Microservices validation: Mjolnirr platform case study

South Ural State University, Russia:
Gleb Radchenko (gleb.radchenko@susu.ru);
Dmitry Savchenko

Lappeenranta University of Technology, Finland:
Ossi Taipale
Microservice Architecture

- **Microservice architecture** is a cloud application design pattern that implies that the application is divided into a number of small independent services, each of which is responsible for implementing of a certain feature.

- Microservices can be considered as meta-processes in a Meta operating system (OS):
  - they are independent,
  - they can communicate with each other using messages
  - they can be deployed, suspended or moved to any computational resource in cloud.
Microservice features

We define microservice as a software-oriented entity, which has following features:

- *Isolation* from other microservices as well as from the execution environment based on a virtualized container;
- *Autonomy* – microservices can be deployed, destroyed, moved or duplicated independently;
- *Open and standardized interface* that describes all communication methods (either API or GUI) supported by the microservice;
- *Microservice is fine-grained* – each microservice should handle its own task.
Monolithic Architecture

Microservice Architecture

- Clients
  - Balancer
  - Monolith application
    - Component 1
    - Component 2
    - Component 3
    - Component 4
  - DB

- Clients
  - Balancer
  - API Gateway
    - Service 1
    - Service 2
    - Service 3
    - Service 4
  - Service 3 DB
  - Service 4 DB
Why Microservice Validation?

• We need an **automated infrastructure** that will provide a life-cycle management, naming, addressing and scaling of microservices depending on the current load

• The need for **continuous integration** of developed and/or modified microservices into the existing infrastructure requires **comprehensive testing** of both *individual microservices* and their interoperability in an *ensemble* with other microservices.
Goals of the Research

• **To analyze** the existing methods of software testing and identify features that are specific to microservice architectures

• **To develop a model** of a microservice and microservice system

• **Develop a methodology** for microservice development and testing, including approaches for component and integration testing
Mjolnirr platform
Mjolnirr Microservice Platform Architecture

- **Load Balancer**
- **Microservice Repository**
- **Resource Manager**
- **Microservice**
- **Container Image**
- **Computational Node**

**Relationships:**
- Load Balancer uses Microservice Repository.
- Resource Manager uses Load Balancer.
- Microservice stores Container Image.
- Container Image is deployed to Computational Node.
- Computational Node monitors load.
- Load Balancer deploys Microservice.
Microservice Unit testing

- **Functional Microservice testing** can be divided into two stages: unit, integration and system validation of the microservice source code or/and internal components (internal testing); self-validation of the microservice interface (container self-testing).

- **Load Microservice testing** is aimed at the individual microservice validation at a certain load.

- **Security Microservice testing** is aimed at the security and isolation validation of the individual microservice.
Microservice Integration testing

- **Functional integration testing** ensures interoperability of the individual microservices, including correctness of microservices message exchange.

- **Load integration testing** is aimed at checking the microservice correctness under automatic deployment, as well as microservice orchestration and choreography.

- **Integration security testing** is aimed at validation of the security of communications between microservices, as well as checking for the interception of messages from outside or inside by anyone, besides the message recipient.
Microservice interface definition

To automate the test cases generation, we propose to provide a formal description of a microservice interface

```json
{
    "name": "test_microservice",
    "inputs": [
        {
            "type": "integer",
            "min": 0,
            "max": 10
        },
        {
            "type": "string",
            "syntax": "he.\?lo"
        }
    ],
    "outputs": [
        {
            "type": "integer",
            "min": 0,
            "max": 10
        }
    ],
    "input_connections": ["test_microservice2"],
    "output_connections": ["test_microservice3"]
}
```
To implement a subsystem that provides a declarative approach for load based testing we are developing a testing platform using the BDD (Behavior Driven Development) approach.

**Feature: Connections**

Given there are 10 000 users connected to a system
When every user send a GET request every 1 second
Then user should get a response within 50 ms

**Feature: Login**

Given there are 10 000 users connected to a system
When I connect to a system
Then I should get my home page 50 ms
Continuous Microservice system stability testing

• It is necessary to implement and deploy to the testing and production environment a "Chaos Monkey" subsystem for the microservice platform.

• **Chaos Monkey** is implemented as a distributed system, which introduces targeted microservice malfunctions (disconnection of services, violation of networking, etc.) at the production environment.

• The microservice system should be designed in such a way that the Chaos Monkey is not breaking the functions of the system.
Mjolnirr Platform Validation Implementation

Production Environment: Microservice Environment
- Chaos Monkey

Validation Environment: Microservice Environment
- Load Validation
- Security Validation
- Integration Validation

Developer: PC
- Container Image
- Container Self-testing solution
Results

• We analyzed the concept of microservice cloud architecture and offered an implementation of this approach on the Mjolnirr platform.
• We investigate methods of software testing and testing features of microservice systems
• We have proposed a methodology for the microservices development and testing on the basis of the Mjolnirr platform.
• As a development of this project, we are planning to provide:
  – implementation of proposed test methodology as a set of software components, integrated in the Mjolnirr platform.

• Contact: gleb.radchenko@susu.ru
Test Design and Implementation Process

## Special aspects of microservice validation

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>Integration</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Load</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Security</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

- - no features for microservice validation
+ - has features for microservice validation