RADON
DevOps Framework

Pelle Jakovits

RadonFramework.eu
Agenda

- Application deployment and delivery challenges
- Introduction to RADON project
- Infrastructure as Code (IaC) and TOSCA
- Framework tools
- Data pipelines as Code
Application Development and Delivery

- Cloud provides ability to programmatically control infrastructure resources and platforms
- **Resource programmability**
  - Creates a looser boundary between development and operations (DevOps)
  - Accelerates the delivery of changes to the production environment
  - Agile delivery tools and model-based orchestration languages (e.g., Terraform, OASIS TOSCA) are increasingly adopted in Cloud applications
  - Help automating lifecycle management, including continuous delivery and continuous integration, application and platform configuration, and testing
- **Platform programmability**
  - Separation of concerns has helped in tackling the complexity of software development and runtime management
  - Application evolution is still a major challenge
Application Development and Delivery Challenges

• How to continuously monitor and iteratively evolve the design and quality of Cloud applications within the continuous delivery pipelines?

• To extend existing software development and delivery methodologies with reusable abstractions
  • For designing, orchestrating and managing IoT, Fog/Edge computing, Big Data, and serverless computing technologies and platforms

• Infrastructure-as-code (IaC) is expected to grow further

• Novel architectures and patterns to tackle cloud application decomposition
  • With implications in security, performance, reliability and operations costs
  • Individual functions are easier to protect and verify than monoliths vs. greater attack surface with FaaS-based architectures
    • The benefits of function-level auto-scaling vs increased network traffic and latency
    • FaaS cheaper to use per function invocation but can incur higher network charges
Rational Decomposition and Orchestration for Serverless Computing
2019-2021 Horizon 2020 EU project (4 mil eur)

Value proposition:
Offer a DevOps framework to help the EU software industry adopting serverless FaaS without vendor lock-in

https://radon-h2020.eu/
https://github.com/radon-h2020
@RADON_2020
From Monoliths to Serverless

<table>
<thead>
<tr>
<th>Monolith</th>
<th>SOA</th>
<th>Microservices</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Simplified arch.</td>
<td>+ Separation of concerns</td>
<td>+ Scalability</td>
</tr>
<tr>
<td>+ Less to deploy</td>
<td>+ Specular to business</td>
<td>+ Cost</td>
</tr>
<tr>
<td>+ Less to manage</td>
<td>+ Pre-cloud</td>
<td>+ Zero admin</td>
</tr>
<tr>
<td>- Inflexible</td>
<td>- No infrastructure focus</td>
<td>- Resource limits</td>
</tr>
<tr>
<td>- Slow updates</td>
<td></td>
<td>- Size limit</td>
</tr>
</tbody>
</table>

SOA (container based): + Container-based + Easy to migrate + Reproducible + Vendor-agnostic + Manual admin + Running costs

Serverless (platform): - Scalability - Cost - Zero admin - Resource limits - Size limit - Vendor lock-in
RADON framework

• A DevOps framework to help developing FaaS-based products without code & data lock-in

• Goals:
  • Avoid data and code lock-in with standard models
  • Quickly learn how to develop serverless - avoiding early-stage adoption issues
  • A range of quality-assurance tools
  • Customizable serverless templates for increased portability
  • Easily re-engineer monoliths for serverless
Benefits for the end user

A DevOps framework to help developing FaaS-based products without code & data lock-in

- Embrace a technology shift
  - Reduce the learning curve

- Low cost and low risk
  - Multi-cloud deployments avoiding code & data lock-in

- Embrace portability across business use cases
  - Model-based approach
Infrastructure as Code (IaC)
The **DevOps** principle for **automated configuration management of system infrastructure** to address issues regarding the manual configurations of infrastructural resources **based on practices from software development** through the definition of **machine-readable files**, aka **blueprints**.
TOSCA

- Topology and Orchestration Specification for Cloud Applications (TOSCA)
- Standard for automating the deployment and management of cloud applications
- Started by major companies, such as IBM, CA, Rackspace, RedHat, Huawei
- Major goals:
  - Automation of Deployment and Management
  - Portability
  - Interoperability
  - Vendor-neutral ecosystem
For more info on TOSCA, check previous webinars in the RADON H2020 Youtube channel.
Advantages of TOSCA

- Standard for describing infrastructure resources, platform configuration and deployed software

- Alternative solutions:
  - Platform agnostic:
    - Ansible, Terraform, Chef, Puppet, etc.
  - Platform specific:
    - AWS CloudFormation, OpenStack Helm charts

- TOSCA can be used in combination with other solutions
  - RADON uses Ansible to implement life cycle and configuration operations

- TOSCA templates & models contain everything an orchestrator needs to orchestrate systems
RADON IDE
Integrated Development Environment
RADON IDE

• Provide a web-based IDE, with all RADON tools integrated
  • Kubernetes
  • Eclipse CHE

• A shared space where different teams can access and develop software and deployment models

• Dev file contains everything needed to deploy RADON IDE and tools
  • No installation, other than having Kubernetes and CHE available
  • All components deployed either as an external SaaS app or inside CHE as a Kubernetes pod

• Fully compatible with Git and existing CI/CD tools (e.g. Jenkins)
• **RADON devfile** defining:

  • A “radon-particles” project which clones in the RADON workspace the TOSCA modeling entities

  • The set of Che plugins and **Kubernetes** components developed to integrate the RADON tools
RADON IDE deployments

Kubernetes Pods

Plugins

RADON Workspace
Keycloak is the identity and access management tool used:

- To create and manage user’s accounts
- To secure the interaction with the xOpera SaaS Orchestrator and Monitoring Tool
- To store secrets, keys and configurations
GMT

- GUI for creating, developing, and modeling TOSCA models and service templates
- Based on Eclipse Winery TOSCA editor
- It supports:
  1. Managing TOSCA types and templates,
  2. Topology Modeler for graphically composing application topologies and configuration properties
  3. File-based backend to store, import, and export TOSCA entities
RADON GMT
Continuous Testing Tool (CTT)
Continuous Testing Tool (CTT)

- Automate continuously assessing functional and non-functional properties
- Define test cases inside the TOSCA model as testing policies
  - Which tests should be executed
    - E.g.: Load tests, smoke tests, etc.
  - How to automatically set up test environment
    - Separate TOSCA models for Software under Test (SUT) and Testing Infrastructure (TI)
- Service for automatically executing tests
  - How to generate required data
  - How to monitor and generate test reports

```yaml
policies:
  - exampleTestPolicy:
      type: radon.policies.testing.HttpEndpointTest
      properties:
        path: "/
        hostname: "google.com"
        method: "GET"
        port: "443"
        expected_status: "200"
        ti_blueprint: "radon.blueprints.testing.DeploymentTestAgentEC2"
        use_https: "true"
        test_id: "exampleTest"
        targets: [ AwsLambdaFunction_0 ]
```
Orchestrator (xOpera)
• TOSCA orchestrator
• Deploys the infrastructure resources and software components described in the TOSCA model
• Depends on Ansible
• CLI and SaaS versions available
• SaaS version supports receiving monitoring trigger events to enable auto-scaling
• SaaS version supports user groups, Secrets management and KeyCloak authentication
RADON orchestrator: xOpera

- SaaS version
- CLI version
Project management: deployment1

Service template: definitions/radonblueprints__ServerlessToDoListAPI.tosca

Enter the service template filename

Inputs (YAML): Drag a file here or browse to upload

Actions:
- Deploy
- Validate service template
- Validate CSAR
- Undeploy project
- Get debug package

Diff & update (alpha):
- Deploy project
- Clean state and deploy

Project health:
- Container: Container healthy
- Connectivity: Connectivity healthy

Project status

Validation result
Defect Prediction
DEFUSE

The RADON tool or IaC Defect Prediction

**Step 1** Connect a GitHub or GitLab repository

**Step 2** Mine failure-prone data

DEFUSE uses **repository-miner** to automatically collect failure data from IaC blueprints, label them as failure-prone or neutral, and calculate metrics to be used as ground truth for defect prediction models.

False-positive data can be checked as so by the user to improve model training.

To extract metrics, DEFUSE uses **ansiblemetrics** or **tosca-metrics** depending on the language selected by the user, that is Ansible or TOSCA.
The Defect Prediction tool supports:

- **Automatic data mining of repositories** to seek defect-prone and defect-free IaC blueprints for continuous re-training;
- **DataOps auto re-training** with feedback provided by tool users (developers and operators) for example by correcting a prediction from defect-free to defect-prone;
- **Performance improvements** will trigger a model re-deployment;
Other Tools
RADON Particles

• Set of reusable and composable TOSCA models
• IaaS life-cycle operations (install, start, configure, stop, delete) implemented by Ansible playbooks
• Can be used by any TOSCA orchestrator

Available templates:
• **Cloud resources**: AWS EC2, OpenStack, Google Cloud, Azure Cloud
• **FaaS**: OpenFaas, AWS Lambda, Azure functions, Google Functions
• **Other Software**: NiFi, Mosquitto MQTT, MySQL, Prometheus, Docker, Kafka, MongoDB, etc.
• **Extensions**: FTP, GridFTP, Kubernetes.
Decomposition Tool

- Optimization problem:
  - **objective**: total operating cost
  - **variables**: memory and concurrency;
  - **constraints**: average response time less than 2.5 sec

- Performance modeling:
  - **benchmarking**: service demand estimation
  - **formalism**: layered queueing networks (LQNs)

- LQN for the thumbnail generation example (simplified)
  - clients (open workload)
  - create_thumbnail (Lambda function)
  - thumbnails (S3 bucket)
  - uploads (S3 bucket)
Express functional and non-functional requirements on a RADON model

- Provides built-in definitions of common runtime issues, such as *deadlocks*, *race conditions*, and *execution loops*, but is also flexible enough to allow users to define their own.

The primary use of the **Verification Tool** (VT) is to verify that a RADON model meets a specification written in the CDL.

- The VT can be used as an aid at design-time, to search for potential issues that could occur at run-time.
Data Pipelines (as Code)
Data Pipelines

- Services for automating the data migration, processing and storage across multi-cloud environments
- Support distributed FaaS applications
  - Where functions are deployed on-premise, HPC or on different cloud providers

- **Goal:** Re-useable data pipeline services that can be composed into more complex data management pipelines
Example data pipeline

Real-time data migration from MQTT to AWS to Google Cloud
Example data pipeline in TOSCA
RADON data pipelines

- Extend TOSCA with support for modelling data migration and management across multi-cloud environments
- Provide a set of reusable node types which can be composed into data pipelines
- Data automation is implemented by:
  - Apache NiFi
  - AWS Data pipeline service
- Ansible is used to install, configure and start the underlying services and cloud resources
Automatic Scaling
Service Model example
Automatic Scaling
Project use cases

• **Ambient assisted living:** Development of a FaaS-based implementation of the SARA solution for health monitoring and (socially interactive) assistance in daily living tasks to the elderly (and their caregivers) at home, in order to prolong autonomy and delay institutionalization of elderly.

• **Managed DevOps:** Implement a Serverless Cloudstash - a DevOps solution providing **automation and management of binaries and artifacts** through the applications delivery process.

• **Travel technology:** Refactor initial monolithic architecture of a **personalized tour planning solution**, amplifying the tourists’ experience when exploring new destinations through continuously evolving and enriching mobile-based applications and solutions.
Conclusions

• While there are many DevOps tools available, continuous application evolution is still a major challenge
• **Avoid data and code lock-in** with standardized and reusable models
• RADON offers
  • Set of reusable and easily composable TOSCA templates
  • Range of **quality-assurance** tools
  • Customizable serverless and data pipeline templates for increased **portability**
  • Tools to simplify **re-engineering monoliths** for serverless
Additional information

• TOSCA open source nodetypes: https://github.com/radon-h2020/radon-particles/
• RADON videos & webinars: https://www.youtube.com/channel/UCgoXX6JZ6bDqTxVBRm4K WNQ
• RADON github: https://github.com/radon-h2020/
• RADON website: https://radonframework.eu/