OpenTOSCA Injector: Vertical and Horizontal Topology Model Injection

Karoline Saatkamp, Uwe Breitenbücher, Kálmán Képes, Frank Leymann, and Michael Zimmermann

Institute of Architecture of Application Systems, University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de

Abstract. The automation of application deployments is supported by various technologies. The TOSCA standard facilitates to describe application deployments in a portable manner by modeling application structures as topology models. The final structure often depends on the target environment and is, therefore, not always known at modeling time. However, a manual adaptation is error-prone and time-consuming. In this paper, we demonstrate the OpenTOSCA Injector for an automated completion of topology models: the extended TOSCA runtime OpenTOSCA for an automated injection and deployment is presented.

Keywords: TOSCA, Deployment Model, Completion Automation

1 Introduction and Motivation

In recent years, several technologies and standards were developed to automate the deployment of cloud applications. This includes configuration management technologies such as Chef, container technologies such as Docker, and standards such as the Topology and Orchestration Specification for Cloud Applications (TOSCA) [5]. TOSCA is an OASIS standard that enables to define application deployments by topology models and management plans, which can be executed automatically by a TOSCA runtime, e.g., the OpenTOSCA runtime [1].

A TOSCA topology model describes the application components and their relations. This includes application-specific components, e.g., PHP applications as well as databases, middleware, and infrastructure components such as web servers or virtual machines. Thereby, application deployments can be described in a vendor-independent and portable manner. However, the available middleware, infrastructure, as well as application-specific components can differ between different environments. When, for example, application deployments are provided for third parties or parts of the IT infrastructure are outsourced, the target environment is not known in advance. Thus, the final topology model is not known at modeling time. However, the manual adaptation for each target environment is time-consuming and error-prone [3]. To enable an environment-independent modeling and an automated environment-specific injection of components during deployment time, the *OpenTOSCA Injector* is developed.

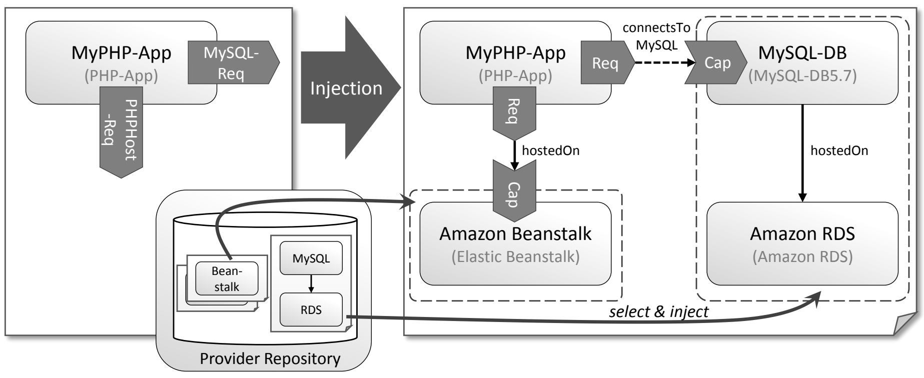

Fig. 1. Topology model with open requirements (left) and injected components (right)

2 TOSCA Fundamentals and Injection Concept

As already mentioned TOSCA enables to describe the automated deployment of applications in a vendor-independent and portable manner. In the following all TOSCA concepts relevant for the OpenTOSCA Injector are introduced. More details about TOSCA can be found in the specification [5].

The structure of an application can be described as *Topology Template*, which is a directed and weighted multigraph as depicted in Fig. 1 on the right. The components are modeled as *Node Templates*, e.g., MyPHP-App, and the relations between them as *Relationship Templates* such as hostedOn. Their semantic is defined by *Node Types*, e.g., PHP-App, and *Relationship Types*, respectively. Types can be derived from other types, thus, inheritance hierarchies can be defined. For Relationship Types valid target and source elements are specified, which can be Node Types or *Requirement Types* and *Capability Types*. For each Requirement Type exactly one required Capability Type is defined. *Requirements* and *Capabilities* of these types can be attached to Node Templates, whereby Requirements can serve as sources and Capabilities as targets for Relationship Templates. Thus, the matching between Requirements and Capabilities is realized, which is the basis for the OpenTOSCA Injector.

On the left side of Fig. 1 an incomplete topology with two open Requirements is shown. The Requirements *PHPHost-Req* and *MySQL-Req* require Node Templates with matching Capabilities. With the OpenTOSCA Injector not only the injection of single Node Templates but also of topology fragments is possible [3]. For each match a suitable Relationship Type has to be found. As seen in Fig. 1 often different types for each match are needed. The suitable Relationship Type is determined by the assigned Requirement and Capability. However, specific types such as *connectsToMySQL* are not always available in the target environment. For this, TOSCA base types are used [6]. For the following two TOSCA Relationship Types the valid target elements are defined: for the *hostedOn* type the *Container* and for the *connectsTo* type the *Endpoint* Capability Type is the valid target element. In any case one of these basis types is selected, because each Capability Type has to be derived either from the Container or the Endpoint Capability Type. After the topology fragments are selected and injected with suitable Relationship Types, the topology is complete and deployable. We implemented the described injection concept and demonstrate it with the OpenTOSCA Injector, which extends the existing OpenTOSCA runtime.

3 System Architecture and Demonstration

The OpenTOSCA runtime is a TOSCA runtime supporting the imperative and declarative processing of TOSCA models for an automated deployment [2]. For the imperative processing the management plans are explicitly defined while for the declarative processing the deployment logic is inferred from the topology model. Our demonstration bases on a declarative processing.

In Fig. 2 the extended OpenTOSCA system architecture is depicted. On the left the existing OpenTOSCA components required for the deployment, on the right the *Container Repository* extending the existing runtime are shown. The Container API is used to upload topology models and all related artifacts, such as JAR files or scripts, for the deployment. The *Control* component is responsible for interpreting the topology and tracking the process. For a declarative processing the *Plan Builder* generates plans based on the topology model. The operations invoked by the plans need Implementation Artifacts (IA) to install and start the application's components. They are part of the upload and processed by the *IA Engine* while the plans are processed by the *Plan Engine*. With the *Management Bus* plans finally invoke different kinds of management operations for the deployment. All data required during the deployment, e.g., model information, and after the deployment such as the instance data, are stored in databases.

For the demonstration of the injection the *Container Repository* is essential. Its source code bases on *Winery*, a modeling tool for TOSCA [4]. Because the injection affects the topology model, the existing Winery capabilities to deal with

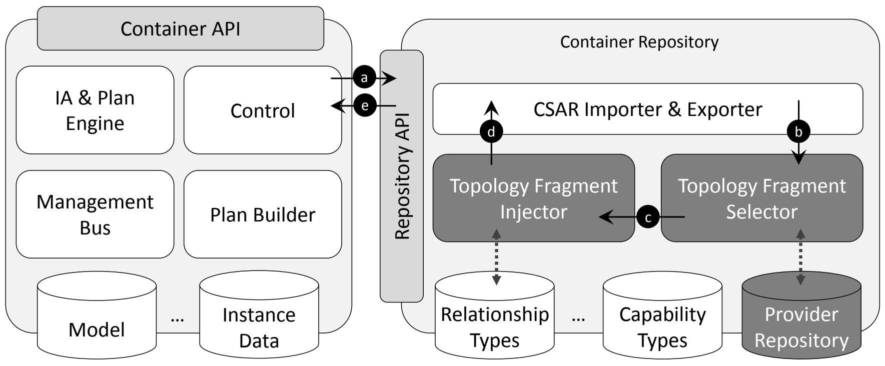

Fig. 2. Extended OpenTOSCA System Architecture and Processing Overview

topology model elements is utilized. The *Topology Fragment Injector*, the *Topology Fragment Selector* component, and the *Provider Repository* extend the existing Winery source code to use it as Container Repository for the injection. For the injection the incomplete topology as depicted in Fig. 1 is uploaded to the Container API and forwarded to the Control component. It checks the topology model for open requirements and in case open requirements are contained, an injection request is sent to the Container Repository (cf. (a) in Fig. 2). The Topology Fragment Selector browses the Provider Repository for topology fragments with matching Capabilities (cf. (b) in Fig. 2). For multiple injection options the user can select the preferred fragment. After the selection suitable Relationship Templates are determined based on the Requirements and Capabilities and used to inject the topology fragments in the model (cf. (c) in Fig. 2). The completed topology model as presented in Fig. 1 on the right is exported and the Control component starts the deployment of the entire application (cf. (d) in Fig. 2).

The demonstrated OpenTOSCA Injector implements the TOSCA concept for Requirement and Capability matching in an automated manner. It facilitates beyond the general matching of Capabilities, the injection of whole topology fragments. Additionally, it supports to restrict the set of considered topology fragments for the injection by target labels attached to Node Templates to express preferences for the matching [7]. With the OpenTOSCA Injector concepts for an environment-dependent and automated application deployment can be realized.

Acknowledgments This work was partially funded by the projects SePiA.Pro (01MD16013F), SmartOrchestra (01MD16001F), and IC4F (01MA17008G).

References

1. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA - A Runtime for TOSCA-based Cloud Applications. In: Proceedings of the 11th International Conference on Service-Oriented Computing. pp. 692–695. Springer (2013)
2. Breitenbücher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., Wettinger, J.: Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA. In: International Conference on Cloud Engineering. pp. 87–96. IEEE (2014)
3. Hirmer, P., Breitenbücher, U., Binz, T., Leymann, F., et al.: Automatic Topology Completion of TOSCA-based Cloud Applications. In: GI-Jahrestagung, GI, vol. P-251, pp. 247–258. GI (2014)
4. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – A Modeling Tool for TOSCA-based Cloud Applications. In: Proceedings of the 11th International Conference on Service-Oriented Computing. pp. 700–704. Springer (2013)
5. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0. OASIS (2013)
6. OASIS: TOSCA Simple Profile in YAML Version 1.0. OASIS (2015)
7. Saatkamp, K., Breitenbücher, U., Kopp, O., Leymann, F.: Topology Splitting and Matching for Multi-Cloud Deployments. In: Proceedings of the 7th International Conference on Cloud Computing and Services Science. pp. 247–258. SciTePress (2017)