Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. ????

strapping Provisioning Engines for On-demand Provisioning in Cloud Environm

Lukas Reinfurt

Studiengang: Informatik

Prüfer: Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Betreuer: Dipl.-Inf., Dipl.-Wirt. Ing.(FH) Karolina Vukojevic
begonnen am: ??.??.2013
beendet am: ??.??.2014

CR-Klassifikation: ???

Abstract

Table of Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

AMI	Amazon Machine Images, page 11
BPEL	Business Process Execution Language, page 10
BPMN	Business Process Model and Notation, page 10
CSAR	Cloud Service Archive, page 11
ESB	Enterprise Service Bus, page 13
OASIS	Organization for the Advancement of Structured Information Standards, page 9
TOSCA	Topology and Orchestration Specification for Cloud Applications, page 9

1 Introduction

Workflow technology and the service based computing paradigm were mostly used in a business context until now. But slowly they are extended to be used in other fields, such as eScience, where business centric assumptions that where previously true aren't reasonable anymore. One of these assumptions is that it makes sense to run services continuously. This made sense in large enterprise where those services are used often. Science, on the other hand, often takes a more dynamic approach, where certain services, for example for simulation purposes, are only used at certain times. In those cases, it would make more sense to dynamically provision service only when they are needed.

1.1 Task of this Diploma Thesis

The task of this diploma thesis is to design a lightweight bootstrapping system that can kick off dynamic provisioning in cloud environments. It should be able to provision various provisioning engines in all kinds of cloud environments. The provisioning engines then handle the actual provisioning of required workflow systems and services. A managing component that keeps track of provisioned environments is also part of this system.

Support for different cloud environments and provisioning engines should be achieved through means of software engineering. A functioning prototype that supports Amazon as cloud environment and TOSCA as provisioning engine should be implemented.

1.2 Structure of this Document

...

2 Fundamentals

2.1 Simtech

2.2 Bootstrapping

2.3 Cloud

2.4 Provisioning Solutions

This section describes some of the provisioning solution available today, in particular TOSCA and Open TOSCA, since those are used in the prototypical implementation later on.

2.4.1 TOSCA

Topology and Orchestration Specification for Cloud Applications (TOSCA) is a standard that is currently being worked on by the Organization for the Advancement of Structured Information Standards (OASIS)[1]. Its development is also supported by various industry partners, which include IBM, Cisco, SAP, HP and others. It's aim is to provide a language that can describe service components and their relations in a cloud environment independent fashion [4].

TOSCA defines an XML syntax, which describes services and their relations in a so called service templates. Figure 2.1 shows that a service template can be made of up of four distinct parts: Topology templates, orchestration plans, reusable entities, and artifact templates.

Topology templates, as seen on the left side of Figure 2.1, model the structure of a service as a directed graph. The vertices of the graph represent nodes, which are occurrences of a specific component, for example, an application server or a database. These nodes are defined

[1] https://www.oasis-open.org/

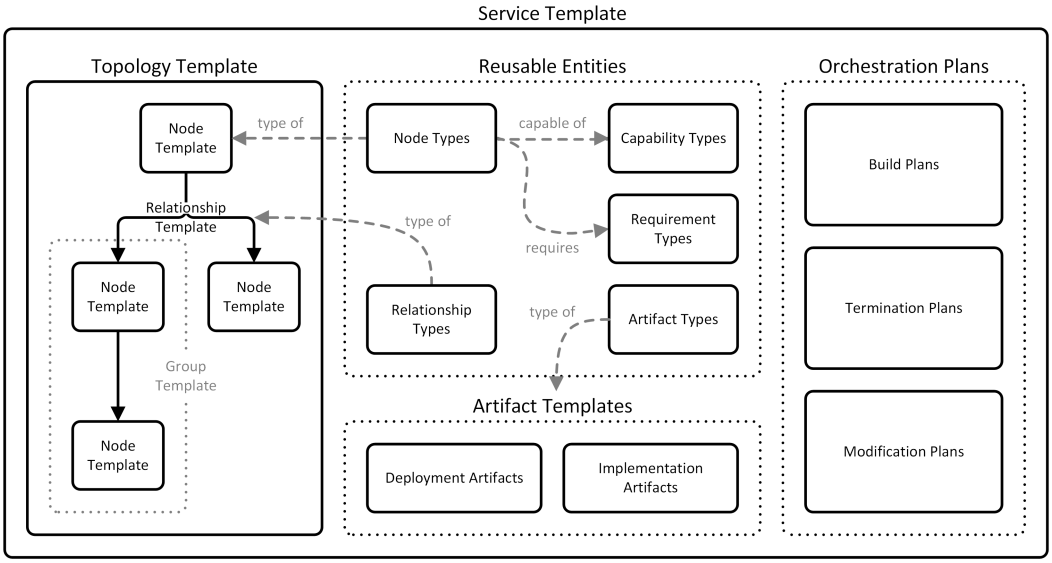

Figure 2.1: TOSCA service template structure [based on 4].

by node types, or by other service templates. Node types are reusable entities, as shown in the top center of Figure 2.1. They define the properties of a component, as well as operations to manipulate a component, so called interfaces. Additionally, node types can be annotated with requirements and capabilities. These, in turn, are defined by requirement and capability types, which also belong to the group of reusable entities. This allows for requirement and capability matching between different components. The edges of the graph represent connections between nodes, which are defined by relationship templates that specify the properties of the relation. An example for such a connection would be a node A, representing a web service, which is deployed on node B, an application server. Relationship types are also used to connect requirements and capabilities.

Orchestration plans, shown on the left of Figure 2.1, are used to manage the service that is defined by the service template. TOSCA distinguishes between three types of plans: Build plans, termination plans, and modification plans. Build plans describe, how instances of a service are created. Termination plans describe, how such a service is removed. Modification plans manage a service during its runtime. These plans consist of one or more tasks, i.e., an operation on a node (via an interface) or an external service call, and the order in which these tasks should be performed. They can be written in Business Process Model and Notation (BPMN) or Business Process Execution Language (BPEL), which are already existing languages to describe process models.

The bottom center of Figure 2.1 shows artifact templates, which represent artifact. Artifacts are things that can be executed directly (e.g.: scripts, archives) or indirectly (e.g.: URL, ports).

TOSCA further distinguishes between two types of artifacts, namely deployment and implementation artifacts. Deployment artifacts materialize instances of a node and are used by a build plan to create a service. An example for this is an Amazon Machine Images (AMI), which creates an Apache server once deployed in a VM. Implementation artifacts represent the interfaces of components. Here, an example would be a node that has an interface for starting the particular component described by the node. This interfaces could be implemented by an implementation artifact like a *.jar* file.

One or more TOSCA service templates are packaged, together with some meta data, into a Cloud Service Archive (CSAR), which is essentially a zip file that contains all files necessary to create and manage a service.bv CSAR files can then be executed in a TOSCA runtime environment, also called TOSCA container, to create the service described within.

2.4.2 OpenTOSCA

OpenTOSCA is an browser based open-source implementation of a TOSCA container, created at the IAAS at University Stuttgart, which supports the execution of TOSCA CSAR archives. Figure 2.2 shows the architecture of OpenTOSCA. The functionality of OpenTOSCA is realized in three main components, which are the Controller, the Implementation Artifact Engine, and the Plan Engine. After a CSAR is uploaded to OpenTOSCA it can be deployed in three steps. In the first step, the CSAR file is unpacked and its content is stored for further use. The TOSCA XML files are then loaded and processed by the Controller. The Controller in turn calls the Implementation Artifact Engine and the Plan Engine. The Implementation Artifact Engine knows how to deploy and store the provided implementation artifacts via plugins. Plans are then run by the Plan Engine, which also uses plugins to support different plan formats. OpenTOSCA also offers two APIs, the Container API and the Plan Portability API. The Container API can be used to access the functionality provided by the container from outside and to provide additional interfaces to the container, like the already existing admin UI, self-service portal, or modeling tool. The Plan Portability API is used by plans to access topology and instance information [1].

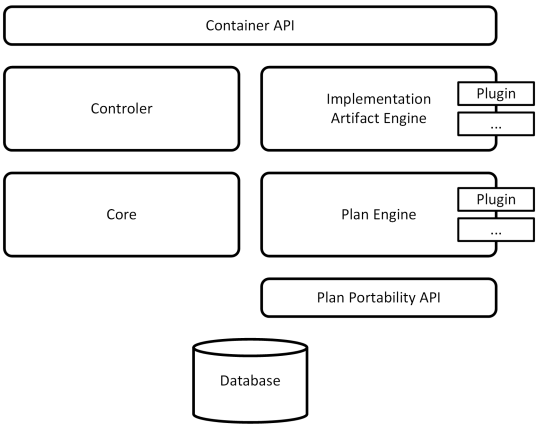

Figure 2.2: OpenTOSCA architecture [based on 1].

3 Related Work

This chapter summarizes related work of other authors that is of interest to this thesis.

Vukojevic-Haupt, Karastoyanova, and Leymann introduce a new binding strategy for services called on-demand provisioning. They also provide a middleware architecture that enables the provisioning of the software stack [5].

Schneider further refines the previously shown middleware architecture by adding a provisioning manager as intermediary between the Enterprise Service Bus (ESB) and the provisioning engines [3].

Kirschnick et al. present an architecture for automatic provisioning of cloud infrastructure and services [2].

4 Constrains and Design

5 Implementation

6 Summary and Conclusion

Bibliography

[1] Tobias Binz et al. “OpenTOSCA -- A Runtime for TOSCA-based Cloud Applications”. In: International Conference on Service-Oriented Computing. LNCS. Springer, 2013. URL: http://www.iaas.uni-stuttgart.de/RUS-data/INPROC-2013-45%20-%20OpenTOSCA_A_Runtime_for_TOSCA-based_Cloud_Applications.pdf.

[2] Johannes Kirschnick et al. “Toward an Architecture for the Automated Provisioning of Cloud Services”. In: *Communications Magazine* 48.12 (2010), pp. 124–131. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5673082.

[3] Valeri Schneider. “Dynamische Provisionierung von Web Services für Simulationsworkflows”. Diploma. University Stuttgart, 2013. URL: ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/DIP-3473/DIP-3473.pdf.

[4] *Topology and Orchestration Specification for Cloud Apllications*. Committee Specification 01. OASIS, 2013. URL: http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.pdf.

[5] Karolina Vukojevic-Haupt, Dimka Karastoyanova, and Frank Leymann. “On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows”. In: Proceedings of the 6th IEEE International Conference on Service Oriented Computing Applications (SOCA 2013). 2013. URL: ???.

All links were last visited on December 19, 2013.

Declaration of Authorship

I hereby certify that the diploma thesis entitled

Bootstrapping Provisioning Engines for On-demand Provisioning in Cloud Environments

is entirely my own work except where otherwise indicated. Passages and ideas from other sources have been clearly indicated. To date, neither this diploma thesis nor essential parts thereof were subject of an examination procedure. Until now, I don't have published this diploma thesis or parts thereof. The electronic copy is identical to the submitted copy.

Stuttgart, December 19, 2013, ...

(Lukas Reinfurt)