HAL
open science

Towards Model-Driven Multi-Cloud Resource Management

Fawaz Paraiso, Stéphanie Challita, Yahya Al-Dhuraibi, Philippe Merle

To cite this version:

Fawaz Paraiso, Stéphanie Challita, Yahya Al-Dhuraibi, Philippe Merle. Towards Model-Driven Multi-Cloud Resource Management. [Research Report] Inria Lille - Nord Europe. 2016. hal-01534785

HAL Id: hal-01534785

https://inria.hal.science/hal-01534785v1

Submitted on 8 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards Model-Driven Multi-Cloud Resource Management

Fawaz Paraiso Stephanie Challita Yahya Al-Dhuraibi Philippe Merle

University of Lille & Inria Lille - Nord Europe
LIFL UMR CNRS 8022, France
Email: firstname.lastname@inria.fr

Abstract

Multi-Cloud computing has established itself as a paradigm of choice for acquiring resources from different providers and get the best of each of them to run their applications. However, managing resources in Multi-Cloud remains a challenging task. Several multi-cloud libraries based approaches include Apache Libcloud, Apache jclouds, δ-cloud, Daseincloud, fog, and pkgcloud exist in the cloud market. But these are still low level as Multi-Cloud management tasks must always be programmed. Therefore, application platforms are need to help developers to succeed. In this paper we give a model-driven approach for managing resources of multiple clouds at high level. We illustrate our proposal by providing a prototype of Multi-Cloud Designer for managing resource from multiple clouds.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

General Terms Systems

Keywords Multi-Cloud, Resource, Resource Management, Model, Docker

1. Introduction

Nowadays, multi-cloud computing [8] is becoming a reality. On the one hand, there is a plethora of public cloud providers such as Amazon EC2, Google Compute Engine, Microsoft Azure to name a few, and of cloud software stacks like VMware ESX, OpenStack, CloudStack, OpenNebula, Eucalyptus to build private or public clouds. Each cloud platform provides its own Cloud Resource Management API (CRM-API) allowing administrators to (de)provision, (re)size and manage their cloud resources. However these CRM-API are very low level, often incarnated as a networked REST API only usable by experts, and are heterogeneous limiting interoperability between clouds. In addition a cloud platform is often equipped with a graphical interface built on top of the CRM-API and appropriated for manual management tasks only. On the other hand, using multiple clouds simultaneously allows one to optimize operational costs, to deal with peaks of service requests, to react to cloud failures, etc. To deal with the heterogeneity of CRM-API, several multi-cloud libraries exist like Apache Libcloud[1], Apache jclouds[2], δ-cloud[3], Daseincloud[4], fog[5], and pkgcloud[6]. But these are still low level as multi-cloud management tasks must always be programmed. Thus, high level abstractions for managing resources of multiple clouds are required.

In this paper we propose a model-driven approach for managing resources in Multi-Cloud environments. Our approach assists users to manage resources from multiple clouds.

The rest of the paper is organized as follows. In Section 2 we describe our approach. Next, in Section 3 we presents the result of experiments of managing an application in Multi-Cloud environments. Then, in Section 4 we discuss some related work. Finally, Section 5 concludes the paper with future direction.

2. Multi-Cloud resource management

In this section we present our approach. We begin by giving an overview of the solution architecture. Then, we present the *Multi-Cloud Designer*. Finally we describe how the Multi-Cloud resources are modeling.

2.1 Architecture overview

As depicted in Figure 1, the architecture presented is divided into five layers: *Multi-Cloud Designer*, *Multi-Cloud Modeling*, *Models@Run.time*, *Multi-Cloud*, and *Virtualization &*

[1] https://libcloud.apache.org/

[2] https://jclouds.apache.org/

[3] https://deltacloud.apache.org/

[4] http://dasein-cloud.sourceforge.net/

[5] http://fog.io/

[6] https://github.com/pkgcloud/pkgcloud

[Copyright notice will appear here once 'preprint' option is removed.]

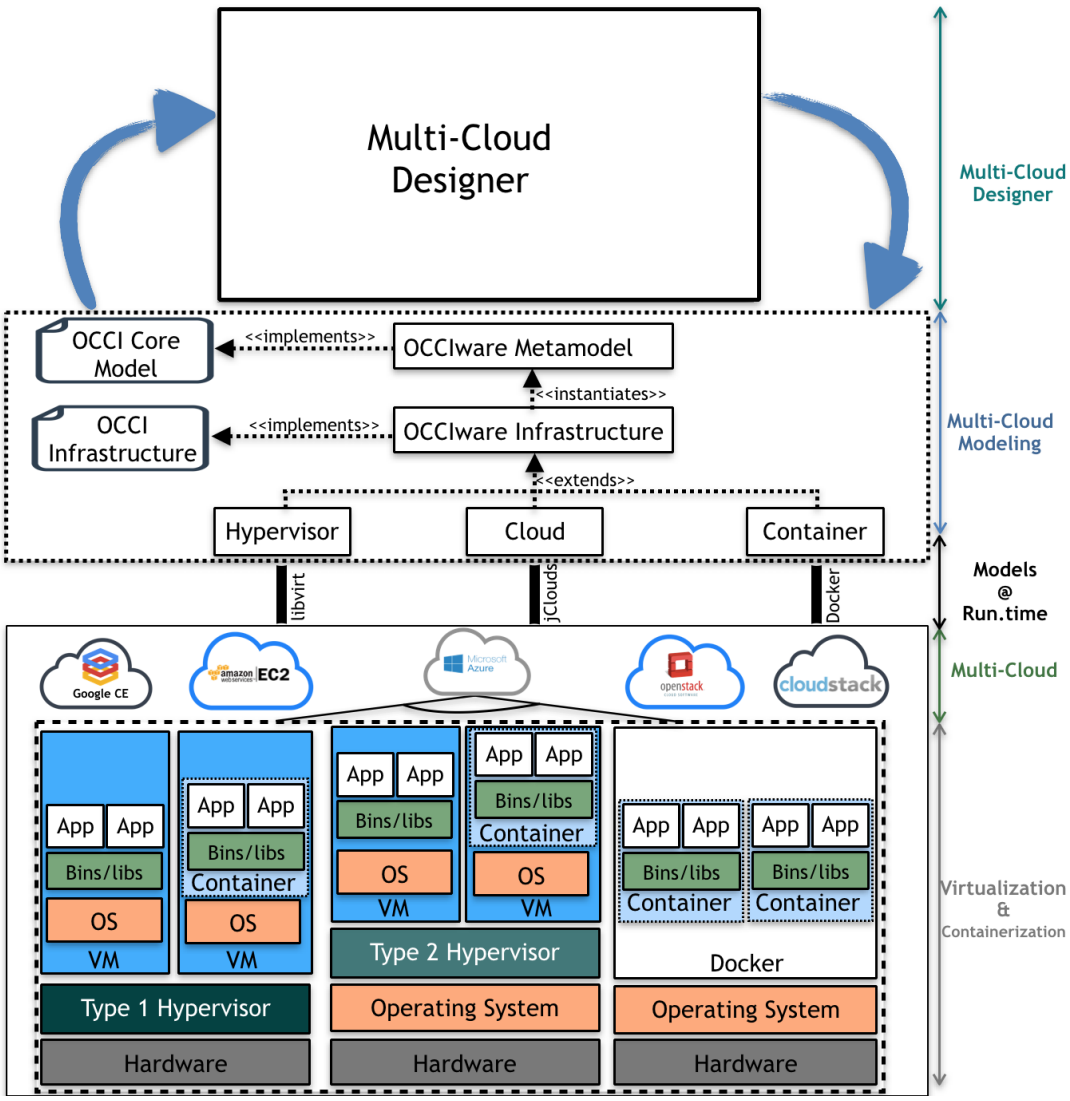

Figure 1. Architecture for managing Multi-Cloud resource.

Containerization. A brief description of these layers (from top to bottom) is provided below:

- **Multi-Cloud Designer**: this layer gives a graphical tool for designing, developing, and managing resources from multiple clouds. This tool interacts with the Multi-Cloud Modeling layer.
- **Multi-Cloud Modeling**: is the model defined that allows users to represent different kinds of resources at levels of Hypervisor, public Cloud, and Container.
- **Models@Run.time**: based on the Multi-Cloud Modeling layer, the Models@Run.time provides the execution of these models with runtime capability.
- **Multi-Cloud**: this layer represents the target cloud environments where the resources are provisioned, and managed.
- **Virtualization & Containerization**: relies on hypervisor-based and container-based virtualization solutions that are used by cloud providers infrastructure.

2.2 Multi-Cloud Designer

The *Multi-Cloud Designer* tool in Figure 1 offers a structured approach and intuitive graphical interface to users for designing, deploying and managing resources across multiple clouds. This tool abstracts management of Multi-Cloud resources at high level. It provides an explicit way to manage resources. Using our Multi-Cloud Designer, users can design and manage resources from multiple clouds. These resources are provided by public clouds, hypervisor and containers. This tool represents all the concepts of Multi-Cloud resources defined in the underline model. Then, users can design a Virtual Machine (VM) and application containers by interacting with the target model.

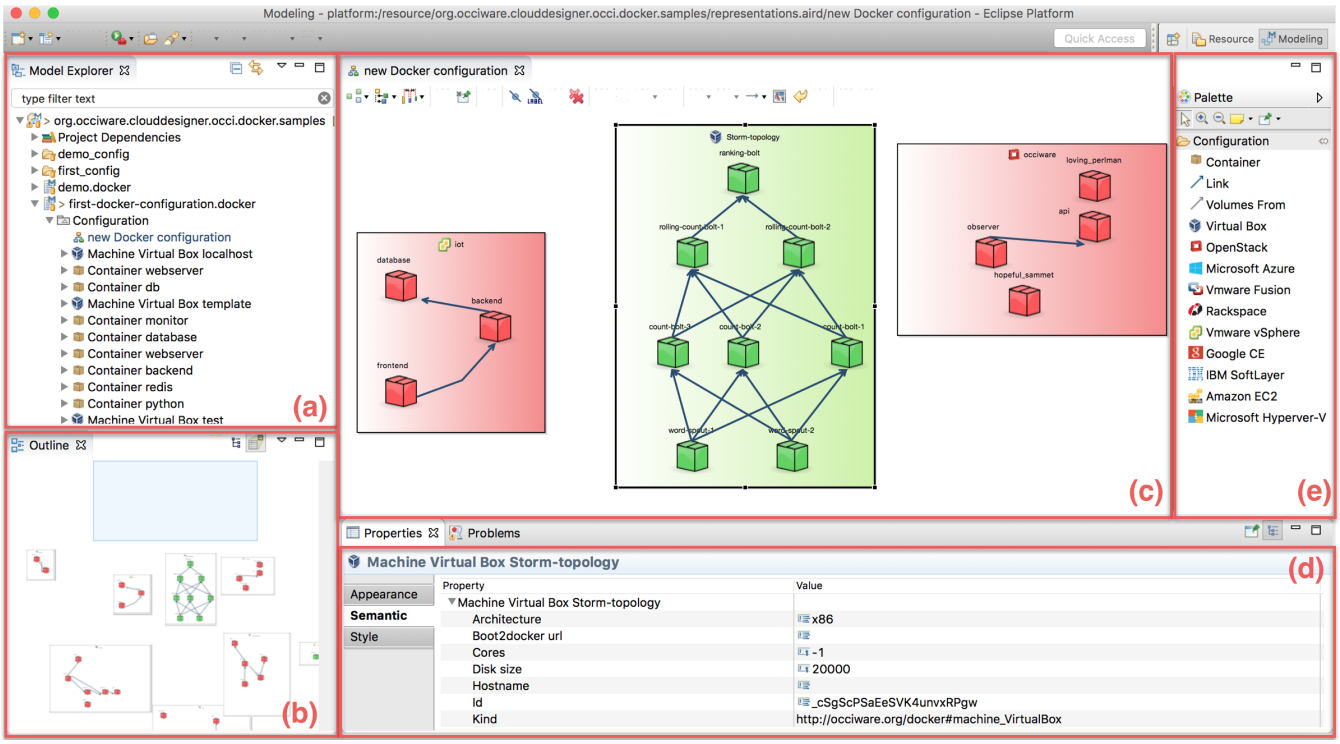

Figure 2. Multi-Cloud Docker Designer.

2.3 Multi-Cloud Modeling

Our metamodel for OCCI, named OCCIware [9] METAMODEL, is based on the Eclipse Modeling Framework (EMF) [10]. In the OCCIware project[7], we promote to use models as high level abstractions for managing multi-cloud resources.

As shown in Figure 1, our model-driven approach is based on a precise metamodel [6] of Open Cloud Computing Interface (OCCI), an OGF's specification defining an open interface for managing any kind of cloud computing resources (IaaS, PaaS, and SaaS). This metamodel is encoded with EMF. Cloud infrastructure resources, aka compute, network and storage, are abstracted by our **Infrastructure** model. The **Infrastructure** model is extended to manage specific aspects of **Hypervisor**s, **Cloud** platforms/stacks, and **Container**s. Together, these three models allow to model resources of multiple clouds simultaneously and seamlessly.

2.4 Models at Run.time

The execution of the models relies on what we called connectors. The connectors provide the connection between models and running systems. Basically, connectors take a source model defined by a metamodel and project it into the running systems. Conversely, connectors monitor and introspect the running systems and transform them into the target models. As shown in Figure 1 we have three connectors: *Hypervisor*, *Cloud*, and *Docker*. The *Cloud* connector is specific to Cloud model, it performs the interactions between the model and the public clouds. The *Hypervisor* connector is specific to Hypervisor model, and the *Docker* connector is specific to container model.

The connectors are based on the following principles:

- **Transformation**: the connectors provide expressive model transformation techniques based on design patterns, which facilitate the specification of translations between the models and running systems.
- **Introspection**: to introspect the running system, the connectors employ Model-Driven Engineering (MDE) techniques, which handle the introspection and analysis of the system at higher level of models. Using MDE techniques, different models describing certain constraints are derived and maintained at runtime.
- **Synchronisation**: the connectors provide incremental synchronization between a running system and models. To detect model modifications, the connectors rely on a notification mechanism that reports when source model element has been changed. To synchronize the changes of model with the running system, the connectors check if model elements are still consistent by navigating efficiently between the source model and running system model using the correspondence model.

[7] `http://www.occiware.org/`

3. Experimental setup

For our experiments, we implemented[8] a prototype of Hypervisor Designer, Cloud Designer, and Docker Designer as a set of plugins for Eclipse Integrated Development Environment (IDE) respectively for Hypervisor, Cloud, and Docker models. Our Multi-Cloud Designer can be downloaded here[9].

In this work, we are going to illustrate only Docker Designer. A screenshot of the current Docker Designer is depicted by Figure 2. Frame (a) in Figure 2 shows the Eclipse Model Explorer used to navigate through a Docker projects containing a Docker Model. Frame (b) in Figure 2 gives a perspective or a global view of the modeled containers. Obviously, this view can be adjusted to provide the most optimal perspective. Frame (c) displays the design area that provides a graphical representation of Docker Model. Frame (c) displays the design area that provides a graphical representation of Docker Model. As shown in Frame (c), the model elements are green or red. The green color of machine or container elements shows the **started** state of containers and host machines. The red color shows the **stopped** state of containers and the host machine. Frame (d) in Figure 2 contains the Eclipse properties editor for visualizing and modifying attributes of a selected modeling element.

The Hypervisor, Cloud and Docker connectors are implemented using Libvirt [3], Apache jclouds [1] and Docker [5] respectively.

4. Related work

Libraries based approaches like Apache Libcloud, Apache jclouds, δ-cloud, Daseincloud, fog, and pkgcloud remain on the code-level, which makes redesign difficult and error-prone. Authors in [4] propose a Multi-Cloud management platform that locates between Cloud users and Cloud sites and provides unified Cloud services from the SOA perspective. However, they achieve the unified management of multiple Clouds at code-level which is not flexible to meet personalized requirements. The authors in [2] propose a model-based framework called CLOUDMF to manage multiple Clouds. Their work are on a higher level, but both lack the support of interCloud network connection.

5. Conclusion

We propose a novel model-driven approach for managing resources of multiple clouds. Through model construction, and model transformation, multiple Clouds resources can be managed in a unified and personalized manner. Our approach encompasses a precise metamodel of OCCI and models for managing hypervisors, cloud platforms/stacks, and containers. In our approach, we also guarantee the synchronization between the customized model, and the target Multi-Cloud environments. Moreover, the models@runtime environment facilitates reasoning about dynamic adaptation of running systems [7] by providing an abstract representation of the system causally connected to the running system.

As for the future work, we plan to leverage for supporting autonomic Multi-Cloud resource management through reuse of model at runtime.

[8] https://github.com/occiware/ecore

[9] http://www.obeo.fr/download/occiware/

Acknowledgments

This work is supported by OCCIware (www.occiware.org) research and development project funded by French Programme d'Investissements d'Avenir (PAI).

References

[1] Apache jclouds. The Java Multi-Cloud Toolkit. Website `https://jclouds.apache.org/`, February 2016.

[2] N. Ferry, F. Chauvel, A. Rossini, B. Morin, and A. Solberg. Managing multi-cloud systems with cloudmf. In *Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies*, NordiCloud '13, pages 38–45, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2307-9. . URL `http://doi.acm.org/10.1145/2513534.2513542`.

[3] R. Hat. libvirt: The virtualization api. _____, *http://libvirt.org*, 2012.

[4] T. Liu, Y. Katsuno, K. Sun, Y. Li, T. Kushida, Y. Chen, and M. Itakura. Multi cloud management for unified cloud services across cloud sites. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 164–169, Sept 2011. .

[5] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014. ISSN 1075-3583. URL `http://dl.acm.org/citation.cfm?id=2600239.2600241`.

[6] P. Merle, O. Barais, J. Parpaillon, N. Plouzeau, and S. Tata. A Precise Metamodel for Open Cloud Computing Interface. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 852–859. IEEE, 2015.

[7] B. Morin, O. Barais, J.-M. Jezequel, F. Fleurey, and A. Solberg. Models@ run.time to support dynamic adaptation. *Computer*, 42(10):44–51, Oct. 2009. ISSN 0018-9162. . URL `http://dx.doi.org/10.1109/MC.2009.327`.

[8] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier. A Federated Multi-Cloud PaaS Infrastructure. In *5th IEEE International Conference on Cloud Computing*, pages 392 – 399, hawaii, United States, June 2012. . URL `https://hal.inria.fr/hal-00694700`.

[9] J. Parpaillon, P. Merle, O. Barais, M. Dutoo, and F. Paraiso. OCCIware-A Formal and Tooled Framework for Managing Everything as a Service. In *Projects Showcase@ STAF'15*, volume 1400, pages 18–25, 2015.

[10] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.