ISSN:0975-9646

Implementation of Repositories in TOSCA-Parser

Mrs T.L.Priyadarsini[#1], Vemula Nandini[*2], Mr Tadepalli Srinivas[#3], Mr Sahdev Zala[*4]

[1] *Assistant Professor, Department of Computer Science and Engineering*
VNR Vignana Jyothi Institute of Engineering& Technology
Bachupalli, Nizampet Hyderabad, Telengana, INDIA

[2]*M.Tech CSE, Department of Computer Science and Engineering*
VNR Vignana Jyothi Institute of Engineering& Technology
Bachupalli, Nizampet Hyderabad, Telengana, INDIA

[3]*Solution Architect, Team Lead*
NextGen R&D, TCS,Hyderabad, INDIA

[4]*Advisory Software EngineerIBM*

***Abstract*— Topology and Orchestration Specification for Cloud Application (TOSCA), is an OASIS open standard provides new ways to enable portable automated deployment of and management of composite applications. TOSCA captures the description of cloud applications and infrastructure services, the relationship between parts of the services and operational behaviour of these services (e.g., Deployment, Configuration, Patch etc.). TOSCA support service template to ported applications over alternative cloud environments so that the services remain interoperable. TOSCA-Parser is a parser for TOSCA simple Profile in YAML. TOSCA has been using the notion of Containers from the very beginning, now TOSCA Committee adding some specific modelling TOSCA capabilities for PaaS Containers, such as ability to link to distributed Repositories, i.e., implementing the feature Repositories in TOSCA-Parser. Repository is a named external Repository which contains deployment and implementation artifacts that referenced within the TOSCA Service Template.**

***Keywords*— TOSCA, Containers, Repositories and Artifacts.**

I. INTRODUCTION

A. OpenStack

OpenStack is a free and open-source software platform for cloud computing, that is mainly deployed as an Infrastructure-as-a-Service (IaaS). OpenStack is a cloud operating system that controls large pools of compute, storage and networking resources that are managed through dashboard that gives administrators control while empowering their users to provision resources through a web interface.

B. HEAT

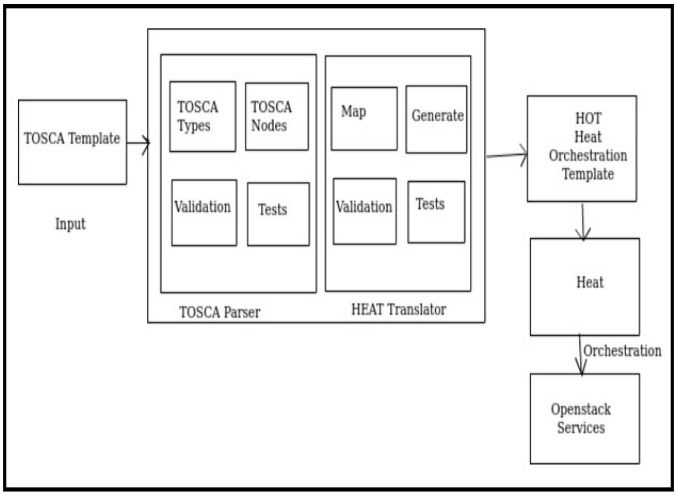
HEAT is an orchestration program to implement orchestration engine in order to launch multiple cloud applications for OpenStack. HEAT provides both a Cloud Formation-compatible Query API and an OpenStack-native ReST API.

C. HEAT-Translator

HEAT-Translator is project that is developed after LIBERTY release of an OpenStack. The project got split in to TOSCA-Parser and HEAT-Translator after LIBERTY release. Concept behind the development of HEAT-Translator is to translate the Non HEAT (TOSCA) templates to Heat Orchestration Template (HOT) files. Here Non HEAT files are translated to HOT files because these HOT files are deployed in OpenStack. Heat-Translator also support different template formats other that TOSCA templates.

HEAT-Translator tool takes an in-memory graph from TOSCA Parser as an input, maps it to HEAT resources and then produces a HOT.

Figure 1shows the Architecture of HEAT-Translator.



FigureI.1: Architecture of HEAT-Translator

D. TOSCA

TOSCA is a new open cloud standard constitutes a unique eco-system, which is supported by a large number of international industry leaders. TOSCA is an OASIS open standard that defines the interoperable description of services and applications hosted on the cloud and elsewhere, including their components, relationships, dependencies, requirements and capabilities, thereby enabling portability and automated management across cloud providers regardless of underlying platform or infrastructure, thus expanding customer choice, improving reliability, reducing cost and time-to-value. TOSCA enables an eco-system where service providers can compete and Differentiate to add value to Your Applications. TOSCA supports automated matching of application requirements to provider capabilities. TOSCA is a non HEAT template standard that is accepted by all cloud providers. TOSCA follows certain standards in writing the templates. If a template is developed according to TOSCA standards then no need of developing a separate template in case of migration from one cloud to other cloud.

TOSCA addresses critical cloud challenges:

- Speed and accuracy moving apps to Cloud
- Consumer Choice of Cloud vendor and technology
- Agility adapting to change (Business and IT)

II. TOSCA-PARSER

A. TOSCA-Parser

TOSCA-Parser is a project that is developed after LIBERTY release of an OpenStack. TOSCA-Parser is a parser that is developed for parsing TOSCA simple profiles that are written in YAML. It takes the Input as a service template after on successful validation a memory line graph is produced as an output. The service Template that is passed as an input is written in TOSCA standards. It creates a line graph of nodes and their relationships. TOSCA-Parser is a parser for TOSCA simple Profile in YAML.

B. TOSCA Template

Topology template defines the structure of a service. It is also known as topology model of a service. Plans are the process models that are used for service life cycle. Plans are used to create, terminate and manage the service through its life time. Topology template consists of relationship templates and node templates together constitute a topology template which may not be a directed graph all time. Every node in the memory graph is represented as a node type in the node template. Node type defines the properties of node like computing, operations that are required to manipulate the components. Node types are defined independently in order to reuse the components. Relationship templates specify the occurrences of relationships between the nodes in a topology template. Relationship types are defined independently for reuse purpose. Service template that is created is an instance for deployed service. These are derived by instantiating topology and service template. Process model contains tasks that refer to operations of interfaces of nodes and relationship templates.

Figure 2 shows the arrangement of a service template.

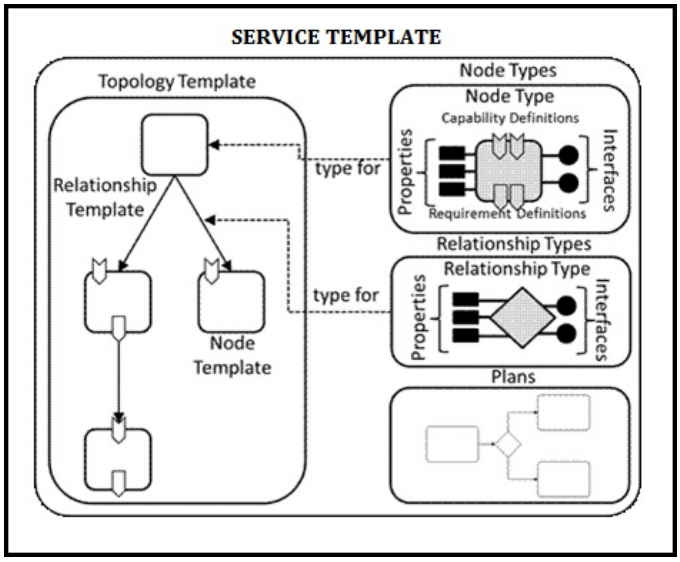


Figure 2: Service Template

III. REPOSITORIES

A Repository definition defines a named external repository which contains deployment and implementation artifacts that are referenced within the TOSCA Service Template.

List of recognized key names for a TOSCA repository definition:

- **Description**: The optional description for the repository.
- **URL**: The required URL or network address used to access the repository.
- **Credential**: The optional Credential used to authorize access to the repository.

Single line Grammar:

<repository_name>: <repository_address>

Multi line Grammar:

<repository_name>:
description: <repository_description>
url: <repository_address>
credential: <authorization_credential>

repository_name: represents the required symbolic name of the repository as a string.
repository_description: contains an optional description of the repository.
repository_address: represents the required URL of the repository as a string.
authorization_credential: represents the optional credentials (e.g., user ID and password) used to authorize access to the repository.

A. Repositories Design and Implementation

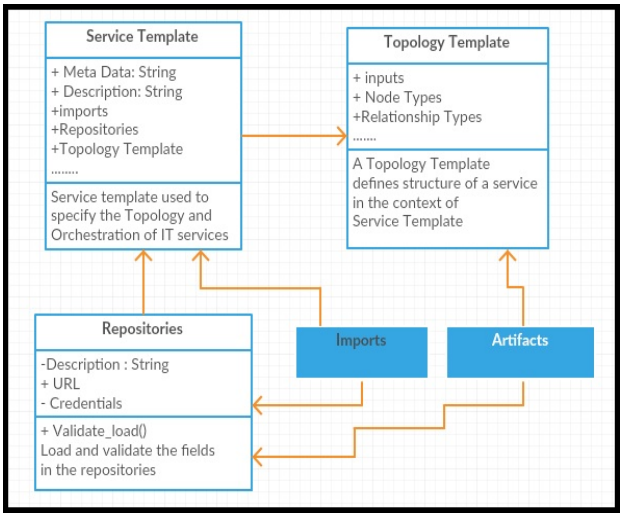


Figure 3: Static view for Repositories Implementation

Initially when TOSCA Template is run, providing template as an input file then the execution starts from the shell.py main function from there which is moved to the Toscatemplate.py file where the repositories class is being called if the parser is met with the repositories key word in the template. All the fields in the repositories section are being validated and then if the parser encounters with import keyword then it is directed to the import file where the fields are validated and loaded If in imports if repository is present then again it is directed to the repository file and it is validated and after on successful validation output is produced.

IV. Repositories In Service Template

Figure 4 shows the service template with Repositories.

```
~/14071D5816/tosca-parser
tosca_definitions_version: tosca_simple_yaml_1_0

description: TOSCA simple profile with repositories validation and imports.

repositories:
  repo_code0: https://raw.githubusercontent.com/nandinivemula/intern
  repo_code1:
    description: My project's code Repository in github usercontent.
    url: https://raw.githubusercontent.com/nandinivemula/intern/master
    credential: #type: Credential
      token_type: basic_auth
      token: myusername:mypassword

  repo_code2:
    description: My Project's code Repository in github.
    url: https://github.com/nandinivemula/intern/master
    credential: #type: Credential
      token_type: basic_auth
      token: myusername:mypassword

imports:
  - sample_import:
      file: tosca_repository_import.yaml
      repository: repo_code1
      namespace_uri: https://github.com/nandinivemula/intern
      namespace_prefix: intern
```

Figure 4: Template with Repositories

In figure 4 service or Tosca template with Repositories section along with imports is included.

Figure 5 shows the Output of the Repositories in TOSCA-Parser.

```
: ~/14071D5816/tosca-parser
warning: no previously-included files found matching '.gitreview'
warning: no previously-included files matching '*.pyc' found anywhere in distribution
writing manifest file 'tosca_parser.egg-info/SOURCES.txt'
copying toscaparser/tests/data/tosca_repositories_test_definition.yaml -> build/lib.linux-x86_6
4-2.7/toscaparser/tests/data
running install_lib
copying build/lib.linux-x86_64-2.7/toscaparser/repositories.py -> /usr/local/lib/python2.7/dist
-packages/toscaparser
copying build/lib.linux-x86_64-2.7/toscaparser/tests/data/tosca_repositories_test_definition.ya
ml -> /usr/local/lib/python2.7/dist-packages/toscaparser/tests/data
byte-compiling /usr/local/lib/python2.7/dist-packages/toscaparser/repositories.py to repositori
es.pyc
running install_egg_info
removing '/usr/local/lib/python2.7/dist-packages/tosca_parser-0.6.1.dev6-py2.7.egg-info' (and e
verything under it)
Copying tosca_parser.egg-info to /usr/local/lib/python2.7/dist-packages/tosca_parser-0.6.1.dev6
-py2.7.egg-info
running install_scripts
Installing tosca-parser script to /usr/local/bin
             :~/14071D5816/tosca-parser$ tosca-parser --template-file="toscaparser/tests/data
/tosca_repositories_test_definition.yaml"

version: tosca_simple_yaml_1_0

description: TOSCA simple profile with repositories validation and imports.
             :~/14071D5816/tosca-parser$
```

Figure 5: Output of Repositories

V. Need For Repositories

- Using repositories section in the service template we can import the various templates that are defined already and stored in repository.
- Import section of the service Template use the repositories feature and imports the templates that exists in repository.
- Artifacts also use the repositories feature in their definition.

VI. Conclusion

In this paper, Repositories feature in TOSCA-Parser is implemented and the repositories are included in the TOSCA-Template along with import sections included. On Successful validation of the Template output is produced. Repositories can also be referenced with artifacts.

Acknowledgment

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. It gives me immense pleasure to acknowledge all my team members who have really share their best experiences with me while writing this paper.

I would like to sincerely thank all those who have read this paper and provided me with their valuable inputs and suggestions. I want to thank them for all their support, interest and valuable hints.

References

[1] https://www.oasis-open.org/committees/download.php/56826/OpenStack%202015%20Tokyo%20Summit%20-%20TOSCA-and-**HEAT**-Translator-TechTalk.pdf

[2] https://www.oasis-open.org/committees/download.php/56126/TOSCA%20Technical%20Marketing%20Slides%20OSCON%202015%20SpeakerScipt-V0_3-20150719.docx

[3] http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csprd01/TOSCA-Simple-Profile-YAML-v1.0-csprd01.html