Cloud Computing – Lecture 13

Cloud service Deployment Models

11 May 2021

Chinmaya Dehury

Chinmaya.Dehury@ut.ee
Outlines

• Deployment models in cloud
• Service deployment in cloud
• Service deployment standard - TOSCA
• Graphical modeling tools
  • Cloudify
  • RADON GMT – Eclipse Winery
Cloud deployment models

1. Private Cloud
2. Public Cloud
3. Hybrid Cloud
4. Managed Private Cloud
5. Hosted Private Cloud
6. Community Cloud Services
Cloud deployment models

1. Private Cloud
2. Public Cloud
3. Hybrid Cloud
4. Managed Private Cloud
5. Hosted Private Cloud
6. Community Cloud Services
How about the service deployment? How would you deploy your own service, e.g. a messaging platform, your own website, etc.
TOSCA
TOSCA standard overview

• Topology and Orchestration Specification for Cloud Applications (TOSCA)
  • TOSCA is developed under the OASIS Open Consortium.
  • Members include Cisco, Fujitsu, Hewlett Packard Enterprise (HPE), Huawei, IBM, NEC Corporation, NetApp, NetCracker, U.S. NIST, Red Hat, SAP and many more.

• Enhance the portability and operational management of cloud and other types of applications and services across their entire lifecycle.

• User will be able to provide
  • Description of interoperability of the application and infrastructure cloud services
  • Relationships between parts of the services
  • operational behavior of the services
    • Deploy, patch, shutdown
TOSCA standard overview (contd...)

• Facilitate higher level of Solution portability:
  • Portable deployment
  • Simplify migration
  • Dynamic, flexible scaling

• Model-driven creation of cloud services

• Standardize deployment

• Process-driven service lifecycle management

• Latest version:
  • TOSCA Simple Profile in YAML Version 1.3: https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.html
TOSCA standard overview (contd...)

Some imp links

• Latest version - TOSCA Simple Profile in YAML Version 1.3:
  • [https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.html](https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.html)
  • [https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.pdf](https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.pdf)

• TOSCA Implementations

• TOSCA Implementation Stories
  • [https://www.oasis-open.org/tosca-implementation-stories/](https://www.oasis-open.org/tosca-implementation-stories/)

• OASIS TOSCA YouTube Video Playlist
  • [https://www.youtube.com/c/Oasis-openOrg](https://www.youtube.com/c/Oasis-openOrg)
Let's start with an example

A VM in private OpenStack Cloud

Install Apache Nifi atop openstack VM
TOSCA modelling

Let’s start with an example

```
tosca_definitions_version: tosca_simple_yaml_1_3
metadata:
target_namespace: "radon.blueprints.examples"
node_templates:
  OpenStack_0:
type: radon.nodes.OpenStack
metadata:
  x: "523"
y: "828"
display_name: "Cloud User"
properties:
  flavor: "m1.medium"
  key_name: "chinalabs"
  image: "13a94b11f001"
  ssh_username: "root"
  name: "NifiHost"
  network: "provider"

install:
  - node:
      name: Nifi
    type: radon.nodes.Nifi
      properties:
        nifi_version: "1.12.1"
    relationships:
      - type: host
        target:
          type: OpenStack
```

A VM
OpenStack Cloud
Install Apache Nifi atop openstack VM
TOSCA modelling

GOAL:

• Cross cloud, cross tools, orchestration of application on the cloud

Basic blocks:

• Topology
• State
  • Nodes
  • relationship
TOSCA modelling

GOAL:
• Cross cloud, cross tools, orchestration of application on the cloud

Basic blocks:
• Topology
• State
  • Nodes
  • Relationship
• Composition
• Requirements – Capabilities
• Lifecycle
• Policy
TOSCA modelling – Service Template

• Templates define the structure of a cloud service.
• Usually described in YAML or in XML format
• Structure of service template
  • Node Type
  • Relationship type
  • Node template
  • Relationship template
TOSCA modelling – Service Template

Service Template Example

tosca_definitions_version: tosca_simple_yaml_1_3

metadata:
  targetNamespace: "radon.blueprints.examples"
  name: "DataPipelineExample"

imports:
  - file: radonnodesVM__OpenStack.tosca

namespace_uri: radon.nodes.VM
namespace_prefix: radonnodesVM

.....

topology_template:
  node_templates:
    OpenStack_0:
      type: radon.nodes.VM.OpenStack
      properties:
        flavor: "m1.medium"
        key_name: "chinmaya"
        image: "13a94b11-98e-ad29-00ae97e8f790"
        ssh_username: "centos"
        name: "NifiHost"
        network: "provider_64_net"
    Nifi_0:
      type: radon.nodes.nifi.Nifi
      properties:
        port: 8080
        component_version: "1.12.1"

requirements:
  - host:
      node: OpenStack_0
      relationship: con_HostedOn_2
      capability: host

relationship_templates:
  con_HostedOn_2:
    type: tosca.relationships.HostedOn
TOSCA modelling – Node Type

• Describe the type of a resource, component. E.g. type of cloud, type of software etc.

• This is similar to declaring a class in different Object Oriented Programming languages.

• Describe the lifecycle of the node type.

```plaintext
tosca.interfaces.node.Lifecycle:
  create:
    description: Basic lifecycle create operation.
  configure:
    description: Basic lifecycle configure operation.
  start:
    description: Basic lifecycle start operation.
  stop:
    description: Basic lifecycle stop operation.
  delete:
    description: Basic lifecycle delete operation.
```
TOSCA modelling – Node Type

- Describe Properties, attributes, capabilities, requirements of a node

```
tosca_definitions_version: tosca_simple_yaml_1_3
node_types:
  radon.nodes.VM.OpenStack:
    derived_from: tosca.nodes.Compute
    attributes:
      public_address:
    properties:
      flavor:
        type: string
        description: OpenStack flavor id (flavor names are not accepted)
      key_name: ..... 
    interfaces:
      Standard:
        type: tosca.interfaces.node.lifecycle.Standard
        operations:
          create:
            inputs:
              key_name: ..... 
              image: ..... 
            implementation:
              primary: /nodetypes/radon.nodes.VM/OpenStack/files/create/create.yml
          delete: ..... 
        artifacts:
          create:
            type: radon.artifacts.Ansible
            file: /nodetypes/radon.nodes.VM/OpenStack/files/create/create.yml
          delete: ..... 
```
TOSCA modelling – Node template

- Define topology with
  - Nodes *(node_templates)*
  - Relationship *(relationship_templates)*

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
metadata:
  targetNamespace: "radon.blueprints.examples"
  name: "DataPipelineExample"
imports: ==
topology_template:
  node_templates:
    ConsGCSBucket_0: ==
    PubGCS_0: ==
    ConsS3Bucket_0: ==
    PubsAzureBlob_0: ==
    OpenStack_0: ==
    Nifi_0: ==
  relationship_templates: ==
```

Cloud Computing - Lec 13: Cloud Service Deployment models
TOSCA modelling – Node template

• Node template
  • To describe components in the topology structure.
  • Set of nodes (resources/components)
• Concrete Node:
  • Have deployment and implementation artifacts
• Abstract Node:
  • Describe only node type, and mandatory capabilities and properties
TOSCA modelling - Node template

- Node template - Example

```
tosca_definitions_version: tosca_simple_yaml_1_3
metadata:
  targetNamespace: "radon.blueprints.examples"
  name: "DataPipelineExample"
imports:

topology_template:
  node_templates:
    ConsGCSBucket_0:
    PubGCS_0:
    ConsS3Bucket_0:
    PubsAzureBlob_0:
    OpenStack_0:
    Nifi_0:
```
TOSCA modelling - Relationship template

• Relationship template
  • To describe connections, dependencies, deployment ordering

• Requirements and capabilities are implicit way to describe the relationship

• Some basic explicit relationship types:
  • DependsOn
  • ConnectsTo
  • AttachesTo
  • RoutesTo
  • HostedOn
TOSCA modelling - Relationship template

Relationship types example:
TOSCA modelling - Relationship template

Relationship types example:
TOSCA modelling - Relationship template

• Relationship template
  • To describe connections, dependencies, deployment ordering

• Requirements and capabilities are implicit way to describe the relationship

• Some basic explicit relationship types:
  • DependsOn: represents a general dependency relationship between two nodes.
  • ConnectsTo: represents a network connection relationship between two nodes.
  • AttachesTo: represents an attachment relationship between two nodes. For example, attaching a storage node to a Compute node.
  • RoutesTo: represents an intentional network routing between two Endpoints in different networks.
  • HostedOn: represents a hosting relationship between two nodes.
TOSCA modelling - Interfaces

• Interfaces are *reusable entities*
• Define a *set of operations*
• Can be included as part of a *Node type* or *Relationship Type* definition.

tosca.interfaces.node.lifecycle.Standard:
  derived_from: tosca.interfaces.Root
create:
  description: Standard lifecycle create operation.
configure:
  description: Standard lifecycle configure operation.
start:
  description: Standard lifecycle start operation.
stop:
  description: Standard lifecycle stop operation.
delete:
  description: Standard lifecycle delete operation.
TOSCA modelling – Interfaces operations

tosca_definitions_version: tosca_simple_yaml_1_3
imports:
node_types:
  radon.nodes.nifi.Nifi:
    derived_from: tosca.nodes.SoftwareComponent
    metadata:
    properties:
    capabilities:
    interfaces:
      Standard:
        type: tosca.interfaces.node.lifecycle.Standard
        operations:
          stop:
            implementation:
            primary: /nodetypes/radon.nodes.nifi/Nifi/files/stop/stop.yml
          start:
            implementation:
            primary: /nodetypes/radon.nodes.nifi/Nifi/files/start/start.yml
          create:
            inputs:
            tarball_version:
              type: string
              required: true
              default: { get_property: [ SELF, component_version ] }
            implementation:
            primary: /nodetypes/radon.nodes.nifi/Nifi/files/create/create.yml
          configure:
            inputs:
            implementation:
            primary: /nodetypes/radon.nodes.nifi/Nifi/files/configure/configure.yml
          delete:
            inputs:
            tarball_version: 
            implementation:
            primary: /nodetypes/radon.nodes.nifi/Nifi/files/delete/delete.yml
TOSCA modelling - Interfaces

• Relationships have interfaces as well

• configure : the basic interface
  • pre_configure_source
  • pre_configure_target
  • post_configure_source
  • post_configure_target
  • add_source, add_target
  • remove_source, remove_target
TOSCA modelling – Requirements & Capabilities

tosca_definitions_version: tosca_simple_yaml_1_3

node_types:
  radon.nodes.datapipeline.MidwayPB:
    derived_from: radon.nodes.datapipeline.PipelineBlock
    requirements:
      - ConnectToPipeline:
        capability: radon.capabilities.datapipeline.ConnectToPipeline
        node: radon.nodes.datapipeline.PipelineBlock
        relationship: radon.relationships.datapipeline.ConnectNifiLocal
        occurrences: [ 1, UNBOUNDED ]
      - host:
        capability: tosca.capabilities.Container
        node: radon.nodes.nifi.Nifi
        relationship: tosca.relationships.HostedByNode
        occurrences: [ 1, 1 ]

capabilities:
  ConnectToPipelineRemote:
    occurrences: [ 1, UNBOUNDED ]
    valid_source_types: [ radon.capabilities.datapipeline.ConnectNifiLocal ]
    type: radon.capabilities.datapipeline.ConnectNifiLocal
  ConnectToPipeline:
    occurrences: [ 1, UNBOUNDED ]
    valid_source_types: [ radon.capabilities.datapipeline.ConnectNifiLocal ]
    type: radon.capabilities.datapipeline.ConnectNifiLocal
TOSCA Implementation

• Cloudify - http://getcloudify.org/
• DICER - https://github.com/DICERs/DICER
• Eclipse Winery - https://projects.eclipse.org/projects/soa.winery
• Opera - https://github.com/xlab-si/xopera-opera
• OpenTOSCA - http://www.opentosca.org/
• see full list...
TOSCA implementation

We will discuss following two
• Cloudify
• RADON Graphical Modelling Tool (GMT)
TOSCA implementation - Cloudify
TOSCA implementation - Cloudify

• Open-source cloud orchestration framework.
• written in the Python programming language.
• Model applications and services and automate their entire life cycle.
• Supports configuration management tools like Chef, Puppet, Ansible for the application deployment phase.
• Each application is describe in a YMAL file called blueprint.
• GUI to model the service blueprint.
TOSCA implementation - Cloudify

Node

Node

Hosted_on relationship

Connectes_to relationship
TOSCA implementation - Cloudify

Image source: https://docs.cloudify.co/4.3.0/images/architecture/cloudify_advanced_architecture.png

Cloud Computing - Lec 13: Cloud Service Deployment models
Cloudify Manager primarily is built with open-source components:

- **Nginx**: high-performing Web server. For the Cloudify REST service, file server to host Cloudify-specific resources
- **Gunicorn**: Web server gateway interface HTTP server
- **Flask**: Web framework
- **PostgreSQL**: object-relational database that stores the application’s model, indexing, and logs’ and events’ storage
- **Logstash**: to pull log and event messages from RabbitMQ and index them in PostgreSQL.
Cloudify Manager primarily is built with open-source components (contd...):

- **RabbitMQ**: Queueing deployment tasks, logs and events, Queueing metrics
- **Riemann**: policy-based decision maker (an experimental feature)
- **Celery**: distributed task queue. Cloudify management worker, the deployment-specific agents and the host agents are based on Celery
- **InfluxDB**: time-series database; to store metrics submitted by the application’s hosts
RADON Graphical Modelling Tool (GMT) - Eclipse Winery
The Graphical Modeling Tool (GMT) is developed based on **Eclipse Winery**

- A web-based environment to graphically model TOSCA-based application topologies.
- Support latest version of TOSCA standard
- It includes
  - a component to manage TOSCA types and templates,
  - a Topology Modeler that enables to graphically compose application topologies and specify configuration properties
  - a file-based backend to store, import, and export TOSCA entities.
Graphical Modelling Tool (GMT) - Overview

- Version control of model development
- Development of
  - Node type
  - Relationship type
  - Capability type
  - Service templates
  - Artifacts type
- Export and importing the service template
  - Export in TOSCA exchange format is a Cloud Service Archive (CSAR)
  - CSAR is essentially a ZIP file
- Commit to remote GitHub repository
Node type development
• What you can do while developing a node type
  • Define properties and attributes
  • Define requirements and capabilities
  • Define Interface, operations and their inputs
  • Readme file, documentations, appearance etc
• What you can’t do
  • Prepare the implementation file (e.g. Ansible files, script files)
Eclipse Winery - Component

HTTP REST API

- TOSCA XML Model Importer & Exporter
- XaaS Packager
- Versioning & Difference Calculation
- TOSCA YAML Model Importer & Exporter
- BPMN4TOSCA Management Plan Importer
- CSAR Packager
- Topology Completion
- Accountability
- TOSCA YAML Model to TOSCA XML Model Transformer
- BPMN4TOSCA to BPEL Transformer

Templates, Types, Plans & CSARs Management

- Consistency Check
- Splitting & Matching
- Compliance Checker
- Key-based Policy Template Generator
- Key & Access Control List (ACL) Management
- Implementation Artifact Generator

Winery Backend System Components

- Templates, Types, Plans & CSARs Repository
- Matching Templates Repository
- Compliance Rules Repository

Functionality provided by the OpenTOSCA Container and usable in the Winery if a Container instance is running

Img src: https://winery.readthedocs.io/en/latest/_images/components.png
Eclipse Winery - Component

Four parts:
- the templates, types, plans, and CSARs management,
- the TOSCA topology model editor,
- the BPMN4TOSCA management plan editor, and
- the repository to store templates, types, plans, etc.

![Component Diagram](https://winery.readthedocs.io/en/latest/_images/components.png)
Four parts:

- **The templates, types, plans, and CSARs management**
  - *Provides* functionality to access, store, or delete TOSCA elements in the *Repository*
  - Enables *managing* all TOSCA types, templates, node types, relationship types, policy types, artifact types, artifact templates, and artifacts such as virtual machine images.

Img src: [https://winery.readthedocs.io/en/latest/_images/components.png](https://winery.readthedocs.io/en/latest/_images/components.png)
Eclipse Winery - Component

Four parts:

• **The TOSCA topology model editor,**
  • Enables the creation of service templates as directed graphs.
  • Modeled service templates can be exported in YAML format
Eclipse Winery – Installation & web interface overview

docker run -it -p 8080:8080 \
  -e PUBLIC_HOSTNAME=localhost \
  -e WINERY_FEATURE_RADON=true \
  -e WINERY_REPOSITORY_PROVIDER=yaml \
  -e WINERY_REPOSITORY_URL=https://github.com/radon-h2020/radon-particles \
  opentosca/radon-gmt

Web interface of RADON GMT

Cloud Computing - Lec 13: Cloud Service Deployment models
Eclipse Winery – Installation & web interface overview

Cloud Computing - Lec 13: Cloud Service Deployment models
Eclipse Winery – Installation & web interface overview
Service template Orchestration – Opera*

- Lightweight orchestrator
- Compliant with OASIS TOSCA
- Support TOSCA Simple Profile YAML v1.3
- Red Hat *Ansible Automation* for interface implementation
- Supported by European Union’s Horizon 2020 research and innovation programme.
- [https://github.com/xlab-si/xopera-opera](https://github.com/xlab-si/xopera-opera)

*More will be covered in next Lecture*
Automation Engine – Ansible*

- IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.
- Founded in 2013
- Bought by Red Hat in 2015.
- Similar technology: Chef, Puppet
- Heavily use SSH infrastructure to connect and control other remote servers.

*More will be covered in next Lecture
Automation Engine – Ansible*

Advantage:

• Open-source tool
• **Agentless:** No extra agent is required on the remote machine.
• Flexible
• Quite easy as it follow YAML format.
• User/developer focus only on the task.

*More will be covered in next Lecture*
Service deployment solutions – A Summary

**Standard Specification**
A standard to follow for service deployment and management: TOSCA

**Graphical Modelling Tool**
A tool to model the TOSCA-based cloud services graphically: RADON GMT - Eclipse Winery, Cloudify

**Orchestrator**
A tool to understand the standard: Cloudify, Opera

**Automation Engine**
To implement the necessary service deployment and lifecycle management tasks: Ansible and Puppet
What next ???
Let’s move to lab session...
(Use of Eclipse Winery, TOSCA, Opera)
References

5. https://docs.cloudify.co/4.3.0/about/manager_architecture/components/
Thank you