Agnostic Cloud Services with Kubernetes

João Bonacho, **Carlos Gonçalves** and **Luís Osório**
ISEL - Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa (IPL), 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. This work proposes an agnostic cloud provider model 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 informatics 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 defining the running configurations for the services and generating container images. Within this context, container technology is deemed as an efficient and standard strategy for deploying 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 manage the life-cycle of multi-container informatics 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 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 multiple cloud providers, public or on-premises.

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

Objectives

This work 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. Therefore, the following requirements must be considered:

- Develop an interface to receive services from organizations to be deployed and managed on cloud platforms;
- Build a component for defining services' execution configurations;
- Deploy and manage services on a Kubernetes (K8s) cluster, ensuring that cluster's components are properly created and maintained;
- Enable effortless vendor switching for organizations, facilitating seamless migration of services across different cloud providers;
- Test and validate the platform employing multiple services, containers, and cloud providers to prove its adaptability to heterogeneous organizations requirements.

System Architecture

The model's architecture for the solution is represented in *Fig. 1*. Summarily, the **Model Interface** component is responsible for interacting and communicating with organizations that intend to deploy services on a remote infrastructure; **Services Configuration** component defines 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 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, leveraging Cloud Controller Manager (CCM) features [1]. In both cases, it uses a K8s client application [2] to communicate with clusters' API server.

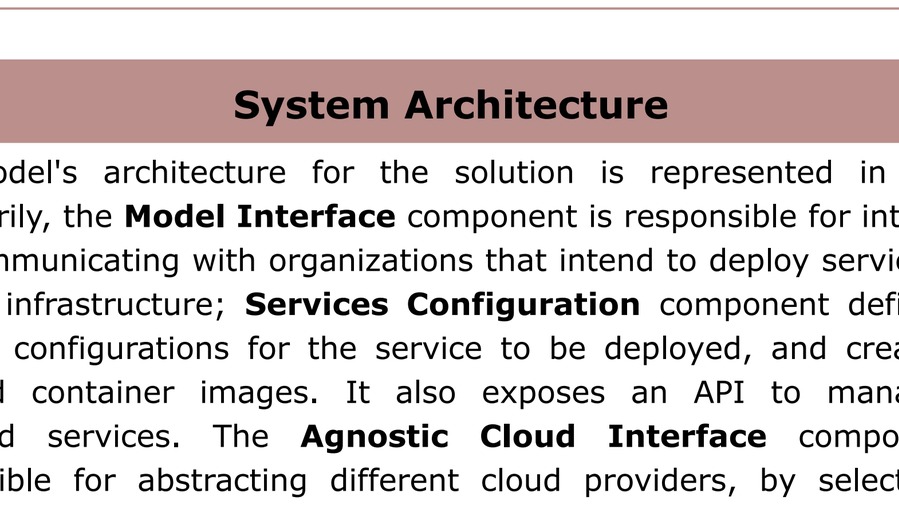

Fig. 1

Implementation / Validation

The reference implementation of the platform was supported by the Java and Maven ecosystem. The platform provides an API, a Java-based Command-Line Interface and a Web Interface for organizations to deploy and manage their services. Both interfaces leverage API functionalities, providing operations to manage services, receiving as input parameters: service name, container image, port, provider, and K8s namespace. Users can deploy, remove, update and migrate a service between providers, as well as list running services and get a specific service information. *Fig. 2* displays a web service configuration opened in the editor of the Web Interface, featuring Microsoft Azure as target provider.

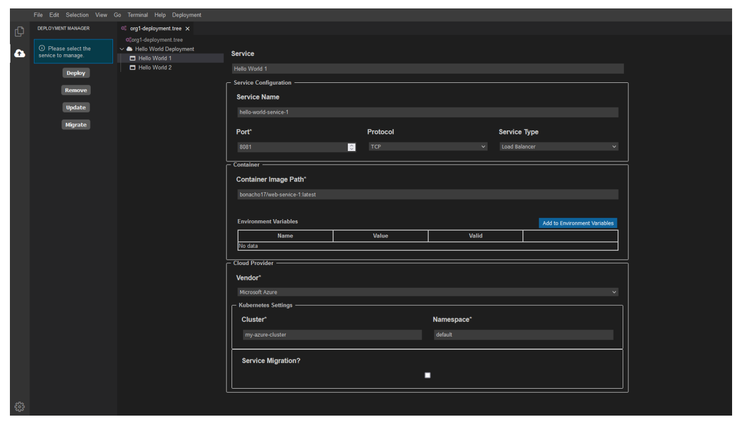

Fig. 2

The selected use cases, reflecting real-world challenges, for the validation of this system include: deploying "Hello World" web services on diverse cloud providers; the migration of services between cloud providers; dynamic services updates triggered by CI/CD pipelines in GitLab repositories; and the deployment of real systems from IPL projects.

Conclusions and Future Work

The developed work was designed to face organizations' difficulty of migrating applications between platforms, aiming 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 cloud-specific configurations. Hence, it mitigates the dependency on any vendor, effectively addressing vendor lock-in challenges [3]. As future work, it is proposed to programmatically build K8s clusters, to extend service management API capabilities, and to have enhanced error control in clusters' connection is also imperative.

References

[1] Cloud Controller Manager - Kubernetes, https://kubernetes.io/docs/concepts/architecture/cloud-controller
[2] GitHub - Java client for Kubernetes & OpenShift, https://github.com/fabric8io/kubernetes-client
[3] 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, 1-18 (2016)