DevOps – Lecture 07

Application Deployment Modelling

27 Oct 2021

Chinmaya Dehury

Chinmaya.Dehury@ut.ee
OUTLINE

• Application Deployment
• Deployment phase in DevOps
• Deployment Methods
  • Blue-Green
  • Canary
• TOSCA: Application Modelling standard
• Winery: Graphical Modelling Tool for TOSCA
• Opera: TOSCA Orchestrator
A quick recap

Monolith Architecture
Microservice Architecture
OpenAPI
Swagger

VM 1

Sensor list
Remove sensor device
Sensor Activity manager
Add sensor device
Sensor activity service
Appliances list
Sensor data preprocess

SmartBuilding App (SmartBuilding.py)

Data
- Data visualization
- Data preprocessing
- Data analysis

SmartBuilding App

Services 1
Data storage

Services 2
Data visualization

Services 3
Data preprocessing

Data analysis

LTAT.06.015 - Application Deployment Modelling
What is Application Deployment

Process of *installing, configuring, accessing* using a URL

Deployment Activities:
• Installation / Uninstallation
• Configuration
• Activation/deactivation
• Update
Deployment phase in DevOps

You basically deploy the software product on a production environment

• Product is ready for
  • different deployment contexts
  • different deployment environments
  • different deployment platforms

• If the release process for deployment to the production environment is also automated, it is referred to as continuous deployment.

• In a continuous deployment process, every validated change is automatically released to users.
Continuous Deployment – Benefits

• Eliminate manual processes for Continuous Delivery
• Improve the focus on the product.
• Automate the repetitive tasks and focus on actual testing.
• Scale from a single application to an Enterprise IT portfolio.
• Quick response to customer feedback
Points to know, while deploying an application

• How to minimize application downtime, if any.
• How to manage and resolve incidents with minimal impact on users.
• How to address failed deployments in a reliable, effective way.
• How to minimize people and process errors to achieve predictable, repeatable deployments.
Some popular deployment methods

- Blue-Green Deployment
- Canary Deployment
- Atomic Deployment
- Shadow deployment
- Rolling deployment
Blue-Green Deployment Methods

Initial State

During Update

Final State
Canary Deployment Methods

Load Balancer

Kubernetes Cluster

Application A (version 1)

Application A (version 2)

Initial State

During Update

Final State

Load Balancer

Kubernetes Cluster

Application A (version 1)

Application A (version 2)

Load Balancer

Kubernetes Cluster

Application A (version 1)

Application A (version 2)

Load Balancer

Kubernetes Cluster

Application A (version 1)

Application A (version 2)
Canary Deployment Methods

- Deploy the change to a small subset of servers
- Test, or wait until satisfied
- Less impact if fail
- Serves as an early warning indicator
  - with less impact on downtime
Blue-Green vs Canary Deployment Methods

- Resource Requirement
- Rollout time
- Downtime
- Failure impact
- Time-to-market
Modelling the applications before deployment

How would you model your application if you have hundreds of components?
Modelling the applications before deployment

How would you model your application if you have hundreds of components?

- Dependencies
- Lifecycle of each component
- Different types of component
- Workflow
- Abstraction
Modelling the applications before deployment

How would you model your application if you have hundreds of components?

- Dependencies
- Lifecycle of each component
- Different types of component
- Workflow
- Abstraction

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 (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
TOSCA modelling

Let’s start with an example
TOSCA modelling

Let’s start with an example

- **Ngixn Container**
- **Docker Engine**
- **Virtual Machine**

Hosted on

Hosted on

Nodes
TOSCA modelling

Let’s start with an example

Virtual Machine

- Hosted on

Docker Engine

- Hosted on

Nginx Container

- Nodes

Edges

Or Relationship

ETAIS
TOSCA modelling

Let’s start with an example

A graph (Nodes + Edges)
Let’s start with an example
TOSCA representation – Node_template

In TOSCA relationship_template

In TOSCA topology_template

TOSCA representation of existing VM

Assumption: VM (remoteworkstation) is already created
TOSCA representation – Node_template

- **In TOSCA relationship_template**
  - **Virtual Machine** hosted on **Docker Engine**, which is hosted on **Remote Workstation**.

- **TOSCA representation**:
  - **Docker Engine** (component)
  - **Remote Workstation** (node)
  - **ssh_username** is set to "centos".
  - **keyFile** is set to "/home/ubuntu/chinmayadehury".
  - **external ip** is set to "172.17.191.11".

- **Assumption**: VM (Remote Workstation) is already created

- **TOSCA representation** of **existing VM**

- **TOSCA representation** of **Docker Engine software component**
TOSCA representation – Node_template

A graph (Nodes + Edges)

**topology_template**

- **TOSCA representation of existing VM**
- **TOSCA representation of Docker Engine software component**
- **TOSCA representation of Nginx container software component**

**Assumption:** VM (remoteworkstation) is already created
TOSCA representation – Node_template

```
 topology_template:
   node_templates:
     containerNginx_0:
       type: radon.nodes.docker.containerNginx
       metadata:
       properties:
         container_port: "80"
         image_name: "nginx:1.13"
         container_name: "mynginxserver"
         host_port: "8082"
       requirements:
         - host:
           node: DockerEngineUT_0
           relationship: con_HostedOn_1
           capability: host

     RemoteWorkstation_0:
       type: radon.nodes.VM.RemoteWorkstation
       metadata:
       properties:
         ssh_username: "centos"
         KeyFile: "/home/ubuntu/chinmayadehury"
         external_ip: "172.17.191.11"
       requirements:
         - host:
           node: RemoteWorkstation_0
           relationship: con_HostedOn_0
           capability: host

     DockerEngineUT_0:
       type: radon.nodes.docker.DockerEngineUT
       metadata:
       requirements:
         - host:
           node: RemoteWorkstation_0
           relationship: con_HostedOn_0
           capability: host
```
TOSCA representation – relationship_templates

In TOSCA

In topology_template

node_templates:

- type: radon.nodes.docker.containerNginx
  metadata:
  properties:
  requirements:
    - host:

- type: radon.nodes.docker.DockerEngineUT
  metadata:
  requirements:
    - host:

relationship_templates:

- type: tosca.relationships.HostedOn
  from: con_HostedOn_0
  to: DockerEngineUT_0

- type: tosca.relationships.HostedOn
  from: DockerEngineUT_0
  to: RemoteWorkstation_0

- type: tosca.relationships.HostedOn
  from: Nginx_0
  to: DockerEngineUT_0

- type: tosca.relationships.HostedOn
  from: RemoteWorkstation_0
  to: Nginx_0

- type: tosca.relationships.HostedOn
  from: ETLAS
  to: DockerEngineUT_0

- type: tosca.relationships.HostedOn
  from: DockerEngineUT_0
  to: ETLAS
TOSCA representation – Node + relationship template

In TOSCA

- topology_template

- node_templates:
  - nginx_container:
    type: os_container
    properties:
    - image: nginx:latest
  - docker_engine:
    type: os_container
    properties:
    - image: docker:latest
  - etais:
    type: os_container
    properties:
    - image: etais:latest

- relationship_templates:
  - con_HostedOn_0:
    type: tosca.relationships.HostedOn
  - con_HostedOn_1:
    type: tosca.relationships.HostedOn

-Hosted on

Virtual Machine

Docker Engine

Nginx Container
TOSCA representation – Node + relationship template

In TOSCA

Graphical representation of topology_template using Winery

LTAT.06.015 - Application Deployment Modelling
TOSCA – In general
TOSCA modelling - **Basic blocks**

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

**Basic blocks:**
- Topology
- Composition
- Requirements – Capabilities
- State (nodes, relationship)
- Lifecycle
- Policy

Graphical representation of topology_template using **Winery**
TOSCA modelling - Basic blocks

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

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

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

Service Template

```
tosca_definitions_version: tosca_simple_yaml_1_3
metadata:
  target_namespace: "radon.blueprints.examples"
  name: "myFirstToscaApp"
imports:
  - file: radonnodesVM_RemoteWorkstation.tosca
  namespace_uri: radon.nodes.VM
  namespace_prefix: radonnodesVM
  - file: radonnodesdocker_containerNginx.tosca
  namespace_uri: radon.nodes.docker
  namespace_prefix: radonnodesdocker
  - file: radonnodesdocker_DockerEngineUT.tosca
  namespace_uri: radon.nodes.docker
  namespace_prefix: radonnodesdocker

topology_template:
  node_templates:
    containerNginx_0:
      type: radon.nodes.VM.RemoteWorkstation
      metadatas: ■
      properties:
        ssh_username: "centos"
        Keyfile: "/home/ubuntu/chinmayadehury"
        external_ip: "172.17.91.1"
    DockerEngineUT_0:
      type: radon.nodes.docker.DockerEngineUT
      metadatas: ■
      x: "448"
      y: "91"
      display_name: "DockerEngineUT"

requirements:
  - host:
    node: RemoteWorkstation_0
    relationship: con_HostedOn_0
    capability: host

relationship_templates:
  con_HostedOn_0:
    type: tosca.relationships.HostedOn
    con_HostedOn_1:
    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.
TOSCA modelling – Node Type

- Describe Properties, attributes, capabilities, requirements of a node
TOSCA modelling – Node template

• Define topology with
  • Nodes (node_templates)
  • Relationship (relationship_templates)

• 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 - Relationships

Relationships

Relationship types

defaultNginxServer_chinm...
(containerNginx)

HostedOn

DockerEngineUT
(DockerEngineUT)

HostedOn

myWorkerVM_chinmaya
(RemoteWorkstation)
TOSCA Relationship types

**AttachesTo:**
- represents an attachment relationship between two nodes.
- *For example,*
  - attaching a storage node to a Compute node
TOSCA Relationship types

**Relationship types**
- **HostedOn**
  - represents a hosting relationship between two nodes.
- **ConnectsTo**
  - represents a network connection relationship between two nodes.
- **DependsOn**
  - represents a general dependency relationship between two nodes.
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 Example

node_templates:

my_block_storage:
type: BlockStorage
properties:
  size: 10
my_web_app_tier_1:
derived_from: Compute
requirements:
  - local_storage:
    node: my_block_storage
    relationship: storage_attachesto_1
my_web_app_tier_2:
derived_from: Compute
requirements:
  - local_storage:
    node: my_block_storage
    relationship: storage_attachesto_2

relationship_templates:

storage_attachesto_1:
type: MyAttachesTo
properties:
  location: /my_data_location

storage_attachesto_2:
type: MyAttachesTo
properties:
  location: /some_other_data_location

relationship_types:

MyAttachesTo:
derived_from: AttachesTo
interfaces:
  some_interface_name:
  some_operation:
    implementation: default_script.sh
TOSCA modelling - Relationship template

- 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 – Properties & Attributes

- **Properties:**
  - *What you provide during modelling*

- **Attribute:**
  - *Runtime variable*

- **Example:**
  - *EC2 node type*
    - *To create an EC2 instance*

Example:

```yaml
node_types:
  radon.nodes.VM.EC2:
    derived_from: tosca.nodes.Compute
    metadata:
      attributes:
      - public_ip:
        type: string
      - id:
        type: string
      properties:
      - image:
        type: string
        default: "ami-00890f614e85c6866"
      - ssh_key_name:
        type: string
      - vpc_subnet_id:
        type: string
      - instance_type:
        type: string
        default: "t2.micro"
      - ssh_key_file:
        type: string
      - ssh_user:
        type: string
```
TOSCA modelling – Requirements and Capability

Required a host

HostedOn

Capability to host

Properties

Requirements

Capabilities

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Relationship</th>
<th>Possible Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependency</td>
<td>@osca.relation...</td>
<td>DependsOn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalable</td>
<td>Scalable</td>
<td>Edit</td>
</tr>
<tr>
<td>endpoint</td>
<td>Admin</td>
<td>Edit</td>
</tr>
<tr>
<td>os</td>
<td>OperatingS...</td>
<td>Edit</td>
</tr>
<tr>
<td>host</td>
<td>Compute</td>
<td>Edit</td>
</tr>
<tr>
<td>binding</td>
<td>Bindable</td>
<td>Edit</td>
</tr>
<tr>
<td>feature</td>
<td>Node</td>
<td>Edit</td>
</tr>
</tbody>
</table>

myWorkerVM.chinmaya
(RemoteWorkstation)
TOSCA modelling – Requirements and Capability

• Typically, **Requirements** are described against a known **Capability Type**

• **Capabilities Types**
  • `tosca.capabilities.Compute`
  • `tosca.capabilities.Network`
  • `tosca.capabilities.Storage`
  • `tosca.capabilities.Container`
  • ...
  • ...
TOSCA Interface types

• Interfaces are **reusable** entities that define a set of **operation**
• can be included as part of a **Node type** or **Relationship Type** definition
• An **operations** may have code or scripts associated with them that orchestrators can execute
• **Possible Operations**
  • Create
  • Configure
  • Start
  • Stop
  • Delete
How would you model graphically?
Winery : Graphical Modelling Tool for TOSCA
Winery – Graphical Modelling tool

Winery – Graphical Modelling tool for TOSCA-based application
Winery – Topology Modeler

Winery – Give a set of

Functionalities

List of node types

Canvas area to model your application/service
Winery – Topology Modeler – node types

• List of node types available in Winery
• Below are some examples
• Note: All these node types are developer under RADON project (https://radon-h2020.eu/)

List of node types under `radon.nodes.VM`

- EC2
- OpenStack
- RemoteWorkstation
- waltclmrOS

List of node types under `radon.nodes.docker`

- DockerApplication
- DockerEngine
- DockerEngineUT
- container
- containerNginx
Winery – Node details - Properties

LTAT.06.015 - Application Deployment Modelling
CSAR : Packaging the service template

• TOSCA YAML CSAR file is an archive file using the ZIP file format
• The structure of CSAR complies with the TOSCA Simple Profile Specification.

CSAR: Packaging the service template

CSAR : Packaging the service template

• Example: *myfirstToscaApp.csar*

• csar directory structure:
CSAR : Packaging the service template

- Example: `myfirstToscaApp.csar`
- csar directory structure:
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/radon-h2020/xopera-opera
• OpenTOSCA - http://www.opentosca.org/
• see full list...
TOSCA Implementation

• Cloudify - http://getcloudify.org/
• DICER - https://github.com/DICERs/DICER
• Eclipse Winery - https://projects.eclipse.org/projects/soa.winery
• xOpera - https://github.com/radon-h2020/xopera-opera
• OpenTOSCA - http://www.opentosca.org/

• see full list...
xOpera orchestrator
TOSCA implementation - xOpera

• 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 program “RADON”.
• [https://github.com/radon-h2020/xopera-opera](https://github.com/radon-h2020/xopera-opera)
TOSCA implementation - xOpera

Ansible Automation

• Recall Automation Lecture;
  • [https://courses.cs.ut.ee/2021/DevOps/fall/uploads/Main/Lec4-DevOpsCourse-21Fall.pdf](https://courses.cs.ut.ee/2021/DevOps/fall/uploads/Main/Lec4-DevOpsCourse-21Fall.pdf)

• 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.
End-to-End Service Modelling and Orchestration

Plan your Application

Model the service using Winery

myfirstToscaApp.csar

xOpera orchestrator

Automation using Ansible

Implementation using
- Python
- Sh
- Java
- ...

export csar

myfirstToscaApp.csar

myfirstToscaApp
Service deployment solutions – A Summary

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

- **Graphical modelling tool**
  - **Winery**
    - A tool to understand the standard: **Opera**, Cloudify

- **Orchestrator**
  - To implement the necessary service deployment and lifecycle management tasks: **Ansible**
Lab Sessions

- Winery
- xOpera
- Implement below application
References

1. https://www.oasis-open.org/
7. https://www.slideshare.net/melsatar/cloud-deployments-models
9. https://www.slideshare.net/knoldus/introduction-to-ansible-81369741
Any Question?

THANK YOU