Vinothek – A Self-Service Portal for TOSCA

Uwe Breitenbücher[1], Tobias Binz[1], Oliver Kopp[1,2], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Germany
[2] Institute for Parallel and Distributed Systems, University of Stuttgart, Germany
lastname@informatik.uni-stuttgart.de

Abstract The TOSCA standard provides a means to describe Cloud applications and their management in a portable way. TOSCA-based applications can be deployed on various standard-compliant TOSCA Runtimes. Vinothek is a Web-based Self-Service Portal that hides the technical details of TOSCA Runtimes and provides end users a simple graphical interface to provision Cloud applications on demand. This demonstration shows how Vinothek supports automated provisioning of applications and how it facilitates integrating TOSCA Runtimes.

Keywords: Cloud Applications; Self-Service Portal; TOSCA; Portability

1 Introduction

The *Topology and Orchestration Specification for Cloud Applications* (TOSCA [4]) is an OASIS standard for automating provisioning and management of Cloud-based applications in a portable and interoperable way. TOSCA provides a generic specification to model topologies that define the structure of Cloud applications. Topologies and all required software artifacts such as virtual machine images or scripts are stored in a package called *Cloud Service Archive* (CSAR). This archive is also standardized and enables vendors to distribute their applications to multiple Cloud providers based on a uniform and portable format.

To run TOSCA-based applications, a so called *TOSCA Runtime* is employed. TOSCA Runtimes consume CSARs, install them, and provide functionalities to provision and manage instances of the application. However, different TOSCA Runtimes provide different management APIs as these are not standardized by the specification: Some TOSCA Runtimes provide management functionalities out of the box, others require so called *Management Plans* which are contained in CSARs. Despite the goal of portability, TOSCA is currently lacking a common API specification to provision new application instances.

This demonstration tackles this issue by presenting the Self-Service Portal "Vinothek", which allows end users to provision new application instances through a simple, graphical user interface. Vinothek hides the technical details and differences of TOSCA Runtimes and provides a single unified interface to provision applications on various connected TOSCA Runtimes. Vinothek is based on Web technologies such as HTML5 and JavaScript and requires no additional software on client side. More details about TOSCA and TOSCA Runtimes are provided by the TOSCA Specification [4], the TOSCA Primer [5], and Binz et al. [2].

2 User Interaction

Vinothek provides a simple and easy accessible Web-based end user interface to provision new application instances on different TOSCA Runtimes. The portal consists of two main screens: The *Overview* page shown in Fig. 1 lists all applications installed in the TOSCA Runtimes connected to the Vinothek. The shown applications can be provisioned by the user. Therefore, the user selects one of the listed applications which leads to the *Application Details* page shown in Fig. 2. This page presents all details about the selected application and offers different *options* to configure the provisioning (shown on the bottom left in Fig. 2). The user provisions a new application instance by clicking the "Start Instance" button. If the provisioning requires additional user input such as payment information, a popup appears that enables filling in this information. After the provisioning is finished, the new application instance is opened in a new browser window. Depending on the type of the application, the user interface, a status page, or a remote desktop of the application instance is shown.

3 System Overview

The Vinothek is implemented following the Web-based client-server architecture shown in Fig. 3. The *Graphical User Interface (GUI)* is based on Java Server Pages and HTML5. It communicates via a *RESTful API* with the server that delegates calls to the *TOSCA Application Lifecycle Manager*, which is currently dealing with the provisioning of applications only. We plan to extend this component in the future to support management and termination of application

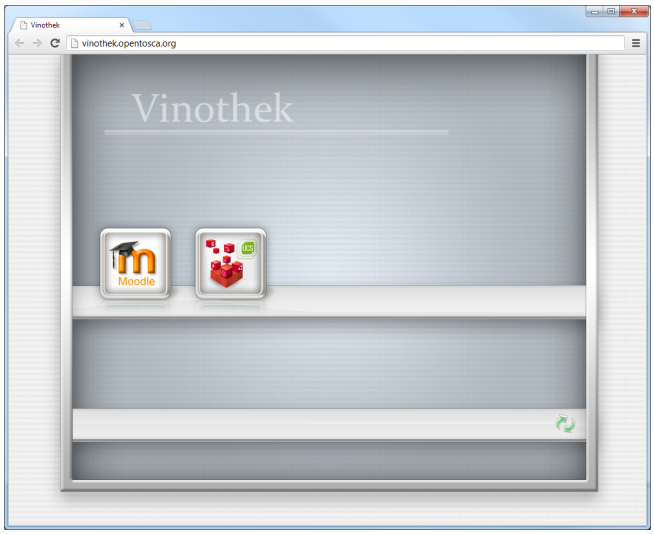

Figure 1. Vinothek Overview page showing all available applications

Figure 2. Vinothek Application Details page showing the selected Moodle application

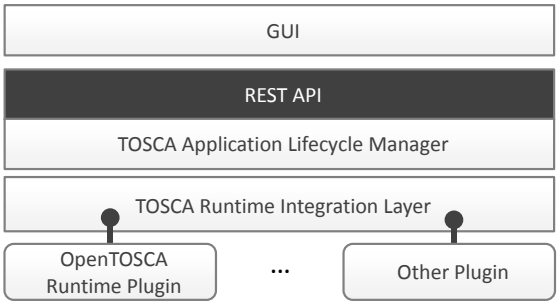

Figure 3. Vinothek System Overview

instances, too. Below this manager, the *TOSCA Runtime Integration Layer* provides mechanisms to plug-in TOSCA Runtimes. Plugins hook into the Vinothek Lifecycle Manager by implementing a certain interface provided by the integration layer and encapsulate all runtime-specific mechanisms to (i) provision a new application instance and (ii) to retrieve available applications that are installed as CSARs in the respective TOSCA Runtime. Thus, if a new application gets installed in a TOSCA Runtime that is connected to the Vinothek by a plugin, the new application is offered automatically by the Vinothek.

Depending on the API provided by the respective TOSCA Runtime, the implementations of plugins differ from each other. We implemented one plugin for the OpenTOSCA Runtime [1], which employs management plans implemented as workflows to provision and manage applications. The OpenTOSCA plugin connects, therefore, to (i) OpenTOSCA's workflow engine to provision new

application instances and to (ii) the RESTful OpenTOSCA Management API for retrieving installed applications.

As the TOSCA Specification does not define how to deal with self-service information, we extended the structure of CSARs by adding a "Meta-SelfService" folder. This folder contains a uniform XML-based application description including marketing information such as text, icons, and screenshots as well as a technical *Deployment Descriptor*. This Deployment Descriptor defines technical information required to provision the application on the respective runtime, i. e., required input parameters and runtime-specific information. When the provisioning of a new application instance is triggered, the Vinothek first requests all specified input parameters from the user via a popup. This information is passed to the plugin that uses these parameters and the technical information contained in the runtime-specific part of the Deployment Descriptor to start the provisioning of the application in the respective TOSCA Runtime. For example, the Deployment Descriptor of the school learning software "Moodle"[3] requires the initial username and password for the admin from the user. In addition, the Deployment Descriptor contains information required by OpenTOSCA to run the plans, e. g., it specifies that both Moodle database and business logic shall run in one single virtual machine. The "Meta-SelfService" folder itself may be created manually or by using the TOSCA modeling tool "Winery" [3].

4 Conclusion and Outlook

We presented the Self-Service Portal "Vinothek", which provides a simple graphical user interface for the provisioning of TOSCA-based applications. The tool also provides a means to integrate different TOSCA Runtimes transparently to end users and hides the technical details. A video of the demonstration is available at `http://demo.opentosca.org`. In the future, we plan to extend the Vinothek to support management functionalities and policies.

Acknowledgements This work was partially funded by the BMWi project CloudCycle (01MD11023). We thank Kálmán Képes for his work on the prototype.

References

1. Binz, T., et al.: OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In: ICSOC. Springer (2013)
2. Binz, T., et al.: TOSCA: Portable Automated Deployment and Management of Cloud Applications, pp. 527–549. Advanced Web Services, Springer (Januar 2014)
3. Kopp, O., et al.: Winery – Modeling Tool for TOSCA-based Cloud Applications. In: ICSOC. Springer (2013)
4. OASIS: OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0 Committee Specification 01 (2013)
5. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0 (January 2013)

[3] `http://www.moodle.org`