Choreographies are Key for Distributed Cloud Application Provisioning

Oliver Kopp[1] and Uwe Breitenbücher[2]

[1] IPVS, Universität Stuttgart, Germany, kopp@ipvs.uni-stuttgart.de
[2] IAAS, Universität Stuttgart, Germany, breitenbuecher@iaas.uni-stuttgart.de

Abstract. The automation of Cloud application provisioning is one of the most important key success factors for Cloud Computing. In case a complex composite Cloud application has to be provisioned across multiple different private Clouds, a single centralized provisioning engine or workflow is not possible: For security reasons, private Clouds typically do not expose their internal provisioning APIs to the outside. Thus, it is impossible to make them callable by an external service. Consequently, distribution of provisioning logic across multiple workflows is required. This paper envisions that choreographies can be used to distribute the provisioning logic.

1 Overview

Cloud computing gains more and more attention due to its economical, organizational, and technical benefits. The reason for this evolution lies in the essence of Cloud computing—the industrialization of IT [1]: application provisioning and management are automated as much as possible and follow well-defined processes to provide industrial properties for virtualized infrastructure, middleware, and software as a service. To enable this, Cloud providers employ Cloud management software to virtualize, manage, and operate their physical infrastructure. One part of this management is responsible for rapid on-demand provisioning of parts of a multi-Cloud application for customers. It has been shown that not all management tasks can be done in a declarative way [3]. Moreover, many applications need to consume different services for their particularities that are not completely provided by one single Cloud provider [10]. Thus, these applications need to be distributed across several different Clouds to gain the needed functionality [11]. Distributing components of one application among multiple Clouds, however, increases the complexity of provisioning. Provisioning applications on a single Cloud (based on internal provisioning engines) helps the Cloud provider to secure the system: most of the required management operations can be executed locally by provisioning engines behind strict firewalls and only coarse-grained facade APIs need to be exposed to the outside. This changes if the application is distributed across multiple Clouds: the provisioning on different Clouds needs to be orchestrated by an external provisioning engine. Thus, Cloud providers have to expose also the low-level management operations, interfaces, and APIs to make

O. Kopp, J. Lenhard, C. Pautasso (Eds.): 9th ZEUS Workshop, ZEUS 2017, Lugano, Switzerland, 13-14 February 2017, published at http://ceur-ws.org/

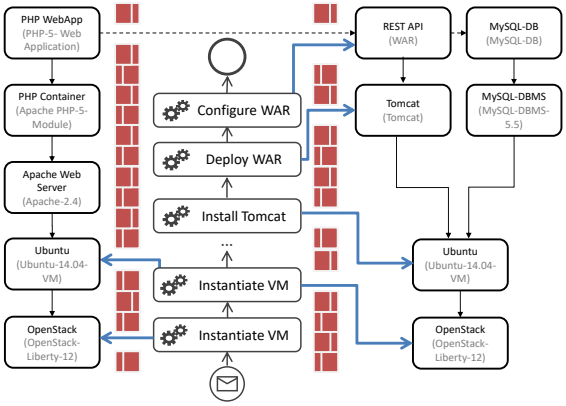

Fig. 1: Orchestration-based Provisioning

them accessible from the outisde. Figure 1 shows this kind of provisioning for an application hosted on a Tomcat as WAR file. First, a virtual machine has to be instantiated, then a Tomcat has to be installed. Afterwards, the WAR file has to be deployed and configured. The calls to the respective management operations have to pass the firewall of the local Cloud provider, which might not be allowed.

2 The Vision

We have the vision of modelling the provisioning of distributed multi-Cloud applications as a choreography model, wherein each participant flow executes the provisioning of one part of the application at one provider.

As a consequence, we need a means to coordinate the provisioning of multi-Cloud applications in a way that the execution of provisioning logic remains on the side of Cloud providers. This avoids that the low-level management functionalities must be made accessible from the outside. Our proposed solution is to use choreographies [9] for application provisioning: the provisioning logic has to be split among the participants, each representing a local Cloud provider. Figure 2 presents such a model for the described example. Now, the remote workflow sends a single message triggering the provisioning within the local Cloud provider. The provisioning is implemented by a workflow running within that Cloud provider.

3 Discussion

The APIs for each component still need to be accessible. With the current approach, they only need to be accessible from local environment and not from the outside. We claim that this shift makes it easier to make the APIs accessible for provisioning workflows. Regarding the firewall setup, one access to the local

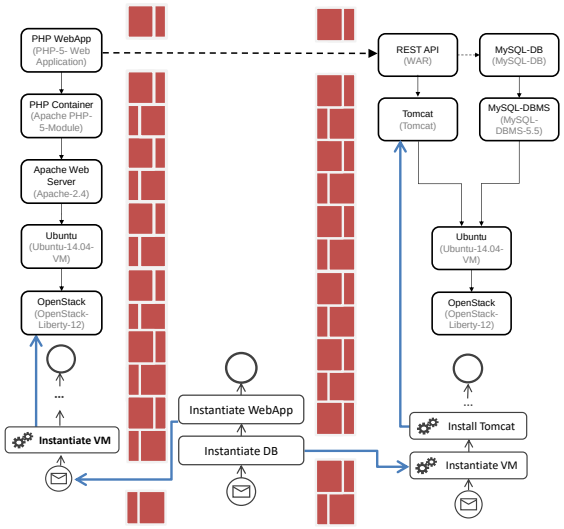

Fig. 2: Choreography-based Provisioning

management infrastructure has to be opened. This endpoint is known in advance. In the orchestration-based provisioning, the concrete endpoints are not known in advance as the reachable components are provisioned and thus not available before. For instance, Tomcat is installed during the provisioning and its IP address is not known in advance.

Herry et al. [4] generate provisioning workflows out of given constrains. Our approach relies on human-generated choreography models. The infrastructure for executing the provisioning workflows does not need to be available all the time: Vukojevic-Haupt et al. [13] showed that it is also possible to provision that middleware on demand.

Schulte et al. [12] identify the state of the art of elastic Business Process Management with a focus on infrastructural challenges. They focus on the runtime of business processes. Their results can also be applied to provisioning.

Another approach is to model a single provisioning workflow and to split this workflow [7] according to the distribution of the topology. For the split, new coordinators might be required [5]. When using a choreography model, the provisioning workflows and their coordination can be generated easily as the coordination of participants is inherently specified by the choreography. We aim to use this model-driven approach to generate the provisioning workflows and their coordination out of the choreography model. Thus, the provisioning choreography is not seen as purely descriptive means, but as a starting point for an automatic processing.

Weber et al. [14] propose to use blockchains in order to enact choreographies in a decentralized fashion. They use the Blockchain concept of “Triggers” to facilitate communication between internal and external APIs.

The idea presented in the paper is not tied to concrete languages or tools. It is possible to model it in an arbitrary language. This includes BPMN4TOSCA [8], which is a language tailored to provisioning with TOSCA [2]. In our setting, we use the domain-specific language BPMN4TOSCA as it allows focusing on the provisioning logic without having to abstract from the nodes in the application topology when modeling a provisioning choreography. To validate the approach, we plan to (i) enable swimlanes in our current modeling tool and (ii) to show that the approach is feasible in our OpenTOSCA eco system.

The provisioning workflows can be seen as subprocesses [6] of the central coordinator as they receive one instantiation message and then no communication is taking place any more. In more complex settings, it might be necessary that the provisioning workflows communicate with the coordinator again and thus forming a choreography with arbitrary interactions between caller and callee [6]. The discussion of this aspect together with possible autonomy grades [6] is part of our future work.

Acknowledgments This work is funded by the BMWi project SmartOrchestra (01MD16001F).

References

1. Binz, T., et al.: Portable Cloud Services Using TOSCA. IEEE Internet Computing 16(03), 80–85 (May 2012)
2. Binz, T., et al.: TOSCA: Portable Automated Deployment and Management of Cloud Applications, pp. 527–549. Advanced Web Services, Springer (Jan 2014)
3. Breitenbücher, U., et al.: Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA. In: IC2E. IEEE (2014)
4. Herry, H., Anderson, P., Rovatsos, M.: Choreographing Configuration Changes. In: CNSM. IEEE (2013)
5. Khalaf, R., Leymann, F.: Coordination for fragmented loops and scopes in a distributed business process. IS 37(6), 593–610 (2012)
6. Kopp, O., Eberle, H., Leymann, F., Unger, T.: The Subprocess Spectrum. In: BPSC. GI e.V. (2010)
7. Kopp, O., Leymann, F., Unger, T., Wagner, S.: Towards the essential flow model. In: ZEUS. CEUR-WS.org (2011)
8. Kopp, O., et al.: BPMN4TOSCA: A Domain-Specific Language to Model Management Plans for Composite Applications. In: BPMN. Springer (2012)
9. Peltz, C.: Web Services Orchestration and Choreography. IEEE Computer 36(10), 46–52 (2003)
10. Petcu, D.: Multi-cloud: expectations and current approaches. In: MICAS Workshop. MultiCloud '13, ACM (2013)
11. Saatkamp, K., et al.: Topology splitting and matching for multi-cloud deployments. In: CLOSER (2017)
12. Schulte, S., et al.: Elastic Business Process Management: State of the art and open challenges for BPM in the cloud. FGCS 46, 36–50 (may 2015)
13. Vukojevic-Haupt, K., et al.: Bootstrapping complex workflow middleware systems into the cloud. In: International Conference on e-Science. IEEE (2015)
14. Weber, I., et al.: Untrusted business process monitoring and execution using blockchain. In: BPM. Springer (2016)