Deliverable D4.3

Final DECIDE ADAPT Architecture

Editor(s):	Lorenzo Blasi
Responsible Partner:	Hewlett Packard Italiana / HPE
Status-Version:	Final – v1.0
Date:	27/05/2019
Distribution level (CO, PU):	PU

Project Number:	GA 731533
Project Title:	DECIDE

Title of Deliverable:	Final DECIDE ADAPT Architecture
Due Date of Delivery to the EC:	31/05/2019

Workpackage responsible for the Deliverable:	WP4 - Continuous deployment and operation
Editor(s):	Hewlett Packard Italiana / HPE
Contributor(s):	Lorenzo Blasi (HPE), Javier Gavilanes Ruano (Experis), Iñaki Etxaniz Errazkin (Tecnalia)
Reviewer(s):	Leire Orue-Echevarria (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP5, WP6

Abstract:	This deliverable describes the final version of the architecture of DECIDE ADAPT tool providing a comprehensive overview of the system using different architectural views to represent different aspect of the system (e.g. Use Case View, Logical View, Process View, Deployment View, and Implementation View).
Keyword List:	Adaptation, monitoring, deployment, stateful containers, Kubernetes, Windows
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's 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	22/02/2019	Defined document ToC	HPE
V0.2	01/03/2019	Agreed sections' ownership	HPE
V0.3	02/04/2019	Added Y3 requirements	HPE
V0.4	19/04/2019	Added ADAPT DO Use Cases, DO architecture and ADAPT integration	HPE
V0.5	24/04/2019	Added analysis on stateful containers	HPE
V0.6	06/05/2019	Added analysis on Kubernetes support	HPE
V0.7	08/05/2019	Added analysis on Windows support	HPE
V0.8	09/05/2019	Added introduction and final architecture	HPE
V0.9	09/05/2019	Integrated Experis contribution	Experis, HPE
V0.10	10/05/2019	Added Executive Summary and Conclusions, updated integration status	HPE
V0.11	10/05/2019	Integrated Tecnalia contribution	Tecnalia, HPE
V1.0	27/05/2019	Integrated further Tecnalia contribution and finalized	Tecnalia, HPE

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

AD	Application Description
CSP	Cloud Service Provider
DB	Database
DO	Deployment Orchestrator
EC	European Commission
MCSLA	Multi-Cloud SLA
MM	Monitoring Manager
NFR	Non-Functional Requirement
REST	Representational State Transfer
SLA	Service Level Agreement
SLO	Service Level Objective
ToC	Table of Contents
TSDB	Time Series Data Base
UI	User Interface
UML	Unified Modeling Language
VH	Violations Handler
VM	Virtual Machine
VPN	Virtual Private Network
WP	Work Package
Yn	Year n

Executive Summary

This deliverable D4.3 (*Final DECIDE ADAPT Architecture*) describes the final version of DECIDE ADAPT's high-level architecture. The architecture is described using standard UML syntax both for the Use Cases diagrams and the component and interfaces diagrams. The document includes only the most relevant functional information, focusing on what has changed from the previous version of this deliverable (D4.2 [1]). Other WP4 deliverables, D4.6 (*Final multi-cloud application deployment and adaptation*) [2] and D4.9 (*Final multi-cloud application monitoring*) [3] will describe in more detail the implemented components. This document also summarizes the current ADAPT integration state and interfaces and includes a preliminary analysis of functionalities that could be added to DECIDE ADAPT in future development cycles.

The main research topics targeted in this third year, along with a further integration of the components, are a specific attention to user interaction, to security and to the extension of current functionalities. Our focus on improving user interaction led to the definition of an ADAPT DO Wizard to be included in the DevOps Framework, to guide developers in the definition of the parameters needed by ADAPT for application deployment and showing at the end a representation of the planned deployment steps. The same focus on improving user interaction resulted in the addition of the ADAPT Monitoring Dashboard component, also integrated in the DevOps Framework and specifically configured to graphically show the evolution of the monitored metrics along with the possible SLA violations. Also targeting a better user interaction for the whole DECIDE Framework has been the idea, proposed by WP4 and then implemented in WP2, to define a state diagram driving the DECIDE workflow. The improvements in security are shown both by the new DO use cases, mostly focused on uploading and configuration of certificates and SSH keys, and by the definition of which sensible information should be stored in the new DECIDE component acting as secrets' repository.

The requirements collected in the first year of the project (Y1) and documented in deliverable D4.1 [4] have been further analysed and merged in Y2, as reported in deliverable D4.2 [1]. This deliverable in section 2 lists the merged requirements introduced in D4.2 and reports about their planned deadline and implementation status. As indicated in the table, most of the requirements have been satisfied or are in the process of final integration and will be satisfied by the end of the project.

The following section 3 describes, using UML Use Cases formalism, the high level functionalities planned to be delivered in the final version for each of the three ADAPT components, which are: manual setup of the deployment configuration (the Wizard and dashboard), adaptation actions in case of high tech risk, and support for extensions of NFRs to be measured. The same section also includes a preliminary analysis of functionalities that could be added to DECIDE ADAPT in future development cycles, which can be intended as a roadmap for future ADAPT functionalities. This section analyses extended monitoring metrics and functionalities, support for stateful containers, for deploying containers on Kubernetes and for the deployment of Windows applications.

Section 4 describes ADAPT architecture, initially at a high level and showing all the external interfaces. The internal architecture of DECIDE ADAPT has not changed significantly from its initial definition. Its main components are still Deployment Orchestrator (DO), Monitoring Manager (MM) and Violations Handler (VH). What has been enhanced, however, is the set of its interfaces with other DECIDE components, indicating an improvement in the interaction and synergy with the DECIDE DevOps Framework itself. The same section then describes in some more detail the internal structure of each ADAPT component. The main changes in the ADAPT architecture with respect to year 2 are: new iSecConfig interface to configure security details for ADAPT DO (see sect. 4.1), new ADAPT DO Wizard / dashboard component, new ADAPT Monitoring Dashboard component (see sect. 4.2), and removal of the iStoreAlertInformation interface as VH now stores violations in a local database.

Section 5 describes the integration status of both internal and external ADAPT interfaces, summarizes the Application Description fields read or written by ADAPT, and indicates the sensitive information to be stored in the new secrets' repository component (built on the Vault open source component).

1 Introduction

1.1 About this deliverable

This deliverable is the result of DECIDE Task 4.1, *DECIDE ADAPT Architecture and Integration* of Work Package 4 (*Continuous deployment and operation*). It describes the updates to DECIDE ADAPT's high-level architecture planned in the third year of the project (Y3) and includes the analysis of some features that could be implemented in future versions. The architecture is described using standard UML syntax, both for the use cases and the component and interfaces diagrams. Other deliverables, D4.6 (*Final multi-cloud application deployment and adaptation*) [2] and D4.9 (*Final multi-cloud application monitoring*) [3] will describe in more detail the implemented components. This document also summarizes the current ADAPT integration state and interfaces. This document is not intended as a user manual for ADAPT operation. It is expected that all Partners participating in DECIDE WP4 will read this deliverable; it will be very useful also for Partners participating in WP2, WP3 and WP5, who should integrate their components with ADAPT, and especially WP6 Partners which are ADAPT's primary users.

1.2 Innovation

The research reported in this deliverable represents a further step of the research reported in the previous ADAPT Architecture deliverables, D4.1 [4] and D4.2 [1]. The main topics targeted in this third year, along with a further integration of the components, are a specific attention to user interaction, to security and to the extension of current functionalities. Our focus on improving user interaction led to the definition of an ADAPT DO Wizard to be included in the DevOps Framework, to guide developers in the definition of the parameters needed by ADAPT for application deployment and showing at the end a representation of the planned deployment steps. The same focus on improving user interaction resulted in the addition of the ADAPT Monitoring Dashboard component, also integrated in the DevOps Framework and specifically configured to graphically show the evolution of the monitored metrics along with the possible SLA violations. Also targeting a better user interaction for the whole DECIDE Framework has been the idea, proposed by WP4 and then implemented in WP2, to define a state diagram driving the DECIDE workflow. The improvements in security are shown both by the new DO use cases, mostly focused on uploading and configuration of certificates and SSH keys, and by the definition of which sensible information should be stored in the new DECIDE component acting as secrets' repository. Finally we think that the definition of a roadmap for new functionalities to be added to ADAPT in the sustainability phase, i.e. after the end of the project, should start now; for this reason this deliverable reports a preliminary analysis of functionalities that could be added to ADAPT in future development cycles.

1.3 Document structure

The first section after this introduction, section 2, lists the merged requirements introduced in the previous deliverable D4.2 [1] and reports about their planned deadline and implementation status. The following section 3 uses the UML Use Cases formalism to describe the additional high level functionalities planned in the final version for each of the three ADAPT components. The same section also includes a preliminary analysis of functionalities that could be added to DECIDE ADAPT in future development cycles. Section 4 describes ADAPT architecture at an high level, including all the external interfaces; the same section then shows into some more detail the internal structure of each ADAPT component, with a description of their new components and interfaces. Section 5 starts reporting both internal and external ADAPT interfaces, their implementation and integration status and a short indication of what has changed from Y2 to Y3; it then summarizes the current status of information exchanged by ADAPT through the Application Description and finally explains which sensible data will be stored by ADAPT in the new component added in the final DECIDE Framework release for the purpose of storing secret information.

2 ADAPT requirements

2.1 Requirements planned for Y3

The **Table *1*** below lists the merged requirements introduced in the previous deliverable D4.2 [1] and shows which have been already satisfied and which are in the development phase to be successfully delivered by the end of the project.

Table 1. Merged requirements status

Req. ID	Description	Component	Deadline	Status
WP4-MR1	DECIDE ADAPT will support the semi-automatic adaptation and dynamic re-deployment of (parts of) multi-cloud applications when certain conditions are not met, by changing the configuration and topology of services at operational time based on continuous monitoring of both the conditions of the application and the CSPs where the application is deployed on.	DO, MM, VH	M24	Implemented. Testing integration
WP4-MR2	A violation is raised when the defined composite multi-cloud application SLA is not being fulfilled, the application is not performing as established or the cloud service providers (CSPs) are violating the contracted SLAs.	MM	M24	Satisfied
WP4-MR3	ADAPT will monitor the objectives indicated in the MCSLA, monitoring the metrics related to the application components (micro-services) and gathering the run-time information related to the CSPs monitoring from other components (ACSmI).	MM	M12	Satisfied
WP4-MR4	DECIDE ADAPT will support automated dynamic deployment of service components	DO	M12	Satisfied
WP4-MR5	DECIDE ADAPT will generate scripts for automating both resource provisioning and deployment for multi-cloud native applications	DO	M12	Satisfied
WP4-MR6	ADAPT will support manual checking of the deployment configuration and scripts	DO	M30	Wizard / dashboard being integrated
WP4-MR7	ADAPT will maintain the current deployment configuration situation; other tools will maintain the history of the previous deployment configurations, so that they can be checked in the re-deployment phase	DO	M12	Satisfied

Req. ID	Description	Component	Deadline	Status
WP4-MR8	In case the technological risk for the application has been defined as low, the multi-cloud application will be redeployed automatically, following a new deployment configuration [provided by triggering OPTIMUS].	VH, DO	M24	Implemented. Testing integration
WP4-MR9	In case the technological risk for the application has been defined as high, once ADAPT has identified the violation(s) that are affecting the malfunctioning of the application, it will both alert the operator and trigger OPTIMUS, sending it the identified violation(s), to simulate a new deployment configuration that could avoid the same problem.	VH	M30	Implemented. Testing integration. Manual redeployment in development phase
WP4-MR10	In case of a violation, ADAPT will report to the operator the NFR (SLO) that are not being fulfilled	MM, VH	M24	Satisfied
WP4-MR11	ADAPT functionalities (deployment, monitoring and adaptation) rely on information about the multi-cloud application obtained from the Application Description	DO, MM, VH	M12 - M24	Satisfied. Application Description still being extended
WP4-MR12	ADAPT will support applications based on composition of stateless (possibly micro) services	DO	M12	Satisfied
WP4-MR13	ADAPT will support composable applications where each composition unit is a containerized service	DO	M12	Satisfied
WP4-MR14	DECIDE (and ADAPT in particular) will support extensions to add more NFRs that need to be measured.	MM	M30	New metrics planned
WP4-MR15	Users will perceive relevant improvements in the business continuity since as soon as there is a problem (i.e. lack of resource due to a peak of requests) the software is automatically re-adapted and re-deployed	DO	M24	Implemented. Testing integration
WP4-MR16	DECIDE ADAPT will support the continuous deployment of the developed apps	DO	M24	Satisfied
WP4-MR17	DECIDE will maintain and store a history of the violations occurred for a deployed application	MM	M24	Satisfied

3 High Level ADAPT Functionalities

This section reports about high level functionalities for the DECIDE ADAPT tool.

3.1 Updates to ADAPT UML Use Cases

The following sub-sections describe the new functionalities planned to be implemented in Y3 by each ADAPT component. The main requirements to be satisfied for the third year are (see sect. 2.1):

- WP4-MR6: ADAPT will support manual checking of the deployment configuration
- WP4-MR9: when tech risk is high ADAPT alerts the operator and triggers OPTIMUS
- WP4-MR14: ADAPT will support extensions to add more NFRs that need to be measured

that however, will be fully implemented and integrated in the final release of the integrated DevOps framework in M33.

3.1.1 ADAPT DO Use Cases

The new functionality planned in Y3 for the Deployment Orchestrator (DO) is a dashboard that allows to check each deployment's configuration and status. In addition to that, DECIDE will also implement some functionalities targeted at increasing both the usability and the security level of the final system. The new DO-related functionalities are described by the UML use cases detailed in fff. Other use cases representing the whole set of ADAPT functionalities are documented in the previous deliverable D4.2 [1].

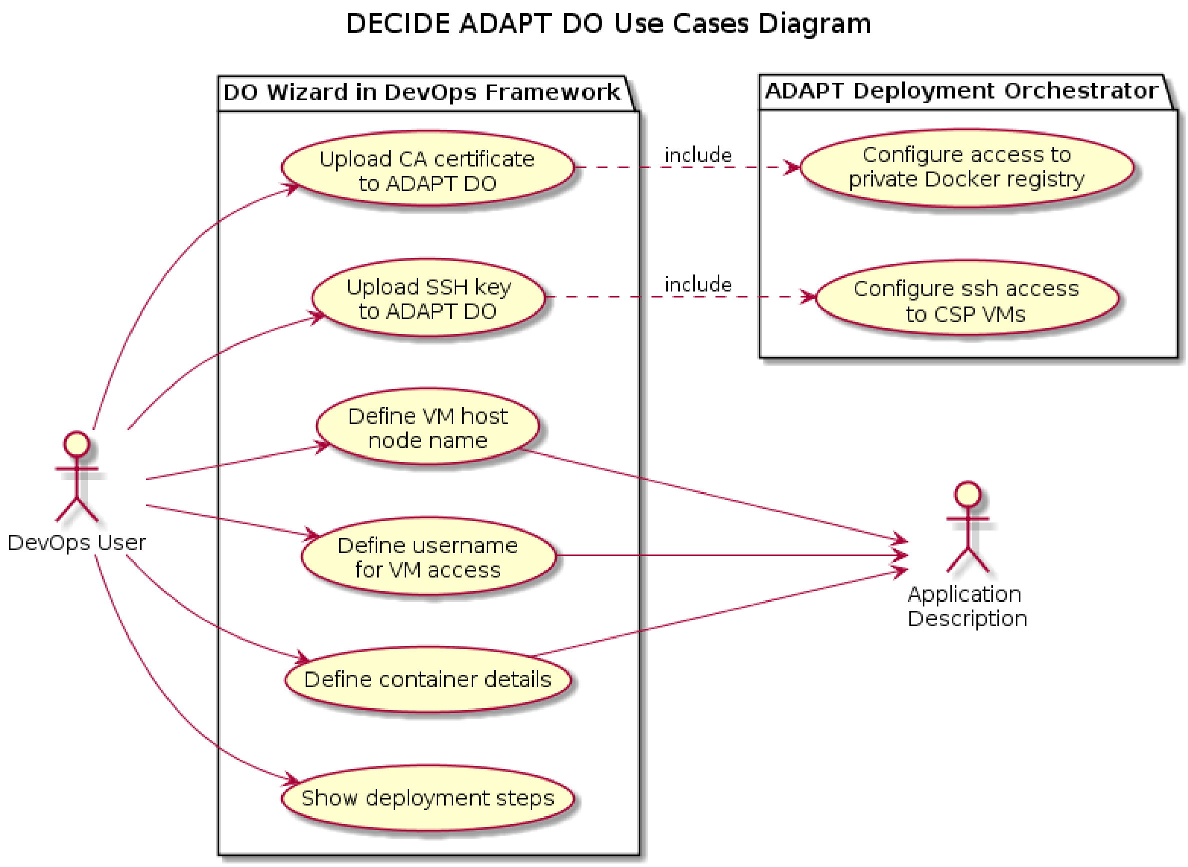

DECIDE ADAPT DO Use Cases Diagram
DO Wizard in DevOps Framework
ADAPT Deployment Orchestrator
Upload CA certificate to ADAPT DO
include
Configure access to private Docker registry
Upload SSH key to ADAPT DO
include
Configure ssh access to CSP VMs
Define VM host node name
Define username for VM access
Define container details
Show deployment steps
DevOps User
Application Description

Figure 1. New Y3 functionalities for ADAPT Deployment Orchestrator

The first Agent on the left in **Figure *1*** is a member of the DevOps team who is in charge of deploying and operating the application.

UCDO301 – Configure access to private Docker registry

A private Docker registry is a repository from which ADAPT can download container images which are not public. If the private registry uses a self-signed certificate to prove its identity when establishing a TLS session, every client must trust this certificate. Therefore, to download container images from such a private registry to a node, that node must trust the registry certificate as a trusted Certification Authority (CA). A dedicated REST API call will allow to upload the registry's certificate file (ca.crt) to ADAPT DO, so that it can install it as a trusted CA on every node which should run containers downloaded from that private registry.

UCDO302 – Configure ssh access to CSP VMs

To allow ADAPT to configure resources created by a Cloud Service Provider (CSP), it is necessary that it can access such resources via ssh authentication. To enable this, ADAPT must have access to the private SSH key related to the profile of the user for the CSP. A dedicated REST API call will allow to upload to ADAPT such a private SSH key.

UCWI301 – Upload CA certificate to ADAPT DO

The DO Wizard user interface within the DevOps Framework, in one of its configuration steps, will allow to select a private registry (previously inserted from user profile management) from a combo box and to upload its public certificate to ADAPT DO, using use case UCDO301.

UCWI302 – Upload SSH key to ADAPT DO

The DO Wizard user interface within the DevOps Framework, in one of its configuration steps, will allow to upload to ADAPT DO the private SSH keys related to the profile of the user for the CSPs that should be accessed during the user's application deployment. The actual upload will be done calling use case UCDO302.

UCWI303 – Define VM host node name

The DO Wizard user interface within DevOps Framework, in one of its configuration steps, will allow the user to give a meaningful name to each of the resources (currently Virtual Machines) provided by ACSmI contracting for deploying the user's application. Each name specified by the user will be written in the Application Description as the value of the key "dockerHostNodeName" for the related VM resource.

UCWI304 – Define username for VM access

When ADAPT DO accesses the VM resources to configure them and deploy the application, it must do that using credentials with the right privileges. The DO Wizard user interface within the DevOps Framework, in one of its configuration steps, will allow the user to specify the username (usually root for Linux) that DO should use when accessing VMs. Each username specified by the user will be written in the Application Description as the value of the key "vmUser" for the related VM resource.

UCWI305 – Define container details

ADAPT DO needs to know some details about the containers to be deployed. At least the container name, its image name:tag and the repository from which to download it must be specified; additionally also the (optional) mapping between network ports on the container and the host can be specified. An optional reverse proxy with a configuration based on a Key/Value store could also be added to (at most) one container to allow dynamical routing of API calls even when containers are redeployed on different nodes. The DO Wizard user interface within the DevOps Framework, in one of its configuration steps, will allow the user to specify the details listed above and will write them in the

Application Description using the keys “containerName”, “imageName”, “imageTag”, “dockerPrivateRegistryIp”, “dockerPrivateRegistryPort”, “portMapping” (array of pairs “hostPort” “containerPort”), and “addConsulTraefikRules”.

UCWI306 – Show deployment steps

The deployment process is composed of three steps (init, plan, apply) for provisioning and configuring the needed infrastructure, and then the same steps are applied to the application, for deploying and starting its containers. The process may take some time and the user should be informed about its progress and the result of each phase. The DO Wizard user interface within the DevOps Framework, will allow the user to see the progress of the deployment process and the status of each of the above mentioned steps.

3.1.2 ADAPT MM Use Cases

ADAPT Monitoring Manager (MM) is in charge of monitoring the multicloud application deployed by ADAPT DO at real time, with respect to the NFRs defined in the MCSLA, and assesses them against the SLOs defined in the MCSLA. In case a violation occurs, it informs the Violation Handler (VH) so that the corresponding actions are launched. ADAPT MM also requests ACSmI to start monitoring the resources (Cloud Services) where the different components of the multicloud application are deployed.

The Figure 2 below shows ADAPT MM’s use cases:

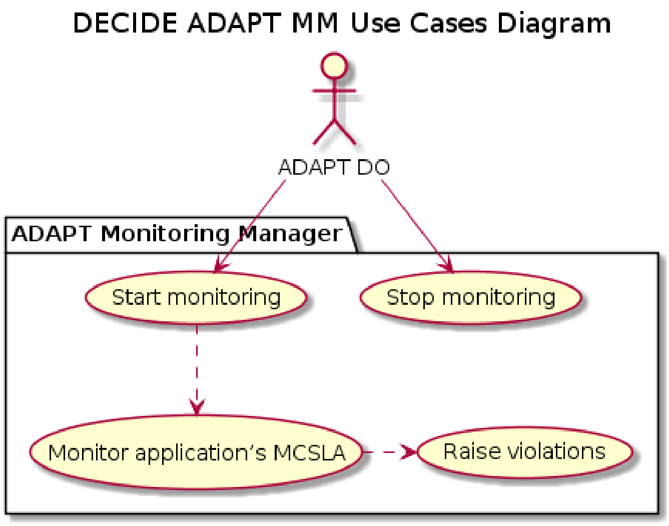


Figure 2. DECIDE ADAPT MM Use Cases Diagram

UCMM01 – Start monitoring

Once the deployment of the application is triggered by the ADAPT Deployment Orchestrator, the monitoring request occurs. The ADAPT Monitoring Manager will receive the request for starting the monitoring of the application. The ‘Start monitoring’ process will include the following actions:

- Request the SLOs and the NFRs to be monitored. These values are obtained from the application description through the application controller.
- Request the list of microservices of the application that have to be measured and monitored. This information has to be previously defined by the user, and is also obtained from the Application Description.
- Configure the data collection means (i.e. agents, DB, etc.) along with the multicloud application components.
- Configure the monitoring dashboards to show the selected measures and thresholds.

- Request ACSmI to start the monitoring of the CSPs.
- Update the aplication description to reflect the monitoring status and the dashboards location. This is done through the Application Controller.

UCMM02 – Monitor application's MCSLA

After the monitoring of the application has been configured, the monitoring process starts. The deployed agents continously provide measurements (sampling of the metrics) that are stored in the time series DB. The ADAPT MM components extract the data from the DB, aggregate the measures of the several microservices of the application through the MCSLA, and compare the resulting values with the established SLOs and MCSLA thresholds. At the same time, the monitored measures are provided to the multicloud application owner through the UI.

UCMM04 – Raise violations

The measured samples of the application data are periodically assessed against the SLOs and thresholds. When a violation is detected, the event is notified to the Violations Handler so that the corresponding actions are triggered.

UCMM05 – Stop monitoring

When ADAPT MM receives the request for stopping the monitoring for any of the components of the application, the following actions need to be performed:

- Stop the monitoring agents and release the corresponding resources.
- Stop the calculation and assessment of the aggregated measurements.
- Send a request to ACSmI to stop assessing the CSP metrics.
- Mark the corresponding registries in the Application Description as not being monitored
- The corresponding measures will not be shown any more in the user interface.

3.1.3 ADAPT VH Use Cases

ADAPT Violations Handler (VH) is the component in charge of receiving and processing violations, which are detected by ACSmI's and ADAPT's monitoring components. When a violation occurs, the Violations Handler notifies the operator and, depending on the technological risk of the application, automatically redeploys the application (low technological risk) or informs the operator that a redeployment is required (high technological risk).

The Figure 3 below shows the VH's use cases:

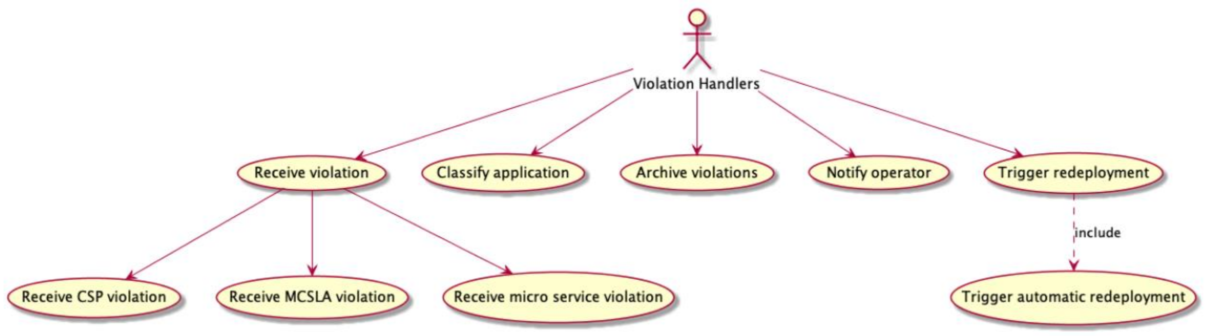


Figure 3. Violations Handler's use cases

UCVH01 – Receive violation

Whenever a violation occurs, it is received by the Violations Handler in order to be processed. These violations, which are sent by either ACSmI Monitoring or ADAPT Monitoring, can be generated due to a failure in the MCSLA, microservice or the CSP itself. The actions that will take place when a violation is received are indicated in the following Table 2. Please note that for an automatic redeployment to take place, the technological risk of the application must be low. If it is high, the redeployment will be performed manually.

Table 2. Actions resulting from received violations

MCSLA failure	Microservice failure	CSP failure	Action
X	X		A redeployment is required but the code must be changed (not automatic)
X		X	Trigger redeployment (execute use case UCVH05)
X	X	X	Trigger redeployment (execute use case UCVH05)
	X		A redeployment is required but the code must be changed (not automatic)
		X	Trigger redeployment (execute use case UCVH05)
	X	X	Trigger redeployment (execute use case UCVH05)

UCVH02 - Classify application

The Violations Handler must obtain the technological risk of the application from the Application Description. Depending on its value, it will automatically redeploy the application (if the technological risk is low) or ask the operator to choose a new deployment configuration before redeploying the application (if the technological risk is high).

UCVH03 - Archive violations

The VH is in charge of storing the received violations, which will be visualized within ADAPT Monitoring Manager. The violations will be stored either in the Application Description or in a local database.

UCVH04 - Notify operator

When the VH receives a violation, the operator will be notified by means of an email message.

UCVH05 - Trigger redeployment

If the application is classified as “low technological risk”, upon receiving a violation, the VH will automatically trigger a new simulation through OPTIMUS, which will, in turn, instruct ADAPT to deploy the application according to the newly found deployment configuration. Otherwise, the following UCVH06 will be executed.

UCVH06 - Trigger automatic redeployment

If the application is classified as “high technological risk”, upon receiving a violation, the VH will give the operator the possibility to choose a new deployment configuration before redeploying the application.

3.2 Analysis of future functionalities

This section offers a preliminary analysis of further functionalities that could be added to ADAPT in future development cycles, either by the DECIDE Alliance or by the Open Source community. The functionalities are not planned to be implemented in Y3, but at least analysing them lays the basis for planning future DECIDE releases. In this section we are considering the following functionalities: new monitoring metrics on redeployment, support for stateful containers, support for deploying containers on Kubernetes, support for Windows applications.

3.2.1 Support for stateful containers

A container storing data on its own filesystem will lose them upon restart: containers are inherently stateless, but real life applications usually need some state, i.e. need some persistent storage for their data. A stateless application, e.g. a calculator, does not need to keep information about its client across requests; a stateful application, e.g. a database, is made to keep information persistent.

Dealing with persitent data in a microservices architecture is not so easy. One can start with an architecture that wraps all the data in a single DataService microservice, but then may end up with one that couples each microservice with its own data [5]. In any case some microservice needs access to persistent storage. Since we deploy microservices into containers, this means that usually at least one container for each stateful application needs some storage.

Docker supports two main ways of writing data to persistent storage from a container: volumes and bind mounts. Bind mounts grant to a container direct access to the host filesystem, which is faster and powerful since allows direct interaction with the host and other non containerized processes, but also has security implications if the container mounts sensitive parts of the filesystem. Moreover, data written in this way are available only to containers running on that specific host. Volumes, as an alternative, allow to store data in a local named area directly managed by Docker or even –through specific plugins– to a remote NFS share or to a cloud storage service such as Amazon EDS.

In the case of a multicloud application, the right way to use storage from a container is to attach a volume to it, and use that volume to access cloud storage from some provider.

Docker allows to define and use volume drivers to provision and manage storage volumes backed by external storage such as cloud storage providers. Multiple open source drivers exist, such as REX-Ray [6] and NetShare [7], both licensed under Apache v2 open source license. REX-Ray for example supports several different cloud storage providers (e.g. Amazon, Google, Azure) and has a client-server architecture: the server should run on some machine with Internet access (e.g. the one hosting the containers) or can even run as a container itself[1]. Volumes handled by such volume drivers should be attached to a container when starting it, using the -v Docker option; see [8] for some examples.

For this to work in ADAPT, some extensions are needed. ADAPT DO should run the volume driver server, either as a container or as an independent process, on the same host where the container that needs the storage should be deployed. Then, when starting that container, ADAPT DO should provide the proper parameters in the `docker run` command to identify the volume driver, the named volume to be attached and its mountpoint within the container. All the information needed, plus those to configure the volume driver and to access the cloud storage, should be defined by adding new specific fields in the Application Description.

Support for cloud storage attached to containers should also be added in other DECIDE tools, for the whole lifecycle to work. The developer should be able to express the need for one or more volumes in

[1] Docker (starting with version 1.13) now supports a new plugin architecture in which plugins (such as the Rex-ray server) can be installed as containers

the DevOps Framework, when defining an application's microservices, along with their Non Functional Requirements; typically the default location of a volume should be the same as that of the microservice to which it should be attached, but the developer should be able to change it. OPTIMUS should be able to find the best provider and storage type (e.g. object storage, block storage, file system) for the given storage requirements (e.g. cost, location, microservice classification) and then add the related storage service into the deployment schema. ACSmI should be able to discover, contract and provision cloud storage providers, adding the related details in the Application Description and their credentials in the application's Vault.

The analysis above indicates what should be implemented to support a first deployment of stateful containers. To also support redeployment for stateful applications, something more is needed. The base simplifying assumption here is that when an application gets redeployed, its storage volumes remain the same, mainly to avoid the burden of moving all the stored data. This means that OPTIMUS, when calculating a new schema, should not define new storage services but will reuse the eisting ones. ADAPT DO will follow the same steps executed for a first deployment, but will avoid formatting the volume (this may be needed for example when using a block storage volume for the first time). Moreover, some syncronization mechanism should be implemented to ask for an application's permission before shutting it down for redeploying it elsewhere, to avoid data corruption.

3.2.2 Kubernetes Support

So far DECIDE ADAPT is able to deploy containers on generic Infrastructure as a Service (IaaS) cloud providers, as long as they are supported by ACSmI. ADAPT DO creates the Virtual Machines (VM) indicated in the Application Description on their respective provider through the ACSmI provisioning interface, installs Docker on them and deploys each container on the VM indicated in the schema.

Another option for running containers in the cloud can be to use some Container-as-a-Service (CaaS) engine, usually based on Kubernetes. Some possible public CaaS services are Google Kubernetes Engine (GKE) [9], Amazon Elastic Kubernetes Service (EKS) [10] and Azure Kubernetes Service (AKS) [11], but an already existing Kubernetes cluster on the customer's cloud may also be used.

OpenShift can also be considered as a deployment target for the application's containers, but since OpenShift is based on Kubernetes, here the more general case of Kubernetes deployment is analysed.

Kubernetes (K8s for short) is an open-source container orchestration system for automating deployment, scaling, and management of containerized applications [12]. To deploy containers it is needed a running K8s cluster, which is composed by at least one master node and multiple worker nodes; nodes represent the virtual (or physical) machines where containers are deployed. The unit of deployment in K8s is called a Pod, consisting of a single container or a small group of containers sharing the same storage and network resources. In particular each Pod has its own IP address, which is one of the major differences between Kubernetes and plain Docker: in Docker containers deployed on the same VM share that VM's IP address, whereas in K8s a node may host multiple Pods, each with a different IP address. Horizontal scaling of an application is obtained by replicating Pods, usually on different nodes; the K8s replication mechanism, handled by the ReplicaSet object, can maintain a given number of instances of a Pod running, restarting them as needed. Multiple ReplicaSets can be managed by a higher level object named Deployment, allowing declarative, server-side rolling updates of ReplicaSets and Pods. Despite the apparent complexity, running a small K8s application can be quite simple [13].

The major issue from the DECIDE point of view is that using Kubernetes an application is deployed on a single cloud, whereas our goal is a multicloud (or, better, cross-cloud) application deployment.

At the moment there are multiple open source tools and products that use the keywords combination "multi-cloud Kubernetes" as a selling point. These tools can be classified in multiple categories. There are tools (e.g. Containership [14], NetApp [15]) that allow to create K8s clusters on multiple clouds and possibly manage the resulting infrastructure, even with a dashboard. Other tools, such as Cloudify [16], allow some degree of multicloud orchestration that allows to set up a multi-cloud infrastructure including Kubernetes and more in a single shot. Finally a multi-cloud controller, such as Crossplane [17], enables workload portability across disparate environments and clouds. Any of the listed tools can be used by ADAPT to create the multi-cloud K8s infrastructure needed to deploy the application.

A further step after creating the infrastructure is to ensure containers' communication across clouds even in case of redeployment and the first obvious problem to be solved is that of *service discovery*. In a single K8s cluster a resource can be accessed by name using the local DNS, but to find services running in a different cluster a higher level DNS is needed, which aggregates the DNS entries of different clusters at least for the application at hand. The first version of the Kubernetes Federation effort did some work in this respect [18], but it is now deprecated and the corresponding v2 functionality is not yet ready [19].

Another possibility to setup K8s cross-cluster communication is to use Istio [20], which also solves the second issue raising immediately after service discovery: the connectivity among the two (possibly overriding) "local" address spaces of the two K8s clusters. The two main possibilities to solve this problem are Gateway Connectivity and VPN Connectivity [21]. In the first case communications from one cluster to the other are directed to the public IP address of the Istio Gateway of the other K8s cluster, whereas in the second case the participating clusters share a common address space through a VPN connection.

Given the richness of available tools there are multiple possibilites for ADAPT of supporting Kubernetes as a target to deploy multi-cloud applications. Three possible solutions are summarized in the following Table 3. Solution A involves different tools for each activity, whereas solution B is mostly based on Cloudify; both A and B rely on Virtual Machines contracted by ACSmI on the providers selected by OPTIMUS. In solution C the idea is to use one or more Container-as-a-Service (CaaS) engines as out-of-the-box K8s clusters, thus skipping the cluster creation phase.

Table 3. Some possible solutions for ADAPT to support Kubernetes

Phase \ Solution	A	B	C
Contracting	ACSmI contracts VMs	ACSmI contracts VMs	ACSmI contracts CaaS
Orchestration config	Terraform generated by DO	Cloudify generated by DO	Terraform generated by DO
Provisioning	Terraform + ACSmI	Cloudify + ACSmI	Terraform + ACSmI
K8s clusters creation	Containership	Cloudify	Not needed
K8s configuration	Generated by DO	Cloudify	Generated by DO
Cross-cluster communication	Istio	Istio	Istio
Containers' deployment	Terraform	Cloudify	Terraform

In summary the contracting phase can be performed by ACSmI, either for VMs (A, B) or for a provider offering a Kubernetes engine (C); ADAPT DO generates the whole orchestration configuration from the given Application Description, either for Terraform (A, C) or for Cloudify (B), and then activates the respective tool to enact the generated configuration; resource provisioning is performed through ACSmI by the selected orchestration tool, assuming that ACSmI will support also provisioning Container-as-a-Service providers; the creation of the Kubernetes cluster infrastructure, when starting from plain VMs, can be performed from an *ad-hoc* tool such as Containership (A) or be part of the Cloudify configuration (B); the K8s configuration could be generated by DO (A, C) or be part of the Cloudify configuration (B); the cross-cluster communication functionalities could be ensured through Istio in all the three possible solutions, and finally the deployment of the applications' containers can be done by Terraform (A, C) like in the current implementation or by Cloudify (B).

3.2.3 New monitoring metrics

In order to provide to the user comparative data about how the different deloyments affect application efficiency (e.g. SLA enhancement/worsening after a redeployment), historical data of the application performance have to be stored. It has to be taken into account that, after a redeployment, the original application is not being monitored anymore, so the data about its performance, availability, or others, must have been previously gathered and inserted in the database. When a comparison is needed, a search has to be done in the DB to extract the previous data and compare it with the actual data of the application. Or simply, both measurements can be shown in the Monitoring UI.

Potential future functionalities regarding metrics monitoring include:

1. Machine learning capabilities to detect and predict abnormal behaviour (e.g. response times) of the application: When operating an application, it is interesting to be able to provide mitigation measures to be able to minimize the risks of being unavailable, even if it is for a very short period of time, with the consequences of risking business continuity and decrease customer satisfaction. A possible way to minimize such risk is to analyse and understand the data and parameters that lead to a malfunctioning of the application. Through machine learning techniques, predictive analysis functionalities could be developed, by analysing the different trends that lead to a problem in the provisioning of the application. Another possibility could be to use predictive analytics to predict performance peaks and propose alternative deployment topologies. The current version of ADAPT MM stores in a time series database, namely InfluxDB, the actual measures but while the current scope is to compare real actual values with the established thresholds and raise alerts, the proposed functionality goes beyond, as it seeks to continuously procress trends and raise an issue whenever an unstable situation is detected. A potential competitor could be Kibana[2]. The baseline technologies of Kibana are somewhat different than the ones used in ADAPT MM but they are complementary.

2. Application performance monitoring (APM) metrics such as transaction duration, and requests per unit of time, and so on. APM has been in the market for a long time. However, the integration of APM with the current monitoring metrics covered by ADAPT MM would provide richer information to the Operator of the multi-cloud application. In this respect, the metrics that could be considered are:
    a. Requests flowing among the different services of the multi-cloud application to understand where the bottlenecks of the application are and optimize it.
    b. Duration of the transactions: to understand on one hand, the user behaviour and improve the design of the application, and on the other, to analyse how much time, e.g., queries need to be successfully executed in order to optimize them.
    c. Requests supported per unit of time: similar to stress tests, to understand how well the application can scale.

[2] https://www.elastic.co/es/products/kibana

To this end, tools like Kibana could be integrated.

3. Monitoring of Kubernetes environments: As one of the potential future functionalities of ADAPT DO, it has been identified the deployment of the multi-cloud application in a Kubernetes cluster. The main goal of ADAPT MM would be to monitor the health of the containers deployed on the Kubernetes infrastructure. To achieve richer metrics, it is necessary that ACSmI and ADAPT MM work together. In this respect ACSmI monitoring should monitor:
    a. the underlying VMs, already covered in the current prototype of ACSmI (see D5.4) with some extensions such as the number of the Kubernetes nodes, the status of such nodes, the number of pods running per node, and the aggregated value of the resource utilization per namespaces, node and pod.
    b. The Kubernetes infrastructure, such as the health of the pods, requests commitment - a request commitment is the ratio of requests, memory and CPU, for all pods running on a node compared to the total resources available on that node, API requests, status of the stateful containers.

    Some of the tools that can be extended to achieve this is Prometheus [3].

3.2.4 Windows Support

At the beginning of the DECIDE project, Containers was a Linux-only technology, aimed at packaging an application into a single environment with all its configuration and dependencies. This package is portable and can run on any host offering the right container runtime. Containers are also lighter than Virtual Machines: a VM emulates the HW and includes the whole environment of a complete machine, whereas a Container runs on the same HW of its host and shares the Linux kernel with it, hiding other workloads running on the same host through the Linux-based mechanisms of namespaces and cgroups. Finally Containers run a read-only image and write on a temporary filesystem which is not persisted on termination, fitting perfectly the stateless workload use case.

The brief description above clearly shows the dependencies of Container technology from the Linux Operating System (shared kernel, namespaces, cgroups, not to mention the layered and Copy on Write filesystem).

Despite this heavy dependency on Linux, the Container technology was very appealing also for Windows customers (we have an example of this in DECIDE with the AIMES use case). The first version of Docker on Windows was based on a Linux VM providing the environment to run only Linux containers. Then Microsoft worked hard to provide Container features on Windows, and reached a first milestone with the Windows 10 and Windows Server 2016 versions of their Operating System.

The kernel-sharing feature of Containers seems a hard wall between the two Operating Systems, constraining Linux Containers to run on Linux and Windows Containers to run on Windows. To obtain a more flexible mix-and-match between the two OSs a further layer would be needed: Linux emulation on Windows and Windows emulation on Linux.

The latter is available since a long time: Wine (whose acronym originally was Wine Is Not an Emulator) is an open source project started in 2003 to provide a “compatibility layer capable of running Windows applications on several POSIX-compliant operating systems, such as Linux” [22]. Running Windows 10 into Wine as a base for hosting Windows containers is unthinkable, but at least running a Windows application over Wine in a Linux container is worth a try: the pre-built Docker-wine image can be useful for this [23].

[3] https://prometheus.io/docs/introduction/overview/

Linux emulation on Windows is available with the Windows Subsystem for Linux (WSL), which is a compatibility layer for running Linux binary executables (in ELF format) natively on Windows 10 and Windows Server 2019 [24]. One thus would expect to be able to run Linux Docker directly on Windows, therefore providing an environment to run native Linux Containers on Windows, but so far the Linux VM is still needed for this (the related issue is still open in GitHub and an answer is still to be expected [25]). In the future it will be possible to run the Linux Docker daemon on Windows (but still with a Linux VM under the hood) since Microsoft has just announced that it will include a Linux kernel within Windows for the version 2.0 of WSL [26].

The four possible combinations of Linux and Windows as Host OS and Container OS are summarized in the following Table 4.

Table 4. Possible combinations of Linux and Windows as Host OS and Container OS

Host OS \ Container OS	Linux Containers	Windows Containers
Running on Linux	Any Linux distribution running a 3.10+ kernel can run Docker containers. The container shares the same kernel with the host.	No way to run directly Windows Containers, the only option seems to be running the desired Windows application over Wine in a Linux container.
Running on Windows	Windows Subsystem for Linux (WSL) is a compatibility layer for running Linux binary executables (in ELF format) natively on Windows 10 and Windows Server 2019 [24]. Docker for Windows already runs Linux Containers using a Linux VM to run the Docker daemon; WSL 2.0 will run the same, just hiding the Linux VM.	Windows 10 and Windows Server 2016 and 2019 can run Windows containers. Two modes are available: Server containers sharing the kernel with the host, Hyper-V containers have their own kernel and run in a light isolated VM.

Since DECIDE is currently running on Linux, the only option for ADAPT to support the deployment of containerized Windows applications is to use Wine. Developers will just have to create their containers layering the application over Wine (the pre-built Docker-wine image can be a starting point for these containers [23]). These containers will be described in the Application Description like normal containers, without even specifying that they run a Windows micro-service Then the normal DECIDE workflow will do the rest for both deployment and redeployment. The only attention point is that to obtain (local) persistence of data a “winehome” volume should be created, as indicated in the docker-wine image documentation, and this volume makes the container stateful (see section 3.2.1 for a discussion of stateful containers).

4 DECIDE ADAPT Final Architecture

The internal architecture of DECIDE ADAPT has not changed significantly from its initial definition. Its main components are still Deployment Orchestrator (DO), Monitoring Manager (MM) and Violations Handler (VH). What has been enhanced is the set of its interfaces with other DECIDE components, indicating an improvement in the interaction and synergy with the DECIDE Framework itself. The ADAPT internal components, along with their internal and external interfaces, are shown in Figure 4.

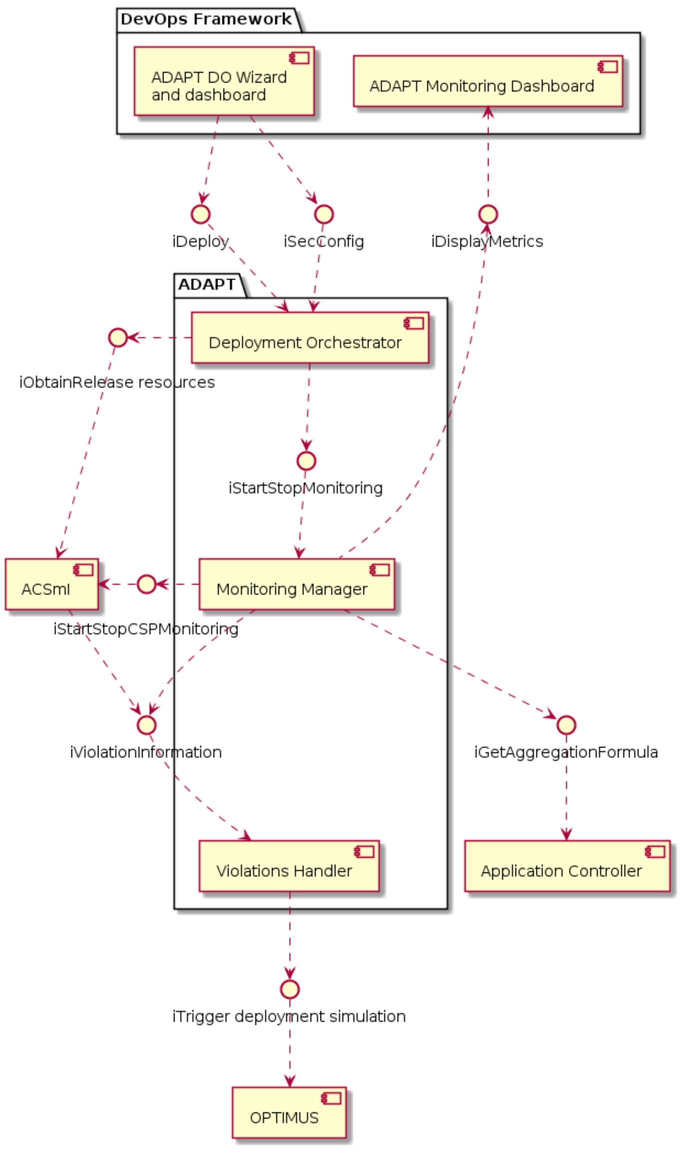


Figure 4. DECIDE ADAPT Final Architecture

The changes from Y2 to Y3, which are visible at this high architectural level, are the following:

- New iSecConfig interface for ADAPT DO (see sect. 4.1)
- New ADAPT DO Wizard and dashboard component (see sect. 4.1)
- New ADAPT Monitoring Dashboard component (see sect. 4.2)
- The iStoreAlertInformation interface has been removed, as VH now stores violations in a local database

4.1 ADAPT DO architecture

The Y3 architecture of ADAPT DO is based on the architecture that was documented in Y2 deliverable D4.2 [1]. An additional component has been added in the DevOps Framework to provide a user interface for configuration and progress/status visualization. An interface, called by the graphical component, has also be added to the REST API existing component. The new component and interface are shown in Figure 5 and are described below.

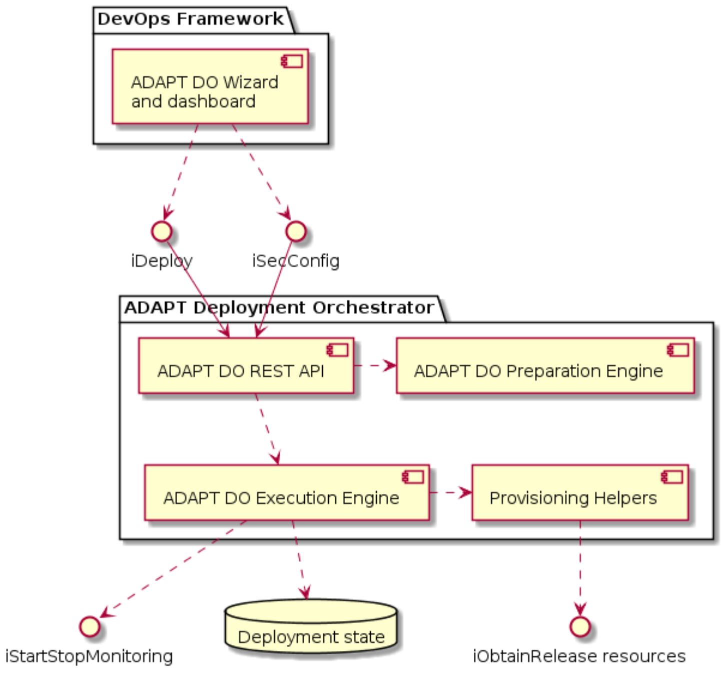


Figure 5. ADAPT DO components and interfaces

ADAPT DO Wizard and Dashboard

This component implements use cases from UCWI301 to UCWI306 and is integrated as an inline frame (iframe) in the DevOps Framework graphical user interface. It has been called Wizard since the idea is to guide the user along the multiple steps that define the configuration deployment details. The last step of the wizard will implement the DO Dashboard showing the progress of the deployment process and the status of each of its steps (use case UCWI306).

iSecConfig

This interface allows to configure some security-related aspects that are needed for the deployment of a user application. It implements use cases UCDO301 and UCDO302.

4.2 ADAPT MM architecture

The architecture of ADAPT Monitoring Manager (MM) presented here is an updated version of the one documented for the M24 prototype, in the previous deliverable D4.2 [1]. No new components have been added or removed to the architecture.

ADAPT MM will gather information stored in the Application Description from the Application Controller (the threshold values for the different metrics to be assessed plus the aggregation formula), and from the Violations Handler (historical information about the alerts).

ADAPT MM will receive requests from ADAPT Deployment Orchestrator to start/stop the monitoring.

Figure 6 below shows the internal sub-components and external interfaces of the ADAPT MM.

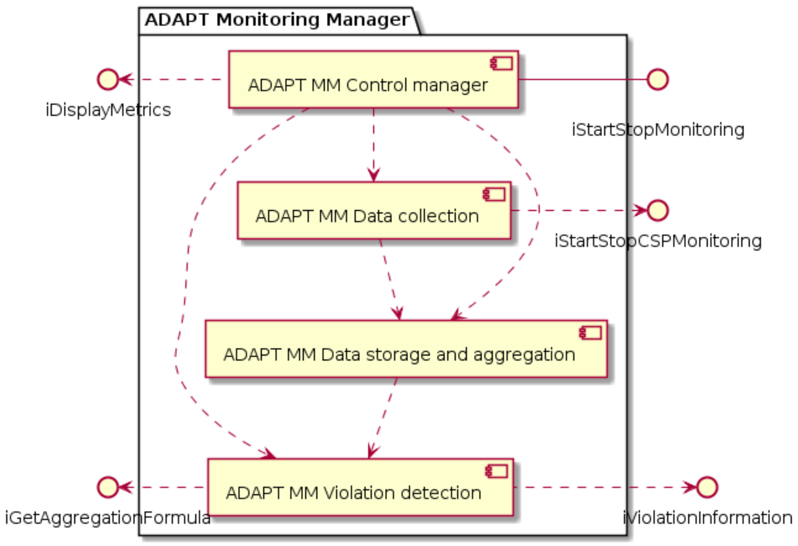


Figure 6. ADAPT MM Internal sub-components diagram

Figure 7 below shows the communications of ADAPT MM with other DECIDE components.

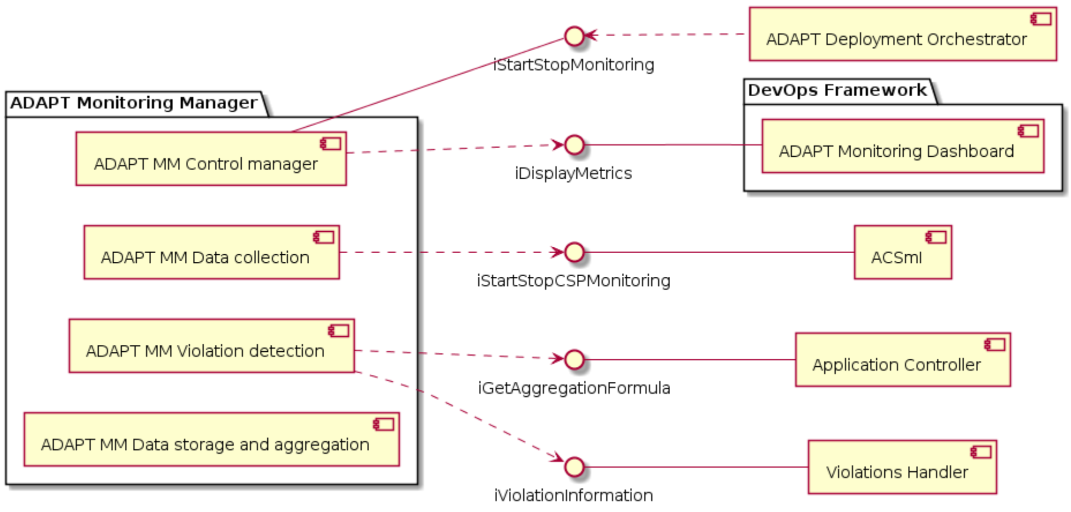


Figure 7. ADAPT Monitoring Manager (MM) external component diagram

ADAPT MM Control manager

This component manages the flow among the components of ADAPT MM. It is in charge of the different processes and requests that need to be triggered, chained and stopped inside the Monitoring manager.

The Control manager will implement the following processes and interfaces:

- iStartStopmonitoring. External interface with Deployment Orchestrator (ADAPT DO)
- iStartStopCSPMonitoring. External interface with ACSml, to start or stop the assessment of the SLOs corresponding to CSPs related metrics.
- Starts the component Data collection
- Starts the component Data storage and aggregation
- Starts the component Violation detection

- Starts the component UI by calling iDisplayMetrics

ADAPT MM Data collection:

This component will collect the data (metrics) from the application microservices. The Data collection component will be based on agents that may need to be deployed withtin the different components and cloud resources. The agents will be pre-defined and pre-implemented to be installed when deploying the multi-cloud application. These time series metrics will be stored for further analysis of the data. ADAPT MM will process and combine the metrics measured in real time.

ADAPT MM Data storage and aggregation

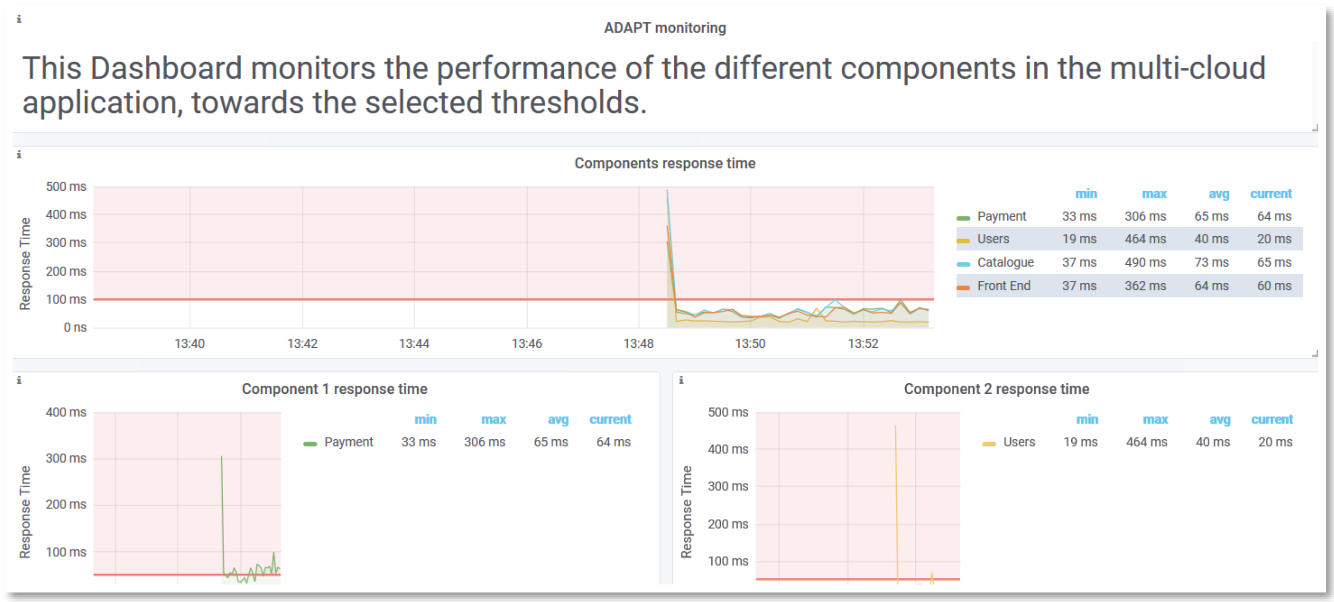
This component will be in charge of storing the data collected from the Data collection component, and aggregate it to create the actual measures that will be assessed by the Violation detection component.

ADAPT MM Violation detection

This component will be in charge of assessing if the required working conditions are or are not being met. The Violation detection needs to get the SLOs for the different metrics and inform the MCSLA about which aggregation formula would be used. If a violation of any of the NFRs or SLOs is detected, ADAPT MM components will inform the Violation Handler component which, in turn, will generate the adaptation action depending on each situation and context.

ADAPT Monitoring Dashboard (UI)

This component is the graphical user interface (UI) for the ADAPT MM component. It will provide the means for the operator of the multi-cloud application to visualize the metrics at run time, and the information of the violations occurred.



ADAPT monitoring

This Dashboard monitors the performance of the different components in the multi-cloud application, towards the selected thresholds.

Components response time

	min	max	avg	current
Payment	33 ms	306 ms	65 ms	64 ms
Users	19 ms	464 ms	40 ms	20 ms
Catalogue	37 ms	490 ms	73 ms	65 ms
Front End	37 ms	362 ms	64 ms	60 ms

Component 1 response time

	min	max	avg	current
Payment	33 ms	306 ms	65 ms	64 ms

Component 2 response time

	min	max	avg	current
Users	19 ms	464 ms	40 ms	20 ms

Figure 8. Example of an ADAPT MM UI

4.3 ADAPT VH architecture

The architecture of the Violations Handler has not changed since the last version of this document. For clarification purposes, the following Figure 9 shows said architecture:

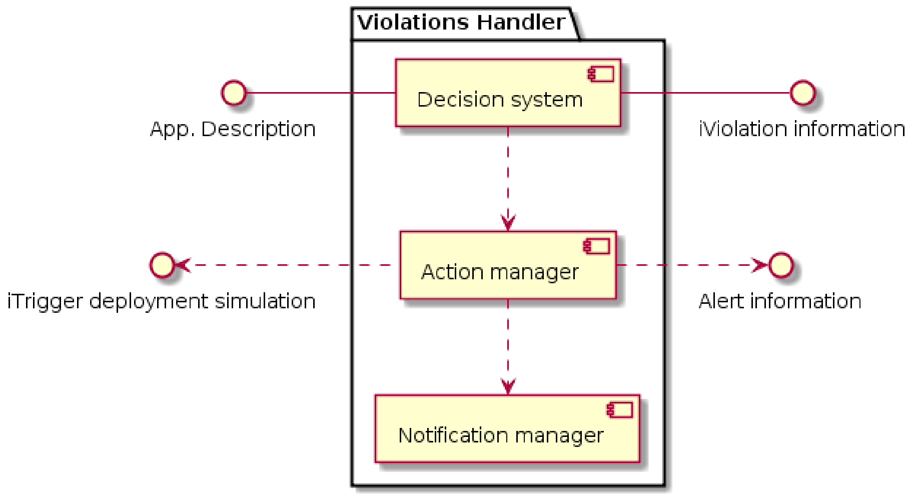


Figure 9. ADAPT Violations Handler component diagram

The *Decision system* receives the violation and requests the Application Description the application's technological risk. Depending on this variable, it will instruct the *Action manager* to perform the corresponding actions:

- Instruct the *Notifications manager* to notify the operator. The notification will be done through an email that will contain information regarding the alert
- Send a message to OPTIMUS, containing the necessary data to start a new simulation process

The *Action manager* will also store the received alerts either in the Application Description or in a local database to keep a history of alerts and to visualize them from ADAPT's UI.

Figure 10 shows these external interfaces:

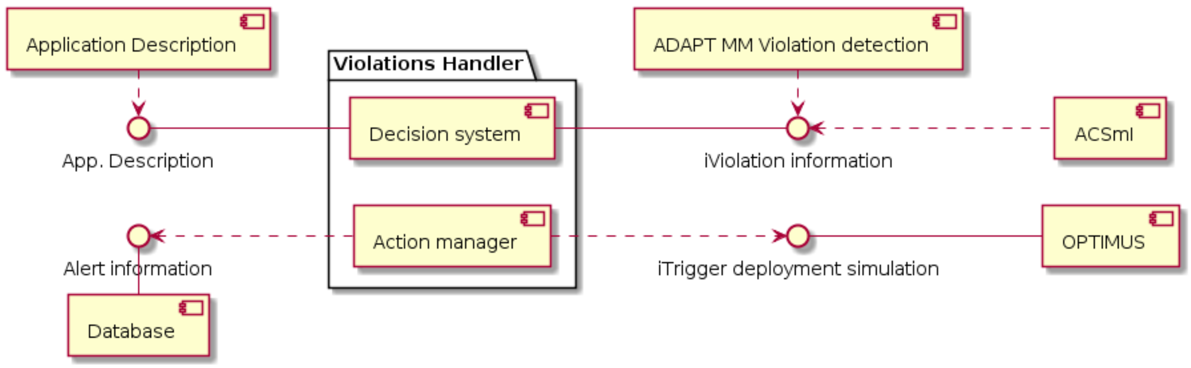


Figure 10. ADAPT Violations Handler external interfaces diagram

5 DECIDE ADAPT Interfaces

DECIDE ADAPT integration is at the final stages and new interfaces will be documented in more detail in the implementation deliverables, namely, D4.6 [2] and D4.9 [3]. This section shows a high level overview of the current status.

5.1 ADAPT integration

This section shows all the integration interfaces for ADAPT, both internal and external, and their implementation and integration status, with a short indication of what has changed from Y2 to Y3.

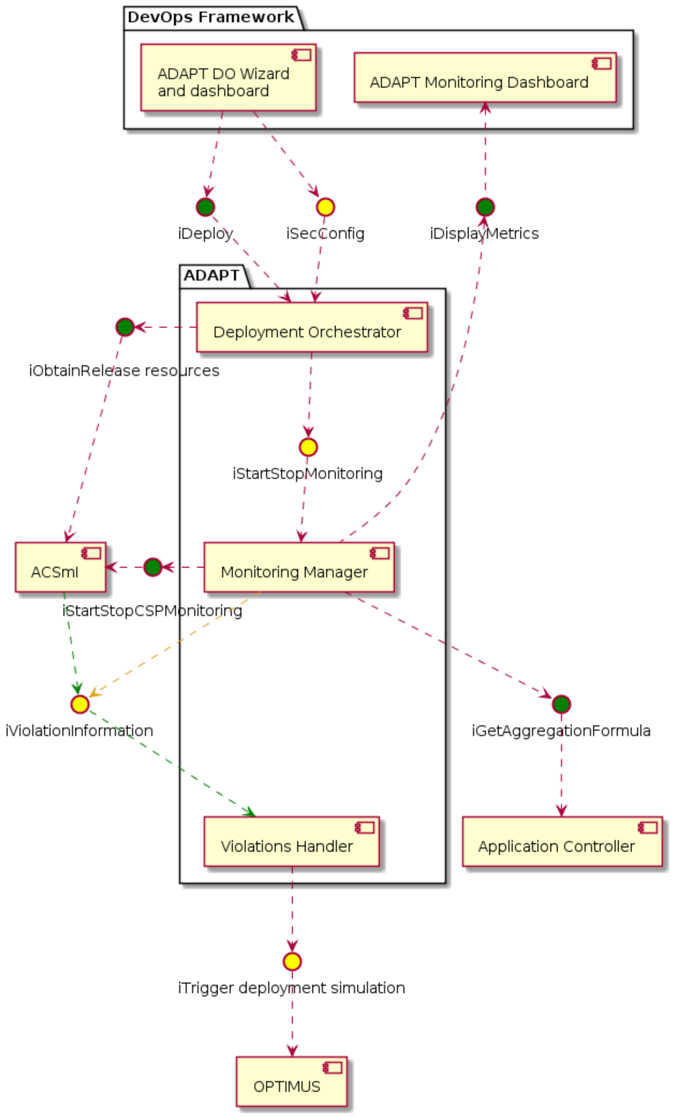


Figure 11. Internal and external ADAPT interfaces and their status

Figure 11 shows, color coded (Green: completed, Yellow: to be integrated), the implementation and integration status of each interface at the time of writing (M30). The following Table 5 shows the current implementation and integration status for ADAPT Internal and External interfaces.

Table 5. Current status of ADAPT interfaces

Interface	I/E	Designed	Coded / Tested	Integrated	Comments
iDeploy	E	Yes	Yes	Yes	Integration to be updated with the new Wizard
iDisplayMetrics	E	Yes	Yes	Yes	URL written in the App Description, DevOps Framework reads and loads it
iGetAggregationFormula	E	Yes	Yes	Yes	The interface calls a library that reads ISO standard metrics description language from Application Description and returns the aggregation formula
iObtainReleaseResources	E	Yes	Yes	Yes	V1 integrated, V2 (extension for firewall rules) implemented, V3 (private cloud) implemented
iStartStopCSPMonitoring	E	Yes	Yes	Yes	Integration approach revised in Y2, from common repository to REST interface
iStartStopMonitoring	I	Yes	Yes	No	Integration ongoing
iTriggerDeploymentSimulation	E	Yes	Yes	No	Integration ongoing
iViolationInformation	E	Yes	Yes	No	Design recently revised, implementation and integration ongoing
iSecConfig	E	Yes	Yes	No	to be integrated with the new Wizard

5.2 Application Description

The Application Description (AD) is the main way of exchanging complex information between all DECIDE tools. The last version of the whole list of AD fields will be documented in appendix A of deliverable D2.8 (*Final DECIDE DevOps Framework Integration*) [27]. This section summarizes the modifications occurred in this last year to information exchanged by ADAPT through the AD.

Microservices.Safemethods:

Safemethods are a list of uris that are defined by the developer to monitor each microservice of the application, without interfering in the state of the application itself. Previously defined as string urls, finally the decision was to represent them only with the pair {path, port} for clarity purposes, and because the rest of the information in the url (protocol, domain) is deemed as not relevant, especially after the deployment of the services.

Used by: ADAPT MM (read)

Example

```
"safeMethods" : {
    "path" : "/customers",
    "port" : "18082"
  },
```

Microservices.name, Containers.containerName:

The name of the microservice and the name of the container in which it is deployed are the same, by consensus. These fields are used to navigate from the microservices to the correspon ding container.

Containers.dockerHostNodeName, virtualMachines. dockerHostNodeName:

These fields of the same name in containers and virtual machines are used by the ADAPT MM to know in wich virtual machine is deployed each container.

virtualMachines. dockerHostPublicIP, :

This field contains the IP of the virtual machine, that is needed by the ADAPT MM to form the safemethods URL after deployment (roughly is IP + port + path), and thus to know where to address the inquiries to monitor the corresponding safemethods.

Containers.portMapping:

This field contains a series of mapping of ports that each container can have defined, so that an incoming TCP request is conducted from the port used in the external call (hostPort) to another port inside the container (containerPort). This mapping is needed for the ADAPT MM to know which port it has to use to call form the outside to a safemethod located in that container.

Used by: ADAPT MM (read)

Example

```
"containerName" : "catalogue"
...
"dockerHostNodeName" : "node-2",
...
"dockerHostPublicIp" : "82.223.2.216",
...
"portMapping" : [ {
      "hostPort" : "8480",
      "containerPort" : "80"
    } ],
```

Nfrs:

This field contains a list of NFRs (Non-Functional Requirements) of the application. ADAPT MM needs to know which are those requirements, which it has to control the measurements against. The type field indicates the kind of nfr, being the “Availability” and “Performance” types those that are controlled by ADAPT MM .

Used by: ADAPT MM (read)

Example

```
"nfrs" : [ {
    "type" : "Performance",
    "tags" : [ "Application" ],
    "abstractValue" : "High",
    "value" : 100.0,
    "unit" : "milliseconds"
  }, {
    "type" : "Availability",
    "tags" : [ "Application" ],
    "abstractValue" : "Medium",
    "value" : 98.0,
    "unit" : "percentage"
  } ],
```

Monitoring:

This part of the AD is a composed field, that contains the “status” of the monitoring (true if the aqpplication is being monitored / false if not); and a list of the Grafana dashboard urls, that is included in the “url” field. This list is used by the Framework to insert these dashboard in its GUI.

Used by: ADAPT MM (write)

Example

```
"monitoring" : {
    "status" : true,
    "urls" : [ "http://85.91.40.245:8093/d/HbAMQcWik/performance-
metrics-for-multi-cloud-application?orgId=1" ]
  },
```

5.3 Secrets stored in Vault

In Y3, a new component, based on the Vault open source provided by Hashicorp, has been added to the DECIDE Framework to store sensible information in a secure way. It is a centralized component, to which all tools have access and can store and retrieve sensitive data as needed.

Information is stored in Vault according to the following path structure:

/[store_type]/[user_name]/[project_name],

where “store_type” depends on the type of data stored. Three types of data are expected: key/value pairs, (SSH) key pairs and certificates. Below, the information contained in Vault is detailed:

- **DevOps Framework:**
  - *gitURL*: URL of the git where the application description is stored
  - *gitToken*: token to access the previous git repository

- **ACSmI Discovery:**
    - *dis_username*: username of the account registered in ACSmI Discovery
    - *dis_password*: the password corresponding to the account registered in ACSmI Discovery

- **ACSmI Contracting:**
    - *con_username:* username to be used for contracting as well as for the account creation within ACSmI.
    - *con_password*: password of the ACSmI user account

- **ADAPT:**
    - *cb_username:* username to access the cloud broker platform APIs
    - *cb_password:* password to access the cloudbroker platform APIs
    - *publicKey:* public key stored in the cloud broker user profile, injected in the VMs started on the ACSmI-managed clouds
    - *privateKey:* private key which allow ssh access to the vms
    - *certificate:* certificate required to access the repo

6 Conclusions

This third year ADAPT architecture deliverable described the updates leading to the final version of DECIDE ADAPT's high-level architecture, which has been described using UML syntax, both for the Use Cases diagrams and the component and interfaces diagrams. The final architecture shows the updated version of both internal and external interfaces, along with the current status of their integration. For some functionalities that could be added to ADAPT in future development cycles, i.e. after the end of the project, a preliminary analysis has been provided.

7 References

[1] DECIDE, "Deliverable D4.2 - Intermediate DECIDE ADAPT Architecture," 2018.

[2] DECIDE, "Deliverable D4.6 - Final multi-cloud application deployment and adaptation".

[3] DECIDE, "Deliverable D4.9 Final multi-cloud application monitoring".

[4] DECIDE, "Deliverable D4.1 - Initial DECIDE ADAPT Architecture".

[5] S. Johnson, "Managing State With Microservices," Medium Corporation, 31 7 2018. [Online]. Available: https://medium.com/@scott_95329/managing-state-with-microservices-8a4952627a40. [Accessed 26 04 2019].

[6] "Rex-Ray, container storage orchestration engine," [Online]. Available: https://rexray.io/.

[7] "NetShare, Docker volume plugin for NFS 3/4, EFS and CIFS/SMB," [Online]. Available: http://netshare.containx.io/. [Accessed 26 4 2019].

[8] "REX-Ray source code on Github," [Online]. Available: https://github.com/rexray/rexray. [Accessed 26 4 2019].

[9] "Google Kubernetes Engine (GKE)," [Online]. Available: https://cloud.google.com/kubernetes-engine/. [Accessed 03 05 2019].

[10] "Amazon Elastic Container Service for Kubernetes (EKS)," [Online]. Available: https://aws.amazon.com/it/eks/. [Accessed 03 05 2019].

[11] "Azure Kubernetes Service (AKS)," [Online]. Available: https://azure.microsoft.com/en-us/services/kubernetes-service/. [Accessed 03 05 2019].

[12] "Kubernetes, Production-Grade Container Orchestration," [Online]. Available: https://kubernetes.io/. [Accessed 03 05 2019].

[13] B. Gunasekara, "Getting Started with Kubernetes: Deploy a Docker Container with Kubernetes in 5 minutes," [Online]. Available: https://codeburst.io/getting-started-with-kubernetes-deploy-a-docker-container-with-kubernetes-in-5-minutes-eb4be0e96370. [Accessed 03 05 2019].

[14] "Containership: multi-cloud Kubernetes made easy," [Online]. Available: https://containership.io/. [Accessed 03 05 2019].

[15] "NetApp Kubernetes Service (NKS)," [Online]. Available: https://cloud.netapp.com/kubernetes-service. [Accessed 06 05 2019].

[16] "Cloudify - Kubernetes Cloud Native Orchestration," [Online]. Available: https://cloudify.co/kubernetes. [Accessed 06 05 2019].

[17] "Crossplane, the open source multicloud control plane," [Online]. Available: https://crossplane.io/. [Accessed 06 05 2019].

[18] "Kubernetes - Cross-cluster Service Discovery using Federated Services," [Online]. Available: https://kubernetes.io/docs/tasks/federation/federation-service-discovery/. [Accessed 06 05 2019].

[19] "Kubernetes Federation v2," [Online]. Available: https://github.com/kubernetes-sigs/federation-v2. [Accessed 06 05 2019].

[20] "Istio - Connect, secure, control, and observe services," [Online]. Available: https://istio.io/. [Accessed 06 05 2019].

[21] "Istio Multicluster Installation," [Online]. Available: https://istio.io/docs/setup/kubernetes/install/multicluster/. [Accessed 07 05 2019].

[22] "WineHW - Run Windows applications on Linux, BSD, Solaris and MacOS," [Online]. Available: https://www.winehq.org/. [Accessed 08 05 2019].

[23] "Docker-wine, a Docker image that runs Wine on your Linux desktop," [Online]. Available: https://github.com/scottyhardy/docker-wine. [Accessed 08 05 2019].

[24] "Windows Subsystem for Linux," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux. [Accessed 07 05 2019].

[25] "Make Docker daemon work on WSL," [Online]. Available: https://github.com/Microsoft/WSL/issues/3255. [Accessed 08 05 2019].

[26] "Microsoft Announcing WSL 2," [Online]. Available: https://devblogs.microsoft.com/commandline/announcing-wsl-2/. [Accessed 08 05 2019].

[27] DECIDE, "Deliverable D2.8 Final DECIDE DevOps Framework Integration".