Towards a Policy-Framework for the Deployment and Management of Cloud Services

Tim Waizenegger, Matthias Wieland
Institute of Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
{waizentm, wieland}@ipvs.uni-stuttgart.de

Tobias Binz, Uwe Breitenbücher, Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
{binz, breitenbuecher, leymann}@iaas.uni-stuttgart.de

***Abstract*—As the adoption of Cloud Computing is growing, the automated deployment of cloud-based systems is becoming more and more important. New standards, such as TOSCA (OASIS), allow the modeling of interoperable Cloud services. It is now possible to build reusable and portable cloud services that can be (semi-) automatically deployed by different cloud-deployment-engines at various Cloud environments. However, there is still an acceptance problem among potential users, especially in the enterprise segment, that stems from security issues like data security. To improve security in automatic Cloud management engines, this paper proposes a framework for processing non-functional requirements of Cloud services.**

Keywords-Cloud Computing; Security; Policy-Framework; TOSCA; Cloud Service; Cloud Management

I. INTRODUCTION

According to the definition of NIST [3], Cloud Computing is a model for enabling ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources. An important aspect is the fast deployment of Cloud Computing resources with minimal management effort. To achieve this goal, the new Cloud standard TOSCA [2] was developed allowing the portable modeling and automatic deployment and management of Cloud services. This enables the effortless migration of Cloud services across different Cloud environments. The TOSCA specification provides a domain-specific language to describe Cloud services based on different components and their relationships using a so-called Service Template (ST), which describes the topology of services. The orchestration via so-called management plans enables automated deployment and management of Cloud services. This allows the deployment of a TOSCA-based service in any Cloud environment that supports the execution of these models. In addition, TOSCA allows defining non-functional requirements of Cloud services based on so-called policy types that we use to define security requirements.

We work on implementing OpenTOSCA - an open source TOSCA container - that enables the automatic deployment of TOSCA based application models. Figure 1 shows an example for such an application. The diagram uses the visual notation Vino4TOSCA [1]. It defines a Cloud service running on two virtual machines with a separate stack for the database and web server. Both machines run the Ubuntu Linux operating system. The diagram shows the web server, PHP module and web application installed on one machine, and the database on the other.

To secure the application, we define different policies for

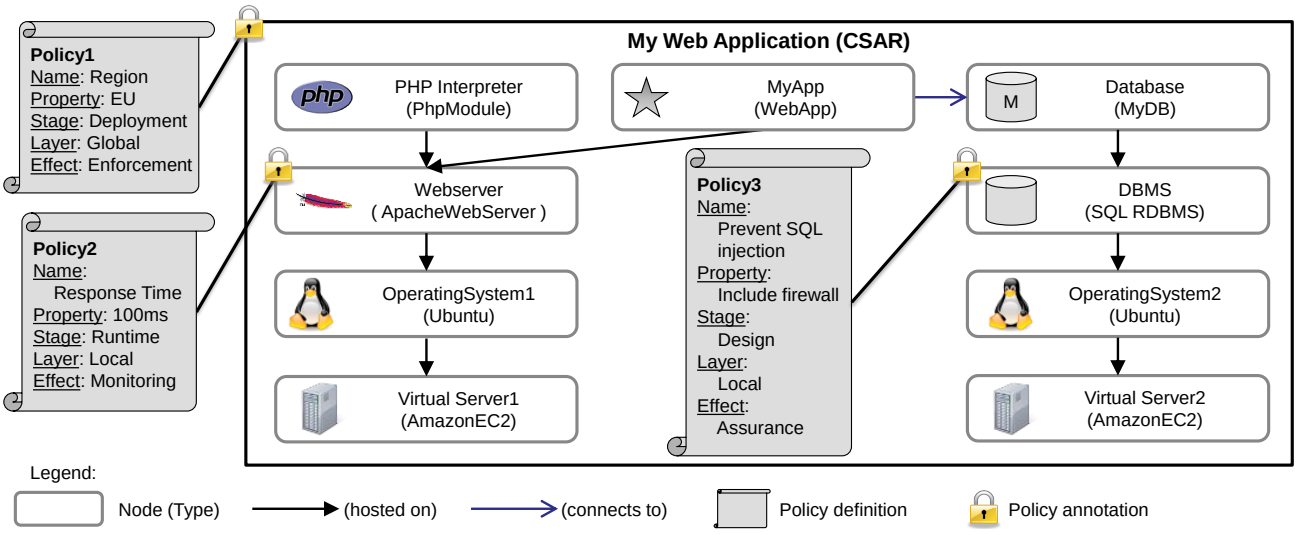

Figure 1. Example Scenario for a Cloud Service Policy in TOSCA.

 ISBN: 978-1-61208-298-1

the service model. *Policy1* requires that all components have to be deployed in a specific geographic region, due to data privacy reasons, as the data managed in the system must not leave Europe. *Policy2* specifies the maximum response time for HTTP requests on the web server in order to stay within customer requirements. When the defined response time is exceeded, the policy will deploy another instance of the web server to provide more capacity. *Policy3* is a security policy for the database system. When enabled, it adds a special application-firewall to the service topology, which provides protection against SQL-injection attacks.

OpenTOSCA does not yet support Policies; so, our next step is to make OpenTOSCA policy-aware. For that purpose, this paper presents the architecture of a Policy-Framework for security related issues in Cloud service deployment and management. The focus of the proposed Policy-Framework is the specification of non-functional requirements and their automatic processing in a Cloud service deployment engine (TOSCA container). OpenTOSCA can already deploy this application but without fulfilment of the policies. In the remainder of the paper we show how we plan to extend the system to realize the implementation of the policies.

This work takes a different approach than other publications that focus on specifying frameworks for implementing different methods to provide security features, e.g., authentication across different providers or trust management [4]. The goal of this paper is to introduce different aspects of policies and to evaluate how to use policies for secure Cloud service deployment and management with TOSCA. It is not meant to be a complete list covering all aspects of policies. We are working towards building a generic framework that supports the aspects mentioned in this paper as well as the ones emerging in future research. The definition of interoperable standard policy types should be addressed in future publications in the context of the TOSCA specification.

The remainder of this paper is structured as follows: Section II explains the ecosystem where policies are used. Section III presents a taxonomy for describing policies. Section IV describes an architecture implementing the different technical aspects of policies for Cloud service deployment. In Section V, we summarize our findings.

II. POLICY ECOSYSTEM

The source of policies, especially in the context of Cloud security, is usually a Service Level Agreement (SLA) between the Cloud provider and the customer. This SLA comprises the business perspective of the policy, whereas the specific implementation is considered the technical perspective.

The business perspective determines the business aspects of the policy such as price, penalties, and legal obligations as well as the subject.

The technical perspective is derived from the requirements given by the business perspective and describes the specific way in which the policy is implemented.

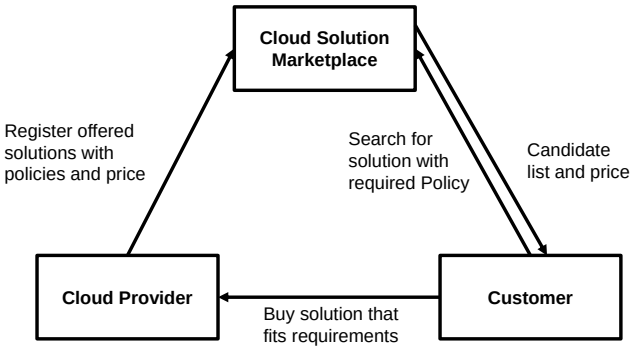

Figure 2. Actors in Ecosystem and Their Perspective on Policies.

Figure 2 shows the actors in this ecosystem. The customer requests the policy that he requires for his service. Then either a marketplace already provides the requested solution with the needed policy or the customer orders such a solution from a service provider. The customer can then buy the solution and contract a Cloud provider to deploy it while complying with the negotiated SLA and, therefore, the resulting policies.

III. TAXONOMY AND ASPECTS OF POLICIES

A policy in its most basic form is a single property that is attached to a service, a component of a service, or a relationship between components. That property is used as a parameter for determining the behavior of a system or process within the lifecycle of the service. A policy is therefore, defined by a) its *property* and b) the *aspects* that define the type of property and for which operation or system it is used during which stage of the lifecycle. We propose the following list of policy aspects as a generic model:

1) Stage in service lifecycle
   a) Design
   b) Provider selection
   c) Deployment
   d) Runtime
   e) Termination
2) Layer in topology
   a) Global policy (for whole Service Template)
   b) Local policy (for specific node or relationship type)
3) Policy effect:
   a) Assurance
   b) Enforcement
   c) Monitoring

A. The Signature of a Policy

The aspects introduced above identify a policy and comprise its signature much like the signature of a program function. The values for name, stage, layer, and effect are fixed since they are determined by the implementation of the specific policy. The policy property is variable and can be selected by the customer in order to fine-tune the behavior of the policy (see response-time policy example below) or if appropriate, it can be a fixed value as well. It should be noted that a property can be an atomic value or a complex

 ISBN: 978-1-61208-298-1

type with multiple values. Policy internal dependencies between property values have to be considered when defining the properties. Dependencies to properties of other policies should be handled by providing dependency-aware policy implementations.

B. Policy Aspects in Detail

For our Cloud policy taxonomy, we define three categories of aspects: *1) lifecycle stage*, *2) affected topology layer*, and *3) policy effect.* The following list gives a detailed description of the aspect categories.

1) Stage in Service Lifecycle

The first aspect defines the stage in the lifecycle of a Cloud service. Figure 3 shows the different stages that every Cloud service undergoes. This aspect describes at which stage the policy has to be fulfilled and implemented. However, a policy implemented at one stage might still affect the service behavior during the subsequent stages.

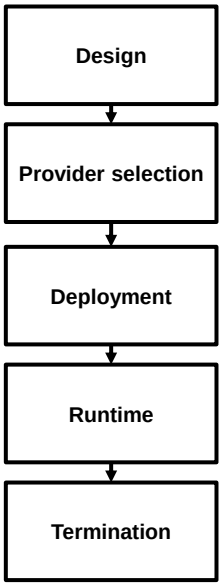

Figure 3. Lifecycle Aspects of Policies.

a) Design

In this phase, a service provider develops the solution as a Service Template (TOSCA model). Policies in this stage of the lifecycle are non-functional requirements either provided by the customer or chosen by the service provider himself. This includes the choice of software components, infrastructure requirements, and topology layout. These policies are not programmatically enforced or defined; they rather comprise a set of guidelines for service development to which the provider adheres.

b) Provider Selection

The provider selection is the first stage where policies are defined in a machine-readable form in order to be automatically evaluated. The requirements defined by these policies are used to determine, which Cloud providers are suited for service deployment. They include capabilities like redundant networking and disaster recovery, off-site backup, and the geographic location of the data center.

c) Deployment

After handling the previous aspects, the selected provider deploys the solution. Policies affecting this lifecycle stage determine the structure of the service topology and the configuration of its components. The service is set up according to the value of the policy property. Policies at this stage may determine characteristics such as the amount of memory and processing that get assigned to a certain node, the type of encryption used in transport protocols or the level of error-logging.

d) Runtime

The runtime phase starts after the service is deployed and available. It ends when service termination is requested. Policies belonging to the runtime stage determine the behavior of the service in a certain situation. This includes the response to events such as a high response time or system load, components becoming unavailable or other events such as suspicious system/user behavior. Therefore, policies at this stage often have a policy effect of the type *monitoring*.

e) Termination

This is the last stage in the service lifecycle. Policies of this type determine the actions that are taken when the service is terminated. This includes secure deletion of customer data and sending shutdown-notifications to connected clients.

2) Layer in Topology and Ecosystem

The second aspect classifies the annotation of the policy in the topology template. Policies may be annotated to a specific node or relationship with a specific type, e.g., a database or database-connection relationship. These are local policies. Global policies are annotated to the whole topology template. These affect the entire service and often recursively apply to many components of the topology. By defining boundary definitions, it is possible to select a subset of a topology and annotate this subset only. This subset is handled in a similar way as a global policy.

3) Policy Effect

This aspect defines what effect a policy has or what operations it triggers.

a) Assurance

This policy effect indicates that the property of a policy is assured by the design of the service or the Cloud provider. It is not programmatically enforced during runtime or deployment but rather the service provider has to engineer the service in a way that it guarantees compliance with the policy. Policies with this effect usually affect the lifecycle stages prior to runtime. Policies with this effect include geographical data center location and redundant networking.

b) Enforcement

Policies with this effect are actively enforced during a certain stage of the service lifecycle. They determine the service behavior according to the set property. They usually occur in the lifecycle stages including and after deployment. Policies with this effect include setting up encrypted transport channels and encrypting stored data.

 ISBN: 978-1-61208-298-1

c) Monitoring

Monitoring policies determine certain parameters of the service that trigger an operation. They usually occur in the lifecycle stages after deployment. Contrary to enforcement policies, they do not determine service behavior by default but rather react to a certain event, although these two policy effects are similar in the way that they trigger operations. Policies with this effect include automatic scaling and error notification. Table I shows three example policies and their main aspects.

TABLE I. EXAMPLE POLICIES WITH DIFFERENT ASPECTS

	Region policy	**SQL-Injection firewall policy**	**Response time policy**
Property	EU	Include firewall	100ms
Lifecycle stage	Deployment	Design	Runtime
Topology layer	Global	Local (DB node)	Local (HTTP node)
Policy effect	Enforcement	Assurance	Monitoring

Aspects of policies often depend on the specific implementation and most policies can be defined and implemented using different aspects. The Region Policy, for example, could be enforced during provider selection by selecting a provider that uses data centers in the EU. It could also be enforced during service design by implementing the service model in a way that it will only deploy on servers within the EU.

IV. ARCHITECTURE OF THE POLICY-FRAMEWORK

Figure 4 shows the architecture consisting of components for the different lifecycle stages of the Cloud service:

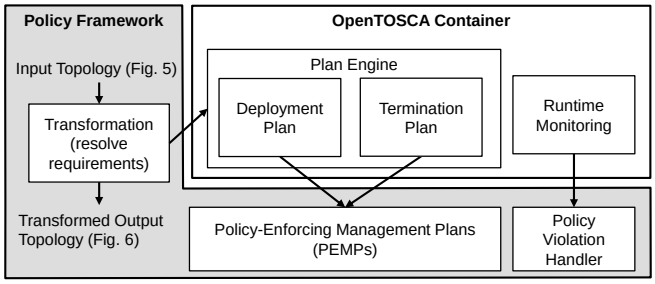

Figure 4. Architecture of the Policy-Framework.

Initially, the Cloud service is described as a solution package using TOSCA. This solution can be annotated with various policies. This TOSCA description is the input for the first stage, the *transformation*. The solution can contain *abstract components* that represent a *requirement* for a certain functionality. These abstract components are refined to *specific components* during the transformation while the annotated policies and functional requirements are considered. We therefore, differentiate between the requirement for a policy and a policy. A policy is provided as part of a system component and it can be enabled and configured according to its property. A requirement for a policy, therefore, limits the choice of components to the ones providing that policy. The transformation also adapts the deployment plan of the solution in order to cope with and install the newly added components.

Figure 5 shows an example for the transformation with an annotated requirement for a policy, here a requirement for SQL injection prevention is annotated. Figure 5 is an excerpt of the main example in Figure 1 and shows the database stack from the example service topology only. The transformation then processes this Service Template and retrieves a security-policy-pattern for the annotated requirement.

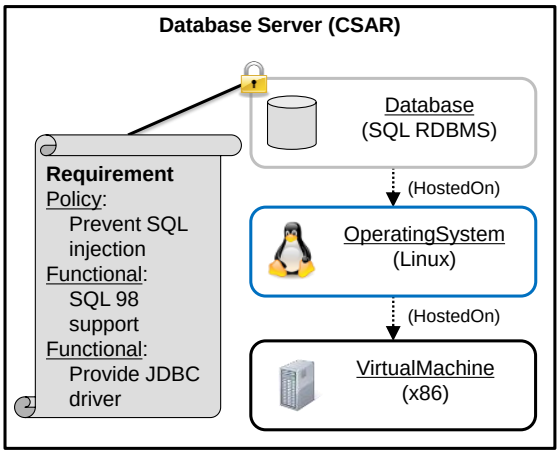

Figure 5. Transformation Example - Input Topology.

Figure 6 shows the result where the abstract database is replaced by a specific one (Oracle 11g). Furthermore, to secure this Database System a new node *database firewall* is added. To show that the new solution implements the policy requirement. Additionally, the new node it annotated by a "Prevent SQL injection" policy that is realized by the design of the system, which documents that the system is now fulfilling the policy.

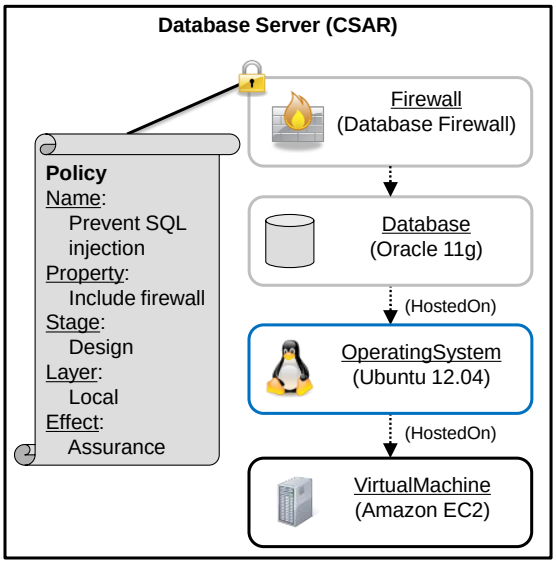

Figure 6. Transformation Example - Resulting Topology.

 ISBN: 978-1-61208-298-1

The next step in the architecture in Figure 4 is the installation of the *transformed solution* to the TOSCA container. For that purpose, the TOSCA container interprets the solution and starts the Cloud service deployment. The plan engine installs the package. Afterwards, Cloud services can be instantiated by using the *deployment plan*.

Policies that have the aspects enforcement and deployment must be enforced during deployment of the service. For that purpose, so-called *Policy-Enforcing Management Plans* (PEMPs) are used, which implement the functionality that guarantees that the defined policies are met.

The next step in the lifecycle is the runtime phase of the Cloud service. This phase starts after the deployment has successfully finished. During the runtime phase, different policies can be in effect, e.g., the *response time policy*. For this purpose, a *Runtime Monitoring* component is used, which continuously monitors the service and detects policy violations. A violation is handled based on the requirements, e.g., stop the service or scale up using a *Policy Violation Handler*.

The last step in the lifecycle is the *termination* of the service. In this step, the same mechanism used during deployment (PEMPs) is used. A typical task for termination policies is to guarantee secure deletion of all customer data.

V. CONCLUSION AND OUTLOOK

The contribution of this paper is to introduce and establish aspects of policies in the context of Cloud service definition and to present a first architecture realizing these aspects in a Policy-Framework. The proposed framework supports non-functional aspects in Cloud service models that are built using TOSCA.

Security is a major concern in using Cloud Computing for outsourcing data and services especially in the enterprise segment. It is, therefore, important to agree upon a common standard for describing, implementing, and realizing security policies in Cloud service models.

The next steps are to implement the different components of the proposed policy-framework, but even more important is the definition of a catalog of security policies and their implementation in topology components using TOSCA.

ACKNOWLEDGMENT

This work was partially funded by the BMWi project CloudCycle (01MD11023).

REFERENCES

[1] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and D. Schumm. Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA. In *CoopIS*, pp. 416–424. Springer-Verlag, 2012.

[2] P. Lipton, S. Moser, D. Palma, and T. Spatzier. Topology and Orchestration Specification for Cloud Applications (TOSCA). http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html, March 2013.

[3] P. Mell and T. Grance. The NIST Definition of Cloud Computing. *Recommendations of the National Institute of Standards and Technology*, Special Publication 800-145:7, 2011.

[4] H. Takabi, J.B.D. Joshi, and G.-J. Ahn. SecureCloud: Towards a Comprehensive Security Framework for Cloud Computing Environments. In *Computer Software and Applications Conference Workshops (COMPSACW)*, IEEE 34th Annual, pp. 393–398, 2010.

 ISBN: 978-1-61208-298-1