an efficient strategy of deploying computational services to cloud providers. The proposed model uses containers instead of virtual machines as containers offer faster deployment and startup, cost effectiveness, and ease of migration. Containers provide a standardized and portable method for packaging and deploying applications along with their dependencies, provisioning a consistent and isolated environment for each service. Furthermore, container orchestration platforms are essential for the effective management of multi-container applications at scale, across cloud providers. Among orchestrators' landscape, the Kubernetes [2] platform (K8s) is recognized as the *de facto* standard and ideal platform for this purpose, particularly because of its Cloud Controller Manager (CCM) [3] component. This mechanism enables the abstraction of cloud providers, allowing for integration across different cloud platforms.

2 Related Work

The related work in this research topic revolves around platforms that offer comprehensive capabilities related to infrastructure automation and management, primarily for the deployment of applications and services in cloud environments. Within this scope, several technologies and platforms can be identified and categorized as follows: Infrastructure as Code (IaC) [4] tools, including platforms as Terraform [5] and CloudFormation [6], which streamline the deployment and management of applications; Multi-cloud orchestration frameworks, including

Cloudify [7] and Heat [8], that provide resource and application orchestrating capabilities across multiple cloud platforms; and Cloud Computing platforms, namely OpenStack [9] and Anthos [10], providing diverse services for efficient application deployment and management in the cloud.

Furthermore, when developing novel solutions for such a rapidly evolving area as the provisioning and management of computing services in cloud environments, a review of the related research in terms of similar academic solutions, key advances and existing gaps is of paramount importance.

3 System Architecture

The proposed solution aims to establish an interface between organizations and IT infrastructures (on-premises or cloud-based platforms), to support the agnostic deployment of organizations' services across different computing platforms.

The model's architecture for this solution is illustrated in Fig. 1. From an general perspective, the **Model Interface** component is responsible for interacting and communicating with organizations that want to deploy their services on a remote infrastructure; **Services Configuration** component defines and creates the running configurations for the service to be deployed, and creates the required container images. It also exposes an API to manage the deployed services. The **Agnostic Cloud Interface** component is responsible for abstracting different cloud providers, by selecting the appropriate cloud provider and configuring the services to conform to provider's policies and specifications. Furthermore, it provides an object for connecting to the Kubernetes cluster, incorporating its configuration and access tokens. Finally, the **Kubernetes Manager** component creates the required Kubernetes objects to fulfill the requirements of organization's services, and instantiates them on a previously created Kubernetes cluster, hosted on the cloud provider's infrastructure or on-premises. In both cases, it uses a K8s client application ([11]) to communicate with clusters' API server.

4 Implementation and Validation

The reference implementation of the platform was supported by the Java ecosystem. Maven ([12]) was also used to manage project's dependencies, and generate the JAR artifacts. However, this implementation can be accomplished using other technological ecosystem, allowing for a modular architecture in order to reduce dependencies and promote flexibility. The platform provides a command-line interface (CLI) and a drag-and-drop User Interface (UI), supported by Eclipse Theia [13], for organizations to deploy and manage their services. Both interfaces provide operations to manage services, receiving as input parameters: service name, container image, port, provider (cloud providers or on-premises), and K8s namespace. Users can deploy, remove, update and migrate a service between providers, as well as list running services and get services' metadata.

The validation of the developed prototype involved the deployment and management of Java web services, built into container images using Docker platform

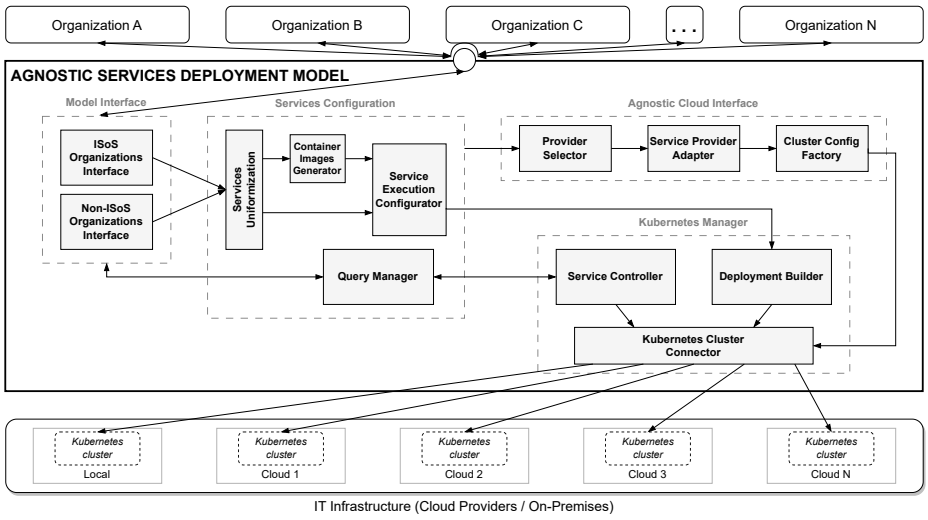

Fig. 1. Model architecture

[14], across multiple cloud platforms, namely Microsoft Azure, Google Cloud Platform (GCP), Amazon Web Services (AWS), and a local development environment (via Minikube [15]), leveraging the aforementioned functionalities offered by the platform. All operations were successfully completed, after closely monitoring clusters' and services' behavior via providers' web interfaces.

5 Conclusions and Future Work

This paper presented and discussed the development of a model for deploying and managing computational services in cloud environments, and simplified implementation and validation of the corresponding prototype.

It is argued that organizations, by adopting the similar platforms mentioned in Section 2, remain dependent on their services and specifications, as each platform requires the writing of configuration files with proprietary syntax and commands. Ultimately, this dependency can make it difficult to migrate applications between platforms. This work is designed to face this issue, as it aims to establish an open standard for an agnostic cloud. The objective of deploying services, through containers, on diverse IT infrastructures was achieved, freeing organizations from worrying about configurations and specific details of cloud platforms. Hence, it minimizes and mitigates the dependency on any vendor, infrastructure or platform, which effectively addresses vendor lock-in challenges.

As future work, it is proposed to programmatically build Kubernetes clusters on cloud providers, to have a more refined error control in clusters' communication, and to extend service management capabilities. Finally, it is imperative to perform additional testing on multiple operations of the platform to prove its adaptability to different services supplied by heterogeneous organizations.

References

1. Opara-Martins, J., Sahandi, R., Tian, F.: Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. Journal of Cloud Computing **1**(5), 1–18 (2016)
2. Kubernetes, https://kubernetes.io/docs/concepts/overview. Last accessed 8 Jun 2023
3. Cloud Controller Manager — Kubernetes, https://kubernetes.io/docs/concepts/architecture/cloud-controller/. Last accessed 10 Jun 2023
4. Infrastructure as Code — IBM, https://www.ibm.com/topics/infrastructure-as-code. Last accessed 9 Jun 2023
5. Terraform — HashiCorp Developer, https://developer.hashicorp.com/terraform. Last accessed 8 Jun 2023
6. Provision Infrastructure as Code - AWS CloudFormation - AWS, https://aws.amazon.com/cloudformation. Last accessed 8 Jun 2023
7. What Is Cloudify? — Cloudify Documentation Center, https://docs.cloudify.co/4.3.0/about/introduction/what-is-cloudify. Last accessed 8 Jun 2023
8. Heat - OpenStack, https://wiki.openstack.org/wiki/Heat. Last accessed 8 Jun 2023
9. Open Source Cloud Computing Infrastructure - OpenStack, https://www.openstack.org. Last accessed 9 Jun 2023
10. Hybrid Cloud Management with Anthos — Google Cloud, https://cloud.google.com/anthos. Last accessed 8 Jun 2023
11. GitHub - fabric8io/kubernetes-client: Java client for Kubernetes & OpenShift, https://github.com/fabric8io/kubernetes-client. Last accessed 8 Jun 2023
12. Maven – Welcome to Apache Maven, https://maven.apache.org/. Last accessed 8 Jun 2023
13. Theia - Cloud and Desktop IDE Platform, https://theia-ide.org/. Last accessed 9 Jun 2023
14. Docker: Accelerated, Containerized Application Development, https://www.docker.com/. Last accessed 9 Jun 2023
15. minikube start — minikube, https://minikube.sigs.k8s.io/docs/start/. Last accessed 9 Jun 2023