Cyclone

Complete Dynamic Multi-cloud Application Management

TOSCA Templates for NFV and network topology description

CYCLONE Technical paper
September 2017

Table of Contents

European Commission

1. OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA)

OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) [TOSCA] allow the description of the applications deployable generally on multiple clouds and including applications servers or VMs, network components, their interconnections and corresponding deployment and configuration workflow. OASIS defines a special TOSCA Simple Profile for Network Functions Virtualization (NFV) [TOSCA-NFV] that can be used for Network Service Descriptor (NSD) and NFV descriptor (NFVD) specification.

TOSCA provides mechanisms to control workflow, describe relationships and reflect dependencies between resources. TOSCA defines a service template, where a "service" for TOSCA is an application or application component, such as a database server, firewall or virtual router. In a networking context, the term "service" implies the customer-specific configuration of a network function. Examples would be an individual Layer 3 VPN router or firewall configured with the customer-specific rules.

Both node and relationship types may define lifecycle operations to implement the behavior an orchestration engine can invoke when instantiating a service template. For example, a node type for some software product might provide a 'create' operation to handle the creation of an instance of a component at runtime, or a 'start' or 'stop' operation to handle a start or stop event triggered by an orchestration engine. Those lifecycle operations are backed by implementation artifacts such as scripts or Chef recipes that implement the actual behavior.

TOSCA based service description and orchestration is defined in a form of Cloud Service Archive (CSAR) file that need to be handled by the TOSCA enabled Orchestrator that is highly favoured by industry. TOSCA can be naturally combined with the YANG [YANG] and Netconf [Netconf] tools[1] where TOSCA can be responsible for the application and NFV based network topology description and deployment of the network elements or functions and YANG can describe actual network nodes/functions and their configuration that can be included in TOSCA template [ref Orchestrators]. YANG is a data modeling language used to describe configuration and state information. YANG has been used to model networking devices and services – i.e., an object and its attributes. YANG defines the data models that are then manipulated through the Netconf protocol.

The adoption of TOSCA is growing both by cloud applications developers and network community. The latter is especially stimulated by the growing adoption of the network virtualusation and Network Function Virtualisation (NFV) in particular.

In current standards development involving MEF, ETSI, TMF, OASIS and IETF, TOSCA has growing popularity and is expected to become an important component for VNF and SDN for higher level applications description and orchestrating. It is currently a part of ETSI NFV MANO architecture, interacts with TMF based OSS/BSS systems and existing MEF OpenLSO implementation[2].

[11] TOSCA and YANG https://www.cisco.com/c/dam/m/en_us/service-provider/ciscoknowledgenetwork/files/581_04_05-16-TOSKA-CKN.pdf

[2] What is the best NFV Orchestration platform? A review of OSM, Open-O, CORD, and Cloudify, by Dmitriy Andrushko, Gregory Elkinbard, March 8, 2017 [online] https://www.mirantis.com/blog/which-nfv-orchestration-platform-best-review-osm-open-o-cord-cloudify/

2. TOSCA Service template

The service template specifies topology (or structure) of services and orchestration (or invocation of management behavior). In other words, services topology defines services and their interconnection, services orchestration defines services interaction when performing their tasks. Orchestration is often represented by the process workflow.

Services are provisioned as a part of an IT infrastructure and their management behavior must be orchestrated in accordance with constraints or policies, e.g. in order to achieve service level objectives.

The TOSCA specification defines a metamodel for defining IT services that includes both the structure of a service and how to manage it. The structure of a service is defined as a Topology Template (also referred to as the topology model of a service). Plans define the process models that are used to create and terminate a service as well as to manage a service during its whole lifetime. The major elements defining a service are depicted in Figure 1. In a networking context, Node Type can be a network, networking device or VNF, Relationship Types can be interconnecting (virtual) links, Plan will define network deployment and configuration workflow.

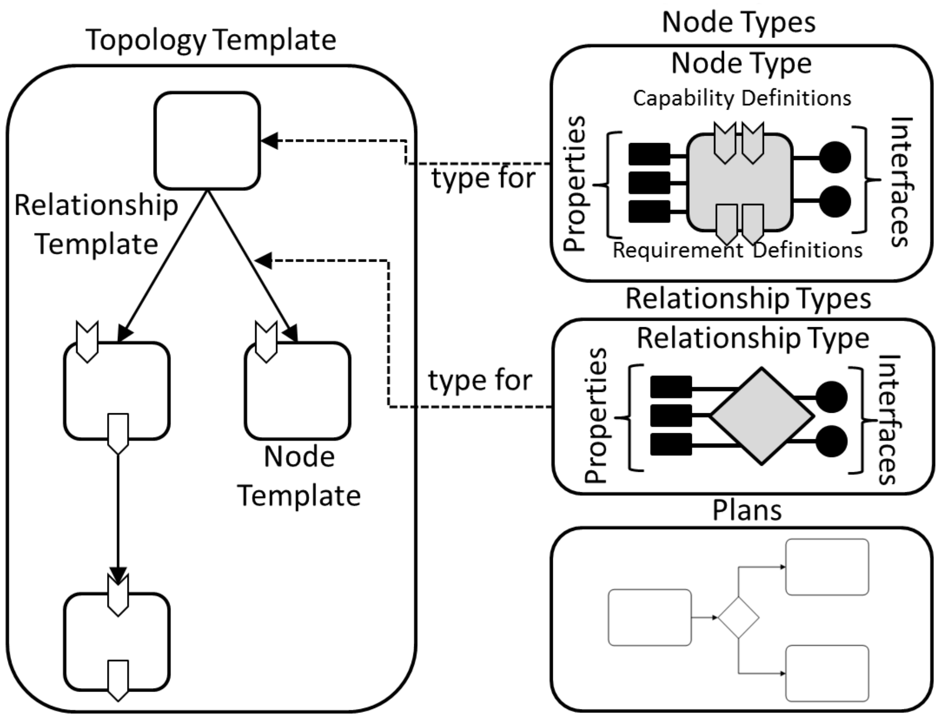


Figure 1. Service template and structural elements (OASIS TOSCA, 2013).

3. TOSCA Core Concepts

A Topology Template consists of a set of Node Templates and Relationship Templates that together define the topology model of a service as a (not necessarily connected) directed graph. A node in this graph is represented by a Node Template. A Node Template specifies the occurrence of a Node Type as a component of a service. A Node Type defines the properties of such a component (via Node Type Properties) and the operations (via Interfaces) available to manipulate the component.

Node Types are defined separately for reuse purposes and a Node Template references a Node Type and adds usage constraints, such as how many times the component can occur.

For example, consider a service that consists of an application server, a process engine, and a process model. A Topology Template defining that service would include one Node Template of Node Type “application server”, another Node Template of Node Type “process engine”, and a third Node Template of Node Type “process model”. The application server Node Type defines properties like the IP address of an instance of this type, an operation for installing the application server with the corresponding IP address, and an operation for shutting down an instance of this application server. A constraint in the Node Template can specify a range of IP addresses available when making a concrete application server available.

Plans defined in a Service Template describe the management aspects of service instances, especially their creation and termination. These plans are defined as process models, i.e. a workflow of one or more steps. Instead of providing another language for defining process models, the specification relies on existing languages like BPMN (Business Process Model and Notation) or BPEL (Business Process Execution Language).

The TOSCA specification defines a corresponding archive format called CSAR (Cloud Service ARchive) that contains all necessary information required to implement that service in a specific environment. This means that beside the service template of the cloud application, the deployment artifacts and implementation artifacts have to be available in that environment, i.e. VM image format, execution environment, application installation or deployable file.

4. Example TOSCA service description

We will discuss an example TOSCA service description provided in the OASIS TOSCA specification (OASIS TOSCA, 2013). Figure 2 depicts a sample Definitions file named Payroll.tosca containing a Service Template of an application. The application is a payroll application written in Java that MUST be deployed on a proper application server. The Service Template of the application defines the Node Template Payroll Application, the Node Template Application Server, as well as the Relationship Template deployed_on. The Payroll Application is associated with an EAR file (named Payroll.ear) which is provided as corresponding Deployment Artifact of the Payroll Application Node Template. An Amazon Machine Image (AMI) is the Deployment Artifact of the Application Server Node Template; this Deployment Artifact is a reference to the image in the Amazon EC2 environment. The Implementation Artifacts of some operations of the Node Templates are provided too; for example, the start operation of the Payroll Application is implemented by a Java API supported by the payrolladm.jar file, the installApp operation of the Application Server is realized by the Python script wsadmin.py, while the runInstances operation is a REST API available at Amazon for running instances of an AMI. Note, that the runInstances operation is not related to a particular implementation artifact because it is available as an Amazon Web Service (https://ec2.amazonaws.com/?Action=RunInstances); but the details of this REST API are specified with the operation of the Application Server Node Type.

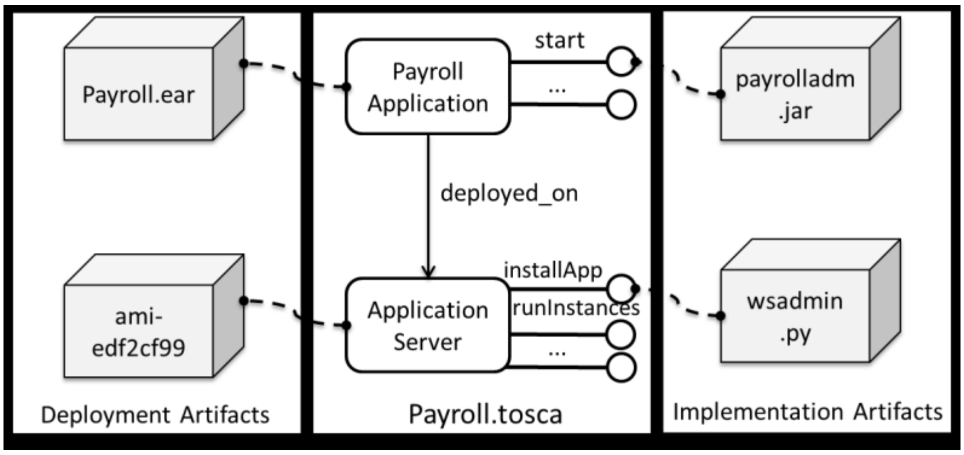
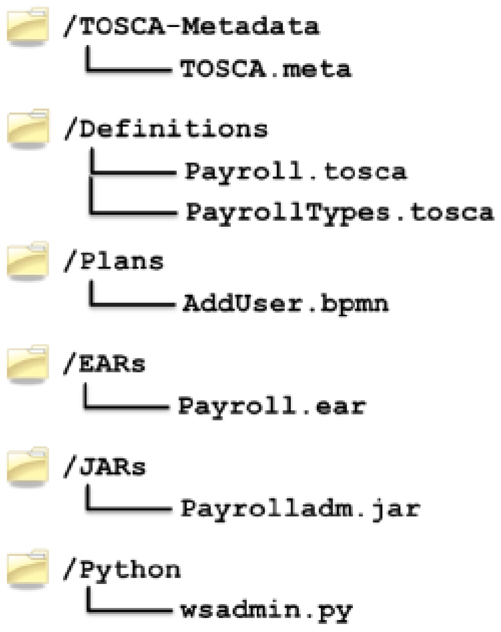


Figure 2. Sample Service Template Payroll Application (OASIS TOSCA, 2013)

The whole set of the Payroll Application components is provided as a container file CSAR (Cloud Service ARchive) that has its standard directories structure illustrated in Figure 3. The TOSCA.meta file is contained in the TOSCA-Metadata directory. The Payroll.ste file itself is contained in the Service-Template directory. Also, the PayrollTypes.ste file is in this directory. The content of the other directories has been sketched before.



```
/TOSCA-Metadata
  └──── TOSCA.meta
/Definitions
  ├──── Payroll.tosca
  └──── PayrollTypes.tosca
/Plans
  └──── AddUser.bpmn
/EARs
  └──── Payroll.ear
/JARs
  └──── Payrolladm.jar
/Python
  └──── wsadmin.py
```

Figure 3. Structure of CSAR sample file.

5. TOSCA Data Model for a network service

TOSCA Data Model for a network service is defined by the TOSCA Simple Profile for Network Functions Virtualization (NFV) [TOSCA-NFV] that provides the deployment template in NFV that fully describes the attributes and requirements necessary to realize such a Network Service. NFV Orchestrator (NFVO) manages the lifecycle of network service via the interface exposed by the VNF Manager (VNFM), and manages virtualized resources via the interfaces exposed by the VIM.

TOSCA defines the deployment template for a network service in NFV is a form of a Network Service Descriptor (NSD) that describes a relationship between VNFs and possibly PNFs that it contains and the links needed to connect VNFs. There are four information elements defined apart from the top level Network Service (NS) information element together with the corresponding descriptors:

- Virtualized Network Function (VNF) information element and VNFD
- Physical Network Function (PNF) information element and PNFD
- Virtual Link (VL) information element and VLD
- VNF Forwarding Graph (VNFFG) information element VNFGD

The mapping between TOSCA and NFV takes the following approach (see Figure 4):

1. NSD is described by using a service template,
2. VNFD, VNFFGD, VLD and PNFD is considered as node templates with appropriate node types.
3. VNFD can be further described by using another service template with substitutable node type.

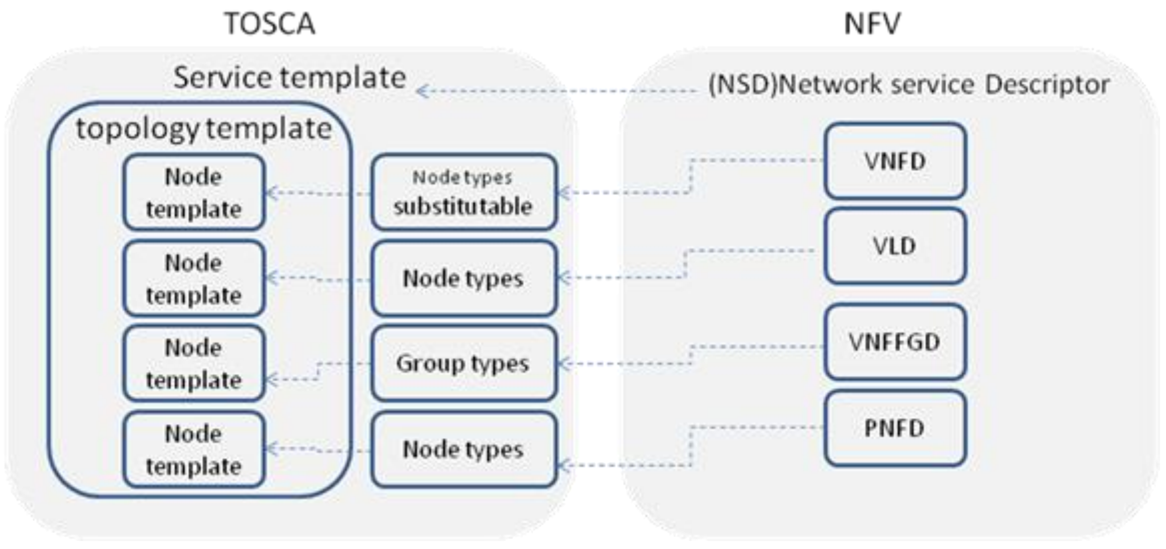


Figure 4. Mapping relationship between TOSCA and NFV.

Figure 5. provides example of the TOSCA Data Model for a network service. In this example, the network service includes three VNFs. Each VNF exposes different number of connection points, which represent the virtual and/or physical interface of VNFs. Virtual link (VL) describes the basic topology of the connectivity (e.g. ELAN, ELINE, ETREE) between one or more VNFs connected to this VL and other required parameters (e.g. bandwidth and QoS class).

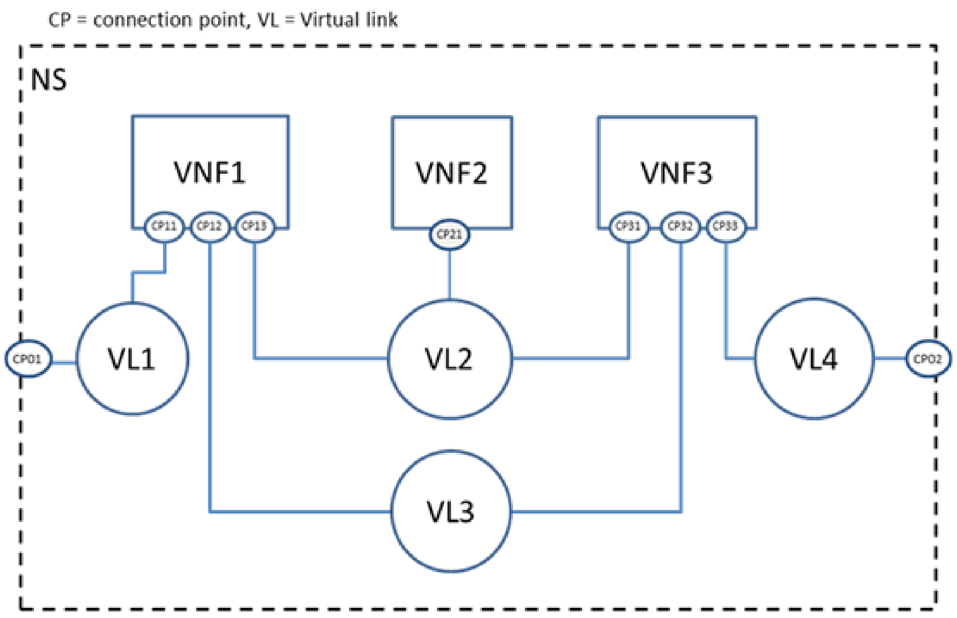


Figure 5. Example of the TOOSCA Data Model for a network service.

Figure 2.6 below provides example of the VNF topology description using YAML (YAML Ain't Markup Language) which is a human friendly data serialization standard used in TOSCA Simple Profile [TOSCA-YAML]. YAML is supported in many programming languages, in particular in Python.

```
tosca_definitions_version: tosca_simple_yaml_1_0

  topology_template:
  description: Template of a VNFD example

node_templates:
VNF1:
  type: tosca.nodes.nfv.VDUComposition.vnf1
  properties:
  # omitted here for brevity
  requirements:
    - virtual_link:VL1
    - virtual_link:VL2
    - virtual_link:VL3

VNF2:
  type: tosca.nodes.nfv.VDUComposition.vnf2
  properties:
  # omitted here for brevity
  requirements:
    - virtual_link:VL_2

VNF3:
  # it can be substituted with a topology provided by another template
  # that exports a virtual_link type's requirement.
  type: tosca.nodes.nfv. VDUComposition.vnf3
  properties:
  # omitted here for brevity
  requirements:
    - virtual_link:VL_2
    - virtual_link:VL_3
    - virtual_link:VL_4

VL_1:
  type: tosca.nodes.nfv.VnfVirtualLinkDesc
  properties:
  # omitted here for brevity
  capabilities:
    - virtual_link

VL_2:
type: tosca.nodes.nfv.VnfVirtualLinkDesc
  properties:
  # omitted here for brevity
  capabilities:
  - virtual_link
```

[VL_3 and VL_4 - skipped]

Figure 2.6. Example of the VNF topology description using YAML.

6. TOSCA Functionality supported in INDIGO Data Cloud

TOSCA NFV and its growing adoption by industry and research community (via INDIGO Data Cloud project) makes it important to be supported by the GEANT Cloud Connectivity service. TOSCA is currently supported by OpenStack Heat-Translator (for IaaS and applications orchestration) and Tacker (Network Function Orchestrator). TOSCA based NFV description and orchestration is supported by a number of Open Source and community projects such as Cloudify, OpenNFV, Open-O, Open Source MANO, and others.

INDIGO Data Cloud Infrastructure Manager (IM) is declared to support TOSCA template for applications orchestration.

Second release of the INDIGO Data Cloud PaaS Layer [INDIGO-DC D5.5] provides functionality to support TOSCA based services description by INDIGO-DC PaaS orchestrator and Infrastructure Manager (IM), which is OpenStack Heat Translator based.

The released PaaS layer provides automatic distribution of applications and/or services over a hybrid and heterogeneous set of IaaS infrastructures. It can be used with both private and public clouds. The PaaS layer is able to accept a description of a complex set, or cluster, of services/applications by mean of TOSCA templates, and is able to provide the needed brokering features in order to find the best fitting resources. During this process, the PaaS layer can also evaluate data distribution, so that the resources requested by the users are chosen by the closeness to the storage services hosting the data.

The Orchestrator delegates the deployment to the Infrastructure Manager (IM), to OpenStack Heat or to Mesos frameworks, depending on the submitted TOSCA templates and on the ranked list of sites.

INDIGO Configuration Management DB (CMDB) Service supports a set of configuration elements that are vital for INDIGO-DataCloud operations:

- providers - organizational entity that owns or operates the services;

- services (both computing and storage) - main technical component description defining type and location of technical endpoints;
- images - local service metadata about mapping of INDIGO-wide names of images, which are necessary to translate TOSCA description into service specific request.

The IM was already available in the first release and has been further enhanced in the context of INDIGODataCloud to support the TOSCA Simple Profile in YAML Version 1.0, an OASIS standard. The IM is used to provide TOSCA-compliant support for OpenNebula-based cloud site, thus introducing local site orchestration. It is also used to deploy TOSCA-compliant definitions of application architectures, i.e. TOSCA templates, on cloud sites external to INDIGO DataCloud, such as public clouds or even on-premises clouds that are not integrated with the IAM service.

In the period between the INDIGO-1 and the INDIGO-2 releases, several new features have been introduced. The complete list the recent developments are available in the releases section of the INDIGO GitHub repository[3]. The following are the main features of the current IM implementation:

- IM supports single-site configuration. This means that it can be configured to specifically deploy the infrastructure requests to the specific OpenNebula/OpenStack site for which it has been configured. In order to provide local site orchestration, the IM is typically deployed on the OpenNebula front-end.
- Support for the Open Telekom Cloud (OTC) run by TSystems has also been included, which, T-Systems being a key industrial partner of the INDIGO-DataCloud consortium, was a very desired functionality.
- Support for understanding VPC-related network information was introduced in the IM. This functionality was required to be able to deploy virtual infrastructures on AWS, since recent AWS accounts include a default VPC on which subnets exist to deploy the virtual machines. This way, TOSCA templates can now be deployed on AWS.

Automatic deployment through Ansible recipes embedded in TOSCA and HOT (Heat Orchestration Template) templates. Docker images have been provided for each component.

[3] https://github.com/indigo-dc/im/releases