D6.3	
Version	2.0
Author	NAEVATEC
Dissemination	PU
Date	31-12-2019
Status	FINAL

D6.3: ElasTest Continuous Integration and Validation System v2

Project acronym	ELASTEST
Project title	ElasTest: an elastic platform for testing complex distributed large software systems
Project duration	01-01-2017 to 31-12-2019
Project type	H2020-ICT-2016-1. Software Technologies
Project reference	731535
Project website	http://elastest.eu/
Work package	WP6
WP leader	Guiomar Tuñón de Hita
Deliverable nature	PUBLIC
Lead editor	Guiomar Tuñón de Hita
Planned delivery date	31-12-2019
Actual delivery date	31-12-2019
Keywords	Open source software, cloud computing, continuous integration, continuous validation,

Funded by the European Union

License

Contributors

Name	Affiliation
Guiomar Tuñón	NAEVATEC
Eduardo de la Iglesia	NAEVATEC
Francisco Ramón Díaz	URJC
Magda Kacmajor	IBM
Piyush Harsh	ZHAW
Felipe Gorostiaga	IMDEA
Luis Miguel Danielsson	IMDEA
Varun Gowtham	TUB
Orlando Ávila	ATOS
Kimon Moschandreou	REL
Francisco Gorostiaga	URJC
Micael Gallego	URJC

Version history

Version	Date	Author(s)	Description of changes
DRAFT00	17/09/2019	Guiomar Tuñón	Revision of the document's FINAL_V1
DRAFT01	25/11/2019	Guiomar Tuñón	Added Info from requirements traceability spreadsheet
DRAFT02	28/11/2019	Guiomar Tuñón, Magda Kacmajor, Andy Edmonds, Luis Miguel Danielsson	E2E tests review from partners.
DRAFT03	04/12/2019	Eduardo de la Iglesia	Added Kubernetes sections
DRAFT04	04/12/2019	Guiomar Tuñón	Added missing end-to-end tests

Table of contents

List of Figures

List of Tables

Glossary of acronyms

Abbreviation	Full definition
APIs	Application programming interfaces
AMI	Amazon Machine Images
AWS	Amazon Web Services
CI	Continuous Integration
CV	Continuous Validation
E2E	End-to-end
ECR	Elastic Container Registry
GUI	Graphical User Interface
OS	Operating System
SW	Software
UI	User Interface
QoE	Quality of Experience
SuT	Software under Test
SiL	Systems in the Large
TiL	Test in the Large
TSS	Test support service
TORM	Test Orchestration and Recommendation Manager
QoS	Quality of Service
UAT	User Acceptance Testing
IPR	Intellectual Property Rights

1 Executive summary

The present document describes the evolution in the design, architecture and maintenance of the ElasTest Continuous Integration (CI) and Continuous Validation (CV) System used in the project. This system has been designed and maintained in the context of the Work Package 6 (WP6) “Continuous Integration & Validation”.

This document describes the evolution in the design, architecture and maintenance of the ElasTest CI environment, completing the previous deliverable 6.1 ElasTest Continuous Integration and Validation System:

- Description of the strategic objectives.
- Description of the environment design, architecture and evolution.
- Description of the available tools in the environment.
- Description of the executed maintenance.
- Description of the CI and CV methodology.

The present version of the document includes the work done during the 18 months of work (July 2018– December 2019) and in some cases, it would refer to previous work described in the 6.1 ElasTest Continuous Integration and Validation System and in specific cases, it would include all the work done in the 33 months (March 2017 – December 2019).

The initial environment devised for running CI/CV tasks for the project started with a single instance that ran the main tools related to the software development process and from the first release of the ElasTest platform it has grown having now four instances, each one with a clear objective regarding the tasks that are meant to be executed over them:

I. Main Instance. Holds the main tools related to the software development process – CI server, repositories, credential generator, etc. –
II. ElasTest Nightly Instance. Hosts the latest developed version of ElasTest (not necessarily stable). This instance main objective is to provide an ElasTest platform where latest changes on the code could be tested.
III. ElasTest Stable Instance. Host the latest stable version of ElasTest. This instance will be used to test the ElasTest Nightly Instance with ElasTest, as specified in the DoA.
IV. ElasTest Nightly K8s Cluster. This “instance” is a Kubernetes’ cluster with two nodes (one master, one slave with the objective of deploying a nightly version of ElasTest so it can be tested nightly and compare executions between Nightly and K8s Nightly in order to grant that both deployments are working, and ElasTest platform and each component works as expected.

In order to have a complete and intensive test suite for the whole ElasTest platform all the components have contributed with specific test suits for their components, these tests have been continuously changing as the set of functionalities of the components have expanded and mutated. In the deliverable these suites are described as they are at the moment of writing, whereas these descriptions could be updated until the last release of the platform.

2 Strategic context and objectives

The ElasTest CI environment and methodology has been designed with the objective of providing the project with a complete set of tools and procedures that must grant the appropriate level of quality of each component and the right integration of all of them.

The CI methodology comprises all the tasks that assure:

- High quality of each of the components from development to release.
- High quality of integration between components.
- High quality of ElasTest as a whole.
- High quality of the CI methodology and CI environment.

The CI environment comprises all the tools that help to achieve and maintain the highest levels of quality in all the steps of the development, testing and release.

The specific configuration of the consortium and the diverse licenses (public/Apache 2.2 and Proprietary) of the components are managed within the CI tasks and tools to grant the appropriate access and dissemination of each component.

The following sections contain the details of the CI Environment [Section 3], the CI and CV methodology [Section 4], and a resume of all the work done and the milestones achieved [Section 5].

3 CI environment

The CI environment is composed by a set of tools managed by Naeva Tec and available to the consortium partners.

The CI environment has two kinds of applications/tools: self-hosted services and provided services. Self-hosted services are those that have been deployed on our own managed servers. Those are fully managed by Naeva Tec. This requires the CI administrator (Naeva Tec) to manage security, access policy, system stability and maintenance (corrective and upgrades). On the other hand, provided services are those that hosted on the providers premises or clouds and serve the technologies and services mainly through an accessible web URL.

During the second part of the project we have been updating the tools but no new tools have been deployed, as the procedures were well defined and accepted by all the components and the initial set of tools where enough.

3.1 Self-hosted Services

The self-hosted Services described in 6.1 ElasTest Continuous Integration and Validation System have been maintained, and we have updated and stabilize the ElasTest Stable instance and added an ElasTest K8s cluster.

3.1.1 ElasTest Stable Instance.

The ElasTest Stable instance contains the ElasTest platform running in single-node mode. This instance is updated with each released version of ElasTest, manually. It can be accessed by all the consortium through a static IP, and it is closed to the rest of the world.

This instance can be used by partners to execute test against the nightly ElasTest manually or through the Jenkins jobs (See 4.2.2. End-to-end tests.)

This instance was remade from 0 on month 20 in order to be launched in a newer and bigger AWS instance.

3.1.2 ElasTest Nightly K8s Cluster.

The ElasTest Nightly K8s Cluster is our newest “instance” created with the sole objective of validating the changes made to the components in order to be compatible with a Kubernetes deployment.

This instance is redeployed nightly in order to assure that the latest changes are tested.

It consists of a single master Kubernetes cluster with a single node on it. It is deployed on both twin servers in AWS with the same capacity that the ElasTest Stable instance. In order to test multi-node distribution of pods on the same cluster, the master is configured to also allow pods to be deployed on it.

The cluster was initially deployed installing Kubernetes manually, through **kubeadm** and controlling the cluster via **kubectl**. As long as Kubernetes does not ships with a default

network implementation, it just defines the model to other tools on how to implement it, we have installed Flannel.

As add-ons to the Kubernetes cluster, we have deployed Fluentd as a data collector, to get all cluster information ready to be exploited by ElasticSearch.

3.2 Tools.

In this section, we make a compendium of all the tools used in the project since the M3 but we will just define the ones added in this second part of the project. For the rest please refer to 6.1 ElasTest Continuous Integration and Validation System.

3.2.1 Tool chain (M3 – M36).

Name	Type	License	Self-hosted	Access	Description
GitHub[1]	Source code repository	Proprietary	No	Public	GitHub is a Web-based Git version control repository hosting service. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.
Jenkins[2]	CI Server	OSS (MIT)	Yes	Consortium only	Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks such as building, testing, and deploying software.
DockerHub[3]	Docker image repository	Proprietary	No	Public	The Docker Hub Registry is free to use for public repositories. Plans with private repositories are available in different sizes. All plans allow collaboration with unlimited people.
OSSRH[4]	Maven and Gradle artifact repository	OSS (Eclipse)	No	Public	Sonatype OSSRH (OSS Repository Hosting) uses Sonatype Nexus Repository Manager to provide repository hosting service for open source project binaries.
Nexus Repository Manager OSS[5]	Maven and Gradle artifact repository	OSS (Eclipse)	Yes	Consortium only	Nexus Repository OSS is a universal repository manager with support for all

					major package formats and types.
Private User Registry[6]	Custom user access manager	Proprietary	Yes	Consortium only	Private User Registry is a service developed by Naeva Tec to manage the access to private Nexus Repository and Amazon ECR, providing access to only Consortium members to the resources published there.
ElasTest[7][7]	--	OSS (Apache)	Yes	Consortium only	An elastic platform to ease end to end testing.
Amazon ECR[8]	Docker image repository	Proprietary	No	Consortium only	Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.
ElasticSearch [9]	Search and Analytics engine	OSS (Elastic)	Yes	Internal	ElasticSearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.
Kibana [10]	Web Console	OSS (Elastic)	Yes	Consortium only	Kibana lets you visualize your ElasticSearch data and navigate the Elastic Stack so you can do anything from tracking query load to understanding the way requests flow through your apps

3.2.1.1

Table 1. CI environment main tools.

ElasTest

ElasTest is the tool developed within this project context, and it is used in two contexts on the CI / CV System:

- As object of the tests, the platform that should be tested before it can be released.
- As part of the tools for testing the SW, the platform that is used for testing.

We use this tool in two different contexts we have deployed the ElasTest platform twice, and we have aliased them as Nightly and Stable.

3.2.1.1.1 Nightly

The Nightly context makes reference to the ElasTest instances (Nightly and Nightly K8s) that are fun with the aim of providing the latest ElasTest version of every component so end-to-end integrated tests can be run.

These ElasTest instances provide all partners a place to test their own components on a production-like environment. Partners can access the ElasTest Nightly (or ElasTest Nightly K8s) UI to do manual testing, check the look and feel, and of course run automated tests with Jenkins jobs. And also automatized tests use these instances as object of the tests as part of the CV procedure.

ElasTest Nightly and ElasTest Nightly K8s have been deployed on AWS following the specification shared in the ElasTest Community [7] validating in this way the correctness of the documentation.

3.2.1.1.2 ElasTest Stable

Since March 2018 we have also a stable version of ElasTest running that is being used for testing the 'ElasTest Nightlies' with ElasTest. This ElasTest is mainly used by the partners through the Jenkins plugin installed in our CI Server.

3.2.1.2

ElasticSearch [9]

ElasticSearch by Elastic is an OpenSource distributed, RESTful search and analytics engine. As the heart of the Elastic Stack, it centrally stores the data from all components of ElasTest and also from the Kubernetes cluster.

ElasticSearch receives inputs from Fluentd and from every log of the cluster, making it available to all Consortium partners so they can check what is happening in their components without accessing to the containers where the components are running. It also aggregates the log from the Kubernetes cluster itself allowing a central point for checking the health of the system.

3.2.1.3

Kibana [10]

Kibana by Elastic is an OpenSource web console that exposes the data collected by Fluentd and aggregated and indexed by ElasticSearch.

Kibana allows the partners to check what is happening in their components in a visual way. They can visualize all data and navigate through the ElasticSearch engine so they can track the work of every element of ElasTest.

3.2.2 Add-ons and auxiliary tools (M3 – M36).

Name	Type	License	Self-hosted	Access	Description
Codecov [11]	Cobertura reports analyser	OSS (Apache 2.2)	No	Public	Codecov provides highly integrated tools to group, merge, archive and compare coverage reports. Whether your team is comparing changes in a pull request or reviewing a single commit, Codecov will improve the code review workflow and quality.
SonarCloud [12]	Code review tool	OSS (LGPL-3.0)	No	Public	Analyse the quality of your source code to detect bugs, vulnerabilities and code smells throughout the development process.
ElasTest Jenkins Library* [13]	Jenkins library for manage ElasTest	OSS (Eclipse)	--	Public	Developed groovy library to be used within Jenkins to help to launch ElasTest and manage ElasTest nodes. Developed in the context of the ElasTest project.
ElasTest Jenkins Plugin [14]	Jenkins plugin to communicate with ElasTest	OSS (Apache 2.2)	---	Public	Plugin to make use of a running ElasTest within a Jenkins job.
Flannel [15]	Kubernetes network implementation	OSS (Apache 2)	---	Internal	Flannel is a virtual network that gives a subnet to each host for use with container runtimes.
Fluentd [16]	Data collector	OSS (Apache 2)	---	Internal	Fluentd is an open source data collector, which lets you unify the data collection and consumption for a better use and understanding of data.

3.2.2.1 ***Deprecated**

Table 2. CI environment auxiliary tools and add-ons.

Flannel [15]

Flannel by CoreOS is an OpenSource implementation of the Kubernetes network model. It is used to communicate Kubernetes nodes and all contained infrastructure as services and pods.

Kubernets does not ship with network implementation, just a model on how to implement it. So to communicate all the elements, a compatible implementation must be installed. We have chosen Flannel because it is one of the simplest implementation available with all the necessary resources.

Flannel is installed using its remote description file thought the kubectl tool:

```
$ sudo kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentatio
n/kube-flannel.yml
```

This installs the Roles, DeamonSets and Services needed to implement the network infrastructure.

Fluentd [16]

Fluentd by Fluent is an OpenSource tool, which lets you unify the data collection and consumption for better use and understanding of data.

3.2.2.2

Fluentd decouples data sources from backend systems by providing a unified logging layer in between. So all data collected by Fluentd is injected as data output to Elastisearch for log handling.

Fluentd is installed along with Elastisearch and Kibana with local scripts based on the provided by Fluent.

3.3 Security and User Access

During this second part of the project, there has been some personnel leaving and others joining. Partners have used the proposed Spreadsheet (GitHub & Component management) to declare these changes and access have been kept updated so people leaving have been maintained as project collaborator in the GitHub repositories, but unlinked from the appropriate Partner Team, so they wouldn't be able to access to the private information and tools, assuring privacy mainly of the private artifacts.

3.4 Maintenance

The CI environment is regularly updated for maintaining the set of tools in a stable, secure and updated state.

The maintenance has been scheduled to be carried every three months with a full upgrade to the latest stable version of each of the hosted tools. Also, Exceptional maintenances have been taken into account for critical or important bugs or security issues. Each of the actuations held on the environment are documented and these documents are kept for future reference and available for all the partners to read.

At the moment the following actuations have been run/ scheduled in the environment:

Date	Name	Cause	Status	Description
03/07/2017	20170703 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx and Docker updates.
02/10/2017	20171002 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker

				Compose, aws-cli updates.
20/11/2017	20171120 - Maintenance Window - Exceptional	Security	Done	OS, Jenkins updates.
22/12/2017	20171222 - Maintenance Window - Exceptional	Security	Done	OS, Jenkins updates.
16/01/2018	20180116 - Maintenance Window	Scheduled	Done with issues	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.
26/02/2018	20180226 - Maintenance Window - Exceptional	Security	Done	OS, Jenkins updates.
02/04/2018	20180402 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.
02/07/2018	20180702 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.
04/10/2018	20181001 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.
24/10/2018	20181024 - Exceptional Maintenance Window	Security	Done	OS, Jenkins updates.
10/01/2019	20190110 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.
22/02/2019	20190222- Maintenance Window - Exceptional	Security	Done	OS, Jenkins updates.
01/04/2019	20190401 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.
01/07/2019	20190701 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker

				Compose, aws-cli updates.
07/10/2019	20191007 - Maintenance Window	Scheduled	Done	OS, Jenkins, Nginx, Docker, Docker Compose, aws-cli updates.

Table 3. Maintenance schedule.

4 Methodology and Procedures

The following section describes the new procedures defined to interact with the environment, tools and maintaining them. Only those that have suffered some changes are described in this document. For Basic Rules and Best practises, Jenkins login, Jenkins Tagging slaves and jobs, Private User Registry, Development environment configuration for AWS ECR and development see 6.1 ElasTest Continuous Integration and Validation System.

4.1 Jenkins Jobs Naming

In addition to the previously described job naming we have added the following rule for those e2e testing jobs using ElasTest to test ElasTest nightly and ElasTest K8s:

- **End to end test nightly/K8s ElasTest jobs:** <component_acronym>-e2e-elastest
- **Comparative nightly vs K8s pipeline:** <component_acronym>-e2e-composed -tests

4.2 Testing

4.2.1 Unitary and integration (component)

During the second part of the project, we have applied the pivot strategy for the unitary and integration testing KPI. While we thought of having a good code coverage with unitary and integration tests in the first releases was a good idea, during first months of 2019 we decided that once the components were quite stable, maintaining the unitary and integration tests updated was very costly for some components and these tests doesn't grant correct functionality, we wouldn't enforce unitary test coverage for components, and focus testing efforts in the End-to-end tests. So code coverage was maintained on best-effort as each team considered appropriate.

Even in best-effort approach, 5 components reached thresholds over 70% and 3 more over 50%.

4.2.2 End-to-end tests.

In this second period of the project, the WP6 has focused on the End-to-end tests for each component and in the platform as a whole.

Following the approach devised at the beginning of the project, we have worked in the 3 stage plan, continuing with the work presented in the first review. Also, we have added a 4th stage at the moment we pivoted to Kubernetes native testing platform, to ensure all the work done in previous stages was reused and applied to this new architecture of the platform.

The stages defined are:

1. **Component end-to-end:** Components provide end-to-end tests that ensure the behaviour of the component and all the services that it makes use of. This kind of tests are usually held against the component API and can be launch against the component running as a part of the ElasTest platform, or against an isolated instance of the component if applies. These kind of tests aren't applicable to all the components, see section 4.2.2.3.
2. **Platform end-to-end, traditional tools:** Most components have defined their own end-to-end tests that test their behaviour within the ElasTest platform through the GUI. These tests reproduce use cases that would make use of the component tested. The assertion clauses are focused on the component tested. Only components that have no GUI are excluded from these end-to-end tests. This way once we run all the component end-to-end tests we have an idea of the actual behaviour of the whole platform. These tests are run by Jenkins jobs nightly.
3. **Platform end-to-end, ElasTest:** All the Jenkins jobs of stage 2 have been converted on TJobs that make use of the ElasTest plugin. These TJobs connect with Stable ElasTest and make use of the advanced features such as browser recording, providing the developers extra possibilities in the analysis of the results. In the Stable ElasTest, all the TJobs executed for all components are in the same project having an overall status of the platform nightly.
4. **Platform end-to-end, ElasTest K8s:** Without major changes, we configured the ElasTest K8s as a new SuT in the ElasTest stable in the same project where stage 3 TJobs where executing. In this way all Jenkins jobs where 100% reused by adding just a configuration parameter selecting which SuT the job should test. With this simple modification we could test the K8s with the same End-to-end tests, and use ElasTest (Stable) feature of execution comparison to check differences in the tests execution between the K8s version and nightly, detecting functionalities that didn't work in the same way in the nightly version and K8s version making it simpler to detect deviations and bugs.

4.2.2.2 *End-to-end tests traceability*

During this second part of the project, we have worked together with WP2 in order to maintain a detailed track of the requirements defined, developed, deployed and tested. With the inclusion of the tests in the traceability procedure we not only traced when a requirement was made available, but we could assure the functionality was behaving as expected, and it wouldn't break on future releases.

We have maintained regular meetings to review the status of the tests and the platform. And focus on those requirements not tested. Some of those requirements are not testable by themselves through the GUI but the functionality is used “behind the scenes” by the tests of other components.

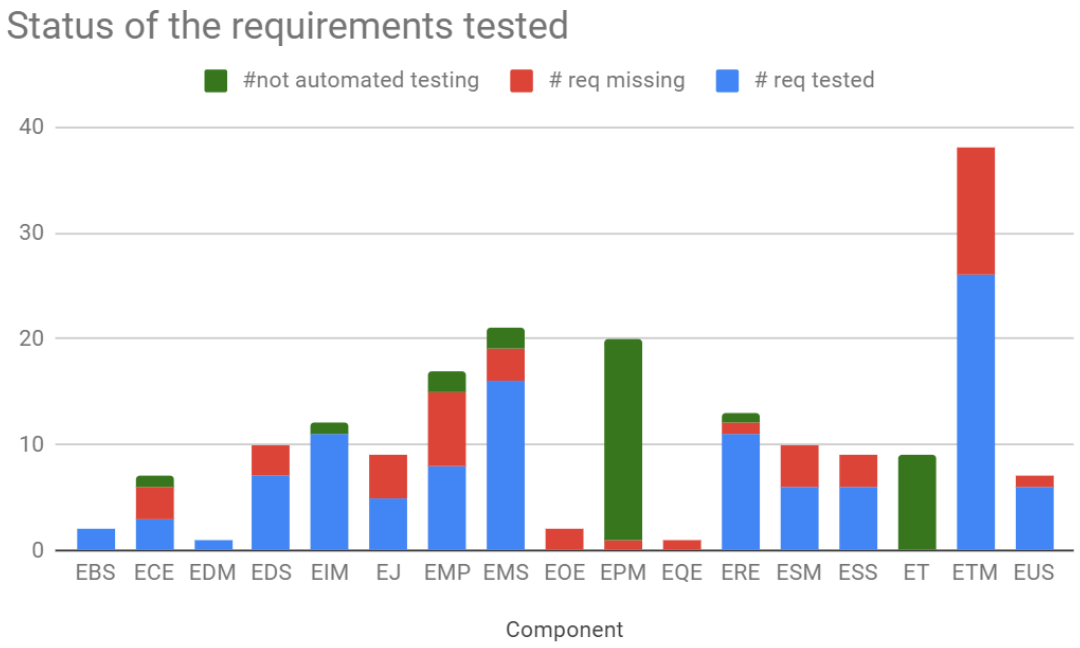


Figure 1. Status of tested requirements

In order to maintain automatic traceability, the provided Requirements Spreadsheet devised by WP2 workgroup, added a dedicated column to the main sheet where a requirement could be declared as not automated tested, and a specific sheet to define the tests developed by each component and which requirement were tested by that test.

In Figure 1. Status of tested requirements a graphic visualisation of the tested requirements is presented. As expected not all the requirements are tested for nearly all the components, this is because with the followed approach for development we first develop the feature, and then the end-to-end test, prioritising the new features over automated tests. It is also true that even if there isn’t an automated test developed all requirements are tested manually and on best effort also tested to avoid regressions.

The tests are described in a table format as follows

Test ID	Traceable test ID	Test Name	Descriptive Name
Requirements tested	List of requirements IDs tested by the test		
Description	Short description of the test		
Step by Step	Step by step of the actions run by the test.		

API End-to-end tests per component.

Each component can be tested as a black box using ElasTest, to assert the validation of the provided APIs, and expected behaviour. Each component can define its own method of validation and test execution scheduled. These tests are mandatory for those functionalities that are executed by API but doesn't have a GUI directly executing these functionalities.

4.2.2.3

In the following subsections, there is a description component by component of all the tests executed in Jenkins with the ElasTest plugin that is executed every night.

4.2.2.3.1 ElasTest Big Data Service

Test ID	API-EBS-001	Test Name	tJobExecutionon_SPARK
Requirements tested	EBS1, EBS2, EDM3		
Description	Test SPARK and EDM functionality		
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Wait for the end of the TJob execution 5. Check if the execution finished correctly 6. Delete TJob and execution		

4.2.2.3.2 ElasTest Cost Engine

Test ID	API-ECE-001	Test Name	RESTDriverTest-testRESTDriver4ETM
Requirements tested	ECE01		
Description	Test whether ETM API service is online or not		
Step by Step	1. Initialize the test system 2. Do a GET request to retrieve a list of registered TJobs from ETM API endpoint 3. Check HTTP status code - 200 representing a successful test		

Test ID	API-ECE-002	Test Name	RESTDriverTest-testRESTDriver4ESM
Requirements tested	ECE02		

Description	Test whether ESM API endpoint is online or not
Step by Step	1. Initialize the test system 2. Do a GET on catalogue list API endpoint of ESM service instance 3. Check HTTP status code - 200 represents a successful test

Test ID	API-ECE-003	Test Name	ControllerTest-getStaticAnalysisDataTest
Requirements tested	ECE03		
Description	Test static analysis form for a selected TJob		
Step by Step	1. Initialize the test system 2. Populate the request parameters with one of preconfigured TJobs 3. Add support services parameters in the http request object 4. Perform a function call on the controller method that controls the display of static analysis resource usage form 5. Check the HTML page being returned to verify correct behaviour		

Test ID	API-ECE-004	Test Name	ControllerTest-showStaticAnalysisTest
Requirements tested	ECE03		
Description	Test static analysis results page for a selected TJob post form submit		
Step by Step	1. Initialize the test system 2. Populate the usage form with mock data values via HTTP request parameters 3. Add support services usage parameters in the http request object 4. Perform a function call on the controller method that controls the display of static analysis results 5. Check the HTML page being returned to verify correct behaviour		

Test ID	API-ECE-004	Test Name	ControllerTest-showDynamicAnalysisTest
Requirements tested	ECE06		
Description	Test dynamic cost analysis generation for a selected TJob		

Step by Step	1. Initialize the test system 2. Populate the HTTP request object with a predetermined TJob 3. Populate the HTTP request object with a predetermined list of support services and associated values 4. Call the controller method that controls the display of true costs for a selected TJob 5. Verify the returned HTML template page name for ascertaining the correct behaviour of the service

4.2.2.3.3 ElasTest Device emulator Service

Test ID	API-EDS-001	Test Name	TestApplication
Requirements tested	EDS1, EDS2, EDS3, EDS4, EDS5, EDS6, EDS7		
Description	A test application to make use of the features of EDS		
Step by Step	1. Minimal EDS is started as a container of image eds-base. It starts the gateway and orchestrator. 2. TJob is able to communicate with minimal EDS. 3. Implemented application logic performs as intended 4. Application logic is able to receive values from the sensor and able to direct actions to the actuator. 5. The EDS orchestrator is able to create, start and teardown devices as required by the user application. 6. Start multiple copies of the same application, still all applications get distinct emulated devices and can perform independently. 7. Reusable code for the emulated device, customizable by the user as required in the application.		

4.2.2.3.4 ElasTest Instrumentation Manager

Test ID	API-EIM-001	Test Name	PacketLossTestsSession
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM15		
Description	Execute API operations [POST, GET, DELETE]		

Step by Step	1. Create a new agent 2. Verify GET operation latency 3. Injection rule **0**% dropped networks 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor 6. Delete agent

Test ID	**API-EIM-002**	**Test Name**	**PacketLossTests0**
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM15		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Verify GET operation latency 3. Injection rule **0**% dropped networks 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor 6. Delete agent		

Test ID	**API-EIM-003**	**Test Name**	**PacketLossTests25**
Requirements tested	EIM1, EIM2, EIM3, EIM4, EIM5, EIM6, EIM15		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Verify GET operation latency 3. Injection rule **25**% dropped networks 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor 6. Delete agent		

Test ID	API-EIM-004	Test Name	PacketLossTests50
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM15		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Verify GET operation latency 3. Injection rule **50**% dropped networks 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor 6. Delete agent		

Test ID	API-EIM-005	Test Name	PacketLossTests75
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM15		
Description	Execute API operations [POST, GET, DELET]		
Step by Step	1. Register an agent 2. Verify GET operation latency 3. Injection rule **75**% dropped networks 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor 6. Delete agent		

Test ID	API-EIM-006	Test Name	ControllabilityMonitoring
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM8, EIM9		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Install monitoring beats [packetbeat, metricbeat, filebeat] 3. Verify GET operation latency 4. Injection rule **25**% dropped networks 5. Verify GET operation latency (SLO latency <=150ms) 6. Unmonitor (Controllability and Monitoring beats) 7. Delete agent		

Test ID	API-EIM-007	Test Name	Monitoring
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM7, EIM8, EIM9		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Install monitoring beats [packetbeat, metricbeat, filebeat] 3. Verify GET operation latency 4. Injection rule **25**% dropped networks 5. Verify GET operation latency (SLO latency <=150ms) 6. Unmonitor (Controllability and Monitoring beats) 7. Delete agent		

Test ID	API-EIM-008	Test Name	CpuCommands1
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM14		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Verify GET operation latency **3. CPU overload: run for 30 seconds with 3 cpu stressors** 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor (Controllability and Monitoring beats) 6. Delete agent		

Test ID	API-EIM-001	Test Name	CpuCommands2
Requirements tested	EIM1, EIM2, EIM3,EIM4, EIM5, EIM6, EIM15		
Description	Execute API operations [POST, GET, DELETE]		
Step by Step	1. Register an agent 2. Verify GET operation latency **3. CPU overload: run for 30 seconds with 68 cpu stressors** 4. Verify GET operation latency (SLO latency <=150ms) 5. Unmonitor (Controllability and Monitoring beats)		

	6. Delete agent

4.2.2.3.5 ElasTest Monitoring Platform

Test ID	API-EMP-001	Test Name	APIOfflineTest-testcreateSpace
Requirements tested	EMP01		
Description	Test to check if monitoring spaces can be created successfully via the EMP REST API		
Step by Step	1. Initialize the test system with a dummy user and preset credentials 2. send a request to create a monitoring space with an empty body, the results should be HTTP 400 code 3. send a request to create a new space and together with valid user credentials, the result should be HTTP 201 status 4. send a request to create the same space, as it is a duplicate space, the result should be HTTP status code 409		

Test ID	API-EMP-002	Test Name	APIOfflineTest-testcreateSeries
Requirements tested	EMP02		
Description	Test to check if monitoring series can be created successfully via the EMP REST APIs		
Step by Step	1. Initialize the test system with a dummy user and preset credentials and an existing monitoring space 2. send a request to create a monitoring series with an empty body, the results should be HTTP 400 code 3. send a request to create a new series and together with valid user credentials, the result should be HTTP 201 status 4. send a request to create the same series, as it is a duplicate series within the same monitoring space, the result should be HTTP status code 409		

Test ID	API-EMP-003	Test Name	APIOfflineTest-testgetEndpointInfo

Requirements tested	EMP03
Description	Test to check if Kafka endpoints can be retrieved for configuration of agents via EMP REST APIs
Step by Step	1. Initialize the test system with a dummy user and pre-set credentials 2. send a request to retrieve agent connection details but without valid credentials, the result should be HTTP status 401 3. send a request again but now with valid credentials, the response should be with HTTP status code 200

Test ID	API-EMP-004	Test Name	KafkaTestProducer-testsend
Requirements tested	EMP04		
Description	Test to check if test messages can be sent to Kafka message bus		
Step by Step	1. initialize the Kafka cluster 2. Initiate a test message sending and check the returned status 3. Is successful, the returned status should be boolean true		

Test ID	API-EMP-005	Test Name	InfluxDBClientTest-.testaddPoint
Requirements tested	EMP05		
Description	Test to check if the InfluxDB endpoints are functional and test samples can be added		
Step by Step	1. Initialize the test system 2. Setup InfluxDB cluster preconfigured with a valid user account, and database with test measurement preconfigured 3. Send a test EMP agent message to the preconfigured InfluxDB endpoint 4. Check the status returned, it should be true for successful insertion		

Test ID	API-EMP-006	Test Name	InfluxDBClientTest-testGetLastPoints

Requirements tested	EMP08
Description	Test to check if InfluxDB interface is functional and can respond to query commands
Step by Step	1. Initialize the test system 2. Setup InfluxDB cluster preconfigured with a valid user account, and database with test measurement preconfigured 3. Initiate InfluxDB DB query using preconfigured credentials against test database and measurement 4. On a successful connection, the last inserted data should be returned.

Test ID	API-EMP-007	Test Name	PingWorkerTest-testrun
Requirements tested	EMP11		
Description	Test to verify whether EMP ping functionality where the liveness of the target system can be ascertained is working as expected or not		
Step by Step			

Test ID	API-EMP-008	Test Name	APIControllerTest-getRootAPI
Requirements tested	EMP14		
Description	Test to quickly verify whether EMP REST server is functional or not by asking for a list of supported API calls		
Step by Step			

4.2.2.3.6 ElasTest Monitoring Service

Test ID	API-EMS-001	Test Name	EMS Double download E2E Test
Requirements tested	EMS1, EMS6, EMS7, EMS8, EMS9, EMS13		
Description	Assertion of valid data retrieved. (Bandwidth)		
Step by Step	1. Create a new project 2. Create a new SuT 3. Create a new TJob		

	4. Execute the TJob that download two files in parallel and assess that it uses twice the bandwidth. 5. Check if the execution finished correctly

Test ID	API-EMS-002	Test Name	EMS Elasticsearch E2E Test
Requirements tested	EMS2, EMS10, EMS13		
Description	Test valid events sent to the elastic search		
Step by Step	1. Create a new project 2. Create a new SuT 3. Create a new TJob 4. Execute the TJob that: 4.1 Sends some events to the EMS. 4.2 The TJob subscribes the Elastcisearch under test. 4.3 It sends more events. 4.4 It unsubscribes the Elasticsearch. 4.5 It sends more events. 4.6 Assesses that only the events in the middle were received. 5. Check if the execution finished correctly		

Test ID	API-EMS-003	Test Name	EMS RabbitMQ E2E Test
Requirements tested	EMS3, EMS6, EMS7, EMS8, EMS9, EMS11, EMS12, EMS13, EMS16		
Description	Test if the RabbitMQ is subscribed to certain channels.		
Step by Step	1. Create a new project 2. Create a new SuT 3. Create a new TJob 4. Execute the TJob that: 4.1 Test if the RabbitMQ under test is subscribed to the correct channels. 4.2 Monitoring machines and stampers are deployed and undeployed while the TJob sends events to the EMS. 4.3 Assertion if only certain events made it to the SuT. 5. Check if the execution finished correctly.		

Test ID	API-EMS-004	Test Name	EMS RPC Orchestration E2E Test
Requirements tested	EMS9, EMS13, EMS16, EMS17, EMS19, EMS21, EMS23		
Description	Test the if-then-else, the previous operator, output JSON data and output through the WebSocket channel		
Step by Step	1. Use Orchestration Library in Jenkins to start a standalone EMS 2. start a SuT 3. configure EMS with the proper specification 4. exercise the SuT with sequential Tjobs (orchestrating Jenkins Jobs) 5 Those Tjobs use the EMS to perform data-driven orchestration 6. the EMS is used to check if the sequence of Tjobs is a good sequence conforming to a use case of the SuT		

Test ID	API-EMS-005	Test Name	EMS-EDS demo
Requirements tested	EMS9, EMS13, EMS17, EMS20		
Description	Test the vector notation in a realistic scenario		
Step by Step	1. Create a new project 2. Create two SuT called 'good' and 'evil' 3. Create four TJob with environment variables: a. 6, linked to 'good' SuT b. 10, linked to 'good' SuT c. 6, linked to 'evil' SuT d. 10, linked to 'evil' SuT 4. Execute all TJobs 5. Check that only TJob '10, evil' fails.		

4.2.2.3.7 ElasTest Service Manager

Test ID	API-ESM-001	Test Name	TestCatalogController-test_catalog
Requirements tested	ESM5		
Description	Basic test of the catalogue		
Step by Step	1. Send GET request against /v2/catalog Validate that the response is successful		

Test ID	API-ESM-002	Test Name	TestCatalogController-test_request_no_version_header
Requirements tested	ESM5		
Description	Bad request – No header		
Step by Step	1. Send GET request against /v2/catalog, excluding the version header Validate response is unsuccessful		

Test ID	API-ESM-003	Test Name	TestCatalogController-test_register_service
Requirements tested	ESM5		
Description	Test of a valid service registration		
Step by Step	1. Send a PUT against /v2/et/catalog containing a new service Validate that the service was successfully registered		

Test ID	API-ESM-004	Test Name	TestCatalogController-test_double_svc_registration_deny
Requirements tested	ESM5		
Description	Bad request – double svc registration		
Step by Step	1. Send a PUT against /v2/et/catalog containing an existing service Validate that the response was unsuccessful		

Test ID	API-ESM-005	Test Name	TestCatalogController-test_store_manifest
Requirements tested	ESM5		
Description	Validate manifest storage		
Step by Step	1. Send a PUT against /v2/et/catalog containing a new service		

	2. Send a PUT against /v2/et/manifest containing a new manifest that is related to the registered service Validate that the response was successful

Test ID	API-ESM-006	Test Name	TestCatalogController-test_update_service
Requirements tested	ESM6		
Description	Validate service update		
Step by Step	1. Create a new service 2. Create a second new service locally 3. Submit the second manifest as the update of the existing manifest Validate that the request was successful		

Test ID	API-ESM-007	Test Name	TestCatalogController-test_update_manifest
Requirements tested	ESM6		
Description	Validate update the service manifest		
Step by Step	1. Create a new service manifest 2. Create a second new service manifest locally 3. Submit the second manifest as the update of the existing manifest Validate that the request was successful		

Test ID	API-ESM-008	Test Name	TestCatalogController-test_get_manifest
Requirements tested	ESM6		
Description	Test get service manifest		
Step by Step	1. Create a service 2. Create a manifest associated with the service 3. Get the manifest Validate the request was successful		

Test ID	API-ESM-009	Test Name	TestCatalogController-test_list_manifests
Requirements tested	ESM6		
Description	Test manifest list		
Step by Step	1. Issue a GET against /v2/et/manifest Validate the request was successful		

Test ID	API-ESM-010	Test Name	TestServiceInstancesController-test_request_no_version_header
Requirements tested	ESM1		
Description	Bad request – no version in the header		
Step by Step	1. Send GET request against /v2/catalog, excluding the version header Validate response is unsuccessful		

Test ID	API-ESM-011	Test Name	TestServiceInstancesController-test_create_service_instance
Requirements tested	ESM1		
Description	Validate the creation of a service instance		
Step by Step	1. Generate a unique ID for the service instance to be created 2. Send a PUT against /v2/service_instances/{instance_id} Validate that the request was successful		

Test ID	API-ESM-012	Test Name	TestServiceInstancesController-test_create_instance_with_same_id
Requirements tested	ESM1		
Description	Bad request – instance with the same id		
Step by Step	1. Generate a unique ID for the service instance to be created 2. Send a PUT against /v2/service_instances/{instance_id} 3. Send another PUT against /v2/service_instances/{instance_id}		

	Validate that the second request was unsuccessful

Test ID	API-ESM-013	Test Name	TestServiceInstancesController-test_create_instance_with_nonexistant_plan
Requirements tested	ESM1		
Description	Bad Request – instance with no plan		
Step by Step	1. Create a service instance without an associated plan and submit the request Validate that the request failed		

Test ID	API-ESM-014	Test Name	TestServiceInstancesController-test_create_service_instance_with_params
Requirements tested	ESM1, ESM4		
Description	Validate the creation of an instance with parameters		
Step by Step	1. Create a service instance with parameters and submit the request Validate that the request succeeded		

Test ID	API-ESM-015	Test Name	TestServiceInstancesController-test_service_bind_unbind
Requirements tested	ESM4		
Description	Validate bind and unbind of a service		
Step by Step	1. Create a service instance with parameters and submit the request 2. Create a binding request against the created service instance 3. Validate that the request succeeded 4. Create an unbinding request against the created service instance Validate that the request succeeded		

Test ID	API-ESM-016	Test Name	TestServiceInstancesController-test_update_service_instance
Requirements tested			
Description	Validate the update of a service instance		
Step by Step	1. Create and submit a new service update request Validate the request was successful		

Test ID	API-ESM-017	Test Name	TestServiceInstancesController-test_all_instance_info
Requirements tested	ESM2		
Description	Validate the population of the instance info		
Step by Step	1. Create a set of new service instances 2. Issue a GET on the newly created service instances Validate that all the service instances information was returned successfully (greater than zero)		

Test ID	API-ESM-018	Test Name	TestServiceInstancesController-test_instance_info
Requirements tested	ESM2		
Description	Validate the correction of the information of the instance		
Step by Step	1. Create a new service instance 2. Issue a GET on the newly created service 3. Validate that all the service information was returned successfully Validate that a networking parameter is present, as a validation test		

Test ID	API-ESM-019	Test Name	TestServiceInstancesController-test_last_operation_status
Requirements tested	ESM2		
Description	Validate retrieving the last operation status		

Step by Step	1. Create a new service instance 2. Issue a GET to get the status of the last operation executed upon the service instance Validate that the response is valid and successful

Test ID	**API-ESM-020**	**Test Name**	**TestServiceInstancesController-test_deprovision_service_instance**
Requirements tested	ESM3		
Description	Validate the deprovision of a service instance		
Step by Step	1. Create a new service instance 2. Issue a DELETE on the service instance endpoint 3. Validate that the request was successful and the service instance no longer exists.		

4.2.2.3.8 ElasTest Test Manager

Test ID	**API-ETM-001**	**Test Name**	**ProjectApiItTest-testCreateProject**
Requirements tested	ETM1		
Description	Creates a new project in ElasTest		
Step by Step	1. Create a new project 2. Check if the project is correctly created		

Test ID	**API-ETM-002**	**Test Name**	**ProjectApiItTest-testGetProjects**
Requirements tested	ETM1		
Description	Retrieves all projects in ElasTest		
Step by Step	1. Create N projects 2. Check if the projects were created		

Test ID	**API-ETM-003**	**Test Name**	**ProjectApiItTest-testGetProjectById**

Requirements tested	ETM1
Description	Gets a project by id
Step by Step	1. Create a project to retrieve 2. Send request 3. Check if returned project name matches with the sent project name

Test ID	API-ETM-004	Test Name	ProjectApiItTest-testDeleteProject
Requirements tested	ETM1		
Description	Deletes a project identified by the id provided		
Step by Step	1. Create a project to delete 2. Send delete request 3. Check if the deletion operation was successful		

Test ID	API-ETM-005	Test Name	SutApiItTest-testCreateSut
Requirements tested	ETM3		
Description	Creates a new SUT in ElasTest		
Step by Step	1. Create a new project 2. Create a new SUT 3. Check if the SUT is correctly created		

Test ID	API-ETM-006	Test Name	SutApiItTest-testModifySut
Requirements tested	ETM3, ETM2		
Description	Modifies an existing SUT		
Step by Step	1. Create a new project 2. Create a new SUT 3. Retrieve the SUT from ElasTest 4. Modify and save the SUT		

	5. Check if the SUT has been modified correctly

Test ID	API-ETM-007	Test Name	SutApiItTest-testGetSuts
Requirements tested	ETM3, ETM2		
Description	Retrieves all SUTs in ElasTest		
Step by Step	1. Create a new project 2. Create n new SUTs 3. Retrieve SUTs in ElasTest 4. Check if they have been retrieved		

Test ID	API-ETM-008	Test Name	SutApiItTest-testDeleteSut
Requirements tested	ETM3, ETM2		
Description	Deletes an existing SUT		
Step by Step	1. Create a new project 2. Create a new SUT 3. Delete the new SUT 4. Check if the SUT has been deleted correctly		

Test ID	API-ETM-009	Test Name	SutApiItTest-testCreateSutWithCommandsContainer
Requirements tested	ETM3, ETM2		
Description	Creates a SuT with commands container		
Step by Step	1. Create a new project 2. Create a new SUT 3. Check if the SUT has been deleted correctly		

Test ID	API-ETM-010	Test Name	TJobApiItTest-testCreateTJob
Requirements tested	ETM4, ETM5		

Description	Creates a new TJob
Step by Step	1. Create a new project 2. Create a new TJob 3. Check if the TJob is correctly created

Test ID	API-ETM-011	Test Name	TJobApiItTest-testModifyTJob
Requirements tested	ETM4, ETM5		
Description	Modifies an existing TJob		
Step by Step	1. Create a new project 2. Create a new TJob 3. Retrieve the TJob from ElasTest 4. Modify and save the TJob 5. Check if the TJob has been modified correctly		

Test ID	API-ETM-012	Test Name	TJobApiItTest-testGetTJobs
Requirements tested	ETM4, ETM5		
Description	Retrieves all TJobs in ElasTest		
Step by Step	1. Create a new project 2. Create n new TJobs 3. Retrieve TJobs in ElasTest 4. Check if they have been retrieved		

Test ID	API-ETM-013	Test Name	TJobApiItTest-testGetTJobById
Requirements tested	ETM4, ETM5		
Description	Retrieves a TJob for a given id		
Step by Step	1. Create a new project 2. Create a new TJobs 3. Retrieve the TJob from ElasTest		

	4. Check if the TJob have been retrieved

Test ID	API-ETM-014	Test Name	TJobApiItTest-testDeleteTJob
Requirements tested	ETM4, ETM5		
Description	Deletes an existing TJob		
Step by Step	1. Create a new project 2. Create a new TJob 3. Delete the new TJob 4. Check if the TJob has been deleted correctly		

Test ID	API-ETM-015	Test Name	TJobExecutionApiItTest-testExecuteTJobWithSut
Requirements tested	ETM6		
Description	Execute a TJob with SUT and check the results.		
Step by Step	1. Create a new project 2. Create a new SUT 3. Create a new TJob that uses the SUT 4. Execute the TJob 5. Wait for the end of the TJob execution 6. Check if the execution finished correctly 7. Delete TJob and execution		

Test ID	API-ETM-016	Test Name	TJobExecutionApiItTest-testExecuteTJobWithoutSut
Requirements tested	ETM6		
Description	Execute a TJob without SUT and check the results.		
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Wait for the end of the TJob execution		

	5. Check if the execution finished correctly 6. Delete TJob and execution

Test ID	API-ETM-017	Test Name	TJobExecutionApiItTest-testExecuteTJobWithoutSutAndGetLogs
Requirements tested	ETM6		
Description	Execute a TJob with SUT and check the results.		
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Check logs 5. Wait for the end of the TJob execution 6. Check if the execution finished correctly 7. Delete TJob and execution		

Test ID	API-ETM-018	Test Name	TJobExecutionApiItTest-testExecuteTJobWithoutSutAndStop
Requirements tested	ETM6		
Description	Execute a TJob and stop it before it is finished		
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Stop TJob execution 5. Check if the execution finished correctly 6. Delete TJob and execution		

4.2.2.4 Integrated GUI end-to-end tests

ElasTest is a set of components that should be tested as a whole and following user paths through the GUI.

As the use of the GUI is an essential part of the GUI end-to-end test, all these tests make use of the ElasTest User impersonation Service (Browsers as a service) from the Stable ElasTest so all these tests have videos available for each of their executions. These tests

have been greatly advanced during the second period as all components with relevant GUI user paths not have their test up and running but make intensive use at least of one of the Services of the ElasTest.

In the document, we have described the test running nightly under the premise of ElasTest in ElasTest and only those of them that have a relevant GUI interaction.

4.2.2.4.1 ElasTest Big Data Service

Test ID	GUI-EBS-001	Test Name	TJob Execution
Requirements tested	EBS1, EBS2, EDM3		
Description	Test EBS and EDM functionality through the Graphic User Interface		
Step by Step	1. Create a new project 2. Create a new TJob. *The TJob downloads a file from https://norvig.com/big.txt* and feeds it to the SPARK engine. The file contains a very long text and SPARK computes how many times occurs each word. The result is stored on Hadoop (EDM 1 and EDM2) 3. Execute the TJob 4. Wait for the end of the TJob execution 5. Check if the execution finished correctly 6. Delete TJob and execution		

4.2.2.4.2 ElasTest Cost Engine

Test ID	GUI-ECE-001	Test Name	ECEElasTestInElasTestTest-check4ece
Requirements tested	ECE1, ECE2		
Description	Verify that ECE is integrated with the ElasTest UI and is accessible via the side navigation panel by end-users.		

Step by Step	1. Reset the test system, set browser dimensions to a pre-specified dimensions 2. Retrieve the ElasTest URL from the environment parameters 3. Using selenium drivers, click the sidebar link for test engines 4. Wait for the page to load 5. Navigate to the ECE link and click on the start engine button 6. Wait for the button state changes to view engine icon 7. Click the View Engine button once the engine has been started 8. Switch the focus to the iFrame which contains the ECE UI 9. Assert that the HTML element corresponding to ECE UI element has been displayed in the browser

4.2.2.4.3 ElasTest Device emulator Service

Test ID	GUI-EDS-001	Test Name	EDS example application execution
Requirements tested	EDS1, EDS2, EDS3, EDS4, EDS5, EDS9, EDS10, EDS11		
Description	Test if EDS is available to the user, if available, request for devices, wire them together with application logic and run the application for a limited duration.		
Step by Step	1. Open ElasTest page. 2. Create a new project and enter into the project. 3. Create a new SuT and configure the SuT. 4. Create a new TJob and configure the TJob and assign the already created SuT to the TJob. 5. Run the test. 6. Test verdict is obtained based on individual verdicts of the tests in the TJob. 7. The video recorded in the GUI test helps in debugging issues during test execution.		

4.2.2.4.4 ElasTest Instrumentation Manager

Test ID	GUI-EIM-001	Test Name	EimTJobE2ETest-testTJob

Requirements tested	EIM-001, EIM-002, EIM-003, EIM-004, EIM-005, EIM-006, EIM-007, EIM-008, EIM-009
Description	Basic TJob creation (base case for packet loss comparison)
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Wait for the end of the TJob execution 5. Check if the execution finished correctly 6. Delete TJob and execution

Test ID	GUI-EIM-002	Test Name	PacketLossTestsSession
Requirements tested	EIM-001, EIM-002, EIM-003, EIM-004, EIM-005, EIM-006, EIM-007, EIM-008, EIM-009, EIM-014		
Description	TJob base test with a % of packet loss injected		
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Verify GET operation latency 5. Injection rule variable% dropped networks 6. Verify GET operation latency (SLO latency <=150ms) 7. Check if the execution finished correctly 8. Delete TJob and execution		

Test ID	GUI-EIM-003	Test Name	EimTJobE2ETest-testTJob-generalTestTJob
Requirements tested	EIM05, EIM06, EIM07, EIM08, EIM09		
Description	Basic TJob creation (base case for CPU command)		

Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Wait for the end of the TJob execution 5. Check if the execution finished correctly 6. Delete TJob and execution

Test ID	**GUI-EIM-004**	**Test Name**	**EIMTjobCpuCommands-testTJob**
Requirements tested	EIM-001, EIM-002, EIM-003, EIM-004, EIM-005, EIM-006, EIM-007, EIM-008, EIM-009, EIM-015		
Description	TJob test with CPU overload		
Step by Step	1. Create a new project 2. Create a new TJob 3. Execute the TJob 4. Verify GET operation latency 5. ***CPU overload: run for X seconds with Y CPU stressors*** 6. Verify GET operation latency (SLO latency <=150ms) 7. Check if the execution finished correctly 8. Delete TJob and execution		

4.2.2.4.5 ElasTest Jenkins

Test ID	**GUI-EJ-001**	**Test Name**	**ElasTestPluginE2ETest-testPipelineJob**
Requirements tested	EJ1, EJ2, EJ5, EJ11, EJ12		
Description	Test that the Jenkins plugin works correctly		
Step by Step	1. Install on Jenkins the plugin from a hpi file 2. Configure plugin		

	3. Create a pipeline Job 4. Execute the Job 5. Go to ElasTest 6. Wait until the TJob is finished

4.2.2.4.6 ElasTest Monitoring Platform

Test ID	GUI-EMP-001	Test Name	EMPElasTestInElasTestTest
Requirements tested	EMP9, EMP16, EMP19		
Description	Verify that EMP preconfigured dashboard is integrated in the ElasTest Torm UI and can be accessible by end-users		
Step by Step	1. Reset the test system, set browser dimensions to prespecified dimensions. 2. Retrieve the elastest url from the environment parameters. 3. Using selenium driver click the navigation sidebar to access the Platform Monitoring dashboard. 4. Get details on the iFrame element which shows the monitoring dashboard. 5. Check the src value of the iFrame object. 6. Check the value to correspond to the expected value that is returned by the emp service.		

4.2.2.4.7 ElasTest Recommendation Engine

Test ID	GUI-ERE-001	Test Name	EreEnd2EndTests-verifyPreprocessUserData
Requirements tested	ERE1, ERE4, ERE6, ERE14		
Description	Verify that user can load a pre-process their training data		
Step by Step	1. Open TORM Dashboard; 2. Select Test Engines in side menu; 3. Click on 'ere' to start it and wait for the spinner to complete; 4. Open More Options menu; 5. Select Admin option;		

	6. Select Pre-process tab; 7. Select Repository Type - Remote; 8. Enter a remote repository url; 9. Select storage type - local/Alluxio; 10. Launch pre-processing and wait for the process to complete; 11. Verify success message 12. Open Submit Dataset tab; 13. Verify that pre-processed dataset is available for submission; 14. Select Delete checkbox 15. Verify that delete action returned success 16. Stop ERE engine

Test ID	GUI-ERE-002	Test Name	EreEnd2EndTests-verifyAskRecommender
Requirements tested	ERE7, ERE8, ERE9, ERE11		
Description	Verify that the user can enter recommendation query and the engine returns a valid result.		
Step by Step	1. Open TORM Dashboard; 2. Select Test Engines in side menu; 3. Click on 'ere' to start it and wait for the spinner to complete; 4. Open More Options menu; 5. Select Default Settings option; 6. Expand dropdown and select a model to query; 7. Save and return to main page; 8. Open New Recommendation wizard; 9. Enter a text into Area field; 10. Enter a text into Task field; 11. Click OK and wait for the spinner to complete; 12. Verify that Generated Testcase pane contains the result; 13. Verify that Reusable Testcases table contains results; 14. Verify first row in Reusable Testcases table: 14-2. Verify class name is present and valid; 14-3. Verify test case name is present and valid;		

	14-4. Verify similarity score is present and valid; 15. Stop ERE engine

Test ID	GUI-ERE-003	Test Name	EreEnd2EndTests-verifyPreprocessInlineHelp
Requirements tested	ERE13		
Description	verify that inline help displays correctly on the pre-processing tab		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Admin Dashboard 3. Navigate to pre-process tab 4. Click on local repository help icon and verify that help message is correct 5. Click on remote repository help icon and verify that help message is correct 6. Click on repository name help icon and verify that help message is correct 7. Click on local storage type help icon and verify that help message is correct 8. Click on Alluxio storage local help icon and verify that help message is correct 9. Click on additional properties help icon and verify that help message is correct		

Test ID	GUI-ERE-004	Test Name	EreEnd2EndTests-verifySubmitDatasetInlineHelp
Requirements tested	ERE13		
Description	verify that inline help displays correctly on data submit tab		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Admin Dashboard 3. Navigate to submit dataset tab 4. Click on datasets help icon and verify that help message is correct 5. Click on delete help icon and verify that help message is correct		

Test ID	GUI-ERE-005	Test Name	EreEnd2EndTests-verifyTrainInlieHelp
Requirements tested	ERE13		
Description	verify that inline help displays correctly on the train tab		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Admin Dashboard 3. Navigate to train tab 4. Click on data collections help icon and verify that help message is correct		

Test ID	GUI-ERE-006	Test Name	EreEnd2EndTests-verifySubmitDataset
Requirements tested	ERE2		
Description	verify that user can submit a pre-processed dataset		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Admin Dashboard 3. Navigate to submit dataset tab 4. From Datasets drop-down list select specific dataset 5. Click on Submit button		

Test ID	GUI-ERE-007	Test Name	EreEnd2EndTest-verifyTrainModel
Requirements tested	ERE3		
Description	verify that user can train submitted dataset		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Admin Dashboard 3. Navigate to train tab 4. From Data Collections drop-down list select specific data collection 5. Click on Train button		

Test ID	GUI-ERE-008	Test Name	EreEnd2EndTests-verifyPreprocessUserDataAlluxio
Requirements tested	ERE1, ERE4, ERE6, ERE14		
Description	Verify that user can load a pre-process their training data		
Step by Step	1. Open TORM Dashboard; 2. Select Test Engines in side menu; 3. Click on 'ere' to start it and wait for the spinner to complete; 4. Open More Options menu; 5. Select Admin option; 6. Select Pre-process tab; 7. Select Repository Type - Remote; 8. Enter a remote repository url; 9. Select storage type - Alluxio; 10. Launch pre-processing and wait for the process to complete; 11. Verify success message 12. Open Submit Dataset tab; 13. Verify that pre-processed dataset is available for submission; 14. Select Delete checkbox 15. Verify that delete action returned success 16. Stop ERE engine		

Test ID	GUI-ERE-009	Test Name	EreEnd2EndTests-verifySubmitDatasetAndDelete
Requirements tested	ERE2		
Description	verify that user can submit pre-processed dataset and delete		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Admin Dashboard 3. Navigate to submit dataset tab 4. From Datasets drop-down list select specific dataset 5. Click on Submit button		

Test ID	GUI-ERE-010	Test Name	EreEnd2EndTests-verifyGetRecommendationsInlineHelp
Requirements tested	ERE13		
Description	verify that inline help displays correctly on the recommendation page		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Default settings and set model 3. Navigate to new Recommendation 4. Insert in Area description “description” 5. Click in Area description help icon and compare help content 6. Insert in Task description “description” 7. Click in Task description help icon and compare help content 8. Click OK 9. Click in Recommended test case help icon and compare help content 10. Click in Test cases recommended for re-use help icon and compare help content		

Test ID	GUI-ERE-011	Test Name	EreEnd2EndTests-verifyGetRecommendationsAllContent
Requirements tested	ERE8		
Description	verify that content and functionality on the recommendation page		
Step by Step	1. Navigate to ElastTest Test Engines 2. Navigate to Default settings and set model 3. Navigate to new Recommendation 4. Insert in Area description “description” 5. Click in Area description help icon and compare help content 6. Insert in Task description “description” 7. Click in Task description help icon and compare help content 8. Click OK 9. Click in Recommended test case help icon and compare help content		

	10. Click in Test cases recommended for re-use help icon and compare help content 11. verify Show details content 12. verify scroll works 13. verify New recommendation button works - wizard display again 14. verify close button works 15. verify description contents displayed correctly

Test ID	GUI-ERE-012	Test Name	GetRecommendationsAllContentTrial-verifyGetRecommendationsInlineHelp
Requirements tested	ERE13, ERE16		
Description	verify inline help in the trial version user interface		
Step by Step	1. Navigate to ElastTest Test Engines 2. Open New Recommendation wizard 3. Find Area input field and insert description of test Area 4. Click on the corresponding inline help icon 5. Verify inline help text displays correctly 6. Find Task input field and insert description of testing Task 7. Click on the corresponding inline help icon 8. Verify inline help text displays correctly 9. Click OK 10. Find the 'Recommended test case' section and click on the corresponding inline help icon 11. Verify inline help text displays correctly 10. Find the 'Tests recommended for re-use' section and click on the corresponding inline help icon 11. Verify inline help text displays correctly		

Test ID	GUI-ERE-013	Test Name	GetRecommendationsAllContentTrial-verifyGetRecommendationsAllContentTrial
Requirements tested	ERE8, ERE16		

Description	Verify the content and functionality on the recommendation page in the trial version
Step by Step	1. Navigate to ElastTest Test Engines 2. Open New Recommendation wizard 3. Find Area input field and insert description of test Area 4, Find Task input field and insert description of testing Task 5. Click OK 6. Wait for the Result Page to display 7. Verify that the Area description displayed in the Details section matches the text inserted in step 3 8. Verify that the Area description displayed in the Details section matches the text inserted in step 4 9. Verify that the Queried Model description displayed in the Details section is: GenericModel. 10. Verify the result displayed in the "Recommended test case" section 11. Verify top result displayed in the "Tests recommended for re-use" section 12. Verify that scroll widget works 13. Verify that Close button works 14. Verify that New Recommendation wizard opens again

4.2.2.4.8 ElasTest Security Service

Test ID	GUI-ESS-001	Test Name	e2e-test.test_create_exec_tjob
Requirements tested	1. ETM1, ETM2, ETM3, ETM4, ETM5, ETM6, ETM7, ETM8, ETM9, ETM18, **ESS3**		
Description	Create a TJob that uses EUS-issued web browser to visit a web site and execute a test		
Step by Step	1. Visit the ElasTest Dashboard. 2. Create a TJob that uses EUS for opening a web browser and testing the login functionality of the FullTeaching web application and start the ESS scan. 3. Tick the EUS and ESS boxes among the Test Support Services for the TJob 4. Executes the TJob and waits for it to finish.		

	6. Checks if the Execution has finished correctly and the ESS test results have been displayed

4.2.2.4.9 ElasTest Test Manager

Test ID	GUI-ETM-001	Test Name	EtmWebappE2eTest-testCreateChromeTest
Requirements tested	ETM1, ETM2, ETM3, ETM4, ETM5, ETM6, ETM7, ETM8, ETM9, ETM11, ETM18, ETM22, ETM26, ETM27		
Description	Test that creates a Project, a SuT and a TJob with EUS and executes it.		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Checks if the Project to create already exists and if not, it is created. 3. Checks if the SuT already exists to create within the Project and if it is not, it is created 4. Checks if the TJob already exists to create within the Project and if it is not, it is created (with SuT and EUS) 5. Executes the TJob and waits for it to finish. 6. Checks if the Execution has finished with FAIL result		

Test ID	GUI-ETM-002	Test Name	EtmWebappE2eTest-testCreateMultiTest
Requirements tested	ETM32, ETM1, ETM2, ETM3, ETM4, ETM5, ETM6, ETM7, ETM8, ETM9		
Description	Test that creates a Project, a SuT and a Multi-Axis TJob with EUS and executes it.		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Checks if the Project to create already exists and if not, it is created. 3. Checks if the SuT already exists to create within the Project and if it is not, it is created 4. Checks if the TJob already exists to create within the Project and if it is not, it is created (with SuT and EUS) 5. Executes the TJob and waits for it to finish. 6. Checks if the Execution has finished with FAIL result		

Test ID	GUI-ETM-003	Test Name	EtmLogAnalyzerE2eTest-testExecuteAndCheckLogsInLogAnalyzer
Requirements tested	ETM6, ETM7, ETM8, ETM10, ETM18, ETM22, ETM27		
Description	Test that executes an existent TJob and checks the logs in LogAnalyzer when the execution has finished.		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Navigates to the Project. 3. Executes the TJob and waits for it to finish. 4. Checks if the Execution has finished with SUCCESS result. 5. Navigates to LogAnalyzer from the button of the Execution. 6. Checks if there are logs.		

Test ID	GUI-ETM-004	Test Name	EtmTestLinkFullteachingE2eTest-tlFullteachingDataTest
Requirements tested	ETM9, ETM12, ETM13, ETM22, ETM26, ETM27, ETM31		
Description	Test that creates sample data in TestLink, syncs them with ElasTest and executes Test Plan.		
Step by Step	1. Checks if TestLink is started and if not, starts it. 2. Create data into TestLink. 3. Navigates to the TestLink section of ElasTest GUI. 4. Syncs TestLink data with ElasTest 5. Checks if data exists in ElasTest (if has been sync successfully) 6. Executes the Test Plan		

Test ID	GUI-ETM-005	Test Name	EtmHelpPageE2eTest-checkElasTestVersion
Requirements tested	ETM15, ETM22, ETM27		
Description	Test that navigates to the Help page and checks ElasTest version		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Navigates to Help page		

	3. Checks the ElasTest version

Test ID	GUI-ETM-006	Test Name	EtmHelpPageE2eTest-checkElasTestMainServices
Requirements tested	ETM16, ETM22, ETM27		
Description	Test that navigates to Help page and checks if the main services table is not empty		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Navigates to Help page 3. Checks if Main Services table is not empty		

Test ID	GUI-ETM-007	Test Name	EtmTestEnginesE2eTest-startAndStopTestEngine
Requirements tested	ETM14, ETM22, ETM27		
Description	Test that starts and stops a Test Engine		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Navigates to Test Engines page. 3. Starts first test engine (ECE) by clicking start button. 4. Waits until 'Ready' status appears 5. Stops the test engine by clicking stop button. 6. Waits until 'Not initialized' status appears.		

Test ID	GUI-ETM-008	Test Name	EtmTestSupportServicesE2eTest-startAndStopTss
Requirements tested	ETM19, ETM22, ETM27		
Description	Test that starts and stops a Test Engine		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Navigates to Test Support Services page. 3. Select EUS TSS. 4. Starts EUS by clicking Create Instance button.		

	5. Waits until 'Ready' status appears 6. Stops the TSS by clicking stop button. 7. Waits until 'Not initialized' status appears.

Test ID	GUI-ETM-009	Test Name	EtmLogComparatorE2eTest-testExecuteAndCompareLogsWithLogComparator
Requirements tested	ETM28, ETM33, ETM5, ETM6, ETM8		
Description	Test that executes a TJob twice with different parameters to obtain a successful and a failed execution and compares its logs with LogComparator.		
Step by Step	1. First navigates to the ElasTest GUI (Dashboard). 2. Navigates to Project. 3. If TJob already exists, deletes it. 4. Creates TJob. 5. Runs TJob with default parameters and waits for success result. 6. Runs TJob with other parameters and waits for fail result. 7. Navigates to TJob 8. Select All executions (2) 9. Click to "Compare Executions" 10. Check that the log comparator is not empty in any of the view/comparison combinations		

4.2.2.4.10 ElasTest User Impersonation Service

Test ID	GUI-EUS-001	Test Name	EusSupportServiceE2eTest-testSupportService
Requirements tested	EUS1, EUS6, EUS8		
Description	Check that EUS works fine as an independent TSS		
Step by Step	1. Navigate to ETM 2. Start a EUS TSS 3. Select a Chrome browser and start a session 4. Wait until the browser is loaded		

	5. Navigate to elastest.io 6. Closer browser 7. View session recording 8. Remove recording

Test ID	GUI-EUS-002	Test Name	EusTJobE2eTest-testTJob-EusTJobE2eTest
Requirements tested	EUS1, EUS6, EUS8		
Description	Check that the EUS works properly together with a TJob		
Step by Step	1. Navigate to ETM 2. Create a new project 3. Create a new TJob that uses the EUS 4. Run the new TJob 5. Wait for the EUS GUI 6. Closer browser 7. Wait until TJob has successfully finished		

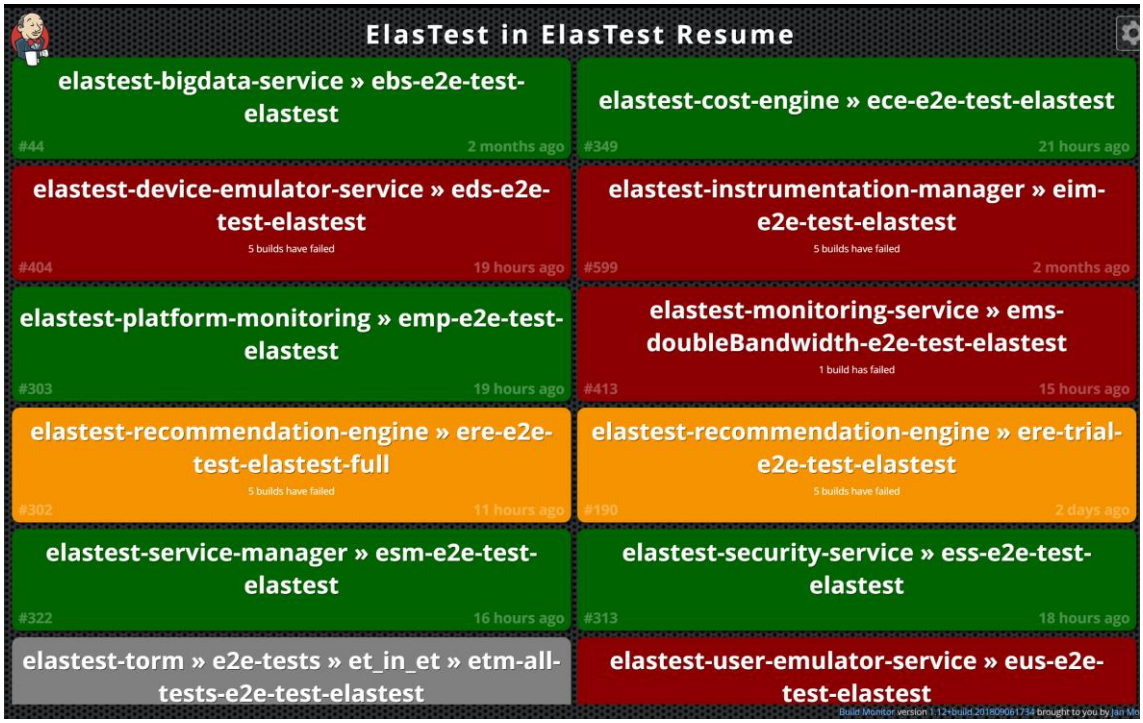
Test ID	GUI-EUS-003	Test Name	EusWebRtcE2eTest-testCreateOpenViduWebRTC
Requirements tested	EUS1, EUS6, EUS7, EUS8		
Description	Check if WebRTC metrics are sent to the ETM from the browser		
Step by Step	1. Navigate to ETM 2. Create a new project 3. Create a new TJob that uses the EUS 4. Run the new TJob 5. Wait until TJob has successfully finished 6. Check if there are webRTC metrics		

Test ID	GUI-EUS-004	Test Name	EusAWSBrowserE2eTest-testBrowserInAWSTest

Requirements tested	EUS1, EUS6, EUS8, EUS11
Description	Check if WebRTC metrics are sent to the ETM from the browser
Step by Step	1. Navigate to ETM 2. Create a new project 3. Create a new TJob that uses the EUS 4. Run the new TJob 5. Wait until TJob has successfully finished 6. Check if there are webRTC metrics

4.2.2.5 End-to-end tests global overview

In order to have a general overview in the status of each component in the nightly version of ElasTest we have created a dashboard in Jenkins that reflects the last end-to-end test job executed for each component



It is common that as shown in the image some components fail whenever other component implements a change, so developers get a notification and they start to look for the bugs using the available tools in ElasTest Stable.

5 Resume & conclusion.

The work done in the context of the WP6 has been quite successful. We have set a complete CI & CV environment fully maintained, providing the consortium partners with a whole set of integrated tools and procedures to test and deploy their developments.
In addition to the most common tools that are used in professional environments, we have also added ElasTest itself as a tool to test each component and the integration itself. This has proved to be an exceptional way to tests not only component by component and the integration between them but a way of having first-hand feedback for the platform itself.
Being the final objective of this project to build a platform to ease software end-to-end, and improve the quality of the Software under Test. We have focused on the quality of the platform itself being the development of automated tests a must for all the components. This has led to a fully usable platform, where the requirements implemented are being tested on a nightly basis.
Regarding the integrations, we have taken into account actual feedback from companies that are already testing intensively their products or offering testing as a service. So we could focus on the integration with their preferred tools i.e. Jenkins and TestLink.

6 References

[1] GitHub - development platform inspired by the way you work. - https://github.com/
[2] Jenkins – Build great thinks at any scale- https://jenkins.io
[3] DockerHub - Dev-test pipeline automation, 100,000+ free apps, public and private registries - https://hub.docker.com/
[4] OSSRH – The Central Repository: Serving Open Source Components Since 2002 - http://central.sonatype.org/
[5] Nexus Repository Manager OSS - The world's only repository manager with FREE support for popular formats. - https://www.sonatype.com/nexus-repository-oss
[6] Private User Registry - Manage the creation of user access to private resource - https://ci.elastest.io/user-registry/
[7] ElasTest - An elastic platform to ease end to end testing – https://elastest.io
[8] Amazon ECR - Amazon Elastic Container Registry - https://aws.amazon.com/ecr/?nc1=h_ls
[9] ElasticSearch - https://www.elastic.co/products/elasticsearch
[10] Kibana - https://www.elastic.co/products/kibana
[11] CodeCov.io - Enhancing development workflows and improving code quality. - https://codecov.io/
[12] SonarCloud - https://about.sonarcloud.io/
[13] ElasTest Jenkins Library - https://github.com/elastest/ci-elastest-jenkins-lib
[14] ElasTest Jenkins Plugin - https://wiki.jenkins.io/display/JENKINS/ElasTest+Plugin
[15] Flannel - https://github.com/coreos/flannel
[16] Fluentd - https://www.fluentd.org

ANNEXES

A1. Maintenance Window Procedure Template (updated)

A1.1. General Information

Affected tools / SW

Template to be filled on each Maintenance Window one row per tool.

SW to be upgraded	Old version	New version	Documentation	Location
<name of the tool>	Currently installed tool	Proposed version	Link to the official documentation of the proposed version	host / container in host / image

Motivation

[Scheduled maintenance window]

[Critical request]

- ¿Who has requested it?
- ¿Why has been defined as critical?

Risks

- [Docker] The images that use the host Docker could fail until their own Docker is upgraded.
- [Jenkins] Some Jobs may need to be reconfigured to work.
- ...

Contact information.

	Partner	User (Name)	Email
Requester			
Upgrade responsible	Naeva Tec		
Notify to	ALL	WP6	elastest-wp6@googlegroups.com

Upgrade Plan

	Date and Time	Duration*
Start	**{_DATE_} 07:15 am**	**2 h**
Init communication	{_DATE_} 07:15 am	5 min
Back Up	{_DATE_} 07:20 am	15 min

Upgrade	{_DATE_}07:35 am	1 h
Test and confirmation	{_DATE_} 08:35 am	25 min
Rollback	{_DATE_} 09:00 am	10 min
Communication of results	{_DATE_} 09:15 am	5 min
End of the Upgrade	**{_DATE_} 09:15 am**	

A1.2. Procedure

A1.2.1.Notification

WP6 users will be notified through the mailing list at the beginning of the upgrade procedure. Expected maintenance time will be reminded in this email.

A1.2.2. System shutdown.

A1.2.2.a Main Instance.

The administrators will check the CI environment is available for the upgrade.

- No jobs are being executed
- No user processes are executing.

If there are slaves up, they will be stopped.

The security group will disable public http access to the instance only access from Naeva Tec will be accepted. ()

A1.2.2.b Slaves.

No actions required

A1.2.2.c ElasTest K8s Nightly.

Stop the service through the script provided:

```
$ /usr/local/bin/kubernetes-evacuate
```

This will finish all pods from both master and slaves

The security group will disable public http access to the instance only access from Naeva Tec will be accepted. (Figure 2. AWS disable inbound rules)

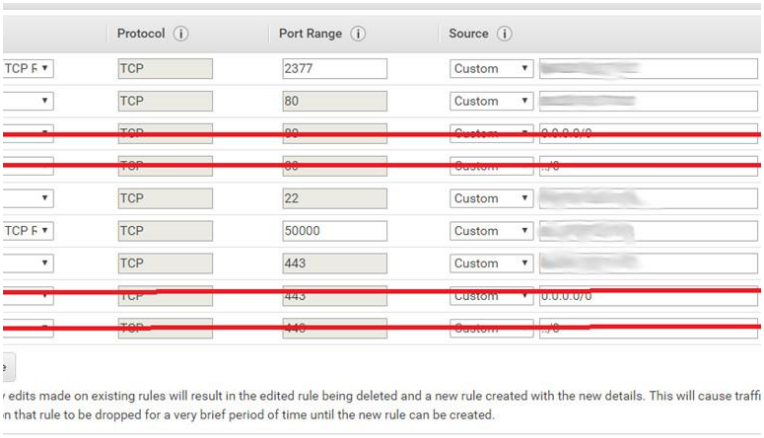


Figure 2. AWS disable inbound rules

A1.2.3.Back-Up

A1.2.3.a Main Instance.

[1] Push user Registry Image:

```
$ $(aws ecr get-login --no-include-email)
$ cd ci-containersEnviroment/private-user-registry
$ docker ps #get the id of the private-user-registry container
$ docker commit <private-user-registry_id> 842800759158.dkr.ecr.eu-west-
1.amazonaws.com/elastest/private-user-registry:AAAAMMDD
$ Docker push 842800759158.dkr.ecr.eu-west-
1.amazonaws.com/elastest/private-user-registry:AAAAMMDD
```

[2] Create Snapshot (Figure 3. AWS EC2. Create Image)

- Select instance: elastest-ci
- AMI Backup -> Image / Create Image. (Figure 4. AWS EC2. Configuration of the Image)
  - **name:** elastestci_AAAAMMDD
  - **description:** AAAAMMDD_Maintenance_Window

[3] Wait until Image status is: available. (Figure 5. AWS EC2. Available Image)

A1.2.3.b Slaves.

No actions required

A1.2.3.c ElasTest K8s Nightly.

The process must be done to all EC2 involved in the cluster, both the master and the node.

[1] Create Snapshot: (Figure 3. AWS EC2. Create Image)

- Select instance: Nightly-K8s-Master
- AMI Backup -> Image / Create Image. (Figure 4. AWS EC2. Configuration of the Image)
  - **name:** nigthly_K8_master_AAAAMMDD

- **description:** AAAAMMDD_Maintenance_Window

Wait until Image status is: available. (Figure 5. AWS EC2. Available Image)

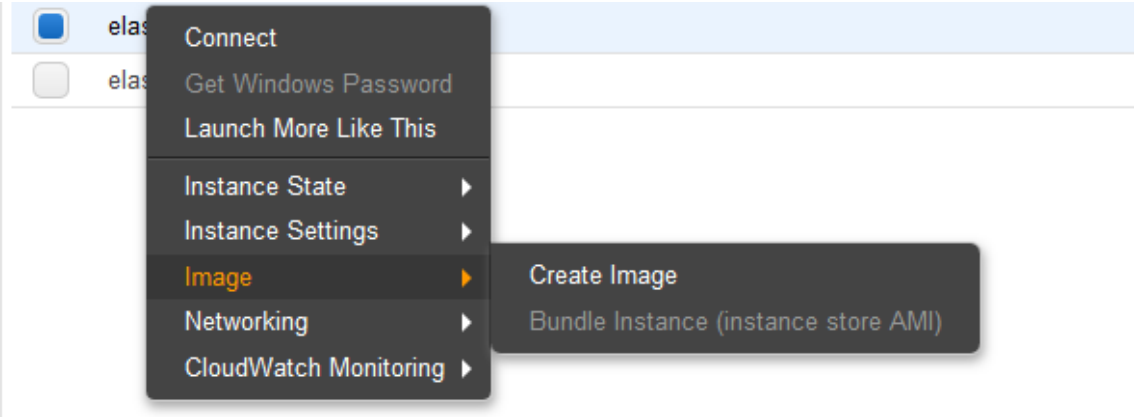


Figure 3. AWS EC2. Create Image

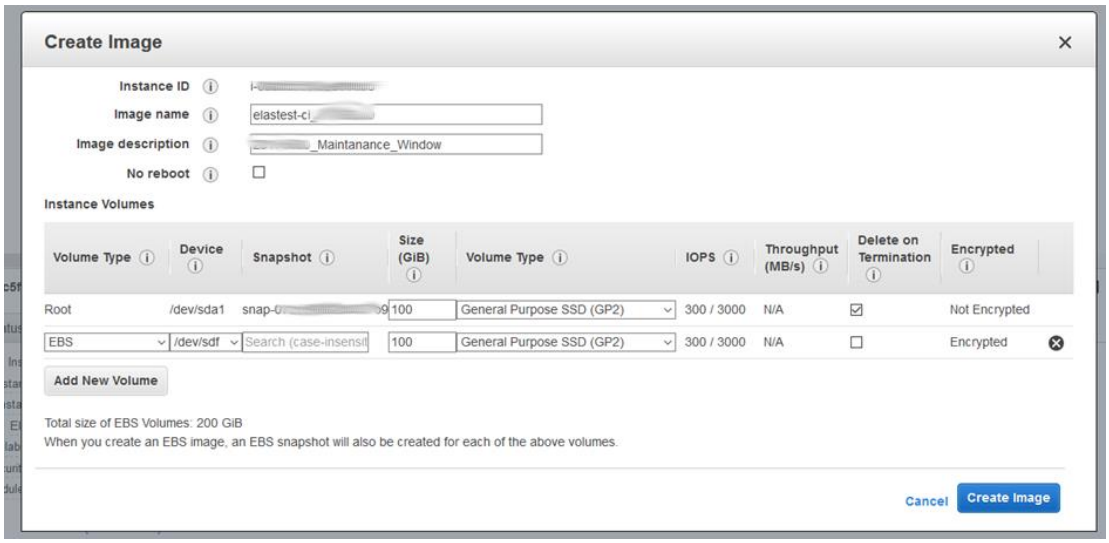


Figure 4. AWS EC2. Configuration of the Image

Figure 5. AWS EC2. Available Image

[2] Create Snapshot: (as sawn in Figure 3. AWS EC2. Create Image)

- Select instance: Nightly-K8s-Slave
- AMI Backup -> Image / Create Image. (Figure 4. AWS EC2. Configuration of the Image)
  - **name:** nigthly_slave_AAAAMMDD
  - **description:** AAAAMMDD_Maintenance_Window

Wait until Image status is: available. (Figure 5. AWS EC2. Available Image)

A1.2.4. Upgrade

A1.2.4.a Main Instance.

- **Kernel (5min):**
  - Update and upgrade

```
$ sudo apt update
$ sudo apt upgrade
```

  - If unused packages:

```
$ sudo apt autoremove
```

  - Reboot

```
$ sudo reboot
```

- **Docker:**
  - All the containers will be stopped.

```
$ Docker stop $(Docker ps -a -q)
```

  - Docker Images will be cleared.

```
$ Docker rmi -f $(Docker images -q)
```

  - Docker will be upgraded in the host with:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
$ sudo apt update
$ sudo apt-get install docker-ce=<DOCKER_NEW_VERSION>
```

- **Docker Compose:**
  - Run this command to download the latest version of Docker Compose:

```
$ sudo curl -L
https://github.com/docker/compose/releases/download/<docker-
compose_NEW_VERSION>/docker-compose-`uname -s`-`uname -m` -o
/usr/local/bin/Docker-compose
```

  - Apply executable permissions to the binary:

```
$ sudo chmod +x /usr/local/bin/Docker-compose
```

  - Test the installation.

```
$ Docker-compose --version
```

- **AWS cli**
  - uninstall old version

```
$ sudo apt-get remove awscli
```

  - install new version

```
$ sudo curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o
"awscli-bundle.zip" &&  sudo  unzip awscli-bundle.zip
&&  sudo   ./awscli-bundle/install -i /var/lib/aws -b /usr/bin/aws
```

- **User-Registry**
  - Log in in aws ecr:

```
$ $(aws ecr get-login --no-include-email)
```

- Modify docker-compose.yml to retrieve backup instead of clean image.
- Start container:

```
$ Docker-compose up -d
```

- **Jenkins: <JENKINS_NEW_VERSION>**
  - Retrieve Dockerfile and setup from GitHub

```
$ git pull
```

  - remove related containers that could be stuck

```
$ Docker-compose rm
```

  - Start and build containers:

```
$ . ./env/generate_docker_env.sh
$ Docker-compose up --build -d
```

  - Plugins and jobs will be upgraded (after nginx start).

- **Nexus**
  - remove related containers that could be stuck

```
$ Docker-compose rm
```

  - The nexus Image will be built and started

```
$ Docker-compose up --build -d
```

- **Nginx:**
  - remove related containers that could be stuck

```
$ Docker-compose rm
```

  - Nginx Image will be upgraded to <NGINX_NEW_VERSION> and the container rebuilt and restarted

- **Jenkins plugins and jobs**
  - Update all Jenkins plugins

A1.2.4.b Slaves.

- **Launch Slaves AMI:**
  - Select launch instance: elastest-slave-basic-AMI
  - Apply the steps 2- 5 into the instance

- **Kernel (5min):**
  - Update and upgrade

```
$ sudo apt update
$ sudo apt upgrade
```

  - If unused packages:

```
$ sudo apt autoremove
```

  - Reboot

```
$ sudo reboot
```

- **Docker:**
  - All the containers will be stopped.

```
$ Docker stop $(Docker ps -a -q)
```

  - Docker Images will be cleared.

```
$ Docker rmi -f $(Docker images -q)
```

  - Docker will be upgraded in the host with:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
$ sudo apt update
$ sudo apt-get install docker-ce=<DOCKER_NEW_VERSION>
```

- **Docker Compose:**
  - Run this command to download the latest version of Docker Compose:

```
$ sudo curl -L
https://github.com/docker/compose/releases/download/<docker-
compose_NEW_VERSION>/docker-compose-`uname -s`-`uname -m` -o
/usr/local/bin/Docker-compose
```

  - Apply executable permissions to the binary:

```
$ sudo chmod +x /usr/local/bin/Docker-compose
```

  - Test the installation.

```
$ Docker-compose --version
```

- **AWS cli**
  - uninstall old version

```
$ sudo apt-get remove awscli
```

  - install new version

```
$ sudo curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o
"awscli-bundle.zip" &&  sudo  unzip awscli-bundle.zip
&&  sudo   ./awscli-bundle/install -i /var/lib/aws -b /usr/bin/aws
```

- **AMI creation:**
  - Select instance: elastest-slave-basic-AMI : (Figure 3. AWS EC2. Create Image)
  - Image / Create Image. (Figure 4. AWS EC2. Configuration of the Image)
    - **name:** elastest-slave-basic-AMI-v<new_version>
    - **description:** AAAAMMDD_Maintenance_Window

- **Jenkins**
  - In the Jenkins substitute old AMI with new AMI

A1.2.4.c ElasTest K8s Nightly Master.

The process must be done for the master and the nodes

- **Kernel (5min):**
  - Update and upgrade

```
$ sudo apt update
$ sudo apt upgrade
```

- If unused packages:

```
$ sudo apt autoremove
```

- Reboot

```
$ sudo reboot
```

- **Docker:**
  - All the containers will be stopped.

```
$ Docker stop $(Docker ps -a -q)
```

  - Docker Images will be cleared.

```
$ Docker rmi -f $(Docker images -q)
```

  - Docker will be upgraded in the host with:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
$ sudo apt update
$ sudo apt-get install docker-ce=<DOCKER_NEW_VERSION>
```

- **Docker Compose:**
  - Run this command to download the latest version of Docker Compose:

```
$ sudo curl -L
https://github.com/docker/compose/releases/download/<docker-
compose_NEW_VERSION>/docker-compose-`uname -s`-`uname -m` -o
/usr/local/bin/Docker-compose
```

  - Apply executable permissions to the binary:

```
$ sudo chmod +x /usr/local/bin/Docker-compose
```

  - Test the installation.

```
$ Docker-compose --version
```

- **AWS cli**
  - uninstall old version

```
$ sudo apt-get remove awscli
```

  - install new version

```
$ sudo curl "https://s3.amazonaws.com/aws-cli/awscli-
bundle.zip" -o "awscli-bundle.zip" &&  sudo unzip awscli-
bundle.zip &&  sudo   ./awscli-bundle/install -i /var/lib/aws
-b /usr/bin/aws
```

- **Kubernetes - kubeadm:**

Only one version can be upgraded at a time. Is possible to upgrade from 1.a to 1.b, and from 1.a.x to 1.a.(x+1), but no to 1.a.(x+2). Must be done step by step

  - Find latest version

```
$apt update
```

```
$apt-cache policy kubeadm
```

- Run this command to check whether the availability of a new version of Kubernetes:

```
# replace x in 1.16.x-00 with the latest patch version
$sudo apt-mark unhold kubeadm && \
$sudo apt-get update && apt-get install -y kubeadm=1.16.x-00
&& \
$sudo apt-mark hold kubeadm
```

- Verify that the download works and has the expected version:

```
$sudo kubeadm version
```

- Drain the control plane node (this cordons the master node):

```
$ sudo kubectl drain $MASTER --ignore-daemonsets
```

- On the control plane node, run:

```
$ sudo sudo kubeadm upgrade plan --ignore-preflight-errors
ControlPlaneNodesReady
```

- If exit on this step is like this, go on with the upgrade. Otherwise, the system cannot be upgraded and has to be migrated to a new fresh Kubernetes installation:

```
…


kubeadm upgrade apply v1.16.0
```

Note: `kubeadm upgrade` also automatically renews the certificates that it manages on this node. To opt-out of certificate renewal the flag `--certificate-renewal=false` can be used.

- Choose a version to upgrade to, and run the appropriate command. (Replace `x` with the patch version you picked for this upgrade):

```
$sudo kubeadm upgrade apply v1.16.x
```

- Test the installation.

```
$ sudo kubeadm version
```

- Apply upgrades to Container Network Interface (flannel)

```
$sudo kubectl apply -f
https://github.com/coreos/flannel/blob/master/Documentation/ku
be-flannel.yml
```

- Uncordon the control plane node

```
kubectl uncordon $MASTER
```

- **Kubernetes - kubectl:**

  - Upgrade the kubelet and kubectl on all control plane nodes

```
$sudo apt-mark unhold kubelet kubectl && \
$sudo apt-get update && apt-get install -y kubelet=1.16.x-00
kubectl=1.16.x-00 && \
$sudo apt-mark hold kubelet kubectl
```

  - Restart the kubelet

```
$sudo systemctl restart kubelet
```

A1.2.4.d ElasTest K8s Nightly Node(s).

The nodes go through a similar process

- **Kernel (5min):**
  - Update and upgrade

```
$ sudo apt update
$ sudo apt upgrade
```

  - If unused packages:

```
$ sudo apt autoremove
```

  - Reboot

```
$ sudo reboot
```

- **Docker:**
  - All the containers will be stopped.

```
$ Docker stop $(Docker ps -a -q)
```

  - Docker Images will be cleared.

```
$ Docker rmi -f $(Docker images -q)
```

  - Docker will be upgraded in the host with:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
$ sudo apt update
$ sudo apt-get install docker-ce=<DOCKER_NEW_VERSION>
```

- **Docker Compose:**
  - Run this command to download the latest version of Docker Compose:

```
$ sudo curl -L
https://github.com/docker/compose/releases/download/<docker-
compose_NEW_VERSION>/docker-compose-`uname -s`-`uname -m` -o
/usr/local/bin/Docker-compose
```

- Apply executable permissions to the binary:

```
$ sudo chmod +x /usr/local/bin/Docker-compose
```

- Test the installation.

```
$ Docker-compose --version
```

- **AWS cli**
  - uninstall old version

```
$ sudo apt-get remove awscli
```

  - install new version

```
$ sudo curl "https://s3.amazonaws.com/aws-cli/awscli-
bundle.zip" -o "awscli-bundle.zip" &&  sudo unzip awscli-
bundle.zip &&  sudo   ./awscli-bundle/install -i /var/lib/aws
-b /usr/bin/aws
```

- **Kubernetes - kubeadm:**

Only one version can be upgraded at a time. Is possible to upgrade from 1.a to 1.b, and from 1.a.x to 1.a.(x+1), but no to 1.a.(x+2). Must be done step by step

  - Find latest version

```
$apt update
```

  - Run this command to check whether the availability of a new version of Kubernetes:

```
# replace x in 1.16.x-00 with the latest patch version
$sudo apt-mark unhold kubeadm && \
$sudo apt-get update && apt-get install -y kubeadm=1.16.x-00
&& \
$sudo apt-mark hold kubeadm
```

  - Verify that the download works and has the expected version:

```
$sudo kubeadm version
```

  - Drain the control plane node (this cordons the node):

```
$ sudo kubectl drain $NODE --ignore-daemonsets
```

  - Upgrade the kubelet configuration:

```
$ sudo kubeadm upgrade node
```

  - Test the installation.

```
$ sudo kubeadm version
```

- **Kubernetes - kubectl:**

  - Upgrade the kubelet and kubectl the nodes

```
$sudo apt-mark unhold kubelet kubectl && \
$sudo apt-get update && apt-get install -y kubelet=1.16.x-00
kubectl=1.16.x-00 && \
$sudo apt-mark hold kubelet kubectl
```

  - Restart the kubelet

```
$sudo systemctl restart kubelet
```

  - Uncordon the node

```
kubectl uncordon $NODE
```

  - Test the installation.

```
$ sudo kubectl version
```

A1.2.5. Test and Confirmation

[Test-Jenkins-01] Login in Jenkins
[Test-Jenkins-02] Run basic jobs.
[Test-Jenkins-02-01] Run job hello-world/mvn-hello-world
[Test-Jenkins-02-02] Run job hello-world/hello-world-Docker-image-pipeline
[Test-Jenkins-03] Plugins?

[Test-Nexus-01] Web interface (Login and query)
[Test-Nexus-02] Publish artifact: run Jenkins job: hello-world/private-mvn-release
[Test-Nexus-03] Retrieve artifact.

[Test-UserRegistry-01] Log In
[Test-UserRegistry-02] Regenerate access.

[Test-DockerSibling-01] Run job hello-world/hello-world-Docker-image-pipeline.
[Test-DockerSibling-02] Run job hello-world/pipeline-Docker-privateRegistry

[Test-AMI-01] After AMI update test reboot and Docker TCP ports (if not check Docker tcp procedure)

A1.2.6.Roll Back

A1.2.6.a Main Instance.

- The instance of the AWS EC2 will be switched off. (elastest-ci)
- A new instance of the AWS EC2 will be launched with the backed-up AMI with the same configuration and IP as the old one.
- Elastic IP will be assigned to the rolled back instance.
- The Docker Images will be rolled back (the Dockerfile recovered and the images recreated with the old values)

A1.2.6.b Slaves.

- New image wouldn't be saved so no extra actions required

A1.2.6.c ElasTest K8s Nightly Master:

- The instance of the Master on AWS EC2 will be switched off. (Nightly-K8s-Master)
- A new instance of the AWS EC2 will be launched with the backed-up AMI with the same configuration and IP as the old one.
- Elastic IP (named Nightly-k8s-Master) will be assigned to the rolled back instance.

A1.2.6.d ElasTest K8s Nightly Node(s):

- The instance of the Node on AWS EC2 will be switched off. (Nightly-K8s-Slave)
- A new instance of the AWS EC2 will be launched with the backed-up AMI with the same configuration and IP as the old one.
- Elastic IP (named Nightly-k8s-Slave) will be assigned to the rolled back instance.

A1.2.7. Open System and Result Notification.

- The instances will be configured to accept external requests

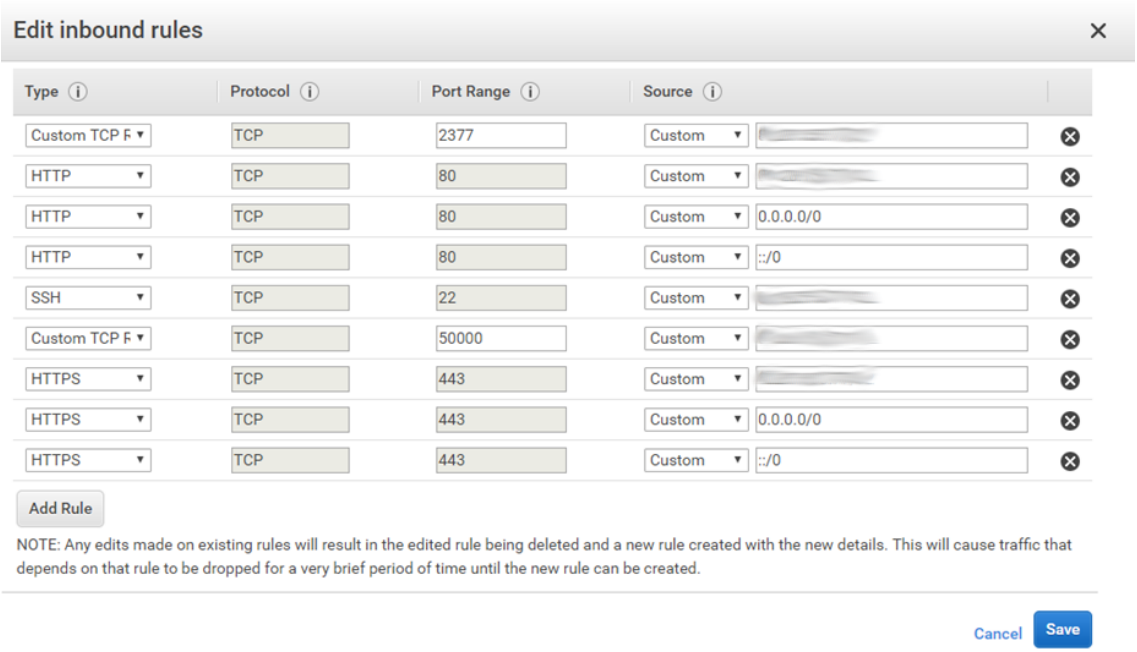


Figure 6. AWS enable inbound rules

A1.3. Results

A1.3.1.Table of results

Phase	Result	Time and duration
Back UP	SUCCESS / FAILURE / WARN	
Upgrade	SUCCESS / FAILURE / WARN	
Test and Confirmation	SUCCESS / FAILURE / WARN	
Rollback	NOT RUN / SUCCESS / FAILURE / WARN	

A1.3.2. Actions to be executed after upgrade

A1.3.2.a Main Instance

A1.3.2.b Slaves

A1.3.2.c ElasTest Nightly

A1.4. Logs

<if applies>

A1.5. Issues

Any issue that is detected and is suspected to be related to the upgrade should be registered here.