TOSCA
Topology and Orchestration Specification for Cloud Applications

Tobias Binz, Uwe Breitenbücher, Oliver Kopp, Frank Leymann

www.opentosca.org
Table of Contents

1. Motivation
2. TOSCA Basics
3. TOSCA Portability
Motivation
Motivation

- Many enterprises outsource their IT into “the Cloud”
- Cloud Computing provides some important benefits
  - Elasticity
  - Pay-On-Demand Computing / Pricing
  - Self-Service
  - Management Automation

- However, business applications have to be adapted to benefit from Cloud Computing advantages
  - Simply installing the whole software stack on a virtual machine and putting this VM into a Cloud is not appropriate in most cases
Motivation

- Migration of an application to the Cloud requires often \textit{fundamental} changes to benefit from the Cloud
- Different Cloud services provide different features
  - IaaS
  - PaaS
  - SaaS
  - DBaaS
  - ...
  - XaaS
- This results in \textit{complex composite applications} that employ multiple types of different Cloud services, middleware components, and software
The Challenges

- How to deploy such applications?
- How to manage such applications?
- How to monitor such applications?
- How to communicate the structure of such applications?
- How to achieve reliable operation?
- How to avoid vendor lock-in?
- How to achieve portability and interoperability?
- ...
The Challenges

- How to deploy such applications?
- How to manage such applications?
- How to monitor such applications?
- How to communicate the structure of such applications?
- How to achieve reliable operation?
- How to avoid vendor lock-in?
- How to achieve portability and interoperability?
- ...
Motivation

- Several technologies available
  - Cloud Provider DSLs and APIs
    - Amazon CloudFormation, Amazon AWS API, Microsoft Azure API, …
  - Cloud Abstraction Layers
    - OpenStack, DeltaCloud, …
  - Proprietary Solutions
    - IBM, HP, …
  - Script-based Configuration Management Technologies
    - Chef, Puppet, Juju, shell scripting, …

→ Heterogeneity, proprietary APIs, different security mechanisms, non standardized data formats, ….
Currently used Technologies and APIs

Currently in use are, for example, all these technologies:
Problems

- Each technology employs its own...
  - ... API(s)
  - ... domain-specific language(s) (DSLs)
  - ... invocation mechanisms
  - ... data model
  - ... wording
  - ... fault handling
  - ... security mechanisms
Motivation

- Integrating these technologies is a difficult challenge
- A lot of architecture and management expertise required
- Many low-level proprietary APIs
  - Difficult orchestration
  - Difficult automation

How to handle all these issues?
TOSCA Basics
OASIS TOSCA

Topology and Orchestration Specification for Cloud Applications

100+ participants from 40+ companies:
OASIS TOSCA

Topology and Orchestration Specification for Cloud Applications

Goals:
- Automation of Deployment and Management
- Portability
- Interoperability
- Vendor-neutral ecosystem
TOSCA Overview

Topology
Service Structure

OperatingSystem (Ubuntu 12.04 LTS)

WebServer (Tomcat)

EC2 VirtualServer (AWS EC2 Server)

Orchestration
Service Orchestration for Deployment & Management

Start VM

Install Tomcat
Graphical Example of an Application Topology

OnlineBookstore (WAR)

WebServer (Tomcat)

OperatingSystem (Ubuntu 12.04 LTS)

EC2 VirtualServer (AWS EC2 Server)
Graphical Example of an Application Topology

**Node Template**
- OnlineBookstore (WAR)

**Node Type**
- WebServer (Tomcat)

**Relationship Template**
- OperatingSystem (Ubuntu 12.04 LTS)

**Relationship Type**
- VirtualServer (AWS EC2 Server)

EC2
Properties

OnlineBookstore (WAR)

WebServer (Tomcat)
User: Admin
PW: 7a!7q

OperatingSystem (Ubuntu 12.04 LTS)

VirtualServer (AWS EC2 Server)
IP: 147.187.2.1
Deployment Artifacts

- Artifacts providing the node’s functionality
- Multiple Deployment Artifacts possible

**OnlineBookstore (WAR)**
- bookstore.war

**WebServer (Tomcat)**
- Tomcat.zip
- Tomcat7 (apt, deb, …)
- http://tomcat.apache.org

**OperatingSystem (Ubuntu 12.04 LTS)**
- Ubuntu.ovf
- ami-d0f89fb9t
- CustomizedUbuntu.img

© University of Stuttgart 19
Define management operations of nodes (and relationships)
- Input & output parameters and their data types
- Implemented by Implementation Artifacts (Web Service, REST-service, Script, ...)

OnlineBookstore (WAR) appSpecific
- start, stop
- deployWAR

WebServer (Tomcat)
- instanceType : String
- region : String
- accessKey : String
- ...

OperatingSystem (Ubuntu 12.04 LTS)
- installPkg
- execScript

VirtualServer (AWS EC2 Server)
- createVM
- terminate
- Instance ID : String
**Capabilities and Requirements**

**Requirement**

I need to be hosted on a Linux Operating System.

**Capability**

I provide a Linux Operating System (Ubuntu).

**Application (WAR)**

(hostedOn)

**WebServer (TomCat)**
TOSCA Overview

Topology
Service Structure

Orchestration
Service Orchestration for
Deployment & Management

Start VM → Install Tomcat →

- WebServer (Tomcat)
- OperatingSystem (Ubuntu 12.04 LTS)
- EC2 VirtualServer (AWS EC2 Server)
Two Flavors of Processing

- **Declarative ➔ What?**
  - Example: “I want this, realize it!”
  - Runtime interprets topology and does deployment

- **Imperative ➔ How?**
  - Example: “First do this, than that.”
  - Management plan explicitly describes each step

---

**Diagram:**
- Start VM
- Install Tomcat
Declarative vs. Imperative – Discussion

**Imperative** Style

- Logic completely contained in Application

**Declarative** Style

- Logic completely implemented by Container

**Hybrid Approach**

- Deployment Flow
- Flexibility & Customizability

- Ease of Modeling
- Low Entry Barrier
Example: Using workflows to deploy an application

Using workflows to deploy an application:

1. **Provision VM and OS**
   - Operating System (Ubuntu 12.04)
   - Web Server (Tomcat)

2. **Install Tomcat Web Server**
   - Deploy WAR on Tomcat

3. **Deploy WAR on Tomcat**
   - Online Bookstore (WAR)

4. **Online Bookstore (WAR)**
   - Web Server (Tomcat)
   - Operating System (Ubuntu 12.04)
   - VM (VirtualMachine)
   - Amazon EC2 (AmazonEC2)
Plans

- Plans are workflows (BPEL, BPMN, etc.)
  - Plans are portable, reusable, and automated
  - Parallel execution
  - Error handling
  - Traceability/auditability
  - Long running processes
  - Recoverability
  - Human tasks

- Management logic shipped together with application
  - Management not hard-wired into the TOSCA container
  - Contained in CSAR (TOSCA packaging format)
What's contained in an **Cloud Service Archive (CSAR)**?

- **Types**
  - Installables
  - Images
  - Services
  - Scripts
  - Deployment Artifacts
  - Implementation Artifacts

- **Management Plans**

- **Topology Templates**
  - OnlineBookstore (WAR)
    - (hostedOn) WebServer (Tomcat)
    - (hostedOn) OperatingSystem (Ubuntu 12.04 LTS)
    - (hostedOn) EC2 (AWS EC2 Server)
TOSCA Portability
Portability Note

- TOSCA deals with portability of Service Templates
- Portability of the ingredients of an IT Service (especially the code artifacts) is not addressed by TOSCA
- Similarly, mobility of data used by a corresponding service instance is not in the scope of TOSCA
Why is this portable?

- Management operations defined in TOSCA standard
- Management Operations, i.e., Implementation Artifacts, are deployed by the TOSCA runtime
- Plans defined using standards (e.g., BPEL, BPMN)
- Plans (1) and Management Operations (2) are “bound” (i.e., connected) by the container
  → Implementation Artifacts on the realization level