Bidimensional Cross-Cloud Application Management with TOSCA and Brooklyn

Jose Carrasco, Javier Cubo, Francisco Durán, and Ernesto Pimentel

Universidad de Málaga, Dept. Lenguajes y Ciencias de la Computación, Spain
{josec,cubo,duran,ernesto}@lcc.uma.es

Abstract. The diversity in the way different cloud providers offer their services, give their SLAs, present their QoS, support different technologies, etc., complicates the portability and interoperability of cloud applications, and favors vendor lock-in. Standards like TOSCA, and tools supporting them, have come to help in the provider-independent description of cloud applications. After the variety of proposed cross-cloud application management tools, we propose going one step further in the unification of cloud services with a deployment tool in which IaaS and PaaS services are integrated into a unified interface. We provide support for applications whose components are to be deployed on different providers, indistinctly using IaaS and PaaS services. The TOSCA standard is used to define a portable model describing the topology of the cloud applications and the required resources in an agnostic, and providers- and resources-independent way. We include in this paper some highlights on our implementation on Apache Brooklyn and present a non-trivial example that illustrates our approach.

Keywords: Cloud applications, multi-deployment, cross-cloud, standards, TOSCA, Brooklyn, IaaS, PaaS.

Trans-cloud application deployment by unifying IaaS and PaaS

In recent years, Cloud Computing has experienced a growth in the demand of its services. The Cloud promotes access on demand to a large number of resources through out three service models, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), which allow cloud providers to offer a set of useful features for current IT requirements of the sector, among which we find scalability and elasticity. Given the interest on this computing model, vendors such as Google, Amazon, Cloud Foundry, etc. have implemented their solutions by developing their own cloud service layers with custom APIs that expose their resources. Most of these providers offer a set of similar services as regards functionality, but developed according to their own specifications. For example, each supplier specifies his own SLA or QoS, supports a concrete set of technologies, etc. The proliferation of these solutions has increased the number of issues to be addressed in cloud computing, mainly related to the diversity of providers and solutions, provoking the vendor lock-in problem, and hampering the portability and interoperability in cloud services.

In order to mitigate this heterogeneity and get a vendor-agnostic solution, independent tools and frameworks have emerged as the result of integrating, under a single interface, the services of multiple public and private providers. Over such interfaces, most of these solutions build models of application topologies, dependencies and used resources, independently of the providers in which services will be executed. Thus, they offer a portable and interoperable environment where developers can describe their systems and select the resources that better fit their requirements, without worrying about technical details of the services' use, and focusing on the required features.

In a very short time, these platforms have evolved according to the mode in which users can take advantage of integrated cloud services to expose and run their systems. Terms such as *multi-cloud*, *cross-cloud*, *federated clouds*, or *inter-clouds* have been used for deployment platforms with the ability of distributing modules of an application using services from different providers. The main differences between these terms are the different ways of handling the connections between modules deployed on different platforms. However, in all these attempts, platforms allow operating simultaneously with a single level of service to deploy applications, i.e., all the components of an application are deployed either at the IaaS level or all at the PaaS level. From this, with the goal of unifying cloud services, we propose a second dimension in which deployment tools integrate IaaS and PaaS levels under a single interface. Then, this will allow developers to deploy their applications combining services offered by providers at any of these levels. Following the evolution in terminology, *multi-/cross-/inter-cloud*, we envision *trans-cloud* management tools without the limitations we currently have. The idea behind *trans-cloud* is to be able to build our applications by using available services and resources offered by different providers, at IaaS, PaaS or SaaS level, using virtual machines or containers, according to our needs and preferences.

In this work, we present our proposal towards *trans-cloud* application deployment by unifying IaaS and PaaS of multiple vendors. Specifically, we propose using a provider-agnostic TOSCA-based model of the topology of applications and their required resources, indistinctly using IaaS and PaaS services, which can be used for their deployment using Apache Brooklyn.

Having an agnostic model of our system may greatly simplify migration, or simply decision change. Indeed, with our approach, each component may be deployed at one level or the other just by changing its location. Much has been said on choosing IaaS or PaaS for our applications. We may decide to go for PaaS when we want to provide flexibility or scalability to our applications without managing it ourselves. Or for IaaS, when we need control over our infrastructure, e.g., to accessing, monitoring or managing our virtual machines, storage or networking. By simplifying so much the change from IaaS to PaaS, and vice versa, for some modules, we are contributing to a real adaptation to our needs by minimizing the effort by the user. The underlying management tool will be in charge of the required provisioning and interoperation.