Towards an Approach for Automatically Checking Compliance Rules in Deployment Models

Markus Philipp Fischer, Uwe Breitenbücher, Kálmán Képes, and Frank Leymann

Institute of Architecture of Application Systems, University of Stuttgart, 70569 Stuttgart, Germany
Email: {fischer, breitenbuecher, kepes, leymann}@iaas.uni-stuttgart.de

***Abstract*—An enterprise's information technology environment is often composed of various complex and heterogeneous systems and is subject to many requirements, regulations, and laws. This leads to the issue that technical experts should also have a firm knowledge about a company's compliance requirements on information technology. This paper presents an approach to ensure compliance of application deployment models during their design time. We introduce a concept that is able to locate compliance relevant areas in deployment models while also specifying how these areas have to be modeled to fulfill the compliance requirements.**

Keywords–Cloud Computing; Compliance; Security; Policies;

I. Introduction

An enterprise's information technology (IT) is subject to many regulations and laws, such as the German *Federal Data Protection Act* [1] or the ISO 27018 standard [2], which is especially concerned with data protection for cloud services (i.e. the privacy of personal data). Modern applications are often composed of various different and heterogeneous systems and form complex composite applications [3]. To avoid failures that are most often caused by human operators [4], there are efforts to automate the deployment and provisioning of applications [5] while also considering non-functional security requirements [6]–[10]. These approaches are implemented in various *deployment technologies* that consume *deployment models* to automatically deploy the described application. These models typically describe the structure of the deployment and the desired configuration, for example that an Java-based web application shall be hosted on an Apache Tomcat that has to be installed on a new virtual machine of a certain type. Unfortunately, companies are subject to a variety of requirements, therefore, it is difficult to ensure that modelers of deployment models are aware of all requirements that must be considered in these models to follow the company's compliance. Moreover, even modelers that are aware of compliance aspects can make mistakes that lead to deployments that are not conformant to the compliance regulations of the company. However, violations of compliance, in turn, can quickly result in serious consequences for the company's business.

In this paper, we present our work in progress about automating compliance ensurance for deployment models based on the *Topology and Orchestration Specification for Cloud Applications (TOSCA)* [11], a standard that allows the description of Cloud applications and their deployment. We introduce a concept for specifying *Deployment Compliance Rules for TOSCA models* that enable compliance experts to define reusable rules that can be used to ensure the compliance of deployment models. The approach enables the automation of compliance checking for deployment models during their design time. As a work in progress, the concept is not yet detailed completely and not implemented but will be integrated into the open source TOSCA modeling tool *Eclipse Winery* [12].

The remainder of this short paper is structured as follows: Section II describes the motivation and presents the concepts of TOSCA. Section III introduces the concept for our approach, followed by Section IV, which introduces selected works that are related. Finally, Section V draws a conclusion on the work in progress and gives an outlook on future work.

II. Motivation

In this section, we describe a scenario to motivate our concept. The described scenario will be used throughout the paper to describe our approach. We also present the fundamentals of TOSCA, the basis of our prototype, in which the presented concept will be realized. Everything needed to understand our approach is described in this section, however, we refer interested readers to the TOSCA Specification [11], the TOSCA Primer [13], and the TOSCA Simple Profile [14] to provide more detailed information about the OASIS standard. TOSCA is a technology-agnostic approach that can be used to orchestrate other deployment technologies, such as *Chef* or Cloud provider APIs. Therefore, it is a suitable basis for a technology-independent compliance checking approach.

The basic structure of an application modeled in TOSCA consists of a *Topology Template*, which is a directed multi-graph. The nodes of the graph represent software or infrastructure components such as a virtual machine, an Apache PHP Server, or an operating system. These components are represented in TOSCA as *Node Templates*. Node Templates can be connected via directed edges that describe the relationship between two adjacent nodes, such as a "hostedOn" or a "connectsTo" relationship. These edges are called *Relationship Templates*. *Node Types* and *Relationship Types* provide the semantics for nodes and edges in the Topology Template. Both are reusable classes that also allow the definition of properties, such as username and password for a virtual machine or a server. Figure 1 depicts our motivation scenario. The scenario consists of a PHP Application that is connected to a MySQL database. In this example, the PHP Application could be a website where users can register themselves with their personal data to receive regular newsletters from the company. The MySQL database is modeled as a Node Template with, for example, a property that semantically defines the type of data that is stored in the database. Both the application stack on the left side (PHP App, Apache PHP Server, Ubuntu 14.04 VM) and the application stack on the right (MySQL DB,

 ISBN: 978-1-61208-582-1

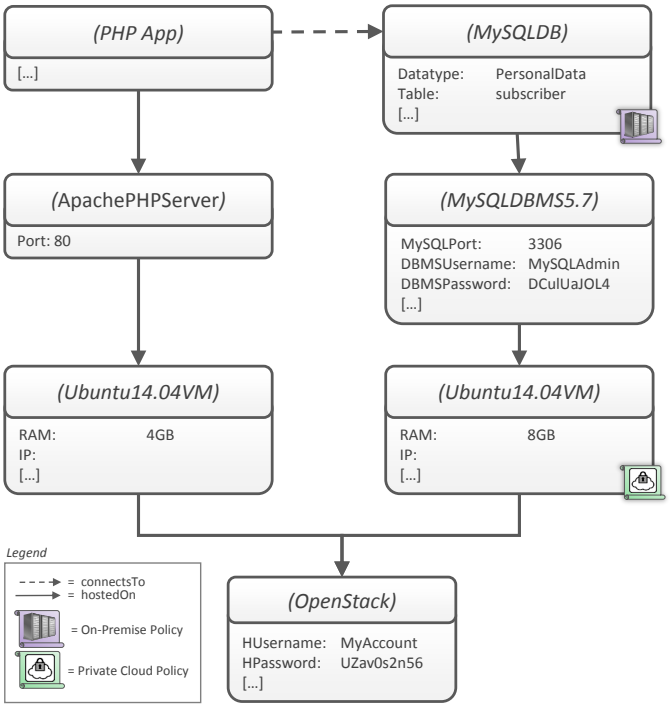

Figure 1. Motivating Scenario with two stacks modeled using the visual TOSCA notation *Vino4TOSCA* [15].

MySQL DBMS 5.7, Ubuntu 14.04 VM) are hosted on a Node Template of Node Type "OpenStack". The Relationship Type "hostedOn" means that the component where the Relationship Template originates, is installed on the target component, while "connectsTo" specifies that the PHP application connects to the MySQL database to store data [13]. The property *Datatype* with value "PersonalData", set in the MySQL database, indicates that the data stored in the database is personal data. If a company using this scenario is located in Germany, it falls under the German Federal Data Protection Act [1], which requires that the data is stored securely and access of third parties to the data is prevented. Another requirement is the adherence to the ISO 27018 standard [2], which is used to certify companies in the area of privacy of personal data in clouds. In our scenario, the company tries to enforce strict data security rules to avoid losing any personal data, to avoid coming in conflict with any law, and also they place importance to discretion. Thus, when personal data is involved, the company's compliance requirement is that it is to be ensured that the data is hosted *On Premise* and in a *Private Cloud*. Deployment models such as depicted in Figure 1 can be automatically executed with TOSCA Runtime Environments, such as OpenTOSCA [5]. Additionally, TOSCA not only allows the specification of application topologies and their orchestration but also enables modeling non-functional requirements using policies, e.g., concerning security or quality of service (QoS). For reusability, TOSCA provides *Policy Types*, which are classes of policies that can also have properties. The actual values of the properties are specified within the *Policy Templates* (*Policy Definitions* in the Simple Profile [14]). A Policy Template is associated with a Policy Type that can be associated with a set of Node Templates the policy can be applied to. The TOSCA Specification does not require a specific format for the policies, so any language is usable. Example policies used in our approach are the *On-Premise Policy* and the *Private Cloud Policy*. The On-Premise Policy is intended to restrict the deployment of components to pre-defined locations that are "On-Premise". This means that the associated components have to be hosted physically on infrastructure that is on the site of a company. This is often practiced with applications that process sensitive data, for example in international customs. With a Private Cloud Policy, a modeler can enforce that any Node Template the Policy is applied to, is hosted in a company-internal data center. This policy is related to the Cloud Computing Pattern *"Private Cloud"* by Fehling et al. [16]. Thus, policies are an instrument to address non-functional concerns such as security and quality of service requirements. However, the knowledge about compliance requirements is often held by compliance experts. Therefore, modelers can be unaware of the requirements, leading to non-compliant applications. It is desired that application models, which are ready to be provisioned, are compliant to the company's requirements. Therefore, each model has to be checked for compliance by experts, which is a time-consuming and error-prone task when done manually, especially for large and complex models. Thus, the issue of compliance checking for deployment models should be automated. In this paper, we present an approach for compliance checking in deployment models on the basis of the TOSCA standard.

III. Towards an Approach for Automatically Checking Compliance Rules in Deployment Models

This section introduces our concept for automated compliance checking of deployment models. We introduce *Deployment Compliance Rules* that can be modeled by compliance experts, which enable to automatically ensure that a certain compliance requirement is satisfied by a deployment model. Deployment Compliance Rules allow compliance experts to model allowed deployment structures, which are compliant to a company's compliance requirements, such as enforcing non-functional security requirements on customer data. Figure 2 shows an example of such a rule. A Deployment Compliance

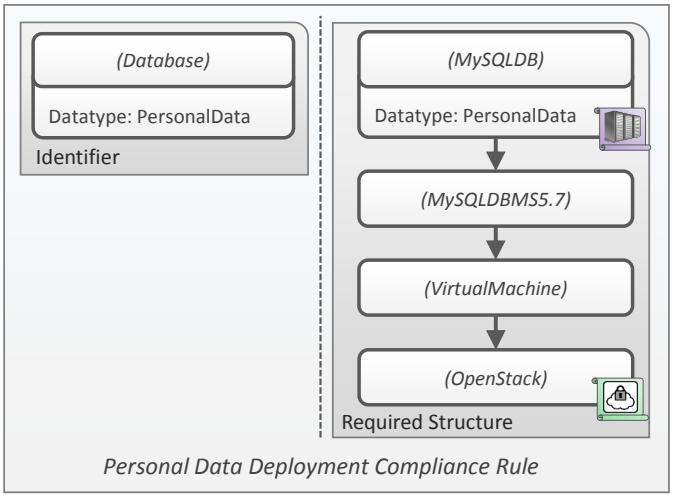

Figure 2. Example for a Compliance Rule.

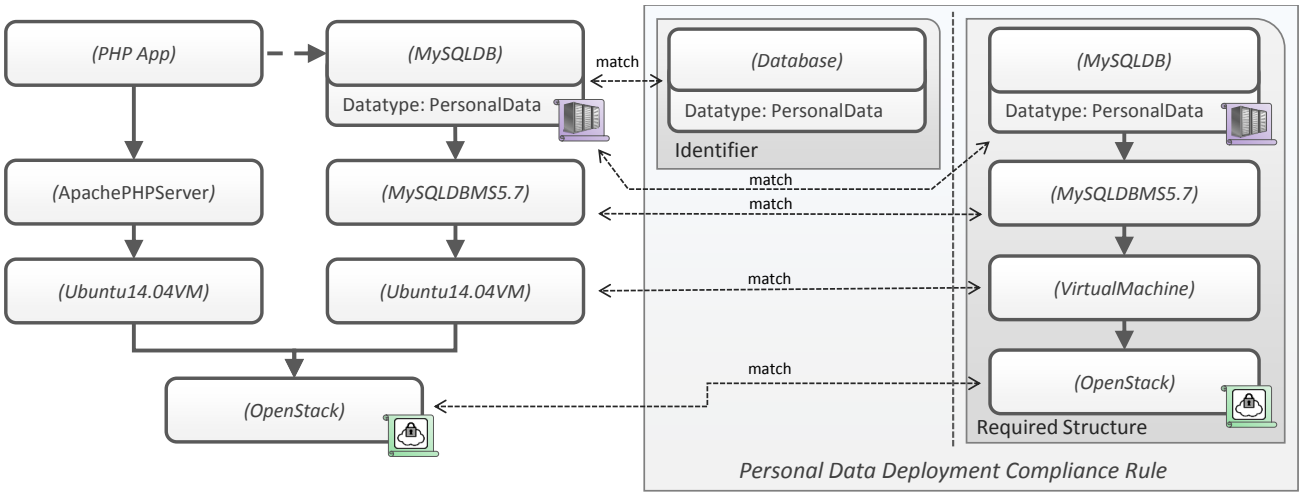

Figure 3. This figure shows how the Identifier and Required Structure of the Personal Data Deployment Compliance Rule is matched to the motivation scenario.

Rule consists of two directed, typed, and attributed multi-graphs, one being the *Identifier* graph (see left-hand side in Figure 2), the other being the *Required Structure* graph (see right-hand side in Figure 2). The basic idea is to use the Identifier to find compliance-relevant areas in the deployment model and to compare them to the defined Required Structure, which defines allowed structures for this Identifier. To find relevant areas and to compare them to the Required Structure, we reduce the problem to a *subgraph matching problem* [17]. The Identifier is used to define compliance-relevant areas of a deployment model as abstract as possible. TOSCA enables this by allowing to specify Node Types that are abstract. Each Node Type inherits the properties of Node Types it is derived from. We explain this matching concept on the basis of the *Personal Data Deployment Compliance Rule* scenario shown in Figure 2. The intention of this rule is to ensure that all databases of applications that store personal data have to be deployed on the local OpenStack of the company. Therefore, the Identifier graph depicted in Figure 2 on the left-hand side consists of a single abstract "Database" Node Template with the property "Datatype" and value "PersonalData". With this construct we ensure that all Node Templates that are derived from the "Database" Node Template are matched to this Identifier. The goal is to identify all areas of the deployment model where the Deployment Compliance Rule has to be applied. The Required Structure graph is used to define the proper structure and semantics of the area of the deployment model the Identifier matches, i.e., how the Deployment Compliance Rule must be fulfilled. In Figure 2 on the right side, the Required Structure specifies that all matching databases storing personal data must follow the structure of a "MySQLDB" NodeTemplate that is connected via "hostedOn" Relationship Templates to "MySQLDBMS5.7", "VirtualMachine", and "OpenStack" Node Templates. Similar to the "Database" Node Template in the Identifier graph, the "VirtualMachine" Node Template is abstract and, thus, allows any kind of virtual machines to be used as operating system for the database. The Policy Templates "Private Cloud' and "On-Premise" are attached to the Node Templates of the Required Structure to specify that the matched parts of the deployment model must be hosted on a private cloud and on premise.

A. Algorithm Sketch

Figure 3 shows the basic idea of the approach. The deployment model is tested for each defined Deployment Compliance Rule separately. In the following, we describe the algorithm for one exemplary rule namely the *Personal Data Deployment Compliance Rule* introduced in the previous section. The first step is to identify all areas where the rule has to be applied, i.e., all areas of the model that match the Identifier of the rule. In Figure 3, the Identifier consists of a single "Database" Node Template that has "PersonalData" as a "Datatype" property. Via subgraph isomorphism the Identifier can be matched to the "MySQLDB" Node Template, which has the same property. The match is indicated by the dashed double arrow between the Identifier and the deployment model. In a second step the Identifier's match has to be checked against the Required Structure. The deployment model is searched for subgraphs that contain the identified subgraph as well as the Required Structure. In Figure 3, the Required Structure consists of a "MySQLDB" Node Template with an "On-Premise" Policy attached and the specified Property and value. The "MySQLDB" Node Template has to be hosted on a Node Template of the specific type "MySQLDBMS5.7", which has to be hosted on a Node Template derived from the abstract Node Template "VirtualMachine". The last Node Template in the Required Structure is an "OpenStack" Node Template that has a "Private Cloud" Policy attached. As indicated by the dashed double arrows in Figure 3 there is a subgraph that matches to the Required Structure. Because the "MySQLDB" Node Template matched to the Identifier is also a part of the subgraph matched to the Required Structure, the Deployment Compliance Rule is fulfilled. The subgraph isomorphism problem is an NP-complete problem, therefore, its execution time has to be discussed. The Deployment Compliance Rule example shown in Figure 2 is defined by two graphs that have to be matched to the overall deployment model and represents a typical use case for the Deployment Compliance Rules defined in this paper. As the two graphs are very small graphs the execution time should be reasonable. However, if the rules become larger, the execution time will also increase. Therefore, the approach is suited for Compliance Rules of a small size to check the compliance of deployment models.

 ISBN: 978-1-61208-582-1

IV. Related Work

In this section, we present works that are related to our approach for compliance checking during design time. Martens et al. [18] propose a Reference Model that allows to capture several aspects of risk and compliance management in Cloud Computing. They use the Unified Modeling Language (UML) as modeling language and define their model as class diagrams. The model provides four different prospectives on risk and compliance management as they separate it by concerns. They provide constructs to characterize a Cloud computing service that also includes concepts such as *private cloud* or the location of a data center. The reference model also provides language constructs to associate a service with business processes, service level agreements, key performance indicators, risk factors, and compliance regulations. Schleicher et al. [19] introduce *Compliance Domains* to model data-restrictions in Cloud environments. Their focus is on the compliant execution of business processes in Cloud environments. Their approach allows to define certain areas of business processes, expressed in the Business Process Model and Notation (BPMN) [20], as Compliance Domains to be annotated by compliance experts with service level agreements and compliance rules, which are written in XPath and are intended to validate, if the data that is entering a Compliance Domain fulfills the compliance requirements. Schleicher et al. use a blood donation process as an example, where the name of a donor is not allowed to be associated with the donation within a certain Compliance Domain. The validation of a modeled business process is done during the design time of the process and the modeler is notified if any rules have been violated, to be corrected. The approach's method is similar to ours since compliance experts are required to specify the rules for compliance checking. They also introduce an algorithm that identifies Compliance Domains in existing business processes based on compliance rules and data flow in the processes. The method is similar to ours with the exception that we integrate locating compliance relevant areas in our Deployment Compliance Rules.

V. Conclusion

In this paper, we presented our work in progress to prevent modelers of deployment models from bothering with compliance requirements. We introduced Deployment Compliance Rules that can validate a deployment model during the design time. We separated the concerns of technical expertise and knowledge of a company's compliance requirements. Compliance experts are enabled to create such rules that can identify compliance relevant areas in deployment models while also providing the modeler with allowed structures that fulfill compliance requirements. In future work, we focus on the implementation and integration of the approach into the TOSCA modeling tool Winery [12] and also provide experimentation results on the execution time with various sizes of deployment models and Deployment Compliance Rules.

Acknowledgement

This work was partially funded by the German Research Foundation (DFG) project ADDCompliance and the BMWi project SmartOrchestra (01MD16001F).

References

[1] Federal data protection act. [Online]. Available: https://www.gesetze-im-internet.de/englisch_bdsg/ [Accessed: 2017-07-17]

[2] ISO/IEC 27018:2014 Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors, International Organization for Standardization Std., 2014.

[3] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and J. Wettinger, "Integrated Cloud Application Provisioning: Interconnecting Service-Centric and Script-Centric Management Technologies," in On the Move to Meaningful Internet Systems: OTM 2013 Conferences (CoopIS 2013). Springer, Sep. 2013, pp. 130–148.

[4] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems (USITS 2003). USENIX, Jun. 2003, pp. 1–1.

[5] T. Binz et al., "OpenTOSCA - A Runtime for TOSCA-based Cloud Applications," in Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013). Springer, Dec. 2013, pp. 692–695.

[6] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and M. Wieland, "Policy-Aware Provisioning of Cloud Applications," in Proceedings of the Seventh International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2013). Xpert Publishing Services, Aug. 2013, pp. 86–95.

[7] U. Breitenbücher et al., "Policy-Aware Provisioning and Management of Cloud Applications," International Journal On Advances in Security, vol. 7, no. 1&2, 2014, pp. 15–36.

[8] T. Waizenegger et al., "Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing," in On the Move to Meaningful Internet Systems: OTM 2013 Conferences. Springer, Sep. 2013, pp. 360–376.

[9] T. Waizenegger, M. Wieland, Tobias, U. Breitenbücher, and F. Leymann, "Towards a Policy-Framework for the Deployment and Management of Cloud Services," in SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies. IARIA, August 2013, pp. 14–18.

[10] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From Security to Assurance in the Cloud: A Survey," ACM Computing Surveys (CSUR), vol. 48, no. 1, 2015, p. 2.

[11] OASIS, Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[12] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann, "Winery – A Modeling Tool for TOSCA-based Cloud Applications," in Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013). Springer, Dec. 2013, pp. 700–704.

[13] OASIS, Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[14] OASIS, TOSCA Simple Profile in YAML Version 1.0, Organization for the Advancement of Structured Information Standards (OASIS), 2015.

[15] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and D. Schumm, "Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA," in On the Move to Meaningful Internet Systems: OTM 2012 (CoopIS 2012). Springer, Sep. 2012, pp. 416–424.

[16] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer, 2014.

[17] B. Gallagher, "Matching structure and semantics: A survey on graph-based pattern matching," AAAI FS, vol. 6, 2006, pp. 45–53.

[18] B. Martens and F. Teuteberg, "Risk and compliance management for cloud computing services: Designing a reference model," Risk, vol. 8, 2011, pp. 5–2011.

[19] D. Schleicher et al., "Compliance domains: A means to model data-restrictions in cloud environments," in Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International. IEEE, 2011, pp. 257–266.

[20] OMG, Business Process Model and Notation (BPMN) Version 2.0, Object Management Group (OMG), 2011.

 ISBN: 978-1-61208-582-1