Building an Open-Source Cross-Cloud DevOps Stack for a CRM Enterprise Application: A Case Study

Sebastian Schork, Feroz Zahid, Dipesh Pradhan, Sébastien Kicin, Antonia Schwichtenberg

Building an Open-Source Cross-Cloud DevOps stack for a CRM Enterprise Application: A Case Study*

Sebastian Schork[1], Feroz Zahid[2], Dipesh Pradhan[2],
Sébastien Kicin[1], and Antonia Schwichtenberg[1]

[1] CAS Software AG, Karlsruhe, Germany
{sebastian.schork,sebastien.kicin,antonia.schwichtenberg}@cas.de
[2] Simula Research Laboratory, Fornebu, Norway
{feroz,dipesh}@simula.no

Abstract. Open Source software solutions play a critical role for the SMEs by enabling easy access to reusable software. Also, with the rapid growth in the popularity of the cloud technologies, computational demands of SMEs are cost-efficiently met by the public clouds as users can dynamically acquire resources on demand according to their needs. However, non-standardized cloud interfaces, lack of inter-cloud transparency, and complex cost models, often result in *vendor lock-in*. Once in vendor lock-in, cloud users have to live with a single cloud provider and accept whatever pricing schemes and SLAs are imposed. Moreover, new regulations covered by the General Data Protection Regulation (GDPR) in Europe require companies to enforce policies regarding secure storage of data in the cloud, as well as restrict moving confidential datasets outside Europe. This situation requires a more transparent use of cloud resources from multiple cloud providers, that conform with user's data privacy needs, service requirements, and budget.
In this paper, we discuss challenges and pitfalls of designing a Cross-Cloud DevOps stack for an *app*-based extension platform of a Customer Relationship Management (CRM) system. The fully-automated DevOps stack, based on open source software tools and technologies, has been developed in close coordination with an open source integration project, *Melodic*. With the help of our DevOps stack, third-party apps in our CRM software are now Multi-Cloud ready, and the data storage in the cloud by the users conforms to potential GDPR requirements. In addition, the deployment time of apps has been reduced to minutes, while the platform is able to scale up and scale down apps efficiently based on the current workload requirements, saving substantial cloud costs.

Keywords: Open Source · Cross-Cloud · DevOps

1 Introduction

The major growth of the ICT industry over the last decade can be attributed towards enabling technologies based on open standards and protocols, making it possible to develop software solutions that can be delivered and integrated on any infrastructure, independent of the vendor. However, with the emergence of the cloud computing paradigm, a step is taken backward [1]. Cloud users are often forced into *vendor lock-in* due to the use of incompatible protocols and standards by the cloud service providers (CSPs). Vendor lock-in or propriety lock-in is an economic condition in which a customer is made dependent on the vendor-specific technology, products, or services by making it fairly difficult and costly to migrate to a competition [2].
The cloud-based customer relationship (CRM) software, *SmartWe*, is designed by CAS Software[1] to support users in an optimal way depending on their role, the business sector they are working in and the respective workflows. With the help of SmartWe's software development kit (SDK) and UI tools, users and CAS development partners can adapt existing apps or develop new apps for the SmartWe platform. The apps can be offered and installed via an App Store. Initially, the SmartWe supports apps to be deployed on a single cloud platform. As we moved on with the development, we found that it was not easy to migrate third-party apps to another cloud platform because SmartWe and its deployment scripts

* This work has received funding from the European Union's H2020 research and innovation programme under grant agreement no. 731664 (MELODIC).
[1] CAS Software AG - https://www.cas.de/en/homepage.html

became heavily dependent on the specific CSP's APIs and management tools. As the new General Data Protection Regulation (GDPR) started being enforced in Europe, the user apps were obliged to conform to the data storage regulations laid down. This was not an issue in the first place for the product SmartWe due to its deployment in secure partner data centers but needs to be considered for the third-party apps that are developed by partners and users. For instance, some datasets used by extension apps, which are confidential to the users according to their data privacy needs, may not be allowed to be migrated to locations outside Europe. Furthermore, the requirements of the apps were dynamic and often updated, requiring manual updates to several scripts and resulting in slower cloud deployments for the third-party apps.

In general, no single CSP is able to provide all the features a user may need in a cost-efficient way, while satisfying the user's security and performance requirements. As mentioned above, even the most dominant CSPs have limited geographical presence. If local legislation requires confidential data to be stored within a country's geographical boundaries, the cloud user will need to rely on cloud providers owning infrastructure locally – or maybe even use a private cloud. Still, the same cloud user would ideally want to take advantage of cheap global cloud offerings for computation and storage when the strict data security requirements do not apply. With these challenges in place, we joined forces together with several other academic and industrial partners to tackle the need of an automated Cross-Cloud DevOps solution under the umbrella of an open source research and innovation project, Melodic[2]. In this paper, we discuss challenges and pitfalls of designing such a Cross-Cloud DevOps stack for SmartWe based on open source tools and technologies. With the help of our DevOps stack, third-party apps management in SmartWe is now fully Multi-Cloud ready bringing our users out of the potential vendor lock-in. In addition, the data storage related to the third-party apps in the cloud now conforms to the GDPR requirements as users can specify their requirements and constraints through an innovate modelling interface. Furthermore, the deployment time for SmartWe and apps has been reduced to minutes, while the platform is able to scale up and down efficiently based on the current workload requirements.

The rest of this paper is structured as follows. In Section 2, we provide details about our SmartWe CRM software. In Section 3, we define the requirements for our automated Cross-Cloud DevOps system, as well as present the process of selecting and integrating the available Open Source Software (OSS), together with outlining the new development under Melodic. The resultant DevOps stack is presented and evaluated in, Section 4 and Section 5, respectively. We conclude in Section 6.

2 The SmartWe CRM

The cloud-based CRM software, SmartWe, is designed to support customers in their daily work by providing a tailored software tool with respect to the particular role of each user. Tailored business solutions support work flows according to the needs of the users and are much more efficient and usable than generic software systems [3]. SmartWe supports different ways to tailor the basic *anything* relationship management (xRM) solution to user role specific tasks. For instance, the user interface can be personalized by adding only a subset of the available apps. Moreover, using the provided SDK, an existing app can be tailored and extended, or a new apps can be developed for fulfilling user-specific requirements. The idea of an app-based xRM cloud software that can be adapted and extended to diverse use-cases and customer needs is highly promising from a marketing and business perspective. However, it is also challenging from the conceptual point of view, and even more so from the developer perspective.

The DevOps stack we present in this paper comes into play both for the transparent deployments of the new or customized apps (such as compute- and data-intensive extensions), as well as to make sure that the third-party apps do not affect the SmartWe system's performance and availability. For example, a CAS partner may develop a new app for either integrating and analyzing existing data from a third-party system or for analytics based on existing data from the primary CRM system, requiring secure data management and on-demand resource availability in the cloud. In these cases, there is a need for a dynamic, customized, and scalable deployment solution for the apps. The aforementioned two cases directly refer to SmartWe's two main pillars:

[2] The Melodic Cloud Project - https://melodic.cloud/

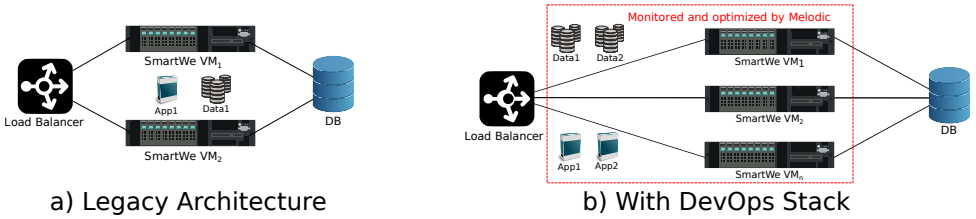

Fig. 1: SmartWe deployment without and with the DevOps stack based on Melodic

– **Third-party apps that extend the SmartWe solution:** Both the data storage and the apps deployments in the cloud need to conform to the specified user-requirements related to the privacy and confidentiality.
– **Scalable SmartWe base deployment:** The dynamic increase in load to the SmartWe base deployment, because of the data-intensive or compute-intensive third-party apps, needs to be tackled automatically.

Both cases and their evaluation with the proposed DevOps stack is described in detail in Section 5. The deployment of SmartWe with and without our DevOps stack, based on an example two application instances with a load balancer and an external third-party app, is shown in Figure 1.

3 OSS Selection and Integration

In order to design a Cross-Cloud DevOps stack for the third-party apps in SmartWe, we first lay down a set of requirements. Next, in the context of our requirements, we surveyed the available OSS, and investigated related integration, compatibility, and licensing issues. Following requirements are specified. **Transparent deployment and execution** is necessary to enable automated app deployment across multiple CSPs. In addition, as both cloud by definition are unpredictive [4], as well as load on SmartWe platform from third-party apps is dynamic, we need a mechanism for **runtime adaptation** for SmartWe and deployed apps. Furthermore, as discussed in Section 1, with the GDPR requirements in place, our DevOps stacks needs to support defining data and component placement requirements and restrictions to cater for the user-specific need of **data privacy and confidentiality**.

3.1 Related Work

The challenges we face for our DevOps stack fall broadly into three areas: Cross-Cloud application deployments, resource management, and modelling/optimization of data-aware applications on heterogeneous infrastructures. In general, cloud federation [5] enables end users to integrate segregated resources from different cloud systems. Popular open-source cloud orchestration solutions, like OpenStack [6], provide mechanisms to complement private cloud infrastructure with dynamically acquired resources from public clouds. Nevertheless, resource management is not well integrated with state-of-the-art federated cloud solutions. Further, none of the current cloud orchestration platforms supports context-awareness needed to optimize application deployments in Cross-Cloud environments, as needed by the SmartWe platform. Furthermore, Cross-Cloud application deployments are subjected to various resource abstraction models offered by different CSPs and a unified approach needed for interoperability is lacking [7, 8].
Recent efforts, such as those in the PaaSage project [9], have targeted model-based approaches for the design, development, deployment, and self-adaptation of Cross-Cloud applications. In particular, cloud modelling frameworks, such as CloudMF [10], are in active development, to equip application developers with capabilities to define a rich set of design-time and runtime attributes like application requirements, Quality-of-Service (QoS) constraints, and security considerations for Cross-Cloud deployments. However, a large number of challenges still remain unaddressed. In particular, support of data-aware deployments in Cross-Cloud environments is still very restricted. Recently, various cluster management solutions have also gain popularity. But most of these solutions work only on statically available cluster and cloud resources, such as Mesos [11], Kubernetes [12] and Docker/Swarm [13]. Some Multi-Cloud deployment solutions, such as CYCLONE [14], are available but lack sufficient advanced reasoning support, as needed for the SmartWe app platform. DC/OS [15]

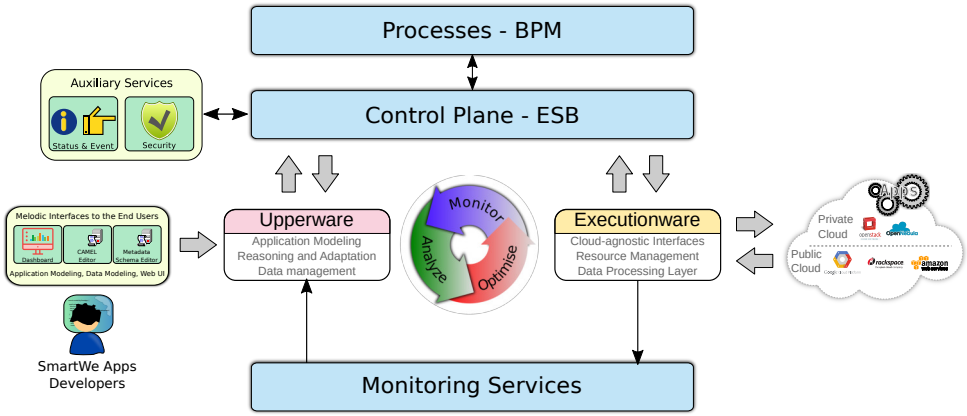

Fig. 2: Overview of the DevOps stack based on Melodic

integrates a range of software, like Mesos and Merathon, to provide an integrated platform for running applications and data services on heterogeneous platforms. However, DC/OS lacks native support for Cross-Cloud deployments and adaptation of applications. Moreover, advanced capabilities for reasoning for efficient resource management, according to the user-defined requirements and constraints, are also missing. Furthermore, many advanced DC/OS features are only available in their closed-source enterprise version.

3.2 Available OSS and Integration

With the related work survey described in Section 3.1, it is quite evident that we need to both integrate the available OSS as well as develop additional capabilities to fulfil the needs of our DevOps stacks for the SmartWe app platform. In the Melodic project, we selected PaaSage OSS as the base Multi-Cloud platform for our stack development. Advanced capabilities related to the data-awareness are implemented as part of the new development in the Melodic OSS on top of the PaaSage code-base.

3.3 Licensing Compatibility

License compatibility is a crucial issue for an OSS integration project built upon existing software. The new components created in the scope of the Melodic itself are released under Mozilla Public License (MPL) v2. The choice of this particular license has resulted from deliberation over its compatibility with the GNU GPL and the licenses used by the Apache Software Foundation. MPL is a *weak copyleft* license designed to address the needs of both proprietary and open source developers [16, 17].

4 A Cross-Coud DevOps Stack

An overview of the our DevOps stack architecture is given in Figure 2. As shown in the figure, the DevOps stack is conceptually divided into three main component groups, the Melodic interfaces to the end users, the *Upperware*, and the *Executionware*. The Melodic interfaces to the end users include tools and interfaces used by the Melodic users to model their applications and datasets and interact with the Melodic platform. These interfaces are exposed to the SmartWe app developers using a modelling language called CAMEL [18]. Applications and data models created in CAMEL are given as input to the Melodic Upperware. The job of the Upperware is to calculate the optimal data placements and application deployments on dynamically acquired Cross-Cloud resources in accordance with the specified application and data models in CAMEL as well as in consideration of the current cloud performance, workload situation, and costs. The actual cloud deployments for the SmartWe apps are carried out through the Executionware. The Executionware is capable of managing and orchestrating diverse cloud resources, and it also enables support of Cross-Cloud monitoring of both deployed apps and the SmartWe base platform.

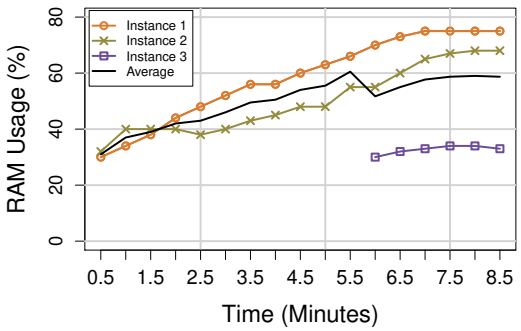

Fig. 3: Autonomic optimization of RAM by Melodic-enabled SmartWe platform

5 Evaluation

We evaluate the DevOps stack designed for apps in SmartWe platform with two different perspectives. First, we assess the usefulness, quality, and maintainability of the open source software developed, in relation to Melodic, to gauge its suitability for the long-term use by the SmartWe partners. Second, we evaluate the SmartWe platform and app scalability features offered through autonomic adaptation and optimization capabilities of the Melodic middleware platform.

CAS found that the initiative of an open source Cross-Cloud DevOps stack is very useful for many SMEs. Besides addressing vendor lock-in, the DevOps stack we developed also enables SMEs to adhere to the security and privacy requirements related to the data storage and processing in the cloud. The platform quickly became a catalyst to federate other companies dealing with similar business scenarios and thus building a developer community around it, bound by a common drive to support and improve open solutions for the cloud development and deployments. The first stages of the Melodic development quickly demonstrated that integrating existing software, with the help of an open source community, quickly realizes the software which requires a large amount of development time otherwise. Moreover, the quality of the software produced is satisfactory promising long-term maintainability. Finally, although Melodic should not be thought just as a *free software*, the fact that the providers require no licensing fees remains a decisive advantage when looking at the potential total cost of deploying the solution across the IT infrastructures.

From the technical perspective, SmartWe platform benefits from various features Melodic offers. Our DevOps stack enables dynamic adaptations to the deployed SmartWe platforms and apps. In the case of SmartWe platform, RAM usage is a critical metric for our live deployments. Increasing numbers of user sessions as well as app's computational complexity leads to higher RAM usage and therefore represent a bottleneck. Based on sensor values and scalability rules, the mechanism autonomously decides on how to best optimize the current deployment. Figure 3 depicts an initial deployment of two instances of the SmartWe system being handled by a central load balancer. A scalability rule was written requiring the average RAM load to be lower than 60% of the total available RAM. The moment when this limit is succeeded, a third application instance is added automatically by the Melodic, and the average RAM load stabilizes, as shown in the figure. A point to note here, however, is that the optimization offered by Melodic uses the concept of *utility function*. Each potential deployment solution determined by Melodic is evaluated regarding its utility before a final selection is made. Utility functions are application and deployment specific and were carefully designed for the SmartWe platform to meet our requirements.

6 Conclusion

In this paper, we discussed the need of an open source Cross-Cloud DevOps stack for our SmartWe CRM solution. With the help of an open source integration project, Melodic, our automated DevOps stack has enabled third-part apps, installable in SmartWe, to counter vendor lock-in. In addition, the user apps in SmartWe are now able to transparently take advantage of distinct characteristics of available private and public clouds, dynamically optimize resource utilization, and conform to the user's privacy needs and service requirements.

References

1. McKendrick, J.: Cloud computing's vendor lock-in problem: Why the industry is taking a step backward. Forbes, November (2011)
2. Opara-Martins, J., Sahandi, R., Tian, F.: Critical review of vendor lock-in and its impact on adoption of cloud computing (2014)
3. Weinhardt, C., Anandasivam, A., Blau, B., Stößer, J.: Business models in the service world. IT professional (2), 28–33 (2009)
4. Schad, J., Dittrich, J., Quiané-Ruiz, J.A.: Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proceedings of the VLDB Endowment **3**(1-2), 460–471 (2010)
5. Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., Kunze, M.: Cloud federation. Cloud Computing **2011**, 32–38 (2011)
6. Sefraoui, O., Aissaoui, M., Eleuldj, M.: OpenStack: toward an open-source solution for cloud computing. International Journal of Computer Applications **55**(3), 38–42 (2012)
7. Petcu, D.: Portability and interoperability between clouds: challenges and case study. In: European Conference on a Service-Based Internet. pp. 62–74. Springer (2011)
8. Taherkordi, A., Zahid, F., Verginadis, Y., Horn, G.: Future Cloud Systems Design: Challenges and Research Directions. IEEE Access **6**, 74120–74150 (2018)
9. Bubak, M., Baliś, B., Kitowski, J., Król, D., Kryza, B., Malawski, M.: PaaSage: Model-Based Cloud Platform Upperware (2011)
10. Ferry, N., Song, H., Rossini, A., Chauvel, F., Solberg, A.: CloudMF: applying MDE to tame the complexity of managing multi-cloud applications. In: IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC). pp. 269–277. IEEE (2014)
11. Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R.H., Shenker, S., Stoica, I.: Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In: NSDI. vol. 11, pp. 22–22 (2011)
12. Mcluckie, C.: Containers, VMs, Kubernetes and VMware, https://cloudplatform.googleblog.com/2014/08/containers-vms-kubernetes-and-vmware.html, accessed January 13, 2019
13. Swarm: a Docker-native clustering system, https://github.com/docker/swarm/, accessed January 13, 2019
14. Slawik, M., Zilci, B.I., Demchenko, Y., Baranda, J.I.A., Branchat, R., Loomis, C., Lodygensky, O., Blanchet, C.: CYCLONE unified deployment and management of federated, multi-cloud applications. In: Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on. pp. 453–457. IEEE (2015)
15. What is DC/OS?, https://docs.mesosphere.com/1.7/overview/what-is-dcos/, accessed January 13, 2019
16. Rosen, L.: Which Open Source license should I use for my software. Open Source Initiative (2001)
17. Rosen, L.: Open source licensing: Software freedom and intellectual property law. Prentice Hall PTR (2004)
18. Rossini, A.: Cloud application modelling and execution language (CAMEL) and the PaaSage workflow. In: Advances in Service-Oriented and Cloud Computing—Workshops of ESOCC. vol. 567, pp. 437–439 (2015)