Testing-as-a-Service Approach for Cloud Applications

Dmitry Savchenko, Nikita Ashikhmin and Gleb Radchenko
System Programming Department
South Ural State University
Chelyabinsk, Russia
Email: savchenkodi@susu.ru, ashikhmin.na@yandex.ru, gleb.radchenko@susu.ru

Abstract—Web-based systems test techniques developed rapidly over the past years. Continuous integration approach demanded rejection of manual testing methods, and transition to fully automated methods, covering all application from the source code of core services to the user interface and API correctness. Every year there are dozens of new approaches to cloud applications testing, many of which, despite their effectiveness, little known to ordinary developers and testers. We propose a Testing-as-a-Service solution to simplify the cloud applications testing by providing a catalog of applicable test techniques and automating testing processes.

I. INTRODUCTION

It is believed that writing documentation and writing tests - the two things software developers hate. Indeed, high-quality testing of complex applications, such as cloud services, requires competence in testing automation [1], knowledge about the state-of-the-art testing techniques (like Twitter Diffy, Chaos Monkey, Fuzzy Testing, etc.) and skills of applying these techniques. Furthermore, development of the cloud service testing methodology is often become a separate complex project that should be conduct concurrently with the main software development project [2]. For all of these reasons, software developers and project managers often pay insufficient attention to the comprehensive testing, sacrificing the quality for the sake of faster product delivery.

We propose a Testing-as-a-Service solution to simplify the cloud applications testing by providing a catalog of applicable test techniques and automating testing processes. Three main features of the proposed service can be distinguished.

• **Testing methods catalog:** Providing users a catalog of testing methods available for their applications. Based on the features of the application under test (like programming language, development framework, application type, etc.), the testing service offers to a user a set of ready-to-use suitable testing methods.

• **Extensibility:** each testing method is implemented as a separate microservice, easily integrated to the catalog of existing testing methods.

• **Pay-as-you-go:** using the cloud approach to implement the software testing provides to the user the combination of cost flexibility and built-in resource scalability.

II. SYSTEM OVERVIEW

According to the features, defined in the introduction, we decided to implement cloud applications testing service as a web-application. Two actors are interacting with the testing service: tester and test techniques developer. **Tester** uses the service to test a cloud application. A tester can browse a catalog of available test techniques, create test projects, define test objectives, run tests and obtain test results. **Test techniques developer** implements test techniques in the testing service.

The interaction of the actors with the testing service is performed using the following key entities, defining features of the application under test and testing process.

**Application attribute** describes a feature of the cloud application under test. For example, a tester can define programming language, basic application framework or database type as an application attribute value. Each application attribute is associated with a set of test techniques, which provide testing for the applications with such attribute value.

**Test technique** is an atomic testing step. It aims the testing of the one single system aspect (like unit-testing, API interface testing, load testing, etc.).

**Test objective** is a set of test techniques focused on a testing of some aspect of the cloud application (e.g. testing of a specific component, interface, performance or stress testing and so on). While defining the test objective, the tester provides a set of cloud application properties (application attribute values) so the testing service could offer him a list of applicable test techniques. The tester picks and configures test techniques for the test objective, providing such information as source code repository URL, web server URL, test cases files, etc. When all test techniques are configured, the tester can run the test objective. When the test objective execution is finished, the tester can get the test report, which contains detailed information about passed and failed tests.

**Test project** is a set of test objectives that represent a comprehensive cloud application testing solution. While creating a test project, a tester defines a set of static (configured once per project, like source code repository URL) and dynamic (must be set on each run) parameters. Test project parameters are passed to underlying test objectives and test techniques when the test project is initializing.
III. SERVICE ARCHITECTURE

We can highlight the following components in the cloud applications testing service (see fig. 1):

- **Authentication and authorization** component identifies users, their roles, and permissions. It provides authentication through the login-password pair, SSH keys, OAuth accounts and authentication tokens (for third-party API users).
- **Test projects, objectives and techniques managers** provide Create, Read, Update, Delete (CRUD) actions for the Test projects, Test objectives, and Test techniques entities respectively.
- **Mapper** associates application attributes to the corresponding testing techniques.
- **Test technique** is independent service, implementing common REST interface providing test configuration, execution, and results delivery.

Let’s describe the process of test configuration and execution (see fig. 2):

- The tester passes the authentication, creates a test project and a test objective, defining application attributes of the cloud application under test;
- According to provided application attributes, testing service offers a list of testing techniques that can be implemented for the application;
- The tester chooses test techniques he want to implement, defines parameters and sets the execution order of the test techniques;
- To initiate a testing process, the tester provides dynamic parameters of the test, executes the test project and receive the testing results.

IV. IMPLEMENTATION AND FUTURE WORK

Testing service was implemented using Ruby on Rails framework and deployed to Heroku cloud platform for evaluation (available on http://cloudrift.x01d.com). It provides a REST API and a web interface for projects, objectives and test techniques configuration. Three test techniques were implemented - **API testing** using Ruby and Airborne; **load testing** using Ruby, Apache Benchmark, and ABCrunch; and **UI testing** for the Facebook React Framework using Node.JS and Carte Blanche.

Let us describe test objective initialization process. First, the tester defines static parameters for the created objective. In a case of API testing, he provides the cloud application domain. Next, the tester fills the tests list by choosing the “API testing” test technique. For each test technique, the tester provides API method path, HTTP verb, test parameters and excepted outputs as JSON. Testing service sends those parameters to the test technique service, and calls the API according to the provided parameters and matches the output with the expected one.

As a further development, we are going to enhance test techniques catalog, increase the number of covered tests techniques; implement automatic application attributes detection for the popular frameworks and programming languages; implement desktop and mobile applications testing support.

ACKNOWLEDGMENT

The reported study was partially supported by RFBR, research project No. 14-07-00420-a and by Grant of the President of the Russian Federation No. MK-7524.2015.9

REFERENCES
