Agenda

• TOSCA Overview and Examples
• TOSCA Concepts Explained
• TOSCA advanced Features
Motivation for TOSCA

Even simple cloud services sometimes have complex software and hardware infrastructures backing them. So how do we ensure portability?
What is TOSCA?

Topology and Orchestration Specification for Cloud Applications

A language for defining Service Templates ...

... including a Topology Template describing the structure of a service

... including the definition of plans for orchestrating the application

Packaging format (CSAR) for packaging models and all related artifacts.

Definition of building blocks for services

... along with the implementation artifacts for manageability operations

... and the definition of deployment artifacts for components

Definition of possible links between components
Scenario 1: Portability of Cloud Applications
Scenario 2: Market-Place of Cloud Applications

1. Create
2. Publish
3. Browse and Select
4. Provision
5. Use

Service Catalogue

Service Template

Cloud Service Archetype (CSA)
TOSCA TC Members

3M
ActiveState
Axway
CA Technologies
CenturyLink
Cisco
Citrix
Cloudsoft
EMC
Fujitsu
Google
HP
IBM
Huawei
Jericho Systems
NetApp
Nokia Siemens
Pricewaterhouse
Primeton
Red Hat SAP
Software AG
VCE
Vnomic
WSO2
Zenoss
1st Interop Example: SugarCRM

Using TOSCA to model Applications

Language: TOSCA v1.0 spec

Node Type

Deployment Artifact

Implementation Artifact

Types

Language: TOSCA v1.0 spec

MySQL

MySQLDBContainer Req.

MySQL

LinuxOS Req.

RPMs: mysql

mysql-server

MySQLDBConnection

Property: DB Name: "mysqldb"

Properties:

- Context root: /mycrm
- HTTP Port: 8080

Templates

Properties:

- Admin User: myadmin

Primary focus for users

"Orchestrator"

Deployed Instances
Role of Relationships in Model Processing

- Use defined lifecycle operations to deploy and manage each component
- Use base relationship types to derive processing order
  - Process a host before a hosted component (HostedOn)
  - Process a provider before a client (DependsOn, ConnectsTo)
- If a component is related to another component, see if relationship injects additional processing logic (e.g. pre-configure endpoint)

... a pretty mechanical process
How is a Topology processed?

- Use base relationship types to derive component processing order
  - First process a host, then process hosted component
  - First process a component that another component depends on, then process the dependent component
  - First process a component that another component connects to, then process the connecting component

- For each component
  - Deploy its Deployment Artifacts
  - Invoke lifecycle operations in right sequence (create, configure, start ...); their can be no-ops

- If a relationship contributes logic, inject it into component operation invocations

**Diagram:**

- Web 1: create VM, perform base OS config, install and configure httpd, install and configure php runtime
- Web 2: create VM, perform base OS config, install and configure MySQL
- Web 3: configure database endpoint properties
- Web 4: run in parallel, create VM, perform base OS config, install and configure httpd, install and configure php runtime
- Web 5: create and configure SugarCRM database
- DB 1: create VM, perform base OS config
- DB 2: perform base OS config
- DB 3: install and configure MySQL
- DB 4: create and configure SugarCRM database

**Relationships:**

- Web 1 depends on DB 1
- Web 2 depends on DB 2
- Web 3 depends on DB 3
- Web 4 depends on DB 4
- Web 5 connects to DB 4

**Notes:**

- relationship injects logic
- creates VM
- deployed to OS
- configured with app
- configure database endpoint properties
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
Declarative vs. Imperative – Evaluation

**Imperative Style**
- Logic completely implemented by Cloud Template

**Declarative Style**
- Logic completely implemented by Container

**Hybrid Approach**

- Deployment Flow
- Flexibility & Customizability (Full Orchestration)

Ease of Modeling
- Low Entry Barrier
- (Deployment & Decommissioning)
How Do Plans Do Orchestration?

• Task of a plan refers to interface of a topology node

• Topology node specifies all interfaces offered to manage it
  • Interface is bound to a concrete implementation

• Implementation is copied from CSAR, or
  • Implementation already available at providers side, or
  • A standardized Cloud Interface (IaaS, PaaS, SaaS) is used, or ...
Some Application

Requirements can be fulfilled explicitly by other components in the model

Database Provider

Requirements can be fulfilled by the runtime

Requirements/Capabilities are base for substitutability
Model Composition

Subsystems can be abstracted in some models.

Other models can define details of subsystems. → separation of concern, re-use
Usage Scenarios for Model Composition & Substitution

Varying deployment options

- Single node web tier
- Scalable web tier

Layering of models

- Custom workload
- Multiple options of middleware deployments
- Multiple options of infrastructure configurations

Application Layer
- App
- DB
- Tomcat
- MySQL

Platform Layer
- VM
- Network
- Storage

Infrastructure Layer
- Custom workload
- Multiple options of middleware deployments
- Multiple options of infrastructure configurations
Learn more about TOSCA

• **TOSCA Specification, Version 1.0**  
  *Committee Specification 02, 9 May 2013,*  
  http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs02/TOSCA-v1.0-cs02.pdf

• **TOSCA Primer, Version 1.0**  
  *Committee Note Draft (CND) 01, Public Review Draft 01, 31 January 2013,*  
  http://docs.oasis-open.org/tosca/tosca-primer/v1.0/cnd01/tosca-primer-v1.0-cnd01.pdf

• **TOSCA Implementer's Recommendations for Interoperable TOSCA Implementations, Version 1.0**  
  *Interoperability Subcommittee, Working Draft 01, Rev. 02, 14 January 2013,*  

• **TOSCA Interoperability Subcommittee, SugarCRM Scenario Sample CSAR**  
  *Preliminary Draft CSAR for Interop. Testing against TOSCA v1.0 Specification,*  
Thank you for your attention!