TOSCA Lightning: An Integrated Toolchain for Transforming TOSCA Light into Production-Ready Deployment Technologies

Michael Wurster[1], Uwe Breitenbücher[1], Lukas Harzenetter[1], Frank Leymann[1], and Jacopo Soldani[2]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Germany
[lastname]@iaas.uni-stuttgart.de
[2] Department of Computer Science, University of Pisa, Pisa, Italy
[lastname]@di.unipi.it

Abstract. The OASIS standard TOSCA provides a portable means for specifying multi-service applications and automating their deployment. Despite TOSCA is widely used in research, it is currently not supported by the production-ready deployment technologies daily used by practitioners, hence resulting in a gap between the state-of-the-art in research and the state-of-practice in industry. To help bridging this gap, we identified *TOSCA Light*, a subset of TOSCA enabling the transformation of compliant deployment models to the vast majority of deployment technology-specific models used by practitioners nowadays. In this paper, we demonstrate TOSCA Lightning by two contributions. We (i) present an integrated toolchain for specifying multi-service applications with TOSCA Light and transforming them into different production-ready deployment technologies. Additionally, we (ii) demonstrate the toolchain's effectiveness based on a third-party application and Kubernetes.

Keywords: Deployment Automation, Cloud Computing, TOSCA, Kubernetes

1 Introduction

Automating the deployment of multi-service applications is crucial, as manually deploying them is time-consuming and error-prone, and since modern software engineering practices (e. g., continuous development and continuous integration) heavily rely on deployment automation [19]. To accomplish this need, various deployment automation technologies have been proposed. Each technology is however typically equipped with its own language for specifying the target deployment for a multi-service application. Such languages either *declaratively* describe the desired application configuration or *imperatively* list the technical tasks to be executed to deploy and configure the application [10]. Even if declarative languages are by far considered to be the most appropriate in practice [19], they are tightly coupled to the corresponding deployment technology, hence limiting the portability of application deployments from one technology to another.

Advanced Information Systems Engineering, CAiSE Forum 2020, Grenoble, France, June 8–12, 2020, Proceedings, pp. 138–145, 2020. https://doi.org/10.1007/978-3-030-58135-0_12.

In contrast, the *Topology and Orchestration Specification for Cloud Applications* [16] (TOSCA) is a standardized modeling language allowing to declaratively specify portable multi-service application deployments. While it is heavily used in research [1], TOSCA is currently not supported by the production-ready deployment technologies daily used by practitioners. As a result, a gap between the academic state-of-the-art and the industrial state-of-practice is arising.

To help bridging this gap, we identified the *TOSCA Light* [20] subset of TOSCA, which can be automatically transformed to the vast majority of deployment technology-specific languages. TOSCA Light identifies the subset of TOSCA that complies with the *Essential Deployment Metamodel* (EDMM), a set of core deployment modeling entities that the 13 most used deployment technologies understand [19]. Any deployment modeling language relying on the entities of EDMM can hence be converted to multiple heterogeneous technology-specific deployment artifacts [18]. TOSCA Light is exactly that subset of the TOSCA specification that complies with EDMM and, thus, multi-service application deployments written in TOSCA Light can be processed to obtain technology-specific deployment models, including required artifacts and templates [20].

Our objective in this paper is to demonstrate the potentials and practical applicability of this approach by introducing the *TOSCA Lightning* toolchain and to show how TOSCA Light favors the portability of multi-service application deployments. We show that TOSCA Light enhances the portability of deployment models as it can be used for devising technology-agnostic specifications that can be translated to different technology-specific deployment artifacts. With TOSCA Light, application developers can indeed specify their application deployment only once, still being able to actually deploy the specified application on multiple production-ready deployment technologies such as Kubernetes or Terraform.

In this perspective, the contribution of this demonstrator paper is twofold. We first present (i) the open-source TOSCA Lightning integrated toolchain. TOSCA Lightning enables specifying multi-service applications in TOSCA validating their compliance with TOSCA Light and generating the technology-specific artifacts for actually enacting their deployment on production-ready deployment technologies. Moreover, we present (ii) an end-to-end case study based on a third-party application. The case study illustrates how to exploit the TOSCA Lightning toolchain to validly specify the application with TOSCA Light and to automatically generate the artifacts for actually deploying the application with one of the supported production-ready technologies, e. g., Kubernetes.

In the following, Sect. 2 presents the integrated *TOSCA Lightning* toolchain and Sect. 3 shows its application to a concrete case study. Section 4 and Sect. 5 discuss related work and draw some concluding remarks, respectively.

2 The TOSCA Lightning Toolchain

We hereby introduce all components forming the integrated *TOSCA Lightning* toolchain and explain the user's workflow. The toolchain consists of four components in total, as depicted in Fig. 1. Two of them were newly developed within the

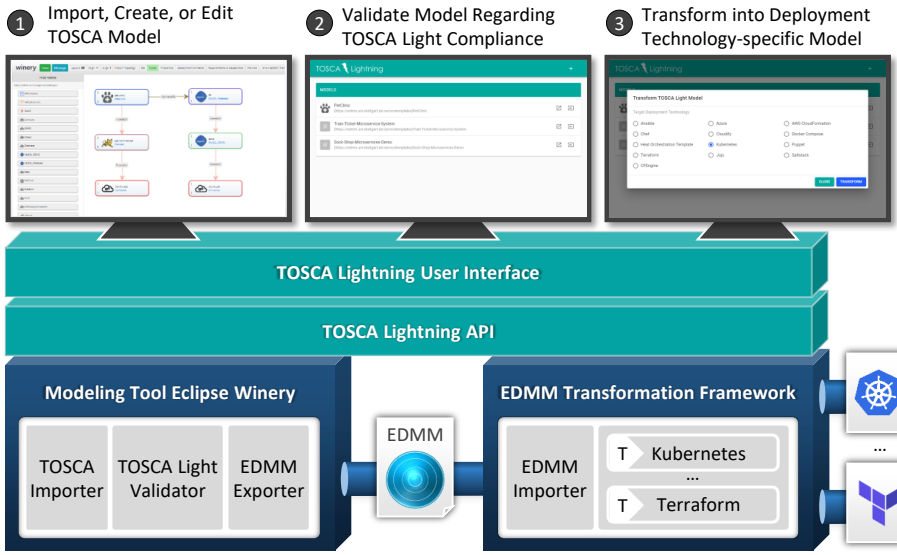

Fig. 1: The TOSCA Lightning Toolchain (new integration components colored green).

scope of this paper, namely the *TOSCA Lightning User Interface* and the *TOSCA Lightning API*. Further, two existing components, namely Eclipse Winery [15] and the EDMM Transformation Framework [18], were integrated into the proposed toolchain by utilizing their corresponding APIs. The TOSCA Lightning toolchain is *open source* and available on GitHub[3], including a demonstration video.

In the scope of this paper, we developed the TOSCA Lightning User Interface and the TOSCA Lightning API to fully integrate the TOSCA modeling tool Eclipse Winery and the EDMM Transformation Framework to enable transformation. The TOSCA Lightning User Interface is the user's main entry point and acts as a dashboard to list, create, change, or transform compliant TOSCA Light models. The TOSCA Lightning API respectively encapsulates the REST APIs over HTTP of Eclipse Winery and the EDMM Transformation Framework to provide a uniform interface for the integrated TOSCA Lightning toolchain.

The TOSCA Lightning User Interface is able to launch the modeling environment Eclipse Winery [15] to import, create, or edit TOSCA Light models. Eclipse Winery is a web-based environment to graphically model TOSCA-based application topologies and provides a *Management Interface* to manage all TOSCA related entities, such as node types, their property definitions, operations, and artifacts. Further, it provides a *Topology Modeler* component which enables the graphical composition of an application and its desired target state. Winery has been extended by various features, which can be enabled by so-called *feature flags*, to provide the necessary TOSCA Light capabilities, e. g., to validate whether imported or created TOSCA models are compliant with TOSCA Light.

[3] `https://github.com/UST-EDMM/tosca-lightning`

The TOSCA Lightning User Interface enables users to execute the transformation of a TOSCA Light model into a *deployment technology-specific model* (DTSM) by selecting one of the supported deployment technologies. For this transformation, the user interface invokes via the TOSCA Lightning API the Eclipse Winery API to export the selected TOSCA Light model as EDMM model, which can be processed afterward by the EDMM Transformation Framework [18]. Exporting TOSCA Light models as EDMM models is possible since TOSCA Light can be directly mapped to EDMM [19, 20]. Thus, the EDMM Transformation Framework enables transforming a given EDMM model into required artifacts, i. e., files and models, to execute the deployment using the selected deployment technology's tooling. The EDMM Transformation Framework is plugin-based and, at the time of writing, supports via the EDMM Transformation Framework the transformation into DTSMs of 13 deployment automation technologies, such as Kubernetes, Terraform, Chef, Puppet, and AWS CloudFormation. We hereafter illustrate how users work with the TOSCA Lightning toolchain:

Import, Create, or Edit TOSCA Model. Eclipse Winery provides the *TOSCA Importer* functionality to easily reuse and adapt existing TOSCA deployment models. Users can create new models or edit existing models by graphically composing the component structure of the desired application. The application's component structure is indeed described declaratively using TOSCA modeling constructs. The resulting model comprises all TOSCA-based definitions and entities describing types, component instances, their properties, operations, and file artifacts required for deploying and operating the application.

Validate Model Regarding TOSCA Light Compliance. The validation of the TOSCA model is performed at *design time* using the *TOSCA Light Validator* component of Eclipse Winery. It checks the TOSCA Light compliance of imported or created TOSCA deployment models. The models are checked against a set of TOSCA Light modeling requirements [20]. In case the model is compliant with TOSCA Light, it is shown in the TOSCA Lightning User Interface. In situations where the model is not compatible with TOSCA Light, the model can be refined according to the provided modification recommendations, presented to the user as a list of violated conditions inside Eclipse Winery.

Transform into DTSM. TOSCA Lightning is able to transform a validated TOSCA Light model into a DTSM by integrating Eclipse Winery and the EDMM Transformation Framework. The model exchange between Eclipse Winery and EDMM Transformation Framework is realized by file transfer, as depicted in Fig. 1. Eclipse Winery's *EDMM Exporter* component produces the output according to the *input file* specifications of the EDMM Transformation Framework. Thus, the output can be directly used for transforming the model into the desired target deployment model format. Notably, the possibility to transform the model is guaranteed by design since the used modeling constructs conform to the essential entities defined by EDMM and, thus, are compliant with TOSCA Light. Users simply select one of the 13 supported deployment automation technologies (e.g., Kubernetes) for the TOSCA Light model that should be transformed. Afterward, users can download the transformation result containing the respective files and

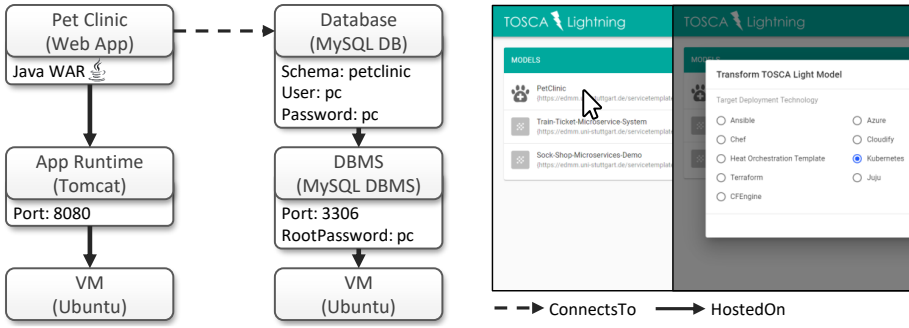

Fig. 2: Case Study: Transforming TOSCA Light to Kubernetes.

templates generated by the EDMM Transformation Framework. At this stage, the generated target deployment model is ready to be deployed using the target technology's tooling. Hence, the actual deployment happens using the actual tools and mechanisms provided by the target deployment automation technology.

3 Case Study: Transforming TOSCA Light to Kubernetes

Today, Kubernetes is one of the fastest-growing open-source projects. Gartner predicts that by 2022, more than 75% of global enterprises will be using containerized applications in production, and Kubernetes will play an important role [8]. Therefore, we want to show how a declaratively modeled TOSCA Light application can be deployed to Kubernetes by generating the required files and template to execute the deployment on a running Kubernetes cluster.

For the sake of demonstration, we prepared a TOSCA deployment model of the Spring PetClinic application that demonstrates the use of the Spring framework (fork of Java's Pet Store application), which represents a simple software for a veterinary clinic. It is a well-known demo web application running on a Tomcat web server while connecting to a MySQL database to store its data. The respective TOSCA model is schematically depicted on the left-hand side in Fig. 2. For reasons of space limitations, we hereby only show the deployment of the depicted PetClinic application to Kubernetes. However, its deployment can be achieved with any of the other 12 deployment automation technologies or any other TOSCA deployment model supported by the TOSCA Lightning toolchain.

The application model is created or imported using Eclipse Winery. The model itself is not specifically composed for Kubernetes as the target runtime environment. Instead, it is modeled in a generic, component-based manner that will be later translated into the respective files and templates required by Kubernetes, e. g., Dockerfiles, Kubernets Deployment, and Kubernetes Service descriptors. The TOSCA Lightning User Interface lists all available TOSCA Light compliant service templates after it has been started with Docker[4].

[4] A Docker Compose file to start the TOSCA Lightning toolchain, a demonstration video, an in-depth quickstart guide, and ready-to-use TOSCA models are available in our GitHub repository: `https://ust-edmm.github.io/tosca-lightning`

Users can open the Topology Editor to display the application's component structure. Further, the Topology Editor is used to set property values which will be used as configuration for instantiating the components at runtime. Eclipse Winery in our demonstration scenario already comes with a set of built-in modeling types, which can be used to model new applications. However, these types follow the proposed normative types by the TOSCA Simple Profile standard [16], while new types can be imported or added manually using Eclipse Winery.

The TOSCA Lightning User Interface and its way to select a transformation target for a certain model is depicted on the right-hand side in Fig. 2. Users can transform the PetClinic application to Kubernetes by selecting the respective entry in the pop-up dialog. The Kubernetes plugin of the EDMM Transformation Framework tries to identify component stacks, a set of tightly coupled components, i. e., components related with "HostedOn" relations. Further, the plugin produces a Dockerfile as well as a Kubernetes Deployment and Kubernetes Service for each component stack. Once the transformation has been performed, users can download the transformation result. From here, users can now use the technology's native tooling, i. e., use Docker and its tooling to build the Docker images based on the translated Dockerfiles and the `kubectl` command-line tool to "apply" the produced Kubernetes descriptor files to a cluster. An in-depth step by step guide as well as a video[5] demonstrating the execution of the Kubernetes deployment is available online and part of our GitHub repository.

4 Related Work

The closest approach to ours is that by Brabra et al. [2], which exploits model-to-model and model-to-text transformations to obtain artifacts for deploying a TOSCA application with four production-ready deployment technologies. Similarly to our approach, Brabra et al. [2] identifies a subset of TOSCA that can be processed by restricting inter-component relationships to *horizontal* dependencies (indicating that a component connects to/depends on another), in order to generate the deployment artifacts for Docker, Juju, Kubernetes, or Terraform. Our solution can deal with a wider set of application topologies, as it also includes *vertical* dependencies (indicating that a component is installed/hosted on another), and it already targets nine more deployment technologies in addition to those targeted by Brabra et al. [2].

Other approaches are aiming at enacting the deployment of TOSCA applications, which can be clustered in three main categories [1]. We can indeed distinguish (i) solutions for *directly deploying* TOSCA applications, (ii) approaches *integrating TOSCA with other standards* for enhancing deployment automation, and (iii) solutions for deploying TOSCA applications on *existing deployment technologies*. The reference approach for (i) is the OpenTOSCA engine proposed by Breitenbücher et al. [3]. OpenTOSCA enables directly deploying TOSCA applications on a target infrastructure, by requiring to get installed on a management

[5] `https://github.com/UST-EDMM/tosca-lightning#video`

node. OpenTOSCA is intended to be itself the orchestrator of the application, and it currently does not support streamlining the deployment of an application to other existing deployment technologies. Similar considerations apply to all other existing approaches for directly deploying TOSCA applications on target infrastructures [9]. All those approaches rely on the availability of full-fledged TOSCA-compliant orchestrators. In contrast, our objective is to enable deploying TOSCA applications by means of production-ready deployment technologies.

Efforts integrating TOSCA with other standards, i. e., concerning (ii), have been published by Calcaterra et al. [6] and Kopp et al. [14]. Both approaches integrate TOSCA with BPMN to imperatively program the deployment of multi-service applications. Additionally, Glaser et al. [11] proposes a cloud application orchestrator based on the integration of TOSCA with OCCI. However, despite the fact that the presented approaches enhance deployment automation by integrating TOSCA with existing standards, they still rely on the installation of some ad-hoc engine for processing the proposed solution.

Lastly, there are the solutions enabling the deployment of TOSCA applications on existing cloud deployment technologies. For instance, Breiter et al. [4] illustrate how to deploy TOSCA applications on the IBM cloud computing infrastructure. Brogi et al. [5] propose the TosKer engine for deploying and managing TOSCA applications on Docker-enabled infrastructures. Carrasco et al. [7] enable trans-cloud application deployment by allowing to run TOSCA application specifications on top of Apache Brooklyn. Additionally, Gusev et al. [12], Katsaros et al. [13] and Tricomi et al. [17] propose different approaches for deploying TOSCA applications on OpenStack cloud infrastructures. However, all these efforts target a precise cloud deployment technology. In contrast, our approach uses transformation and enables the deployment of TOSCA applications using the 13 most used production-ready deployment technologies, such as Terraform, Chef, or Puppet.

5 Conclusions

In this paper, we presented the open-source *TOSCA Lightning* toolchain, which enables the specification of multi-service applications in TOSCA, validating their compliance with TOSCA Light, and generating artifacts for enacting their deployment on 13 production-ready deployment technologies. We also presented an end-to-end case study illustrating how to exploit the *TOSCA Lightning* toolchain to specify the deployment of a third-party application and to automatically obtain the artifacts for effectively running the application with Kubernetes.

We plan to further evaluate TOSCA Light and the *TOSCA Lightning* toolchain in practice, by applying them to real-world industrial case studies. Further, as immediate future work, we plan to extend the TOSCA Lightning toolchain by a *Deployment Technology Integration Framework*. Based on the TOSCA Lightning transformation result, the goal is to directly trigger the automated deployment by unifying and encapsulating respective deployment technology APIs.

Acknowledgments. Work partially funded by projects *RADON* (EU, 825040), *SustainLife* (DFG, 379522012), and *DECLware* (Univ. of Pisa, PRA_2018_66).

References

1. Bellendorf, J., Mann, Z.A.: Specification of cloud topologies and orchestration using TOSCA: A survey. Computing (2019)
2. Brabra, H., Mtibaa, A., Gaaloul, W., Benatallah, B., Gargouri, F.: Model-Driven Orchestration for Cloud Resources. In: 2019 IEEE Int. Conf. on Cloud Computing (CLOUD). pp. 422–429 (2019)
3. Breitenbücher, U., et al.: The OpenTOSCA Ecosystem – Concepts & Tools. European Space project on Smart Systems, Big Data, Future Internet - Towards Serving the Grand Societal Challenges – Volume 1: EPS Rome 2016 (2016)
4. Breiter, G., et al.: Software defined environments based on TOSCA in IBM cloud implementations. IBM Journal of Research and Development 58(2/3), 9:1–9:10 (2014)
5. Brogi, A., Rinaldi, L., Soldani, J.: TosKer: A Synergy Between TOSCA and Docker for Orchestrating Multicomponent Applications. Software: Practice and Experience 48(11), 2061–2079 (2018)
6. Calcaterra, D., Cartelli, V., Di Modica, G., Tomarchio, O.: A Framework for the Orchestration and Provision of Cloud Services Based on TOSCA and BPMN. In: Cloud Computing and Service Science. CCIS, vol. 864, pp. 262–285. Springer (2018)
7. Carrasco, J., Durán, F., Pimentel, E.: Trans-cloud: CAMP/TOSCA-based bidimensional cross-cloud. Computer Standards & Interfaces 58, 167 – 179 (2018)
8. Chandrasekaran, A.: Gartner Report: Best Practices for Running Containers and Kubernetes in Production (2019)
9. Cloudify: TOSCA Orchestration & Training. `https://cloudify.co/tosca` (2020)
10. Endres, C., et al.: Declarative vs. Imperative: Two Modeling Patterns for the Automated Deployment of Applications. In: Proc. of the 9th Int. Conf. on Pervasive Patterns and Applications. pp. 22–27. Xpert Publishing Services (2017)
11. Glaser, F., Erbel, J., Grabowski, J.: Model Driven Cloud Orchestration by Combining TOSCA and OCCI. In: Proc. of the 7th Int. Conf. on Cloud Computing and Services Science - Volume 1: CLOSER,. pp. 672–678. SciTePress (2017)
12. Gusev, M., Kostoska, M., Ristov, S.: Cloud P-TOSCA porting of N-tier applications. In: 2014 22nd Telecommunications Forum Telfor (TELFOR). pp. 935–938 (2014)
13. Katsaros, G., et al.: Cloud Application Portability with TOSCA, Chef and Openstack. In: 2014 IEEE Int. Conf. on Cloud Engineering. pp. 295–302 (2014)
14. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: BPMN4TOSCA: A Domain-Specific Language to Model Management Plans for Composite Applications. In: Business Process Model and Notation. pp. 38–52. Springer Berlin Heidelberg (2012)
15. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – a modeling tool for tosca-based cloud applications. In: Int. Conf. on Service-Oriented Computing. pp. 700–704. Springer (2013)
16. OASIS: TOSCA Simple Profile in YAML Version 1.3 (2019)
17. Tricomi, G., et al.: Orchestrated Multi-Cloud Application Deployment in OpenStack with TOSCA. In: 2017 IEEE Int. Conf. on Smart Computing. pp. 1–6 (2017)
18. Wurster, M., et al.: The EDMM Modeling and Transformation System. In: Service-Oriented Computing – ICSOC 2019 Workshops. pp. 294–298. Springer (2019)
19. Wurster, M., et al.: The Essential Deployment Metamodel: A Systematic Review of Deployment Automation Technologies. SICS Software-Intensive Cyber-Physical Systems (2019)
20. Wurster, M., et al.: TOSCA Light: Bridging the Gap Between the TOSCA Specification and Production-Ready Deployment Technologies. In: Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020). pp. 216–226. SciTePress (2020)