ElasTest: An Elastic Platform for E2E Testing Complex Distributed Large Software Systems*

Juan Francisco Ribera Laszkowski[1], Andy Edmonds[1], Piyush Harsh[1], Francisco Gortazar[2], and Thomas Michael Bohnert[1]

[1] Zuerich University of Applied Sciences, Switzerland
{ribr,edmo,harh,bohe}@zhaw.ch
[2] University of Rey Juan Carlos, Spain francisco.gortazar@urjc.es

Abstract. As systems get more complex testing has also increased not only in complexity but in the total IT cost, which is estimated to increase even more by 2020. Testing large complex distributed applications is hard, time consuming and lacks tooling. Given that the digitisation of business has proved to be a key aspect for improving the productivity of developers in the delivery of the service to end-users, in this paper we present early results showing how these capabilities can also be provided to testers of software and services, by adopting standard interfaces and leveraging the tools provided by an early research open-source platform, capable of efficiently testing large scale systems, ElasTest.

Keywords: cloud · testing · e2e · large scale · distributed · computing · systems.

1 Introduction

The demand for larger and more interconnected software systems is constantly rising, and recently more and more architectures opt for a microservices-oriented architecture [21], many of which are cloud native applications [20]. This increase has also caused that the development, operations and management, and complexity of such microservice-based systems to increase. However, the skills of software developers and testers must satisfy the rate at which these large systems are appearing [18], especially when, in this case, larger and more complex systems demand themselves more efficient testing processes.

The ElasTest project aims at significantly improving the efficiency and effectiveness of the testing process throughout the software development life cycle and, with it, the overall quality of large software systems. A set of required tests that are modern and accepted, are defined by [17], who also shows that testing a full cloud native application is not a mean or small task. This is how

* This work is partially funded by the Swiss State Secretariat for Education, Research and Innovation (SBFI) in association with the European Union Horizon 2020 research and innovation programme via grant agreement #731535, for the ElasTest project [5].

the ElasTest platform comes to be, with the purpose of not only developing an open-source, modern test orchestration theory and toolbox, to allow the creation of complex test suites as the composition of simple testing units; but also, to create an impact in the community and become a worldwide reference in the field of large-scale software testing, providing sustainability of the project generated results.

In this context of testing and delivery of service, the challenging questions to be asked are:

1. How can a complete and large scale cloud native application be effectively tested and supported by a test platform?
2. How can the investments made in testing be minimised such that the organisation (e.g. SME, corporation) tasked with delivering the application can deliver the same reliability and assurance in the same or reduced time frame?

This paper attempts to tackle these questions and therefore, reports the work-in-progress of an efficient large scale system testing and open source research platform, ElasTest [7], which provides the means to test applications (or systems) "in-the-small" as well as the same systems deployed "in-the-large", through the support of industry standards.

We organize this paper by starting with an overview of the ElasTest Project, followed by the Architecture of the system which describes how services are composed and delivered to the tests that operate against an System under Test (SUT). Then, the current set of services offered by ElasTest are briefly described and we finalize with the Discussion and Future Work.

2 ElasTest Overview

The ElasTest project stands for Elastic Platform for Testing complex distributed large software systems. It is being developed by a consortium of European academic institutions, research centers, large industrial companies and also SMEs. The project began January 2017 and will run for 3 years. Its members are [6]:

- **Universidad Rey Juan Carlos**: is the project coordinator and has key involvements in overall plaform design, delivery of components related to test orchestration and implementing one of the key test support services.
- **Fraunhofer FOKUS**: provides their expertise both in telecommunication core networking systems and Industrial IoT (IIoT). For the telecommunications aspect, they provide a vertical demonstrator.
- **Technische Universitaet Berlin**: provide one of the core systems in ElasTest, specifically the system that provides all virtualised resources used by the platform and executed tests. Also provided is an emulated IIoT test support service. They also provide a vertical demonstrator.
- **Consiglio Nazionale delle Ricerche**: oversee and design the evaluation of the ElasTest platform based on their extensive experience of the theoretical foundations of Software Engineering for Testing.

- **IMDEA Software Institute**: provide two key test support services in ElasTest, namely the Security and Monitoring services.
- **ATOS Spain**: oversee the complete delivery of the cloud platform that ElasTest is built upon. Specifically this includes the components related to platform instrumentation.
- **Zuerich University of Applied Sciences**: deliver components related to the core ElasTest platform including Nexus for service delivery, Sentinel for system and service monitoring and a cost estimation engine.
- **Naevatec**: is a SME that provides a reliable continuous integration and delivery system and also provide one of the vertical demonstrators used to evaluate ElasTest.
- **IBM Ireland**: is the key partner providing the component related to test recommendations. These recommendations are based on the set of end-user provided tests and suggests new tests using machine learning.
- **Relational**: is a SME which delivers one of the test support services, the big data analysis service.

ElasTest is funded by the European Commission under the ICT-10-2016 topic of the Horizon 2020 programme [10]. ElasTest [5] is an open source platform with the objective to reduce the complexity of carrying out end-to-end tests of large scale distributed systems. The Systems under Tests (SUTs) can be applications and services. ElasTest gives developers and testers of systems the means to assess their tests in a way that is cost and performance sensitive. ElasTest's capabilities are built on common open source technologies, and ensures to provide elasticity and compatibility to integrate with multiple technologies.

Moreover, ElasTest supports both a lightweight deployment profile, suitable for laptops to be used for testing in the small, to a complete deployment profile, suitable for testing complete large scale systems in an automated fashion. Finally, ElasTest attempts to ease integration with existing continuous integration and deployment systems and currently has a custom Jenkins plugin.

Currently, the project having delivered its core open source platform is now improving and using it for the base line of research activities. Importantly, that platform will be enhanced not only by input from identified end-users but also by on-going evaluation activities of the platform. This evaluation will be based upon a set of vertical demonstrators. There are four of these demonstrators: an e-commerce demonstrator, a 5G carrier-grade network system implementation, a open online class course demonstrator focusing on real-time communication and an Industrial IoT demonstrator.

2.1 Elastest vs. Other Solutions

Elastest solves a variety of problems, but each individual feature has been solved by many different systems. Even though no single system delivers a solution that provides all the features that Elastest does, there are other options available to solve some of these problems. Testcraft[3] provides a codeless selenium testing

[3] https://www.testcraft.io

framework with artificial intelligence. It provides a set of workflow creation tools for non-programmer to develop tests. Cypress[4] is a web service Javascript testing framework developed from the ground up, contrary to a selenium-based framework, which delivers a fast performing testing framework and a vast amount of information when a failure happens, for the problem to be fixed. Other examples include Robot Framework[5], which is a generic test automation framework written in Python which uses selenium to simplify the testing processs, Appium[6], which provides a testing framework written in NodeJS for iOS and Android, etc.. That said, Elastest is not a replacement for other testing tools. Each developer has their own field or programming language where they feel comfortable. Nevertheless, where Elastest excels is at bringing together all the testing tools together and provide real End-to-End testing, starting from the deployment of the infrastructure of your applications, to deploying the Support Services to emulate other variables, and finally, providing a powerful Log Analyzer and a Recommendation System (among others).

3 ElasTest Architecture

ElasTest manages the full testing life cycle, deployment and monitoring of the SUT, execution of the end-to-end tests, all with the specified support services, and exposure of the results to the end-users. In order to use the ElasTest system, the user must first be granted access. Then, the user communicates the testing requirements through the concept of a "T-Job" (testing-job). A T-Job typically consists of a set of tests to be executed against a system endpoint and a set of estimated resources to execute those tests on. It is important to note that *services* are the ones that can be attached to the T-Jobs, whereas *engines* run irrespectively of the T-Jobs, collecting metrics, analyzing execution patterns, costs, etc.

Service delivery and composition is done through the ElasTest Service Manager (ESM), which manages the delivery (deployment, provisioning and execution, incl. destruction) of the Test Support Services (TSS). These TSSs are reusable services that are typically used by developers for creating their T-Jobs. These are deployed as part of the ElasTest infrastructure and shall autoscale to adapt to the tester needs. These TSS follow a Software as a Service (SaaS) delivery model in the sense that developers do not need to worry about how to deploy, provision or scale them.

Their capabilities are reachable through service-dependent APIs, defined by OpenAPI specification (as per ElasTest architectural principles).

Currently the ESM provides all TSSs through an implementation of the Open Service Broker API specification [13], which is currently seen as an industry standard in delivering functionality as a service. It is used in platforms such as Cloud-

[4] https://www.cypress.io
[5] http://robotframework.org
[6] http://appium.io

Foundry [3], OpenShift [14] and Kubernetes [11]. Currently it is implemented against the 2.12 specification version and has specific ElasTest extensions.

The ElasTest Project currently provides five TSSs through the ESM. The motivations and also objectives of these TSS is as follows:

- Enable testers focus on their systems core functionality
- Simplify the creation of T-Jobs validating individual functions or SiS (i.e., deliver TSS capable of helping in the creation of T-Jobs)
- Increase reusability provide reusable capabilities for creating T-Jobs involving common testing tasks
- Decrease significantly the marginal (i.e. unit) cost of testing
- To be provided on-demand as cloud native services
- Provide access to capabilities/functionality through APIs that might be consumed by the T-Jobs or by the test orchestration engines.

In other words, these ElasTest Test Support Services are meant to act as SaaS services designed for helping developers in the creation of simple tests (called T-Jobs in this proposal) by providing capabilities commonly required for typical testing processes.

Each component within the microservice-based architecture of ElasTest has its own internal peculiar architecture and, for the sake of brevity, the internal architecture of these will not be detailed. Each of the components within the architecture are components that are implemented and developed by the ElasTest consortium. Below, we describe the macro architecture of ElasTest as shown in Fig. 1.

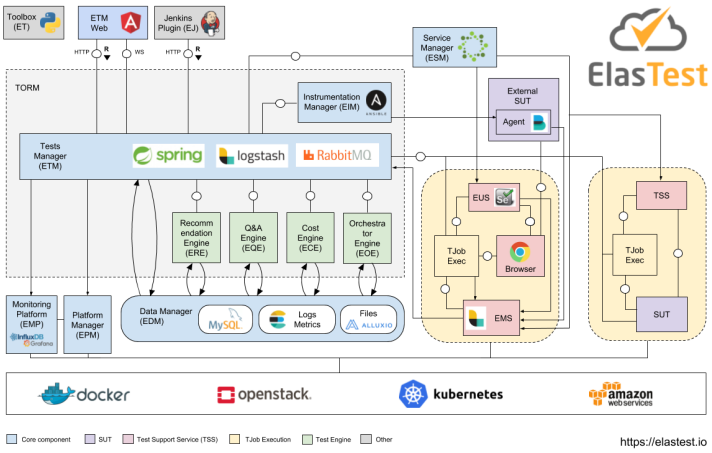

Fig. 1. ElasTest Architecture using the Fundamental Modelling Concept (FMC) notation [9].

- **Test Orchestration and Recommendation Manager (ETM):** provides access to a user through the user interface and/or the programmatic API. It manages all other components within a deployment of ElasTest.
- **ElasTest Service Manager (ESM):** allows the creation of on-demand instances delivered as services, which execute the specified T-Job. This component enables efficient testing and allow for the rapid creation of these tests. The TSS supported by the ESM are:
  - **User impersonation Service (EUS):** This service enables the impersonation of end-users in their tests through GUI (Graphical User Interface) instrumentation and through mechanisms for QoS and QoE evaluation.
  - **Sensor, actuator and device impersonation Service (EDS):** This service is useful for enabling tests to emulate customized device behavior at the time of testing IoT (Internet of Things) applications.
  - **Monitoring Service (EMS):** This service leverages runtime verification ideas (in turn inspired by formal verification) to represent the system behavior as sequences of events that can be monitored in universal ways.
  - **Big Data Analysis Service (EBS):** enables the collection, analysis and visualization of large volumes of logs.
  - **Security Check Service (ESS):** for security vulnerability checking targeting specifically the problems of the main large scale deployed system
- **ElasTest Platform Manager (EPM):** The EPM is responsible for providing and managing the resources on which the various ElasTest components run on. The currently supported cloud infrastructures are: OpenStack [15], Amazon Web Services [2], Docker [4] and Kubernetes [11]. For orchestrating the SUT and the network services within the ElasTest platform, OpenBaton is used [19].
- **ElasTest Instrumentation Manager (EIM):** allows for dynamic modifications of the system under test and to inject real world behaviors over the system (e.g., connection latency or disconnects), while gathers information for the assessment at runtime as well as for later inspection.
- **ElasTest Data Management (EDM):** stores the logs and metrics which are generated during the execution of a T-Job. This component leverages the technologies of MySQL[12], Elasticsearch[8] as search engine, and Alluxio[1] as virtual distributed storage system.

Unlike the services described above, engines can run irrespectively of the T-Jobs, collecting metrics, analyzing execution patterns to report failures, costs, etc., and are described below:

- **ElasTest Cost Engine (ECE)** allows static estimation of test execution costs based on the estimated resource consumption matrix provided by the test developer. This module empowers test authors to optimize their test parameters iteratively and make the tests economical in the long term. ECE also tracks key lifecycle events and correlates them to actual resources consumed by a test execution to compute the true cost of the execution. The

ECE utilizes the plan cost models registered by each support service in ESM in offering cost estimation and actual cost tracking for finished test executions.
- **ElasTest Recommendation Engine (ERE):** recommends new tests to the end-user based on the set of tests they have presented to the the ETM. This component uses machine learning and artificial intelligence to generate a set of additional recommended tests that the end-user can choose to include in an updated version of their T-Job. This just like the use of external services through the ESM and cost estimation through the ECE are facilities that aid rapid system testing with the focus of cost and time-to-market.
- **ElasTest Test Orchestration Engine (EOE):** through the definition an orchestration notation for the graph of T-Jobs, where the edges provide the execution logic and the nodes act as checkpoints, it allows the synchronization of the T-Jobs and automatic verification on its incoming edges. Furthermore, it allows the test augmentation through custom operational conditions and the definition of sub-graphs basing on combinatorial techniques.
- **ElasTest Question&Answer Cognitive Engine (EQE):** enables the reuse of testing knowledge across software projects. Supports a dialog manager to support dynamic interaction between user and system, and generates answers for questions about designing new test cases. A GUI to enable conversation is also supported.

Furthermore, to complement, the **ElasTest Monitoring Platform (EMP)** is a general purpose monitoring framework that accords first class status to metrics and logs. It provides:

- advanced analytics and reporting / alerting functions which is used to monitor the health of service instances managed within the ElasTest Platform and therefore remediation of states based on red flags' alert.
- advanced queries such as *give me list of all hosts ordered by available RAM/CPU/Disk* which can be used by ESM's planner for scheduling.

In unison with the EMP some of the potential planned functionalities include online SLA monitoring, fault tracing and correlated queries over multiple metric streams. Finally, an Anomaly Detection Component is planned to integrate as well into these components to conjunctively allow ElasTest to achieve the smooth and fine grained life-cycle controlled execution of the test environments.

4 Discussion and Future Work

The work within the ElasTest platform has rapidly progressed yet there is still much work to be done. From the research perspective the following questions are seen to be answered by future work:

- Reviewing new stakeholders or evaluating existing competitors in order to provide ElasTest as SaaS;

- Evaluation of not only market conditions, but also the market positioning of ElasTest through a feature comparison with leading competitors;
- Near real-time service delivery and update with minimal wait times in having network-based access to a particular service. This will require the consideration of resource frameworks that far exceed the performance of current container-based technologies such as Docker;
- In order to make such services reliable and measurable suitable observability mechanisms will be needed and this work on observability tooling is being carried out by the EMP;
- With information collected on the running services of ElasTest, the question of how to leverage machine and deep learning arises. Such approaches can be used to detect anomalous behaviors that could indicate runtime errors and prompt remediation actions;
- One of the areas where a need has already been seen from the current five ElasTest TSSs is enhanced support for debugging of services. What is the best means to provide such capabilities: is logging sufficient or such approaches seen in OpenTracing [16] activities be investigated?
- From a business perspective, how can scaling of delivered services by the ESM be supported yet respect financial cost targets set by the end-user. Indeed how can SLAs of delivered services be defined in such a way to be self-validating?

The work in ElasTest, specifically on service delivery provides the impetus for further research in these areas and also provides not only an architecture but an implementation upon which new technologies can be applied against. Current efforts in test and quality assurance research and engineering still have to be furthered in order to address some of the needs of outlined here, within ElasTest and beyond it. Importantly for ElasTest will the evaluation of the platform with the previously noted vertical demonstrators.

References

1. Alluxio. https://www.alluxio.org/, accessed: 04-05-2018
2. Amazon Web Services. http://aws.amazon.com, accessed: 04-05-2018
3. CloudFoundry. https://cloudfoundry.org, accessed: 04-05-2018
4. Docker. https://www.docker.com, accessed: 04-05-2018
5. ElasTest. http://www.elastest.io, accessed: 04-05-2018
6. Elastest Consortium. https://elastest.eu/consortium.html, accessed: 23-07-2018
7. ElasTest Software Repositories. https://github.com/elastest, accessed: 04-05-2018
8. ElasticSearch. https://www.elastic.co/, accessed: 04-05-2018
9. Fundamental Modeling Concepts. http://www.fmc-modeling.org, accessed: 04-05-2018
10. Horizon Programme. https://ec.europa.eu/programmes/horizon2020/en/what-horizon-2020, accessed: 23-07-2018
11. Kubernetes. https://kubernetes.io, accessed: 04-05-2018
12. MySQL. https://www.mysql.com/, accessed: 04-05-2018

13. Open Service Broker API Specification. https://github.com/openservicebrokerapi/, accessed: 04-05-2018
14. OpenShift. https://www.openshift.com, accessed: 04-05-2018
15. OpenStack. https://www.openstack.org/, accessed: 04-05-2018
16. OpenTracing. http://opentracing.io, accessed: 04-05-2018
17. Testing Strategies in a Microservice Architecture. https://martinfowler.com/articles/microservice-testing/, accessed: 04-05-2018
18. Top Trends for the Future of IT Procurement. https://www.gartner.com/smarterwithgartner/top-trends-for-the-future-of-it-procurement/, accessed: 23-07-2018
19. Carella, G.A., Magedanz, T.: Open baton: a framework for virtual network function management and orchestration for emerging software-based 5g networks. Newsletter **2016** (2015)
20. Gilbert, J.: Cloud Native Development Patterns and Best Practices. Packt Publishers (2018)
21. Newman, S.: Building Micro Services Designing Fine-Grained Systems. O'Reilly Media, Inc. (2014)