decide

MULTICLOUD APPLICATIONS TOWARDS
THE DIGITAL SINGLE MARKET

Deliverable D4.7

Initial multi-cloud application monitoring

Editor(s):	Juncal Alonso
Responsible Partner:	TECNALIA
Status-Version:	Final – v1.0
Date:	30/11/2017
Distribution level (CO, PU):	CO

Project Number:	GA 726755
Project Title:	DECIDE

Title of Deliverable:	Initial multi-cloud application monitoring
Due Date of Delivery to the EC:	30/11/2017

Workpackage responsible for the Deliverable:	WP4 – Continuous deployment and operation
Editor(s):	TECNALIA
Contributor(s):	Juncal Alonso, Gorka Benguria, Marisa Escalante, Maria Jose Lopez, Iñaki Etxaniz (TECNALIA), Luis Miguel, Javier Gavilanes, Gema Maestro (Experis)
Reviewer(s):	HPE
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3, WP5, WP6.

Abstract:	This deliverable describes the initial version of the technical details of the engine implemented for continuously monitoring multi-cloud applications enumerating the off-shelf software products selected for the implementation, how they are integrated and the implemented interfaces.
Keyword List:	Monitoring, working conditions, micro-service, multi-cloud application.
Licensing information:	The software is released under MIT license. The document itself is delivered as a description for the European Commission about the released software, so it is not public.
Disclaimer	This deliverable reflects only the author's views and views and the Commission is not responsible for any use that may be made of the information contained therein

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	08/10/2017	First draft version.	TECNALIA
v0.2	10/10/2017	Content included in section 2.1.1, 2.1.2.1, 2.1.2.2.	TECNALIA
V0.3	26/10/2017	Content updated in section 2.1.1 in accordance to D4.4, inclusion of pictures and references. Content included in section 2.1.2.3. Content included in section 2.2 Conclusions included	TECNALIA
V0.4	31/10/2017	Content updated in section 2.2.2	TECNALIA
V0.5	08/11/2017	Content updated in section 2.1.1 (metrics detail). Content updated in section 2.1.2 (Sock Shop App details) Content updated in section 2.2.2 Content updated in section 2.2.3	TECNALIA
V0.6	9/11/2017	Content updated in section 2.2.3	TECNALIA
V0.7	20/11/2017	Content updated in section 2.1.2.1 Explanation about Sock Shop app deployment added in section 2.2.2.	TECNALIA
V0.8	22/11/2017	Content included in section 3	Experis
V0.9	24/11/2017	Review comments addressed	TECNALIA
V1.0	24/11/2017	Final version without comments	TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

D	Deliverable
EC	European Commission
F	Functionality
M12	Month twelve
M24	Month twent- four
M30	Month thirty
MTBF	Mean Time Between Failures
MTTR	Mean Time to Recovery
NFR	Non-Functional Requirement
SRC	Source Code

Executive Summary

This document describes the two components, Adapt Monitoring and Violations Handler forming the current version of the Initial multi-cloud application monitoring prototype, which monitors the deployed multi-cloud application in order to verify the working conditions and manages the actions to be performed when a violation occurs.

The document presents the missions, scopes, functional descriptions, technical approaches, download & installation instructions and user manuals of these components comprising the prototype.

This document will be updated in each release of the multi-cloud application monitoring component, that is in M24 and M30, including the new functionalities implemented in each case.

1 Introduction

1.1 About this deliverable

This document is the complement to the delivered software as prototype in the specified date and deliverable named at the head of the document.

1.2 Document structure

This document is divided into two main sections describing two components of the prototype. Architectural and implementation details of the components and how to use and download them are described in detail in the following sections, section 2 (Adapt Monitoring) and section 3 (Violations Handlers).

2 ADAPT Monitoring

2.1 Implementation

2.1.1 Functional description

DECIDE ADAPT monitoring components monitor the deployed multi-cloud based application and verify that the non-functional requirements and the SLOs are being fulfilled. If a violation of any of the NFRs or SLOs is detected, ADAPT monitoring components will inform the violation handlers (described in section 3) component which will generate the proper actions depending on each situation and context. If the violation occurs, information saying that the working conditions are not met will be sent to the operator. If the application is low technology risk, the "adaptation" process will be launched, through the violation handler component.

The main functionalities of the ADAPT monitoring components are:

F1. Collect data from the deployed multi-cloud application and the underlying cloud resources: The ADAPT monitoring component needs to get the data from the deployed components and their NFRs as well as from the underlying cloud services to collect data from their service level metrics at real time. The data related to the cloud services will be collected from ACSmI. The NFRs to be covered in the context of DECIDE project are [1]: Performance, Availability, Scalability, Security /Legal, Cost, Location, and State.

F2. Store the data collected from the deployed multi-cloud application and the underlying cloud resources: To assess the required working conditions of the multi-cloud application ADAPT monitoring will deal with data at real time. These time series will be stored for further analysis of the data.

F3. Create the aggregated data to be assessed: from the raw metrics, the ADAPT monitoring component will need to create aggregated data for assessing the violations.

F4. Visualize the measurements: DECIDE ADAPT monitoring will provide the user with an interface to visualize the monitored data, from the multi-cloud application and from the underlying cloud services. DECIDE ADAPT monitoring will also provide monitoring information of the violation occurred.

F5. Detect when a violation occurs: DECIDE ADAPT monitoring will detect violations on the working conditions (NFRs/SLOs), send the corresponding alert to the operator and launch the "adaptation" process (through the Violation Handler). The adaptation process launched will include the actions related to stop the monitoring of the previous components and resources.

F6. Inform about a violation to the Violation Handlers component: ADAPT monitoring will inform to the Violation Handlers component that a violation on the working conditions has occurred.

F7. Create new measurements to be collected: Apart from the metrics defined in the context of DECIDE project (see F1). ADAPT monitoring will provide means to declare new metrics to be monitored.

The existing approaches for application monitoring [1] do not tackle the monitoring of applications that monitor different elements of the application (composed of components) and which are deployed into disperse cloud resources. They face the problem monitoring single components, or types of components (i.e. software applications). In DECIDE ADAPT monitoring, the working conditions of a multi-cloud application with respect to specific NFRs (but with an extendable approach) will be supported and addressed at different levels [2] [3] (software, resource) and combining them at run time.

ADAPT Monitoring will be implemented following an incremental approach adding features in the different releases of the tool (D4.8 in M24 and D4.9 in M30). In this first release the following functionalities are implemented, using an example application (Socks Shop, see section 2.1.2.1) as a target: **F1**, **F2, F4, F6 and F7.**

The following table 1 details the relationship between the deployment requirements indicated in deliverable D4.1 and the implemented functionalities, with a description of the coverage for each functionality.

Requirements covered by the prototype:

Table 1. Functionality covered by the M12 ADAPT monitoring prototype.

Functionality	Req. ID	Coverage
F1	WP4-REQ4, WP4-REQ18, WP2-DEVOPS-REQ3.	The component prototype measures the metrics related to one of the NFRs selected in the project (availability), for each of the components of the application (sock shop application).
F2	WP4-REQ4, DEVOPS-REQ3	The component prototype stores the metrics in a time series data base so they can be visualized by the operator in the user dashboard.
F3	DEVOPS-REQ3, WP4-REQ4, WP4-REQ18	None
F4	WP4-REQ4, WP4-REQ27, DEVOPS-REQ3.	The current prototype provides a dashboard for the operator of the application to assess the metrics that are being monitored.
F5	WP4-REQ4, WP4-REQ18, DEVOPS-REQ3.	None
F6	DEVOPS-REQ9, DEVOPS-REQ43, WP4-REQ29, WP4-REQ19,	The current prototype is able to detect errors occurred in the working conditions with respect to availability.
F7	WP4-REQ41	The selected technologies provide the means to create new metrics to be monitored as they are needed

Functionality that this prototype offers:

The delivered prototype is the first version of the ADAPT Monitoring component and contains the following functionality:

- Get availability/performance related metrics: For this first version, ADAPT monitoring will be focused on getting measurements related to availability and/or performance of the components of the micro-services of the multi cloud application Shock Shop (see section 2.1.2.1 for details). Availability/performance of the multi-cloud application will be impacted by the metrics related to the availability/performance of the software component (or micro-service) itself and by the metrics related to the availability/performance of the underlying (cloud) resources where the application is deployed. In this prototype, ADAPT monitoring will monitor only the metrics related to the availability/performance of the software components themselves, without considering the information of the availability/performance related to the different (cloud) resources where the components are deployed. This information will come (in future versions) from the ACSmI [4].

The metrics gathered to be monitored for Availability/Performance for this prototype are:

- http_response_code (int): The code received from the http request.
- result_type (string): The possible results types are success, timeout, response_string_mismatch, connection_failed.
- Response time: Component response time, including:
    - Average response time in s.
    - Minimum response time in s.
    - Maximum response time in s.
    - Current response time in s.

These metrics are monitored per Multi-cloud app component (see section 2.1.2.1 for Sock Shop app details). In the next versions of the prototypes the aggregation of these metrics will be monitored in order to set up the final Availability [2] metric of the Multi cloud application:

- MTBF: Mean time between failures.
- MTTR Mean time to recover.

- Stores the availability/performance metrics in a time series DB: The measurements gathered every second, are stored in a DB, for each component.
- Shows the availability/performance metrics in a graphical dashboard: The dashboard designed includes the following graphics:
    - Graphic 1: Mean response time per app component (it includes the current, maximum and minimum values too).
    - Graphic 2: Generic relevant info of the multi-cloud app including:
        - Sock shop application generic info
        - Number of components in the application being monitored
    - Graphic 3: Maximum response time in the application (considering the different components).
    - Graphic 4/5/6/7: Maximum response time in the components (considering metrics in the last hour).
    - Graphic 8: The type of the response received, per component (each 10s). The possible types of response message are: ssuccess, timeout, response string mismatch, connection failed.

2.1.1.1 *Fitting into overall DECIDE Architecture*

ADAPT monitoring is one of the final steps in the DECIDE workflow [5]. It supports the operation phase of multi-cloud aware applications by providing means for run-time monitoring of the current deployment with respect to selected non-functional requirements and SLOs and re-deployment adaptation when needed. The Adapt Monitoring is part of the ADAPT key result of the DECIDE architecture, indeed it is one of the components inside this package (see figure 1).

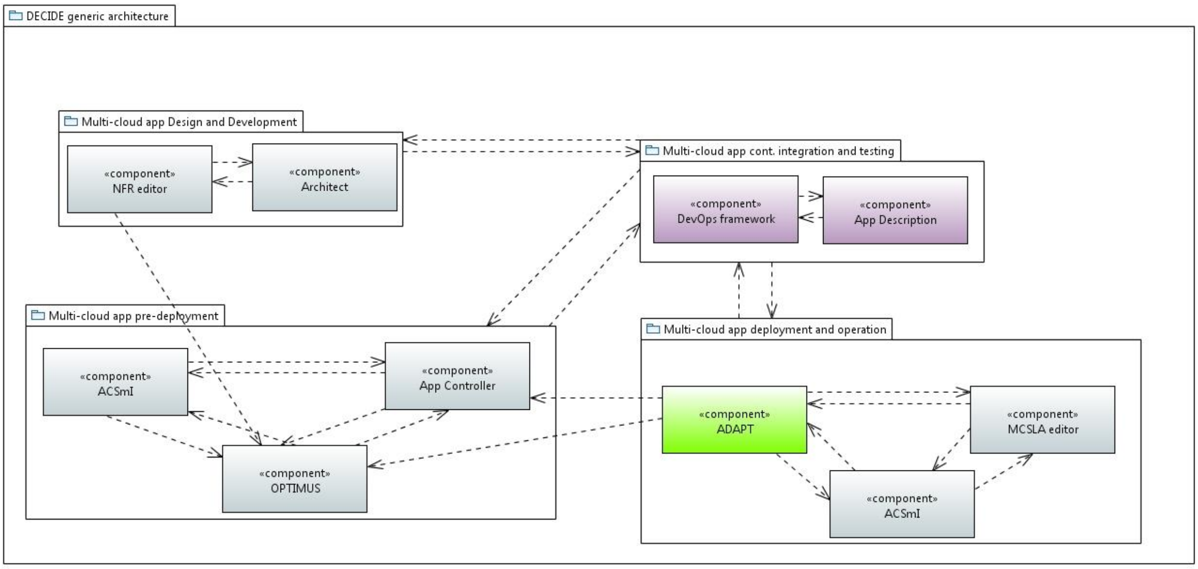

DECIDE generic architecture
Multi-cloud app Design and Development
«component» NFR editor
«component» Architect
Multi-cloud app cont. integration and testing
«component» DevOps framework
«component» App Description
Multi-cloud app pre-deployment
«component» ACSmI
«component» App Controller
«component» OPTIMUS
Multi-cloud app deployment and operation
«component» ADAPT
«component» MCSLA editor
«component» ACSmI

Figure 1. ADAPT monitoring in DECIDE architecture**.**

ADAPT Monitoring interacts with the two components of ADAPT (Violation Handler and ADAPT deployment orchestrator) as well as with other tools in the DECIDE ecosystem [5]. ADAPT monitoring will communicate on one hand with the MCSLA editor/ App description for gathering the threshold values for the different metrics to be assessed and on the other hand, with the Violation Handlers to provide information about any violation. The ADAPT deployment orchestrator will send the requests to start/stop the monitoring and the ACSmI will provide the monitoring metrics coming from the Cloud Service Providers.

In the following picture, the general architecture of the ADAPT monitoring components is shown, for detailed description check D4.1 [6]:

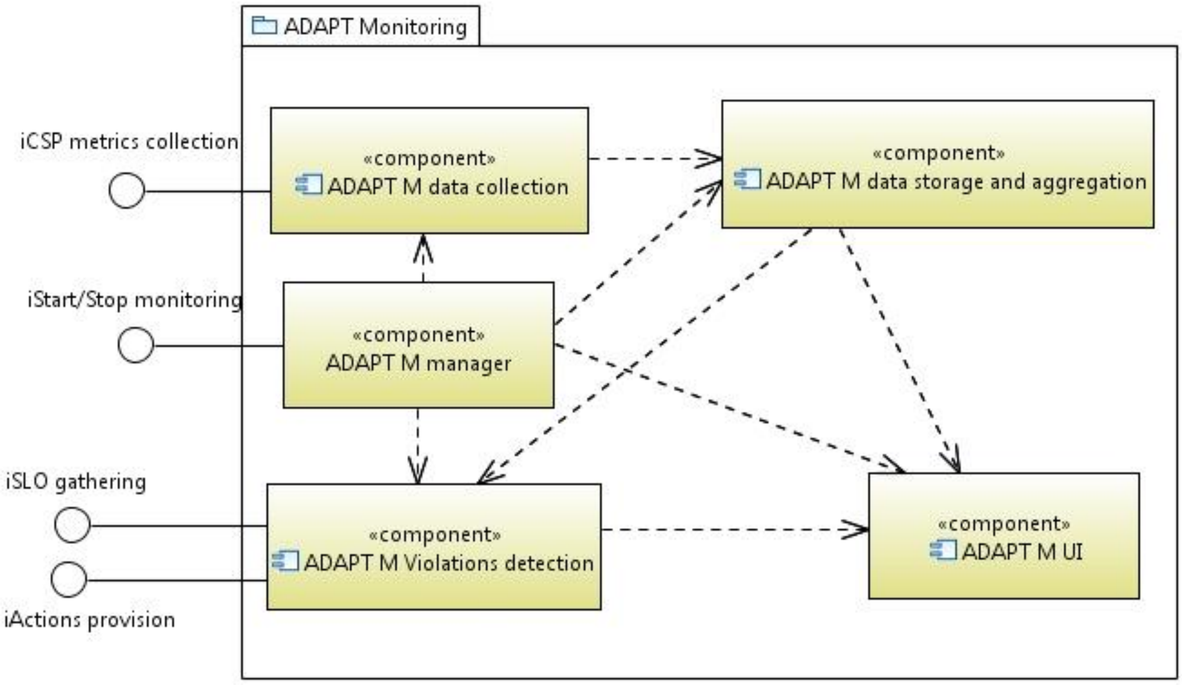

ADAPT Monitoring
iCSP metrics collection
iStart/Stop monitoring
iSLO gathering
iActions provision
«component» ADAPT M data collection
«component» ADAPT M data storage and aggregation
«component» ADAPT M manager
«component» ADAPT M Violations detection
«component» ADAPT M UI

Figure 2. General architecture of ADAPT monitoring.

2.1.2 Technical description

This section describes the technical details of the implemented software for the current prototype of ADAPT monitoring.

2.1.2.1 *Prototype architecture*

The current prototype (M12) of ADAPT Monitoring architecture consists of two tiers:

- Server side: Two Docker containers deployed in a virtual machine, containing the ADAPT M data storage and aggregation and the ADAPT M UI, and the ADAPT M Data collection deployed. The ADAPT M Data collection agents get the selected measurements from the different components of the Sock Shop application.
- Local side: A virtual machine with the Sock Shop application[1] deployed into several Docker containers (each microservice deployed into one docker container). For this prototype and from the "components" architecture for the sock shop application (based on the Docker Compose instance of the application [7]) we have considered the following components to be monitored:
    - Component 1: front-end
    - Component 2: catalogue (the associated *catalogue-db* is not being monitored)
    - Component 3: carts (the associated *carts-db* is not being monitored)
    - Component 4:orders (the associated *orders-db* is not being monitored)

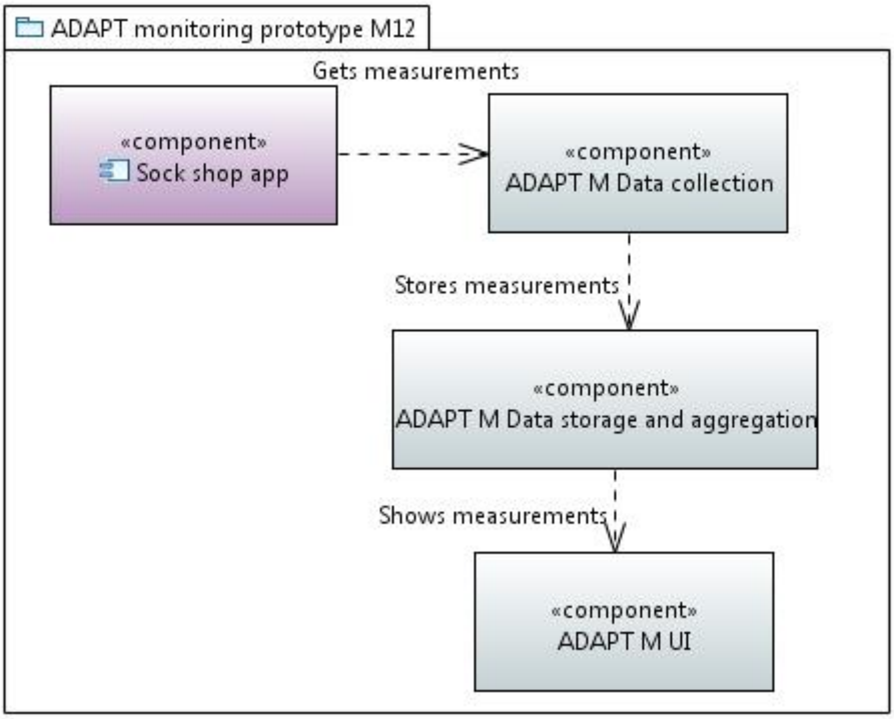


Figure 3. ADAPT Monitoring M12 prototype High level architecture

2.1.2.2 *Components description*

The current prototype includes 3 of the 5 components envisioned for ADAPT monitoring [6].

---

[1] Shock Shop application is a micro-services based application used to illustrate micro-services architectures. Generic description is included in [10].

- ADAPT M Data collection: This sub-component collects the data from the Sock Shop application with respect to its working conditions. In the current prototype the Data Collection sub-component has been configured to get the availability related metrics from the different components of the Sock Shop application. The Data collection component gets these metrics and stores them in the ADAPT M Data Storage and aggregation every one second.
- ADAPT M Data storage and aggregation: This sub-component is in charge of storing the data collected from the ADAPT M Data collection and aggregating them to create the actual measures that will be assessed by the ADAPT M violation detection. In the current version of the prototype the availability related measurements are stored directly in the DB, without any computation nor aggregation.
- ADAPT M UI: This is the graphical user interface for the ADAPT monitoring component. In the current version of the prototype this graphical interface for the visualization of the availability related metrics of the different components in real time has been implemented.

2.1.2.3 Technical specifications

ADAPT monitoring prototype for M12 has been implemented as a monitoring stack, covering the data collection, the data storage and the data visualization.

- ADAPT M Data collection: For the data collection the Telegraf[2] open source technology is used. Telegraf is a plugin-driven server agent written in Go for collecting, processing, aggregating, and reporting metrics. It is a compiled and standalone binary that can be executed on any system with no need for external dependencies. For the purpose of ADAPT monitoring Telegraf plugins have been configured to acquire the following measurements:
    - Input plugins: http_response plugin, which tests HTTP/HTTPS connections to a given list of urls in this case the Sock Shop app components urls), with a given periodicity, from a set interface. This plugin gets the following metrics:
        - response_time (float, seconds)
        - http_response_code (int) (the code received)
        - result_type (string) (success, timeout, response_string_mismatch, connection_failed).
    - Output plugins: Influx DB plugin. This plugin sends the metrics gathered by the agent to a specific Influx DB instance (with a given url). The name of the database, login parameters, etc. need to be specified.
- ADAPT M Data storage and aggregation: For the storage of the metrics, Influx DB storage technology have been implemented. InfluxDB is used as a data store for cases involving large amounts of timestamped data, like monitoring applications or micro-services which is the case of ADAPT monitoring. Influx DB supports plugins for data ingestion protocols like Telegraf. The measurements are injected by the Telegraf plugins in the Influx DB database.
- ADAPT M UI: For the generation of the user interface where the metrics related to the current working conditions of the application can be monitored, Grafana is used. Grafana is an open platform to visualize data. It provides means to configure specific dashboards with different graphics. For the first version of ADAPT monitoring a basic dashboard for showing the selected metrics per component (micro-service) has been created. Grafana has been configured to read the information from the InfluxDB

  As a supporting framework, Vagrant[3] is being used to automatize the virtualized environment where to deploy ADAPT monitoring and the Sock Shop application to be monitored..

---

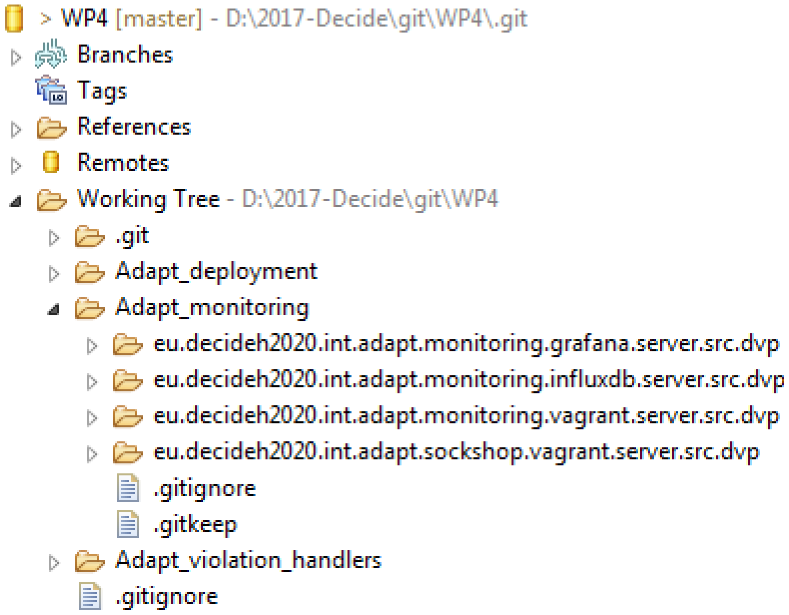
[2] https://www.influxdata.com/time-series-platform/telegraf/

[3] https://www.vagrantup.com/

2.2 Delivery and usage

2.2.1 Package information

Delivered package consists of the below folder structure.

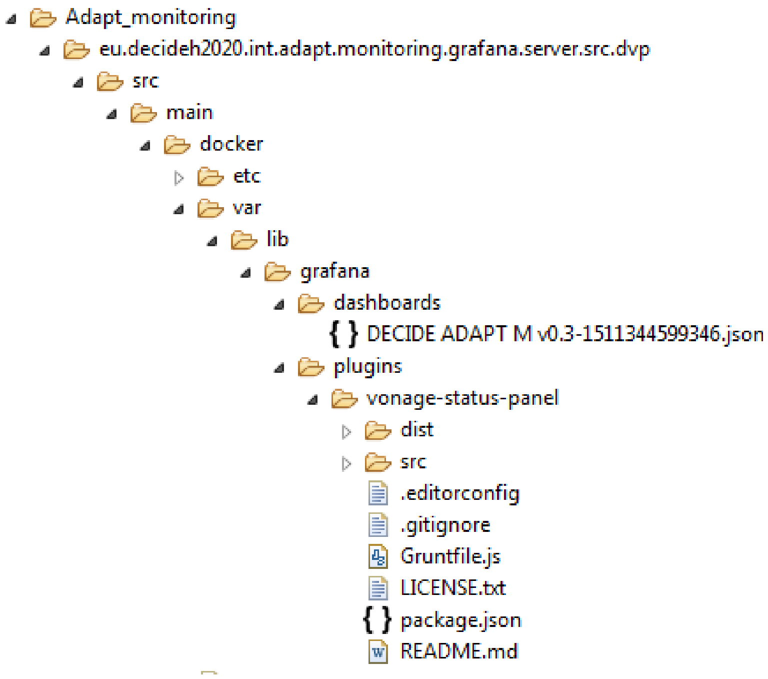


```
> WP4 [master] - D:\2017-Decide\git\WP4\.git
▷ Branches
  Tags
▷ References
▷ Remotes
◢ Working Tree - D:\2017-Decide\git\WP4
  ▷ .git
  ▷ Adapt_deployment
  ◢ Adapt_monitoring
    ▷ eu.decideh2020.int.adapt.monitoring.grafana.server.src.dvp
    ▷ eu.decideh2020.int.adapt.monitoring.influxdb.server.src.dvp
    ▷ eu.decideh2020.int.adapt.monitoring.vagrant.server.src.dvp
    ▷ eu.decideh2020.int.adapt.sockshop.vagrant.server.src.dvp
      .gitignore
      .gitkeep
  ▷ Adapt_violation_handlers
    .gitignore
```

Figure 4. Source folder structure of ADAPT Monitoring component in M12.

ADAPT monitoring folder is composed of four main sub-folders with the corresponding src:

- eu.decideh2020.int.adapt.monitoring.grafana.server.src.dvp: Contains the src for the ADAPT M UI. (see figure 5).
- eu.decideh2020.int.adapt.monitoring.influxdb.server.src.dvp: Contains the src for the ADAPT M storage and aggregation (see figure 6).
- eu.decideh2020.int.adapt.monitoring.vagrant.server.src.dvp: Contains the src for the ADAPT M data collection and the vagrant file for creating the virtualized environment where ADAPT monitoring is deployed (see Figure 7).
- eu.decideh2020.int.adapt.shockshop.vagrant.server.src.dvp: Contains the src for the Multi-cloud application (Shock Shop application) to be monitored (see Figure 8).

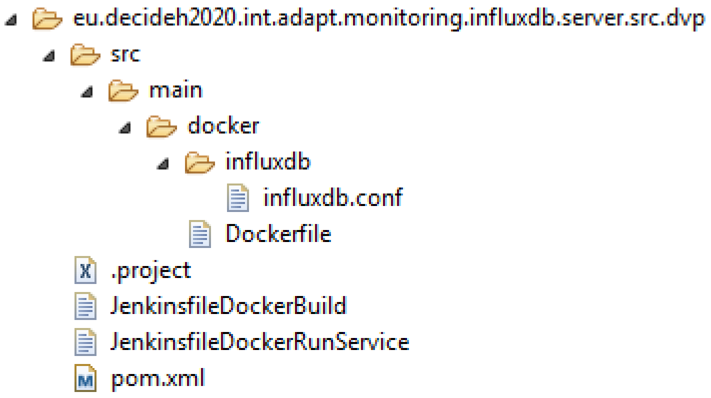


```
Adapt_monitoring
  eu.decideh2020.int.adapt.monitoring.grafana.server.src.dvp
    src
      main
        docker
          etc
          var
            lib
              grafana
                dashboards
                  DECIDE ADAPT M v0.3-1511344599346.json
                plugins
                  vonage-status-panel
                    dist
                    src
                    .editorconfig
                    .gitignore
                    Gruntfile.js
                    LICENSE.txt
                    package.json
                    README.md
```

Figure 5. Source folder structure of ADAPT M UI sub-component

Inside the eu.decideh2020.int.adapt.monitoring.grafana.server.src.dvp the following relevant folders and packages are included:

- src: Contains the source files and their packages. Here the most relevant file is:
    - DECIDE ADAPT M v0.3-1511344599346.json: Json file with the specification of the graphical Dashboard for the DECIDE M UI.

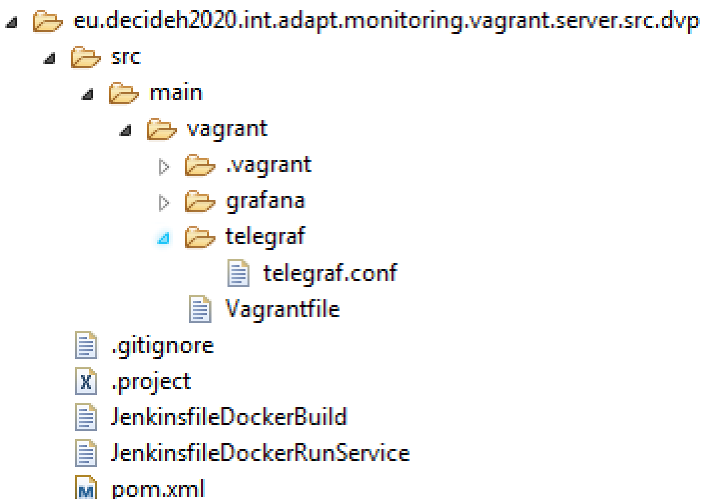


```
eu.decideh2020.int.adapt.monitoring.influxdb.server.src.dvp
  src
    main
      docker
        influxdb
          influxdb.conf
        Dockerfile
  .project
  JenkinsfileDockerBuild
  JenkinsfileDockerRunService
  pom.xml
```

Figure 6. Source folder structure of ADAPT M Data Storage and Aggregation

Inside the eu.decideh2020.int.adapt.monitoring.influxdb.server.src.dvp the following relevant folders and packages are included:

- src: Contains the source files and their packages:
    - Influxdb.conf: This file contains the configuration of the Data Base for storing the selected measurements in DECIDE ADAPT monitoring.

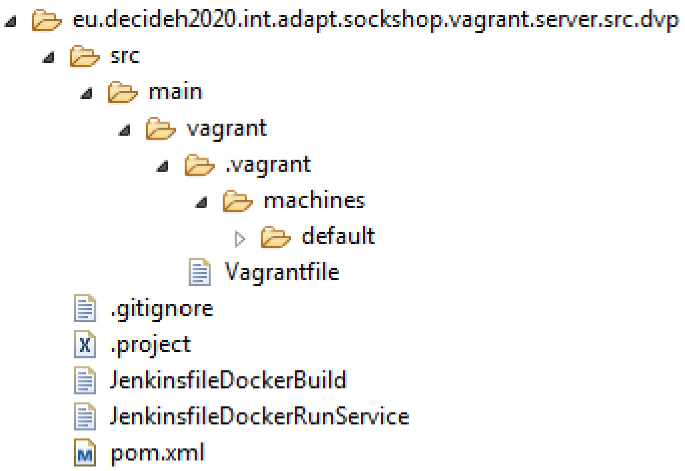


```
eu.decideh2020.int.adapt.monitoring.vagrant.server.src.dvp
    src
        main
            vagrant
                .vagrant
                grafana
                telegraf
                    telegraf.conf
                Vagrantfile
    .gitignore
    .project
    JenkinsfileDockerBuild
    JenkinsfileDockerRunService
    pom.xml
```

Figure 7. Source folder structure of ADAPT M Data Collection.

Inside the eu.decideh2020.int.adapt.monitoring.vagrant.server.src.dvp the following relevant folders and packages are included:

- src: Contains the source files and their packages:
    - telegraf.conf: This file contains the configuration of the agents where the inputs and output plugins for getting the metrics from the application are configured.
    - Vagrant file: Configuration file for facilitating the setting up of the monitoring environment (automatizing the creation of a virtual machine with the corresponding operating system and the containers server).



```
eu.decideh2020.int.adapt.sockshop.vagrant.server.src.dvp
    src
        main
            vagrant
                .vagrant
                    machines
                        default
                Vagrantfile
    .gitignore
    .project
    JenkinsfileDockerBuild
    JenkinsfileDockerRunService
    pom.xml
```

Figure 8. Source folder structure for the Shock Shop application.

Inside the eu.decideh2020.int.adapt.shockshop.vagrant.server.src.dvp the following relevant folders and packages are included:

- Vagrant file: Configuration file for facilitating the setting up of the Shock Shop application to be monitored (automatizing the creation of a virtual machine with the corresponding operating system and the containers server).

2.2.2 Installation instructions

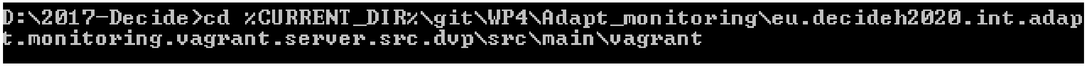
The prototype has been installed and tested using the following software (see Pre-requirements in section 2.2.2.1):

- Vagrant for creating the VM and the Docker server where to create the containers to deploy Grafana, Influx DB, Telegraph and the Sock Shop sample application.
- Virtual Box for virtualizing the local environment for the monitoring stack and for the Shock Shop application.

The following steps need to be executed to get the prototype up and running:

1. Download the src from the DECIDE repository (see section 2.2.5).

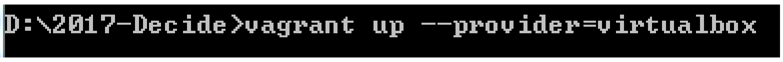
2. Switch to the folder where the vagrant file is located.



```
D:\2017-Decide>cd %CURRENT_DIR%\git\WP4\Adapt_monitoring\eu.decideh2020.int.adap
t.monitoring.vagrant.server.src.dvp\src\main\vagrant
```

Figure 9. Locating into the directory where the vagrant file is located.

3. Execute vagrant up and assign it to the Virtual Box "provider":

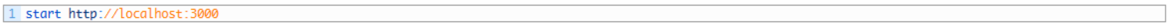


```
D:\2017-Decide>vagrant up --provider=virtualbox
```

Figure 10. Executing vagrant up and virtualizing the local environment.

Now, ADAPT monitoring is up and running.

The local environment is hosting the ADAPT monitoring in port 3000.

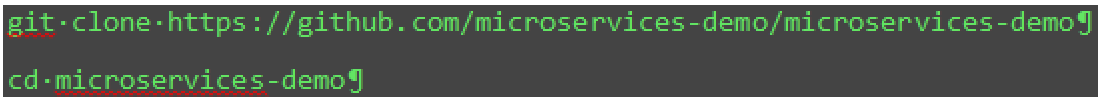


```
start http://localhost:3000
```

Figure 11. Checking that ADAPT monitoring is up and running.

The current version of the ADAPT monitoring, monitors the availability related metrics of the components of the Sock Shop application. For that reason, an instance of the Sock Shop application needs to be deployed:

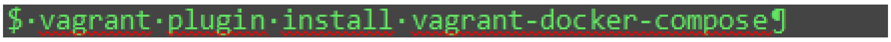
1. Download the src of the Sock Shop app.



```
git·clone·https://github.com/microservices-demo/microservices-demo¶
cd·microservices-demo¶
```

Figure 12. Download the sample application to be monitored.

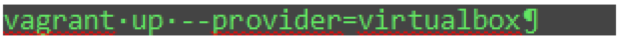
2. Install the Vagrant plugin for the Docker compose.



```
$·vagrant·plugin·install·vagrant-docker-compose¶
```

Figure 13. Install Docker compose plugin in vagrant.

3. Execute vagrant up and assign it to the Virtual Box "provider":



```
vagrant·up·--provider=virtualbox¶
```

Figure 14. Executing vagrant up and assigning it to the Virtual Box

Now the Sock Shop application is up and running.

The local environment is hosting the Sock Shop application in port 80.

2.2.2.1 Pre-Requirements

The following lists the minimum requirements to get the component working.

- Virtual Box: For Virtualizing the local environment.
- Vagrant: For creating the virtual box from one instruction. It has been configured to create the virtual machine and install the O.S (Ubuntu Xenial 16.04) and the Docker server.
- JRE

2.2.3 User Manual

Once ADAPT monitoring is installed it can be used to monitor the Sock Shop application components.

First the user needs to open the browser and visit the following URL : http://localhost:3000

Then, login to Grafana:

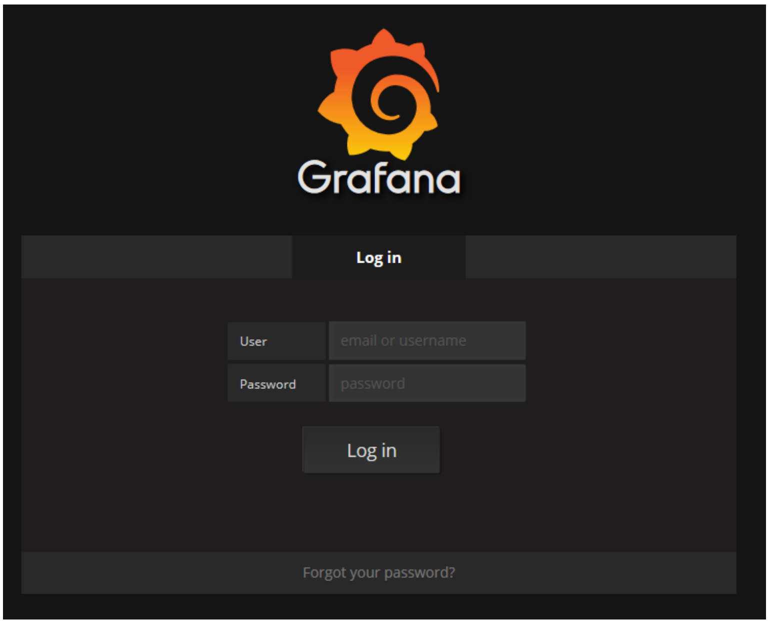
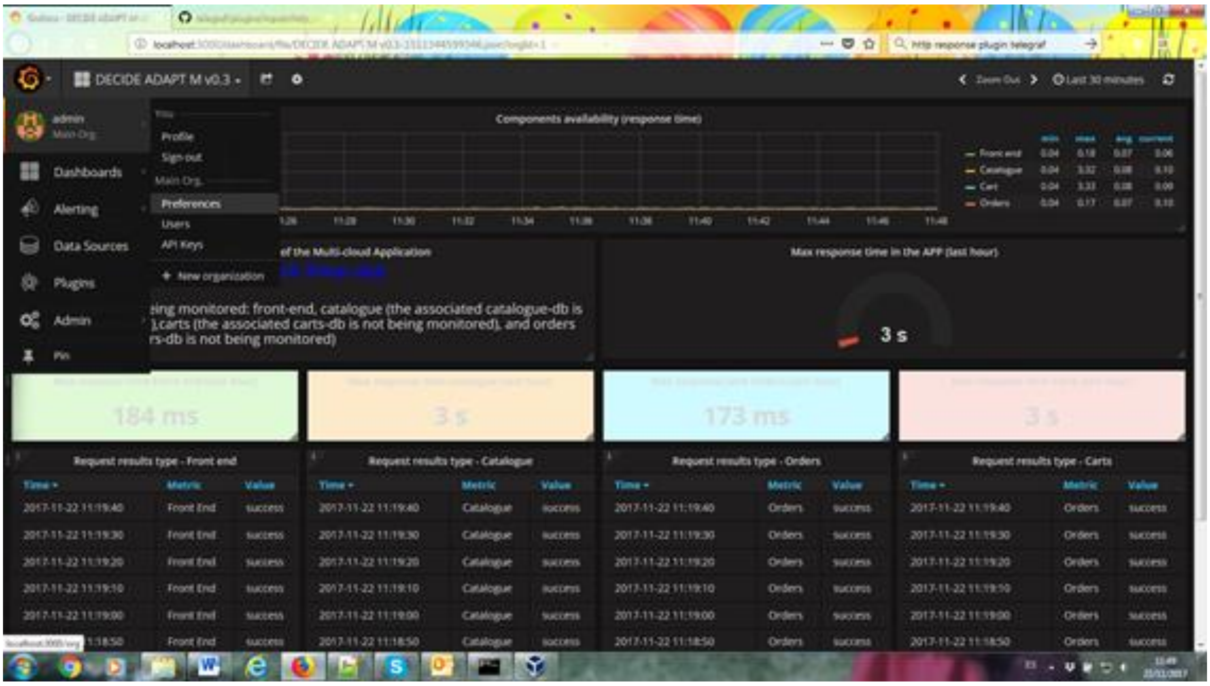


Figure 15. Grafana Log in interface.

A default user has been set up for the ADAPT Monitoring:

- User: admin
- Password: admin

For a better visualization of the metrics, user may change the preferences of the dashboard to the “light” mode:



...eing monitored: front-end, catalogue (the associated catalogue-db is ...),carts (the associated carts-db is not being monitored), and orders ...rs-db is not being monitored)

Request results type - Front end

Time	Metric	Value
2017-11-22 11:19:40	Front End	success
2017-11-22 11:19:30	Front End	success
2017-11-22 11:19:20	Front End	success
2017-11-22 11:19:10	Front End	success
2017-11-22 11:19:00	Front End	success
1:18:50	Front End	success

Request results type - Catalogue

Time	Metric	Value
2017-11-22 11:19:40	Catalogue	success
2017-11-22 11:19:30	Catalogue	success
2017-11-22 11:19:20	Catalogue	success
2017-11-22 11:19:10	Catalogue	success
2017-11-22 11:19:00	Catalogue	success
2017-11-22 11:18:50	Catalogue	success

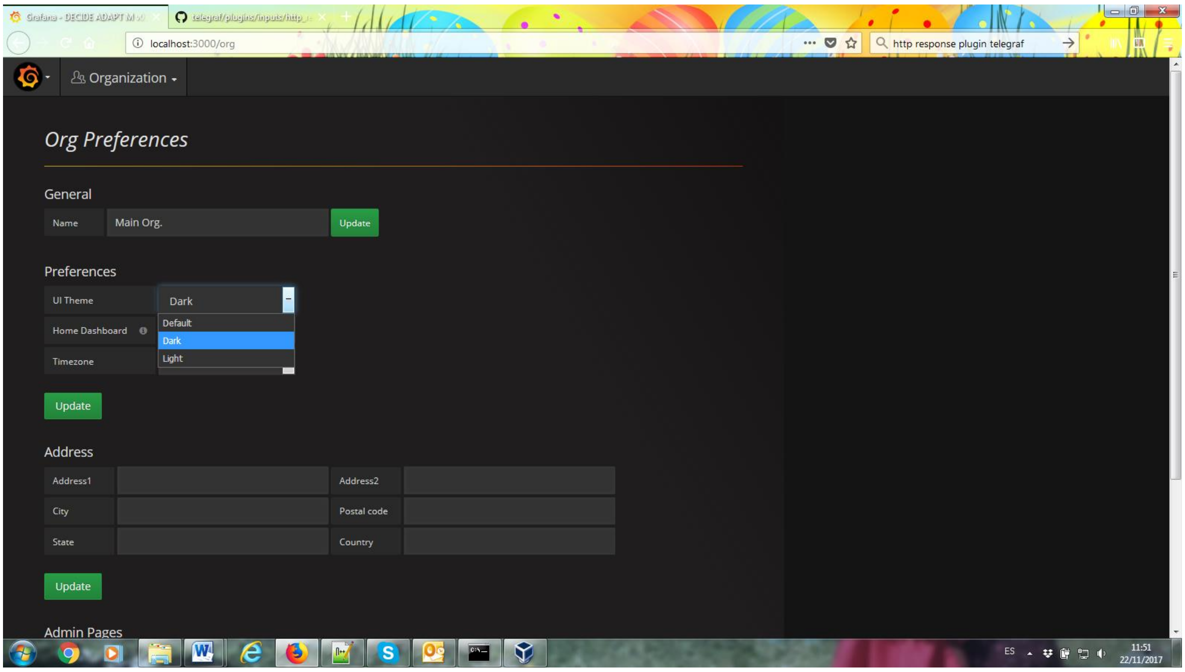
Request results type - Orders

Time	Metric	Value
2017-11-22 11:19:40	Orders	success
2017-11-22 11:19:30	Orders	success
2017-11-22 11:19:20	Orders	success
2017-11-22 11:19:10	Orders	success
2017-11-22 11:19:00	Orders	success
2017-11-22 11:18:50	Orders	success

Request results type - Carts

Time	Metric	Value
2017-11-22 11:19:40	Orders	success
2017-11-22 11:19:30	Orders	success
2017-11-22 11:19:20	Orders	success
2017-11-22 11:19:10	Orders	success
2017-11-22 11:19:00	Orders	success
2017-11-22 11:18:50	Orders	success

Figure 16. Preferences Tab in the ADAPT monitoring UI



Org Preferences

General

Name: Main Org.

Preferences

UI Theme: Dark

Home Dashboard

Timezone

Address

Address1

Address2

City

Postal code

State

Country

Figure 17. Preferences menu

Then, in the dashboards tab, the user needs to select the DECIDE dashboard v3 (current version of the ADAPT monitoring UI M12):

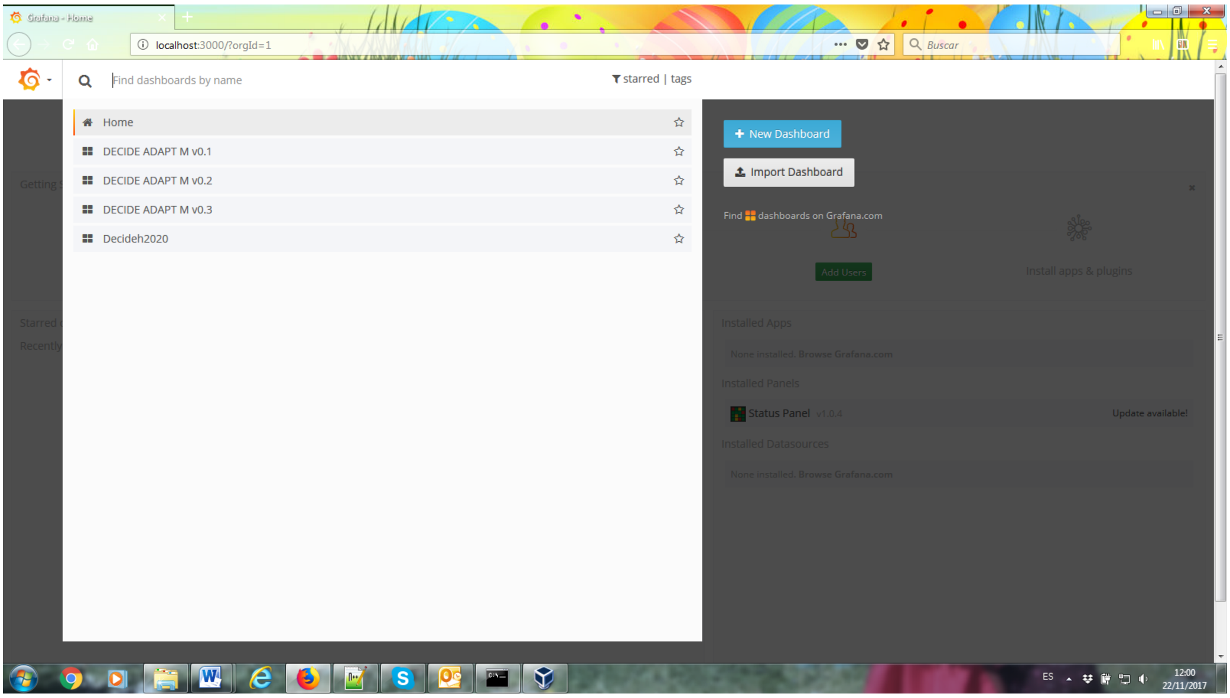
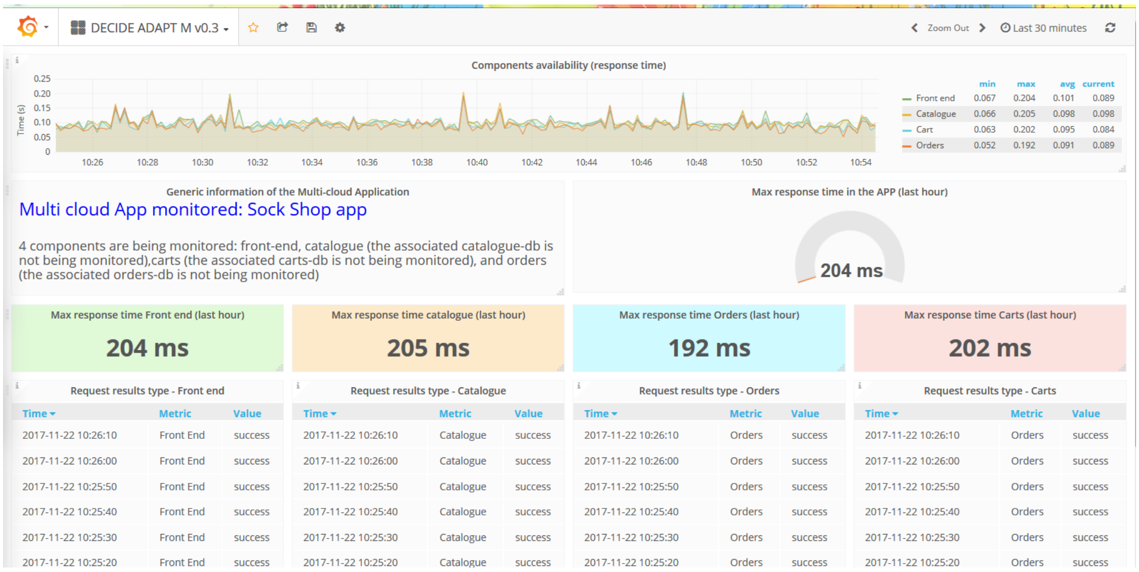


Figure 18. Dahsboards tab

In the dashboard, the established metrics are monitored for the Sock Shop application.



	min	max	avg	current
Front end	0.067	0.204	0.101	0.089
Catalogue	0.066	0.205	0.098	0.098
Cart	0.063	0.202	0.095	0.084
Orders	0.052	0.192	0.091	0.089

Generic information of the Multi-cloud Application

Multi cloud App monitored: Sock Shop app

4 components are being monitored: front-end, catalogue (the associated catalogue-db is not being monitored),carts (the associated carts-db is not being monitored), and orders (the associated orders-db is not being monitored)

Max response time Front end (last hour): 204 ms

Max response time catalogue (last hour): 205 ms

Max response time Orders (last hour): 192 ms

Max response time Carts (last hour): 202 ms

Request results type - Front end

Time	Metric	Value
2017-11-22 10:26:10	Front End	success
2017-11-22 10:26:00	Front End	success
2017-11-22 10:25:50	Front End	success
2017-11-22 10:25:40	Front End	success
2017-11-22 10:25:30	Front End	success
2017-11-22 10:25:20	Front End	success

Request results type - Catalogue

Time	Metric	Value
2017-11-22 10:26:10	Catalogue	success
2017-11-22 10:26:00	Catalogue	success
2017-11-22 10:25:50	Catalogue	success
2017-11-22 10:25:40	Catalogue	success
2017-11-22 10:25:30	Catalogue	success
2017-11-22 10:25:20	Catalogue	success

Request results type - Orders

Time	Metric	Value
2017-11-22 10:26:10	Orders	success
2017-11-22 10:26:00	Orders	success
2017-11-22 10:25:50	Orders	success
2017-11-22 10:25:40	Orders	success
2017-11-22 10:25:30	Orders	success
2017-11-22 10:25:20	Orders	success

Request results type - Carts

Time	Metric	Value
2017-11-22 10:26:10	Orders	success
2017-11-22 10:26:00	Orders	success
2017-11-22 10:25:50	Orders	success
2017-11-22 10:25:40	Orders	success
2017-11-22 10:25:30	Orders	success
2017-11-22 10:25:20	Orders	success

Figure 19. ADAPT Monitoring dashboard.

2.2.4 Licensing information

This component is offered under MIT license

2.2.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D4.7.

The first release is available in the DECIDE open git repository, more precisely at the following address:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/master/ADAPT/Monitoring

3 Violations Handler

3.1 Implementation

3.1.1 Functional description

DECIDE ADAPT violation handlers are in charge of receiving a violation alert from ADAPT monitoring components and, depending on the application's technological risk, they will notify the operator and automatically trigger a new simulation through OPTIMUS (low technological risk) or notify the operator and give them a chance to modify settings before triggering a new simulation (in case of high technological risk).

The violation handlers will also archive the violation to keep a history of alerts.

The main functionalities of the ADAPT violation handlers are:

F1. Receive violation alerts: when a violation occurs, the monitoring components will notify the violation handlers, which will process the violation.

F2. Retrieve technological risk: when the violation handlers receive a violation, they will retrieve the application's technological risk from the Application Description, in order to decide what action to carry out.

F3. Evaluate action to carry out: if the application's technological risk is low, the violation handlers will automatically trigger a new simulation. If it is high, the operator will have to confirm the new simulation. In both cases, the operator has to be notified of the violation.

F4: Notify operator: when a violation is received, the operator will receive an email notification containing relevant information about the alert.

F5: Send OPTIMUS a redeployment order: if a violation has occurred, and the technological risk of the application is low, the violation handlers will automatically trigger a new simulation through OPTIMUS.

F6: Offer new simulation configuration: if a violation has occurred, and the technological risk is high, the violation handlers will offer a screen from which the operator can modify the redeployment settings and confirm a new simulation through OPTIMUS.

F7: Archive violations: the Violation handlers will store a history of received violation alerts.

ADAPT violation handlers will be implemented following an incremental approach adding features in the different releases of the tool (D4.8 in M24 and D4.9 in M30). In this first release the following functionalities are (partially) implemented: **F1**, **F2 and F3**.

The following table details the relationship between the deployment requirements indicated in deliverable D4.1 and the implemented functionalities, with a description of the coverage for each functionality.

Requirements covered by the prototype:

Table 2. Functionality covered by the M12 Violation Handler prototype**.**

Functionality	Req. ID	Coverage
F1	WP4-REQ19, DEVOPS-REQF9	The prototype will be able to receive an alert and extract the relevant information from it.

Functionality	Req. ID	Coverage
F2	WP4-REQ21, WP4-REQ22	Partial. The prototype will retrieve the technological risk from a local database.
F3	WP4-REQ21, WP4-REQ22	Depending on the received alert, the prototype will decide the action that will be carried out.
F4	WP4-REQ22, WP4-REQ27	None
F5	WP4-REQ19, WP4-REQ21	None
F6	WP4-REQ19, WP4-REQ22, WP4-REQ43	None
F7	DEVOPS-REQF9	None

Functionality that this prototype offers:

This prototype is the first version of the Violation handlers component and offers the following functionalities:

- Receive and process alert: the prototype will receive an alert (with the same format as those that will be sent by the Monitoring components, even though there will be no integration at this stage), and will be able to parse this alert to extract the relevant information.
- Retrieve technological risk: the prototype will check the technological risk of the component associated to the received alert. For this version, the technological risk is stored in a local database. For future versions, it will be stored in the Application Description.
- Decide action to carry out: depending on the value of the technological risk, the prototype will decide what action will be carried out (automatic simulation if the risk is low, and manual simulation if it is high).

3.1.1.1 Fitting into overall DECIDE Architecture

The Violation handlers are a component of ADAPT (figure 1 shows a diagram of how ADAPT fits in DECIDE as a whole). This component is in charge of deciding what actions will take place whenever a violation occurs.

As mentioned before, when the Violation handlers receive a violation from the Monitoring components, they will request the technological risk from the Application Description. Depending on this value, a new simulation will be triggered automatically or will need the operator's input. In both cases, the operator will be notified.

The following figure shows the general architecture of this component. For a more detailed description, check D4.1 [6].

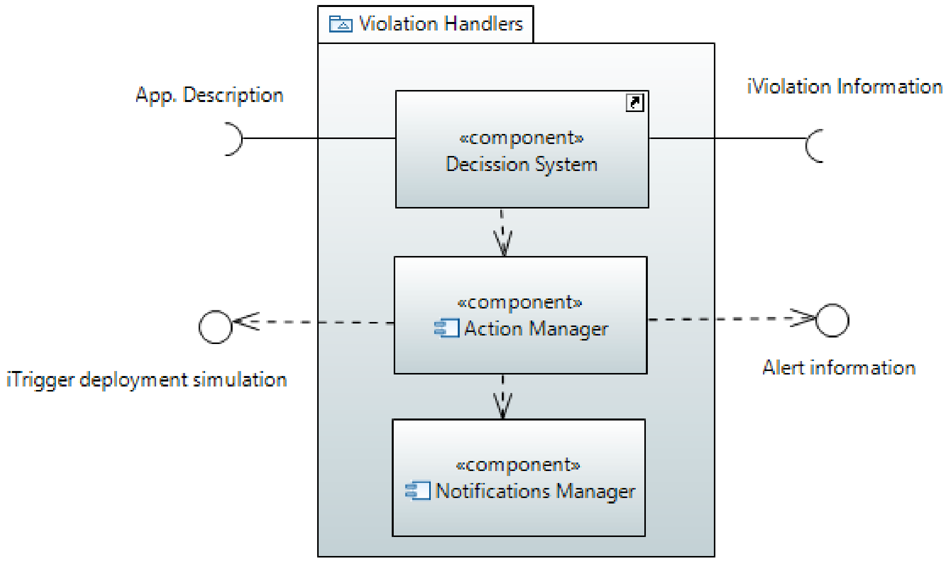


Figure 20. General architecture of the Violation handlers

The Violation handlers communicate with ADAPT Monitoring components to receive the violation notifications; with the Application Description to obtain the technological risk; and with OPTIMUS, to start a new simulation.

3.1.2 Technical description

This section describes the technical details of ADAPT Violation handlers.

3.1.2.1 Prototype architecture

From the proposed architecture that can be found in deliverable D4.1 [6] and that is shown in Figure 20, the component *Decision System* will be implemented. This component is deployed in a Docker container and accessed through a REST API.

3.1.2.2 Components description

- Decision System: receives and processes an alert, to extract the relevant information from it. Then, it requests the Technological risk in order to decide what action will take place.

3.1.2.3 Technical specifications

The Development of the Violation handlers prototype is divided in three parts:

Integration test archetype (*decide-**): this module is used solely to test the developed class. It is a standard development archetype used by Experis for their projects, and it is based on the Spring framework. Only the services for REST requests and database integration are used. Thus, this module is not considered a part of the DECIDE specific developments, since the Violation handlers class will be autonomous and will not need any of the mentioned services.

Database (MySQL): the database used for the prototype contains six columns (*idviolations, title, ruleID, ruleName, state, risk*). The first five columns correspond to the variables found in the sample alert that is received by the component. The last one, *risk*, is associated to the application name (*title*) and simulates the application's technological risk that will be found in the application description. The database receives a request from the module *decide-dbmodel* which, based on the application name, returns the technological risk. For the final version, this component will be substituted for one that requests the technological risk from the Application Description.

Violation handler (*eu.DECIDEh2020.adapt.violationhandler*): this component, developed in Java, is prepared to be integrated with any Java system. In this version of the prototype, the component receives a POST request with a JSON file, with the body of a Grafana webhook call (http://docs.grafana.org/alerting/notifications/#webhook). The variables contained in the JSON file are mapped, and, when the alert is received, the component requests the value risk from the database and returns it, along with the rest of variables and values of the alert.

3.2 Delivery and usage

3.2.1 Package information

The development of the Violation handlers is oriented to having just a class that will be integrated in the project by calling the method:

violationHandler.decision();

The delivered package consists in the below folder structure:

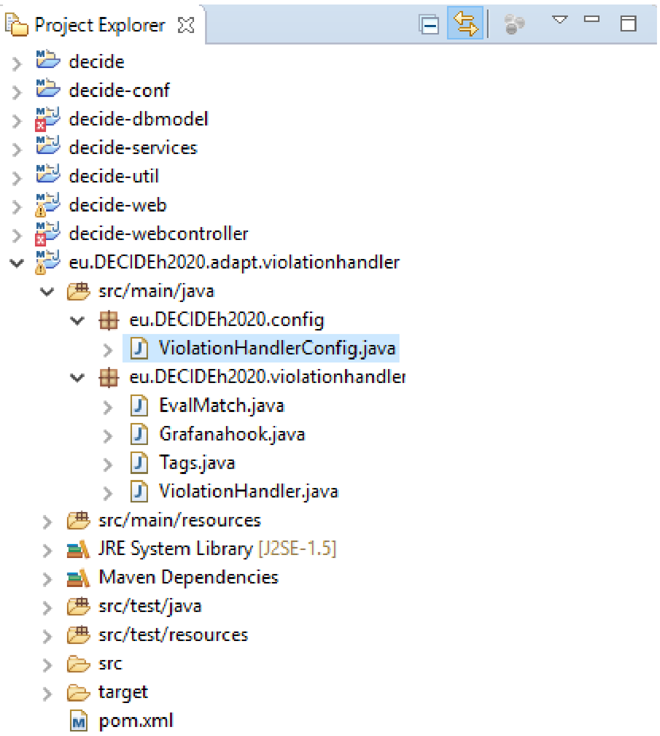
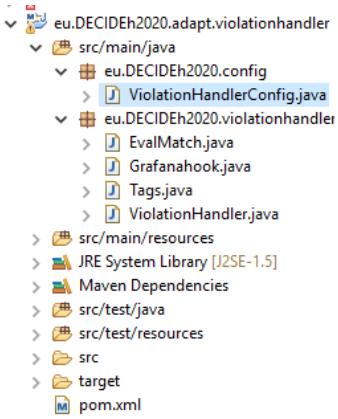


Figure 21. Source code structure of the Violation handlers

It is important to note that the class *eu.DECIDEh2020.adapt.violationhandler* is the Violation handlers class, the rest are for supporting the tests.

A detailed description of the classes that belong to *eu.DECIDEh2020.adapt.violationhandler* can be found below:



- eu.DECIDEh2020.adapt.violationhandler
  - src/main/java
    - eu.DECIDEh2020.config
      - ViolationHandlerConfig.java
    - eu.DECIDEh2020.violationhandler
      - EvalMatch.java
      - Grafanahook.java
      - Tags.java
      - ViolationHandler.java
  - src/main/resources
  - JRE System Library [J2SE-1.5]
  - Maven Dependencies
  - src/test/java
  - src/test/resources
  - src
  - target
  - pom.xml

Figure 22. Structure of the *eu.DECIDEh2020.adapt.violationhandler* class

The main class is *ViolationHandler.java*, which requests the application's technological risk when invoked. This value depends on the field *Title* that is included in the received alert notification. Notifications are received as a JSON file with the format of Grafana, the monitoring application that will be used for creating alerts, at least at this stage of the project. This is an example of a Grafana notification:

```
{
  "title": "APP1",
  "ruleId": 1,
  "ruleName": "CPUAlert",
  "ruleUrl":
"http://url.to.grafana/db/dashboard/my_dashboard?
panelId=2",
  "state": "alerting",
  "imageUrl": "http://s3.image.url",
  "message":  "Load  is  peaking.  Make  sure  the
traffic is real and spin up more webfronts",
  "evalMatches": [
                        {
      "metric": "requests",
      "tags": {},
      "value": 122
    }
  ]
}
```

Figure 23. Grafana notification example

The files *EvalMatch.java, Grafanahook.java* and *tag.java* are part of a class that maps the received JSON to functions to ease data read and handling.

In this first proof of concept, the requests to the database are configured in the file *decide-dbmodel > ViolationsDAO.java*, in the function *findAction*, but this file exists only to support the proof of concept. In the future, this connection will not exist and will be integrated in the same package.

3.2.2 Installation instructions

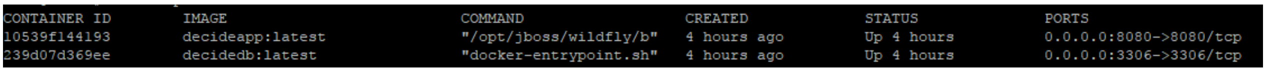
The Violation handlers have been developed in Java, and is supported by a MySQL database. To install it, it is first required to configure a Docker environment configured with the following deployment steps (assuming that there is already a Docker environment and that the compiled file *decide-webcontroller.war* is available):

First, the images must be loaded to the environment:

```
1. docker build -t decideapp -f dockerfileapp
2. docker build -t decided -f dockerfiledb
```

Once the images are loaded, the composer can be executed:

```
3. docker-compose up -d
```



```
CONTAINER ID        IMAGE                            COMMAND                  CREATED             STATUS              PORTS
10539f144193        decideapp:latest                 "/opt/jboss/wildfly/b"   4 hours ago         Up 4 hours          0.0.0.0:8080->8080/tcp
239d07d369ee        decidedb:latest                  "docker-entrypoint.sh"   4 hours ago         Up 4 hours          0.0.0.0:3306->3306/tcp
```

Figure 24. Violation handlers' docker containers running

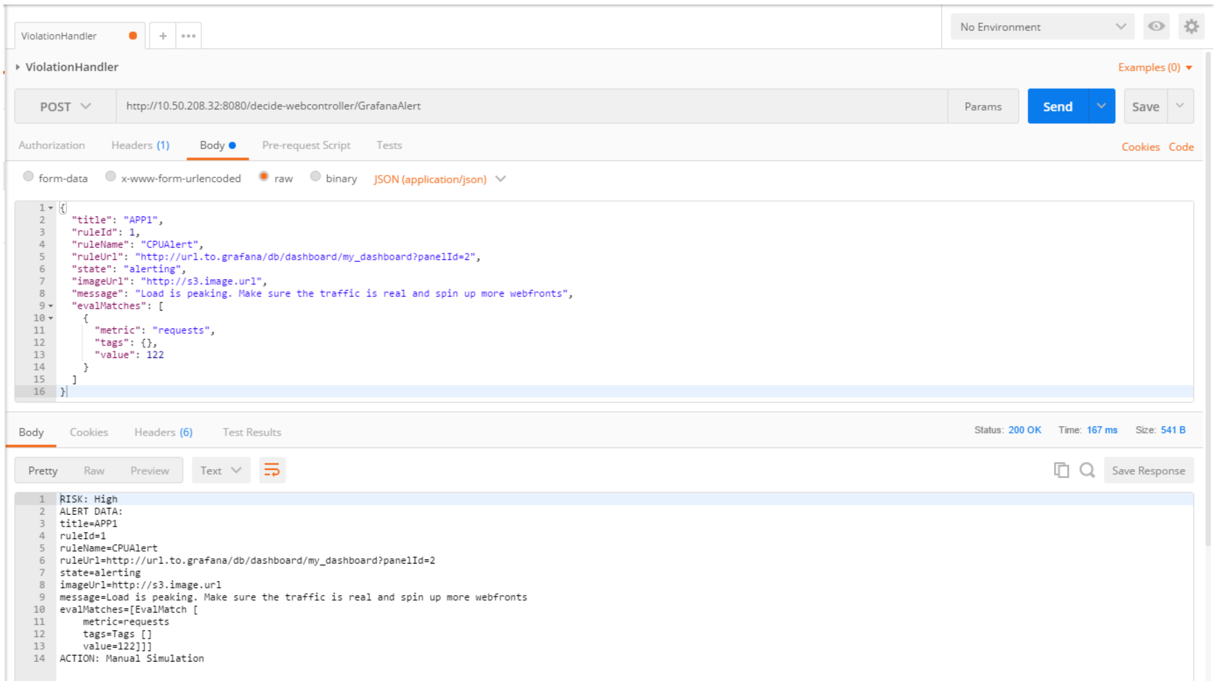
3.2.3 User Manual

The application is used by making a HTTP POST request to http://IP:8080/decide-webcontroller/GrafanaAlert, with the following message body format:

```
{
  "title": "APP1",
  "ruleId": 1,
  "ruleName": "CPUAlert",
  "ruleUrl":
"http://url.to.grafana/db/dashboard/my_dashbo
ard?panelId=2",
  "state": "alerting",
  "imageUrl": "http://s3.image.url",
  "message": "Load is peaking. Make sure the
traffic is real and spin up more webfronts",
  "evalMatches": [
    {
      "metric": "requests",
      "tags": {},
      "value": 122
                         }
  ]
}
```

Figure 25. HTTP post request to the application

This operation can be done using Postman, configured as shown below:



```
http://10.50.208.32:8080/decide-webcontroller/GrafanaAlert
```

```
{
  "title": "APP1",
  "ruleId": 1,
  "ruleName": "CPUAlert",
  "ruleUrl": "http://url.to.grafana/db/dashboard/my_dashboard?panelId=2",
  "state": "alerting",
  "imageUrl": "http://s3.image.url",
  "message": "Load is peaking. Make sure the traffic is real and spin up more webfronts",
  "evalMatches": [
    {
      "metric": "requests",
      "tags": {},
      "value": 122
    }
  ]
}
```

```
RISK: High
ALERT DATA:
title=APP1
ruleId=1
ruleName=CPUAlert
ruleUrl=http://url.to.grafana/db/dashboard/my_dashboard?panelId=2
state=alerting
imageUrl=http://s3.image.url
message=Load is peaking. Make sure the traffic is real and spin up more webfronts
evalMatches=[EvalMatch [
    metric=requests
    tags=Tags []
    value=122]]]
ACTION: Manual Simulation
```

Figure 26. Postman configuration to access the Violation handlers prototype

As seen in the response, the application will return the variables contained in the alert and, depending on the technological risk of the application, it will return one of two actions: *Manual Simulation* or *Automatic Simulation.*

3.2.4 Licensing information

This component is offered under the MIT license.

3.2.5 Download

The source code is available in the EC portal for deliverables, included in the zip file for D4.7.

The first release is available in the DECIDE open git repository, more precisely at the following address:

https://git.code.tecnalia.com/DECIDE_Public/DECIDE_Components/tree/master/ADAPT/ViolationsHandler

4 Conclusions

In this document, the Initial multi-cloud application monitoring is presented. The functional and the technical details of the components comprising this version: Adapt Monitoring and Violations Handlers are described and their installation processes and usage guides are detailed as well. The next release of this deliverable (D4.8 [8]) will include new sub-components and also new functionalities and improvements for the current components.

In the next release, for ADAPT Monitoring, new metrics will be included to support the different DECIE NFRs and MCSLA monitoring, and aggregation of metrics will be implemented. Furthermore, the integration with other components such as ACSmI and Violations Handler will be tackled.

References

[1] DECIDE consortium, «DECIDE DoA (Description of action),» 2016.

[2] C. McLuckie, “Perspective on multi-cloud (part 3 of 3) — Availability and multi-cloud,” [Online]. Available: https://blog.heptio.com/perspective-on-multi-cloud-part-3-of-3-availability-and-multi-cloud-5018762d2702. [Accessed 25 10 2017].

[3] ARTIST Consortium, «D7.1 Definition and extension of performance,» 2014.

[4] DECIDE consortium, «D5.2-Initial Advanced Cloud Service meta-Intermediator (ACSmI),» 2017.

[5] DECIDE consortium, “D2.4 Detailed architecture v1,” 2017.

[6] DECIDE Consortium, «D4.1 Initial DECIDE ADAPT Architecture,» 2017.

[7] WeaveWorks, “Sock Shop - Docker compose,” [Online]. Available: https://microservices-demo.github.io/microservices-demo/deployment/docker-compose.html. [Accessed 08 11 2017].

[8] DECIDE Consortium, «D4.8-Intermediate multi-cloud application monitoring,» 2018.

[9] «Perspective on multi-cloud (part 3 of 3) — Availability and multi-cloud,» [En línea]. Available: https://blog.heptio.com/perspective-on-multi-cloud-part-3-of-3-availability-and-multi-cloud-5018762d2702. [Último acceso: 08 11 2017].

[10] Weaveworks, “Sock Shop: A microservices demo application,” [Online]. Available: https://microservices-demo.github.io/. [Accessed 25 10 2017].