Grant Agreement No: 101016941

Full Title:	Open cooperative 5G experimentation platforms for the industrial sector Network Applications
Start date:	01/01/2021
End date:	30/06/2024
Duration:	42 Months
Project URL:	https://www.5g-induce.eu/

Handbook

Onboarding network applications to 5G-INDUCE platform

Document type	Handbook		
Title	Onboarding network applications to 5G-INDUCE platform		
Contractual due date	Not contracted	**Actual submission date**	20/09/2024
Nature	White paper	**Dissemination Level**	Pulic
Author	5G-INDUCE Consortium		
Contributions from	Q. Wang, J. M. A Calero, I. M. Alpiste, G. Golcarenarenji, J. Diez Tomillo (UWS); N. Lampropoulos, N. Giannakakos, S. Mazarakis (UNIS); D. Klonidis, T. Xirofotos (UBITECH); M. Arbesu, S. Garcia, M. M. López (YBVR); M. Fuentes, P. Trelis (5COMM); S. Koussouris, G. Constantinou (SUITE5); P. Zikos, N. Thomadakis (ILINK); J. Kämpfer (OCULAVIS); L. Korsic (ININ); L. Gonzalex (ABB/ASTI), C. Lessi (OTE), D. S. Cristóbal (ERC), F. Davoli, C. Lombardo (CNIT), G. Rizzi (WI3), E. Bosani (BEKO/WHMAN)		

Revision history

Version	Issue Date	Changes	Contributor(s)
V0.6	08/02/2023	ToC and drafting of Sections 2 &3 based on materials in D4.1. Discussed and agreed in the project's weekly telco on 08/02/2023	UWS
V0.7	31/03/2023	First draft ready, except Section 4	All the partners as assigned in V0.6
V0.8	12/03/2024	Clean and consolidated version considering CNIT's feedback, asking for inputs from UC leaders to Section 4	CNIT and UWS
V1.0	19/09/2024	Final version	UWS and CNIT

Table of Contents

List of Figures

List of Tables

Glossary of terms and abbreviations used

Abbreviation / Term	Description
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AR	Augmented Reality
CPU	Central Processing Unit
CS	Cloud Service
CVM	Common Validation Metric
DN	Data Network
DRAM	Dynamic Random Access Memory
E2E	End-to-End
ExFa	Experimentation Facility
GPU	Graphics Processing Unit
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MEC	Multi-Access Edge Computing
ML	Machine Learning
NAO	Network Application Orchestrator
NAT	Network Address Translation
Network Application	Network Application
NIC	Network Interface Controller
OSS	Operation Support System
PLC	Programmable Logic Controller
PNF	Physical Network Function
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RSSI	Received Signal Strength Indicator
SLA	Service Level Agreement
SLAM	Simultaneous Localization and Mapping
STUN	Session Traversal Utilities for NAT
TCP	Transmission (Transport) Control Protocol
TURN	Traversal Using Relays around NAT
UAV	Unmanned Aerial Vehicle

Abbreviation / Term	Description
UC	Use Case
UDP	User Datagram Protocol
UE	User End
VNF	Virtual Network Function
VR	Virtual Reality
WSS	Web Socket Server

Executive Summary

This 5G-INDUCE handbook aims to support potential third-party developers of Network Applications to prepare and onboard their Network Applications, in line with the 5G-INDUCE framework, to the 5G-INDUCE platform to create vertical services in a rapid and cost-efficient manner.

More specifically, the handbook includes examples and instructions regarding: a) the definition of the required Network Application components, b) the developments and onboarding of component images in the form of containerised functions, c) the component operational parameters that are provided to the 5G-INDUCE platform frontend, d) the definition of the network parameters through the Network Application graph composition, e) the initial definition of operating and application related policies, f) the definition of the full chain of the onboarding process.

The handbook assists in the proper preparation of Network Application operational, network, monitoring and policy requirements, in compliance to the onboarding methodology. It also provides a description of a Network Application use case from the deployment point of view and the generation of the respective graph.

The onboarding methodology and procedure describes the exact steps that a developer follows, in order to register and compose a Network Application graph and provide the component parameters, registration of the images, its deployment and the management of runtime policies.

Finally, a list of recommendations is presented at the end to offer the main takeaways of this handbook.

1 Introduction

1.1 5G-INDUCE project overview

5G-INDUCE offers an open, standard (ETSI NFV, etc.) compatible, 5G orchestration platform for the deployment of advanced 5G Network Applications. The platform's unique features provide the capability to the Network Application developers to define and modify the application requirements, while the underlay OSS can expose the network capabilities to the end users on the application level, without revealing any infrastructure related information. This process enables an application-oriented network management and optimization approach that is in line with the operator's role as manager of its own facilities, while it offers the development framework environment to any developer and service provider through which tailored applications can be designed and deployed over 5G networks, for the benefit of vertical industries and without any indirect dependency through a cloud provider.

1.2 Network Application concept

In 5G-INDUCE, in close alignment with other relevant projects, as well as the consolidated efforts of the 5GPPP Software Networks WG, a Network Application is defined as follows[1]:

Network Application is a set of networked virtualizable functions (also referred to as Application Components), together with the required resources (compute, storage, network, etc.), deployable and operating over 5G and beyond network infrastructures, and being distributed across the network continuum.

The **virtualizable functions** (application components) in a 5G system are split in the following three types:

1. The Customer-facing service virtualizable functions, denoted as virtual application functions VAFs – referring to the application functions that are typically developed by vertical service developers for particular vertical use cases (see section 4.5) and linked together in order to provide the required service functionality.
2. The Network service and resource-facing virtualizable functions, denoted commonly in standards as virtual network functions VNFs – referring to the standardised functions provided by the network operator and infrastructure owner for fulfilling the networking and connectivity services (e.g., User Plane Function (UPF), Access and Mobility Function (AMF), Session Management Functions (SMF), etc.).
3. The Value-added (middleware) VNFs or non-standardised VNFs – referring to network-level functions that are added in support of the networking operations and services either to satisfy specific end user network function requests (i.e., intermediately on demand) or as a response to network changes (i.e., automatically). Examples may include encryption and decryption VNFs for secure communications, video processing VNFs, and so on.

The considered distributed **network infrastructure** spans from Edge, Transport network, Core network and Data Networks (DNs) to Cloud networks.

➔ A Network Application SHALL employ VAFs, and MAY additionally utilise value-added VNFs.

➔ A Network Application SHALL be deployable over any part of the network infrastructure continuum

REMARK – A: Use case applications are not Network Applications. They become a Network Application once they are deployed over a network infrastructure and include specific networking and policy requirements. Therefore, Network Applications are linked to applications but refer to the deployed versions of applications that are further extended with the required resources and VNFs over the targeted infrastructures.

[1] 5G-INDUCE Deliverable, "D2.1 – 5G platform design and requirements in support of Industrial sector Network Applications".

REMARK – B: The applications are composed by one or typically many software components. These applications' components in containerised deployments are becoming the VAFs. Therefore, the VAFs have practically the same notion as the application components. Once the Network Application is deployed, the linked and deployed VAFs are becoming the Network Application.

For further enhancing the definition of Network Applications, the following requirements are identified for the creation of Network Application:

- A Network Application is deployed on demand as requested by a vertical end user. A service provider may manage the deployment on behalf of a vertical end user customer.
- An infrastructure owner (typically a network operator) deploys the requested Network Application over the infrastructure according to the Network Application's deployment requirements and the polices.
- A Network Application must be cloud-native compliant, to allow automated cloud-based deployment and benefit from the cloud deployment such as built-in elasticity.
- A Network Application may operate over a network slice (over a single administrative domain or multiple ones).
- A Network Application may have embedded intelligence in its components (i.e., virtualizable functions) and/or benefit from intelligence provided by Multi-access/Mobile Edge Computing (MEC) and/or Core network, etc.
- A Network Application must be compatible with various networks such as private 5G and beyond networks, and hybrid private and public 5G and beyond networks.
- A Network Application may have resource and network requirements in terms of hardware, such as memory, CPU or GPU usage, H/A availability, etc.
- The composition and deployment of a Network Application should be compatible with the various relevant standard architectures, interfaces and APIs such as ETSI NFV-MANO, 3GPP Service Based Architecture and related network functions for network exposure, network slice selection, and so on.
- For security considerations, general network security requirements and mechanisms in future networks SHOULD be applied, and Network Application specific security network functions MAY be included as part of a Network Application.

Further Network Application definitions are provided below:

- A Network Application may comprise one or more application components.
- Each application component is a standalone entity that executes part of the application logic.
- App components are organized as a direct acyclic graph. Dependencies/Connections between app components are called edges.
- Application providers may choose which app components will be accessible by users.

1.3 Network Application onboarding and orchestration platform (NAO)

The actions for onboarding vertical Network Applications inside the 5G-INDUCE NAO Platform are highlighted in Figure 1.

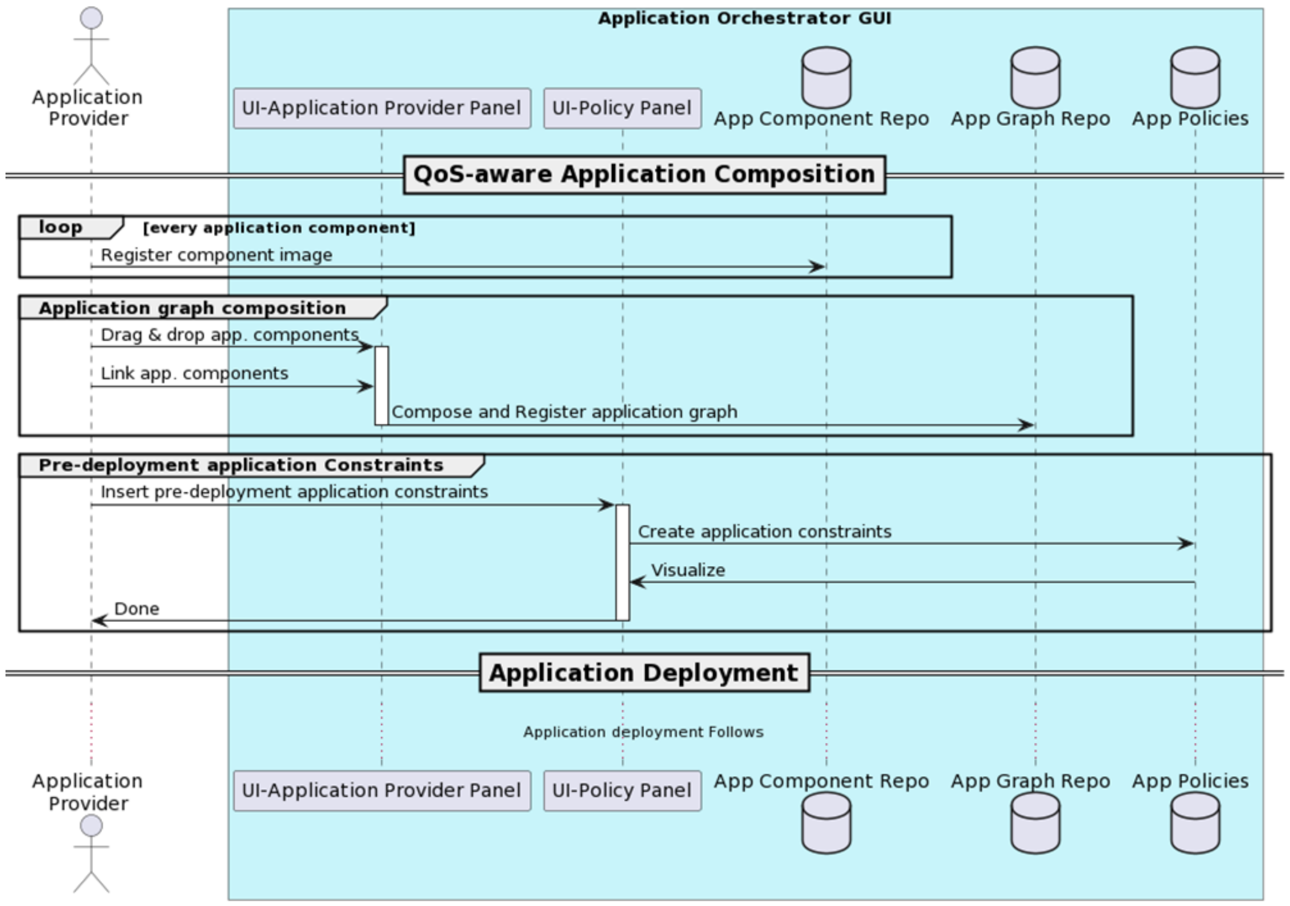

Application Orchestrator GUI
Application Provider
UI-Application Provider Panel
UI-Policy Panel
App Component Repo
App Graph Repo
App Policies
QoS-aware Application Composition
loop
[every application component]
Register component image
Application graph composition
Drag & drop app. components
Link app. components
Compose and Register application graph
Pre-deployment application Constraints
Insert pre-deployment application constraints
Create application constraints
Visualize
Done
Application Deployment
Application deployment Follows

Figure 1. Workflow for onboarding the vertical applications inside the 5G-INDUCE NAO Platform

The overall onboarding process includes the following steps, which are executed through the NAO user interface (NAO-UI):

STEP1: Authentication of registered users to enter the platform.

STEP2: Registration of individual application components through the “Components” view of the NAO-UI. This view displays information about already onboarded applications components, while also allowing the creation of new application components, including updating/deleting the existing ones. The component onboarding interface permits the editing of component requirements such as (i) supported hardware architecture (e.g., x86, x64, arm64, etc.) (ii) location (link) to where a container image resides, (iii) minimum execution requirements in CPU, memory, storage, hypervisor type for OpenStack-based providers, and special hardware dependencies (i.e., GPU, FPGA), (iv) health checks, (v) required and exposed interfaces (e.g., a given TCP-UDP port) of the application component, (vi) associated monitoring plugins, (vii) environmental variables that should be passed to the application component when deployed, and (viii) advanced application component configuration related to Linux kernel capabilities, maximum locked-in-memory address space, and advanced docker execution capabilities.

STEP3: Composition of the application graph through the “Application” view of the NAO-UI. Here the information about the already composed applications is displayed, while allowing to compose new applications, update/delete existing applications, and create instances of existing applications. New applications are created through a wizard, in which the user can search for and select available application components. Once selected, an application component is visualized as a graph node, while edges can be drawn between two nodes, indicating an interface between them. Once all application components are inserted and all their interfaces are drawn, the entire application graph is composed and can be saved in the NAO's database. Then, the user can create an instance of this application resulting in a new application instance being added into the "Instances" view and sent to the multi-domain resource manager to proceed with the deployment of the application.

STEP4: Verification of Network Application deployed instance and live check of selected monitoring parameters through the "Instances" view. Here, the deployed applications can be verified, while also the runtime of existing application instances can be modified. The runtime knobs offered by the NAO are: (i) a "view graph" button which visualizes the deployed application graph and real-time logs from the deployed application components, (ii) an "elasticity policy" button which allows to create new runtime policies for one or more application components, (iii) a "Security Operation Center (SOC) policy" button which allows installing security rules, upon the trigger of which the NAO can raise an event or apply a policy, and (iv) an "un-deploy" button which tears a running application instance down.

1.4 Experimental facilities (ExFas)

1.4.1 Experimental Facilities in Spain

In this section, the particularities of ExFa Spain regarding Network Application onboarding and orchestration are summarized. As the NAO platform is running at the CNIT lab in Genoa, Italy, interconnectivity with the other sites (Ford's I4.0 Factory in Valencia, Spain; the premises of *Universidad Politecnica de Valencia* (UPV) in Valencia, Spain; and 5Tonic lab in Madrid, Spain) has been implemented, granting the communication between the NAO and the compute and networking resources where the Network Applications will be running. Figure 2 shows the high-level end-to-end architecture of ExFa Spain, highlighting in green the interconnections between sites that are more relevant for the Network Application onboarding and orchestration.

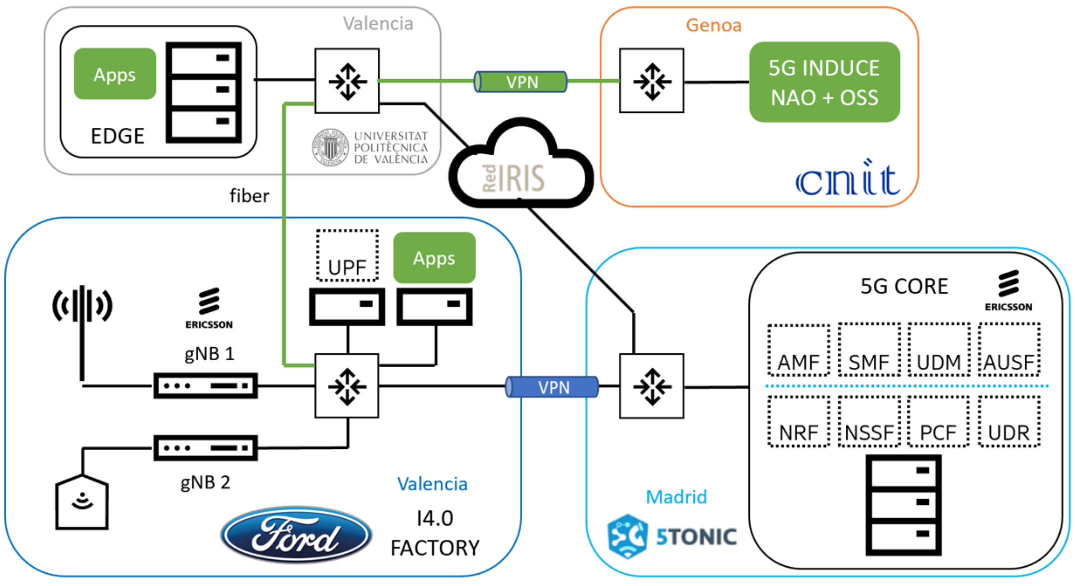


Figure 2. High level end-to-end architecture of ExFa in Spain

Thanks to this implementation, there are two alternatives with regard to the computationalresources: to use the server which is installed at Ford's Factory, for what could be called "on-premises computing" operation; or to use the server installed at UPV premises located 20 km away from Ford's Factory, for the option that could be called "Far Edge computing" operation.

During the deployment process of the Network Application from the NAO, Ford's Factory or UPV premises is selected, which will make an instance of the Network Application to start on the desired location. Either way, when the use cases are running, their traffic will travel through the underlying 5G network. When the traffic coming from the end user devices reaches the UPF, then it will be directed towards the right server according to the IP destination address appearing in the IP packets.

1.4.2 Experimental Facilities in Italy

The ExFA-IT will be held in the Whirlpool industrial site of Cassinetta di Biandronno, in the northern part of Italy, leveraging on Wind3 data center, near Milano, and CNIT data center in Genova (Figure 3.). The overall Experimentation will be deployed with 2 different architectural design solutions for 5G connectivity in the shop floor:

1. Connectivity to NAO on CNIT server via public wind3 network
2. Connectivity to NAO on CNIT server via mobile testbed (CNIT microcell)

All 4 use cases of Italian Experimentation Facility will be tested with both options.

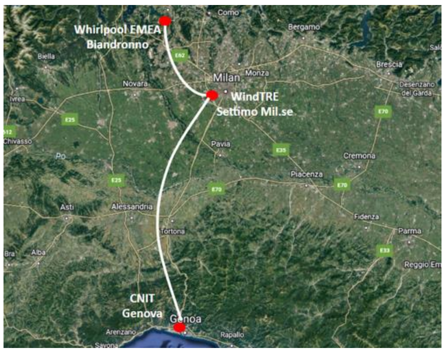


Figure 3. High level end-to-end architecture of ExFa in Italy

Option1: public Wind3 Network

In testbed deployed in Italy connectivity is guaranteed with W3 commercial network, 5G NR covers Cassinetta site while core network is still 4G, W3 core network architecture is No-Stand-Alone. The infrastructure permits connectivity between devices in Whirlpool building and NAO platform running at the CNIT lab in Genoa. SIM cards are configured with public IP address as required in trial, therefore APN configured in devices is **myinternet.wind**.

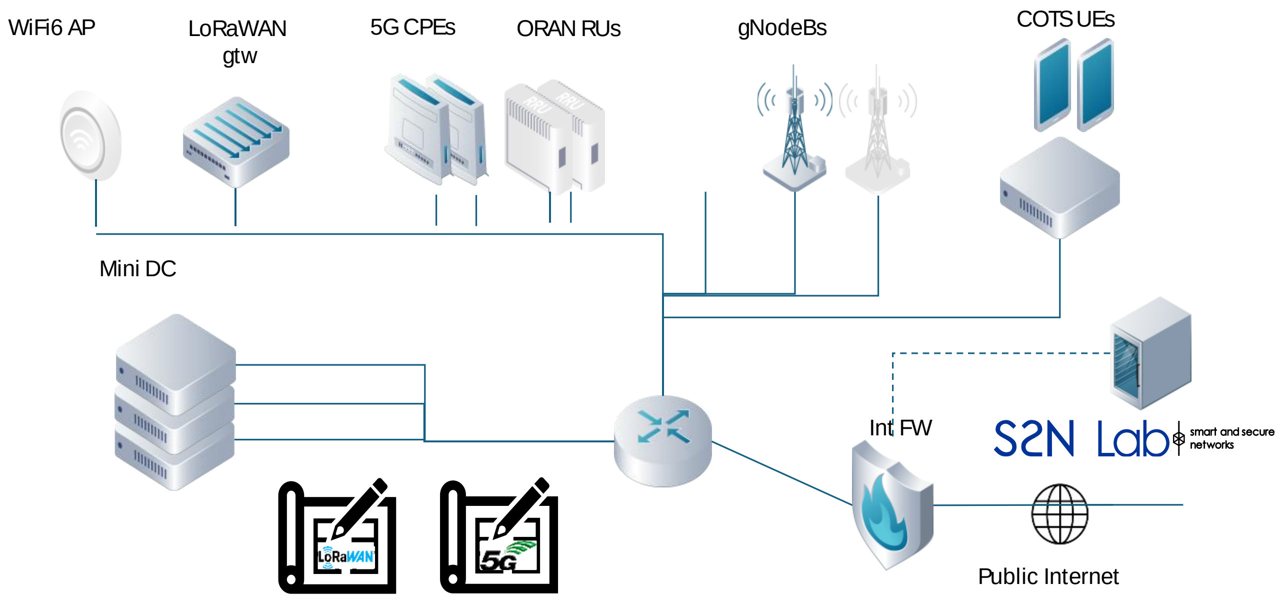


Figure 5: The physical infrastructure of the mobile testbed.

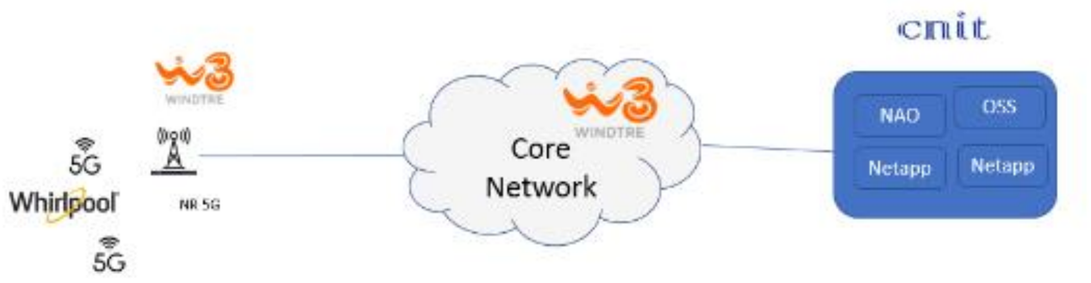


Figure 4: The physical infrastructure of the public network testbed.

Option2: CNIT mobile microcell

The mobile testbed represents a portable “miniature” replica of the main physical infrastructure. The mobile testbed, shown in Figure 5, was born as a satellite of the main testbed in order to perform demonstrations outside of the laboratory, for validating projects’ outcomes and promoting our research achievements at conferences.

It retains the same programming features as the main testbed, i.e., small or large server access management mechanism at the hardware level and allows more or less complex custom installation, est. operating system or more complex programs such as OpenStack.

Since the resources are limited, it can only host one tenant at a time and its management is centralized making it a full-fledged federated testbed, but it can also be used in a standalone fashion but in that case again the management capabilities are limited.

The interconnectivity among the servers, the access and user devices and towards the internal firewall of the main testbed are handled by the Mikrotik L3-switch in the center of Figure 5 allows for LAN management and supports multiple VLANs and L3 routing protocols. A Teltonika acts as the Edge gateway of mobile radio technologies. The stack of servers at the bottom left has the exact same role as in the main testbed but with more limited capacities. It is worth noting that the access and user devices depicted in this figure are only an example and can vary according to the specific use case to be demonstrated.

1.4.3 Experimental Facilities in Greece

The 5G testbed deployed in Greece for the needs of 5G-INDUCE project is located in Athens. It includes network components based on the 3GPP Rel. 16 architecture, the orchestration layer, the proper user equipment, as well as the software and hardware that is required for the Network Applications development.

As shown in Figure 6, the two main sites of the implementation are the Core site (in OTE premises), where the packet core is installed, and the RAN site (in PPC premises), where the access part of the network is installed. The distance between the two sites is about 15km.

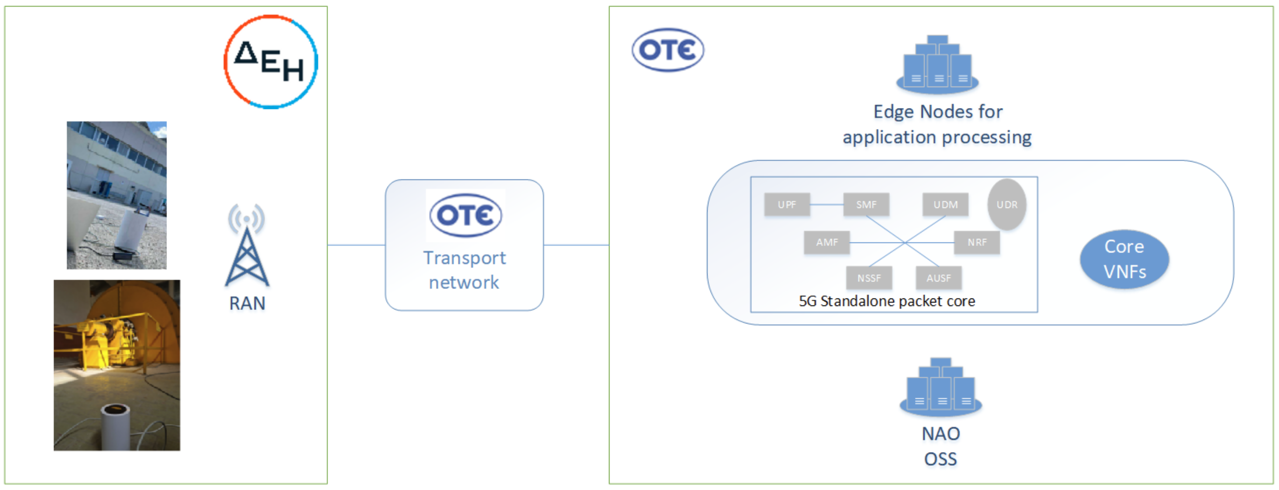


Figure 6. High-level end-to-end architecture of ExFa in Greece.

2 Network Application Preparation

2.1 Example Network Applications

Inspection and surveillance Network Applications for critical infrastructures are used as an example use case, as shown in Figure 7.

2.1.1 Network Application component 1: Video Proxy VNF

Video proxy delivers the video to the other VNFs with high performance, to mitigate the transmission latency drawback of the cloud solution. The NGINX server was used.

2.1.2 Network Application component 2: Intruder VNF

Intruder VNF: Automatic UAV-based area surveillance monitoring for intruder detection using advanced AI-based algorithms to avoid undesired presence of humans or animals.

Intruder detection VNF comprises three pipelines:

- Pre-processing pipeline: Prepares the images using OpenCV.
- AI-based Intruder detection model implemented in TensorFlow (UWS modified YOLO).
- Post-processing pipeline: Prepares results in JSON format.

2.1.3 Network Application component 3: Corrosion VNF

Corrosion VNF: Automatic UAV-based inspection for early corrosion detection in critical infrastructure using advanced AI-based algorithms.

Corrosion detection VNF comprises three pipelines:

- Pre-processing pipeline: Prepares the images using OpenCV.
- AI-based Intruder detection model implemented in TensorFlow (UWS modified YOLO).
- Post-processing pipeline: Prepares results in JSON format.

2.1.4 Network Application component 4: (Message Bus VNF)

Message Bus VNF: receives the detection results in JSON format and sends them back to the Android UAV control application and to other web-based application tools. Rabbit MQ was used.

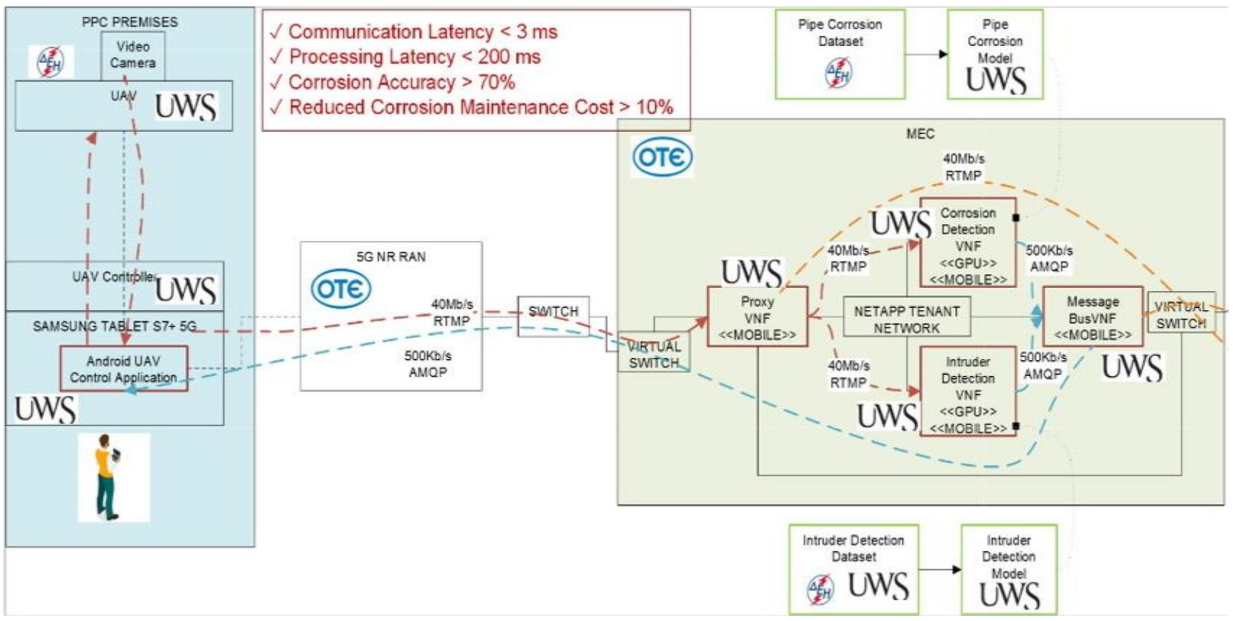


Figure 7. Description and VNFs links for the example use case of two Network Applications

2.2 Operational requirements per Network Application component

The following table summarises all the operational requirements that apply for each component separately.

Table 1. Operational requirements per Network Application for the example use case

	Component 1 Video proxy VNF	Component 2 Intruder detection VNF	Component 3 Corrosion detection VNF	Component 4 Message bus VNF
	Resource parameters			
CPU	2	4	4	2
GPU	No	Yes	Yes	No
Memory	2GB	8GB	8GB	2GB
Storage	20GB	20GB	20GB	20GB
Architecture	x86	x86	x86	x86
	L4 connection parameters			
Protocol type	RTMP	RTMP/AMQP	RTMP/AMQP	AMQP
Port	1935			5671
	Operational parameters			
Dependency	None	{Component-1, Component-4}	{Component-1, Component-4}	None
Mobility	Yes	Yes	Yes	No
Scalability	No	No	No	No
	Other			
Reliability	Normal	Normal	Normal	Normal
Security	Normal	Normal	Normal	Normal

2.3 Network requirements per Network Application graph

The following table summarises the network requirements.

Table 2. Network requirements per Network Application graph for the example use case

	Link: Comp1-to-Comp2	Link: Comp1-to-Comp3	Link: Comp2-to-Comp4	Link: Comp3-to-Comp4
Bandwidth	40Mb/s	40Mb/s	500Kb/s	500Kb/s
Latency	<40ms	<40ms	<10ms	<10ms
Jitter	no	no	no	no
Packet loss	no	no	no	no
Radio service type*	eMBB	eMBB		
Bit rate guarantee type*	Delay critical / Guaranteed bandwidth	Delay critical / Guaranteed bandwidth		

* applicable only for those components that reside in the access part (i.e., having the 'mobility' indication 'ON' in the operational requirements).

2.4 Policy requirements

Operational policies:

-If average GPU utilization metrics (GPU and Memory utilization) exceeds 80% of usage for more than 1 minute, at the component 2 and 3 (corrosion and intruder VNFs), then add another GPU (Resource Scale up).

- If average GPU utilization metrics (GPU and Memory utilization) is below 80% of usage for more than 1 minute, at component 2 and 3 (corrosion and intruder VNFs), then revert to one GPU (Resource Scale down).

-If CPU exceeds 80% of usage in all the components, then add another CPU.

- If RAM exceeds 80% of usage in all the components, then increase RAM.

Application specific policies:

- Corrosion Detection: if average inference time exceeds 55 ms for more than 30 seconds, then increase CPU and GPU cores.

- Intruder Detection: if average inference time exceeds 65 ms for more than 30 seconds, then increase CPU and GPU cores.

- If the average video bandwidth is below 8 Mbps for more than 30 seconds, then increase the number of CPU cores in VideoProxy server.

- If the average video bandwidth is below 8 Mbps for more than 30 seconds, then add a new network slice.

- If an extra drone/video feed is attached into the system, then deploy a new instance of component 2 and 3 (corrosion and intruder VNFs) (Application Scaling Out).

- If a video feed from a drone is not received for more than 30 seconds, delete component 2 and 3 (corrosion and intruder VNFs) instance associated (Application Scaling In).

3 Network Application Onboarding Methodology and Procedure

This section describes the finalised onboarding methodology applicable to all use cases through the NAO end-user interface. The procedure is described in steps, from the registration of the individual Network Application components to the deployment commands sent and the runtime policy-based reconfiguration.

3.1 Registration and composition of a Network Application

The actions for registering, composition and deployment of the vertical Network Applications inside the NAO Platform are highlighted in Figure 1.

An application may comprise of one or more application components. Each app component is a standalone entity that executes part of the application logic. App components are organized as a direct acyclic graph. Dependencies/Connections between app components are edges. Application providers may choose which app components will be accessible by users.

For starters the technical integration with the NAO is to bring the different components to be deployed and communicate using dedicated interfaces or a common message broker in case this is needed. Proper formats for declaring all this information with regard to vertical applications are **Docker-Compose**, **Helm Charts.**

3.2 Entering the web platform

Figure 8 shows entering the web platform. Role-based access control features enable specific users to use specific features and specific views of the web platform (Figure 9).

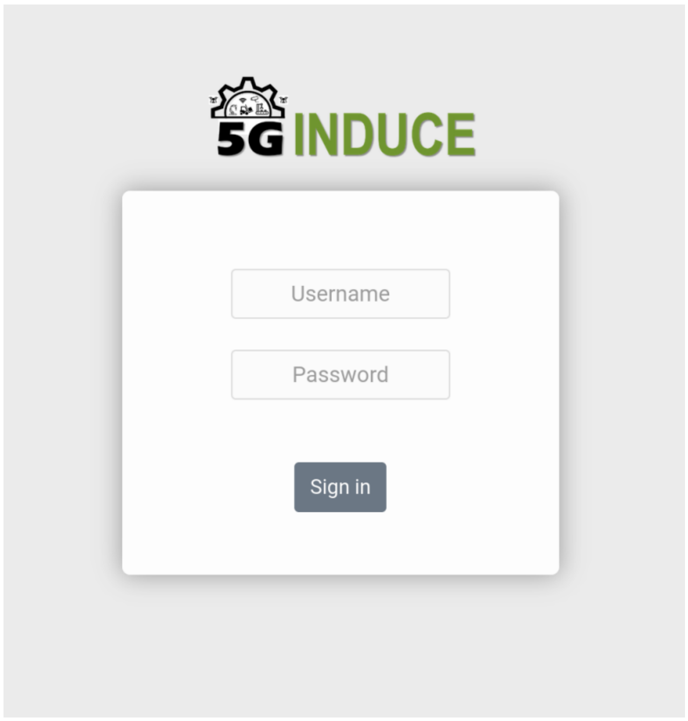
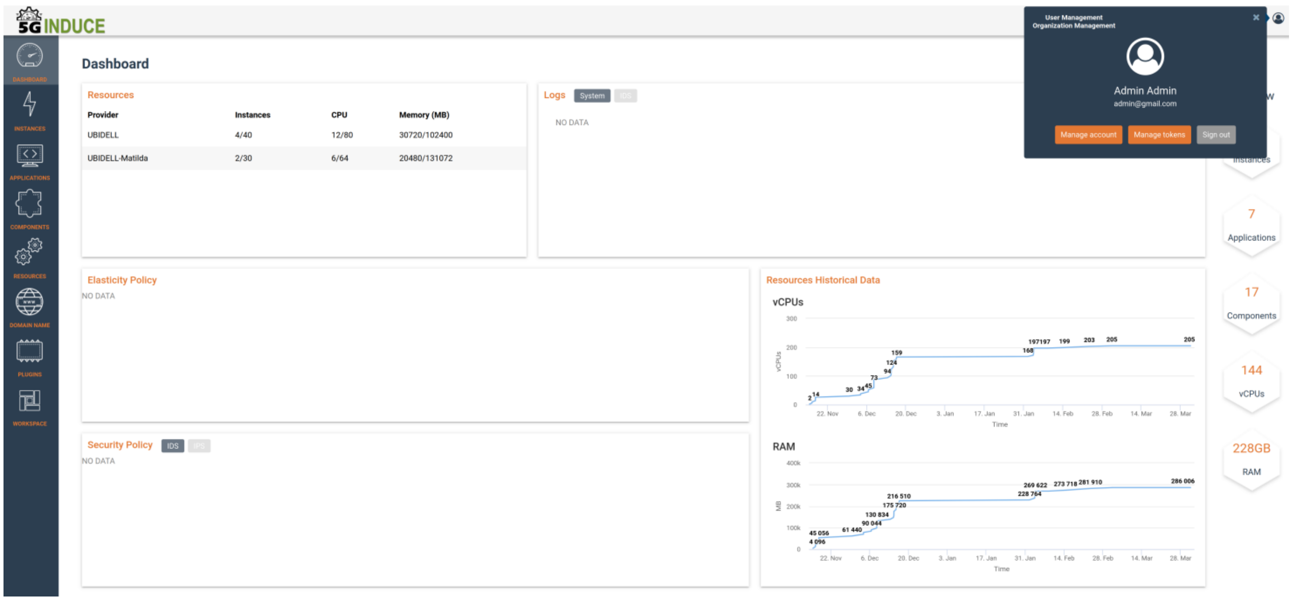


Figure 8. Entering the 5G-INDUCE web interface



Dashboard

Resources

Provider	Instances	CPU	Memory (MB)
UBIDELL	4/40	12/80	30720/102400
UBIDELL-Matilda	2/30	6/64	20480/131072

Logs

NO DATA

Elasticity Policy

NO DATA

Security Policy

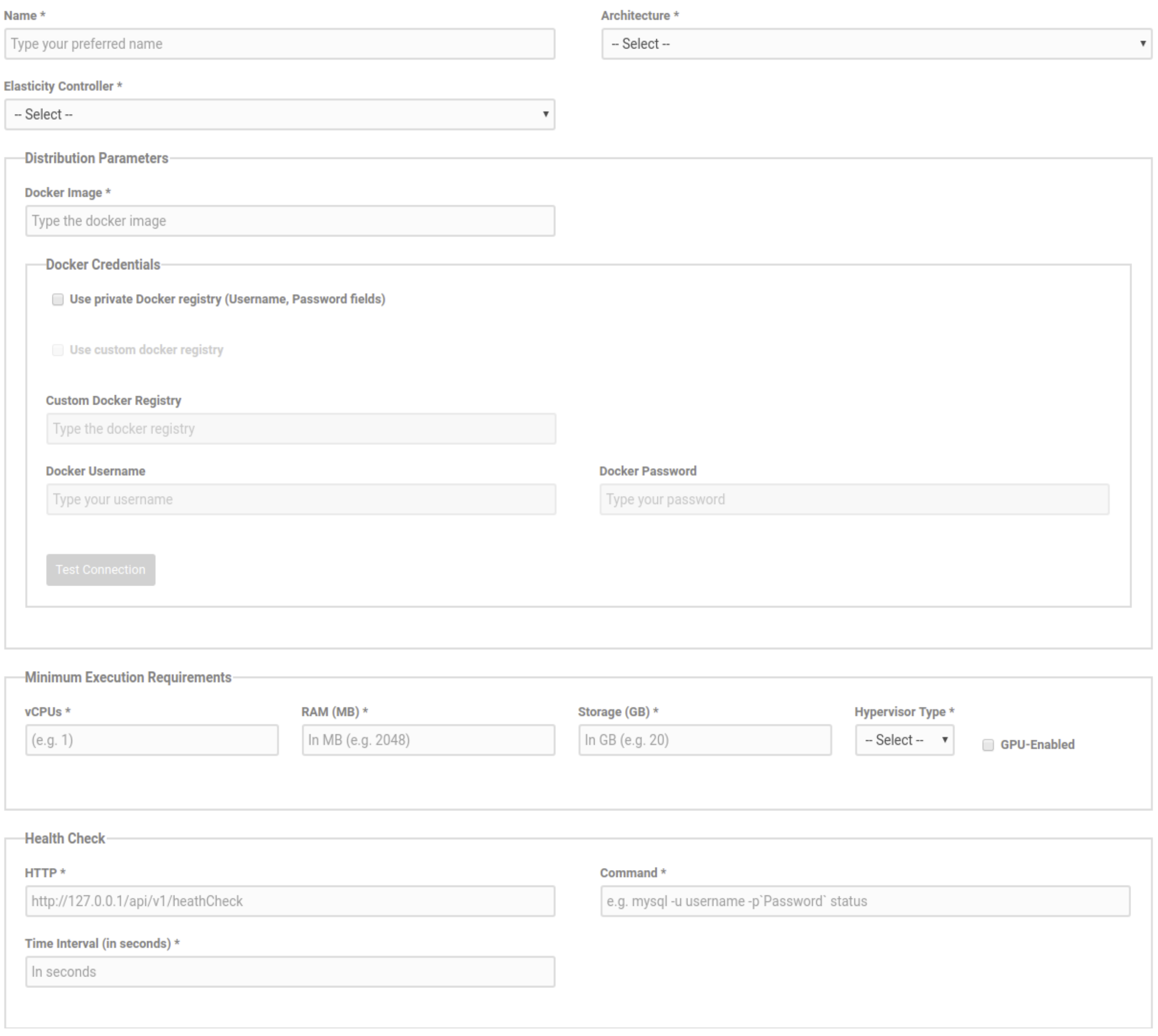
NO DATA

Admin Admin
admin@gmail.com

Figure 9. Roles and Logging into the web platform

3.3 Registration of individual Network Application components

To onboard an application, the NAO offers a UI. For the creation of the application components, we need first to fill the following forms with the corresponding values. This type of information can be found in the application's docker-compose or helm charts descriptor (if such descriptors exist) that the user may have. What must be thought very thoroughly before starting the onboarding of the microservices, is the order in which the components will be created. That can be achieved by creating the acyclic graph of the application, where each link between two microservices represents the dependency that they have. So, for instance, if our application consists of a MySQL and a PhpMyAdmin microservices, in our acyclic graph there will be one link that will go from the PhpMyAdmin to the MySQL and it will represent that the first needs the second in order to start and work properly; so, in that case, we should create first the MySQL component and then the PhpMyAdmin, in order to be capable to refer to the dependency of the MySQL component during the creation of the PhpMyAdmin component. After one has concluded with the order, one can start onboarding the microservices as components to the platform by filling the following forms (Figure 10, Figure 11 and Figure 12):



Name *
Type your preferred name

Architecture *
-- Select --

Elasticity Controller *
-- Select --

Distribution Parameters

Docker Image *
Type the docker image

Docker Credentials

☐ Use private Docker registry (Username, Password fields)

☐ Use custom docker registry

Custom Docker Registry
Type the docker registry

Docker Username
Type your username

Docker Password
Type your password

Test Connection

Minimum Execution Requirements

vCPUs *
(e.g. 1)

RAM (MB) *
In MB (e.g. 2048)

Storage (GB) *
In GB (e.g. 20)

Hypervisor Type *
-- Select --

☐ GPU-Enabled

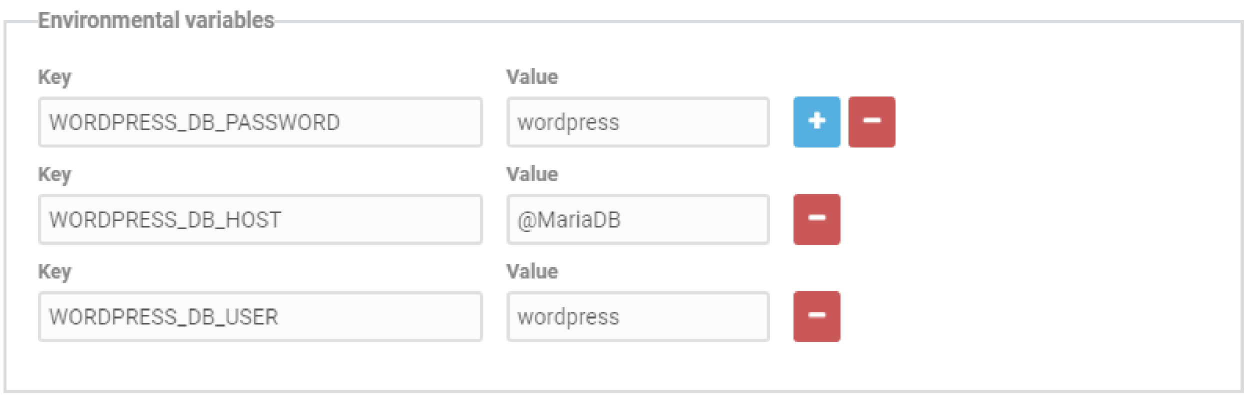
Health Check

HTTP *
http://127.0.0.1/api/v1/heathCheck

Command *
e.g. mysql -u username -p`Password` status

Time Interval (in seconds) *
In seconds

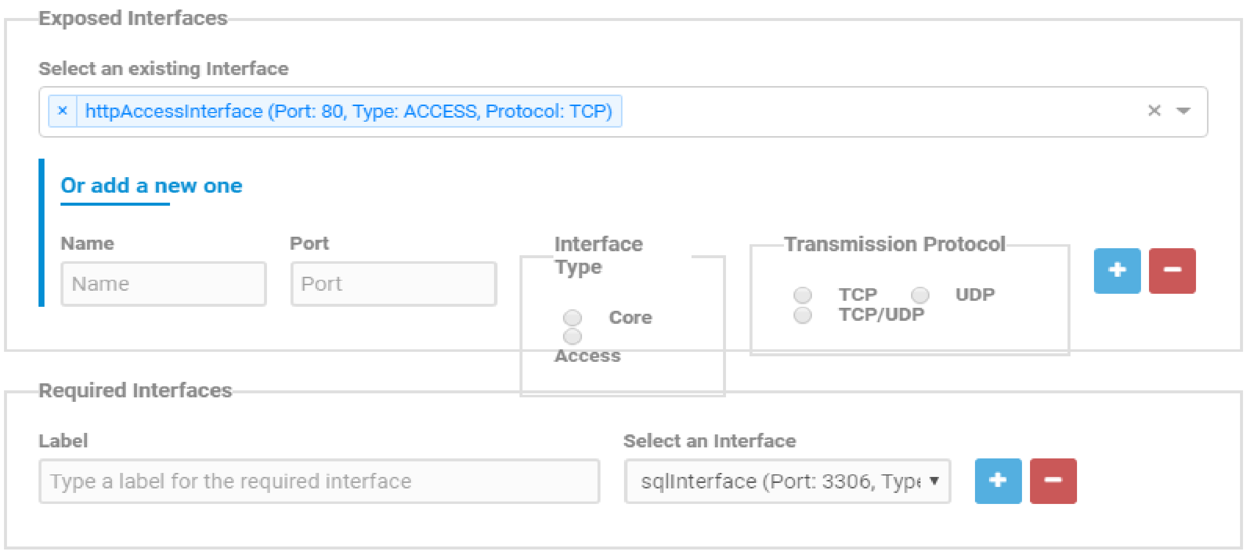
Figure 10. Network Application components' details



Environmental variables

Key	Value
WORDPRESS_DB_PASSWORD	wordpress
WORDPRESS_DB_HOST	@MariaDB
WORDPRESS_DB_USER	wordpress

Figure 11. Network Application components' environmental variables declaration


Exposed Interfaces

Select an existing Interface

× httpAccessInterface (Port: 80, Type: ACCESS, Protocol: TCP)

Or add a new one

Name	Port	Interface Type	Transmission Protocol
Name	Port	○ Core ○ Access	○ TCP ○ UDP ○ TCP/UDP

Required Interfaces

Label	Select an Interface
Type a label for the required interface	sqlInterface (Port: 3306, Type

Figure 12. Network Application component's registration

In Figure 10, Figure 11 and Figure 12, one can find all the basic configurations that are needed in order to create a component, like the docker image and registry of it, the minimum execution requirements that the container needs, as also the health check that it exposes in order to find out if the microservice is running properly. The user can configure the environmental variables, the exposed ports as exposed interfaces, the dependencies on other microservices as required interfaces, the volumes, etc.

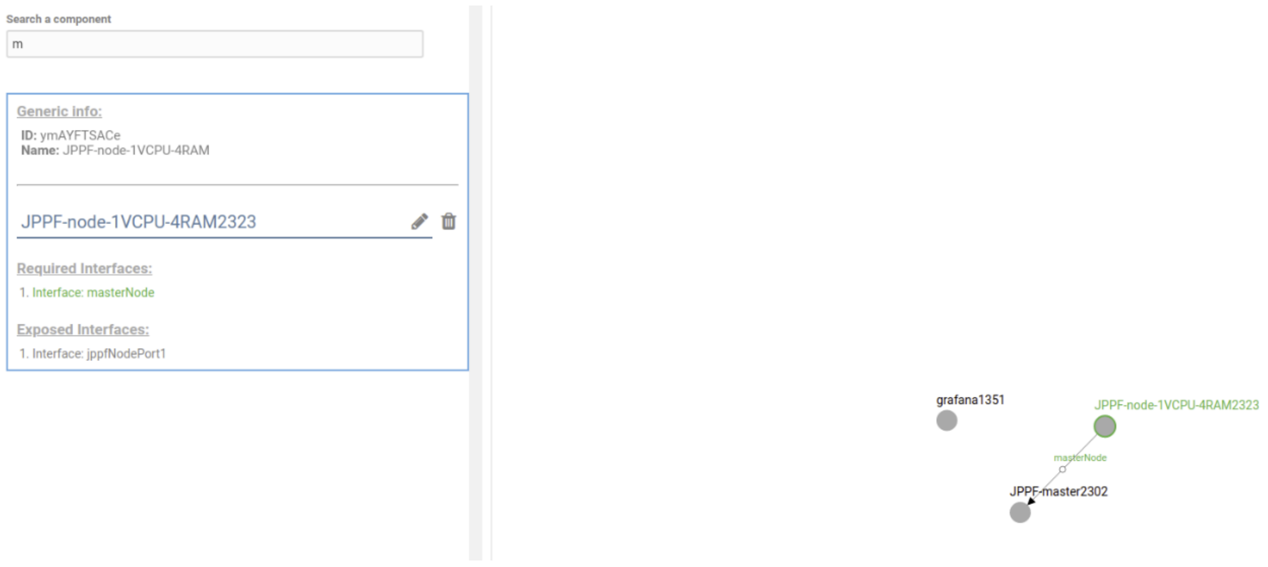
Specific reference should be made on the graph link constraints that Figure 13 shows. Such specifications target the requirements that have to be satisfied regarding the quality of the virtual link which is established between two components. During the deployment of an application component all interfaces that it exposes are bound to network identifiers that are indicated by the infrastructure provider. Through this configuration, the materialized link between components must satisfy some network requirements in terms of delay, jitter, packet loss and throughput. At this point, it should be clarified that a component that participates in a service mesh may expose two types of interfaces. The first type is the CORE and the other is the ACCESS type. CORE interfaces are the ones that are used among the components, while ACCESS interfaces are the ones that interact with the UEs. It should be clear that Graph Link constraints refer to the interconnection of CORE interfaces only.

On the other hand, ACCESS interfaces entail a completely different metamodel.

ACCESS interfaces are like a label for an application component to require a deployment with a network attached to radio equipment. Specifically, the type of constraints that can be provided relate to latency, bandwidth, and the QoS Class Identifier (a.k.a. QCI) like in Figure 16.

3.4 Compose and register the Network Application Graph

After the application component registration ends, the NAO provides a view for selecting and link all the necessary application components that constitute the application graph. Figure 13 demonstrates how the application graph can be created. The Application Provider can just search for the component needed to add to the application graph and then just link the components by grabbing the spheres around the components and dropping them on the component one wants to be linked.


Search a component

m

Generic info:

ID: ymAYFTSACe
Name: JPPF-node-1VCPU-4RAM

JPPF-node-1VCPU-4RAM2323

Required Interfaces:

1. Interface: masterNode

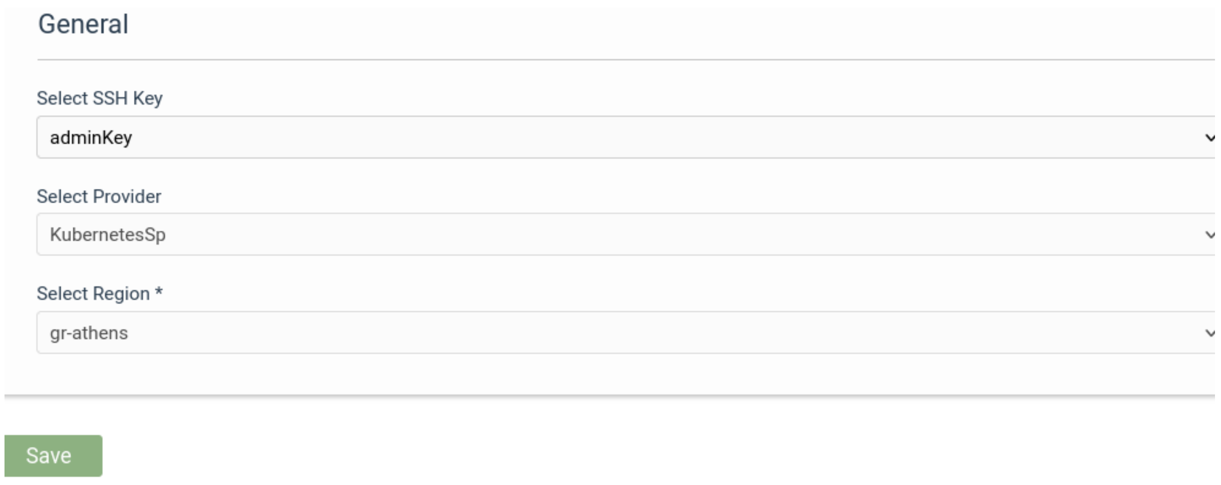
Exposed Interfaces:

1. Interface: jppfNodePort1

Figure 13. Network Application graph's creation

Pre-deployment Application Constraints

After the composition of the application graph, the NAO provides a set of user interfaces supporting the declaration of a set of resource and high-level network constraints that have to be fulfilled during the placement of the application, in order to provide the desired network functionalities. Figure 14, Figure 15 and Figure 16 demonstrate these user interfaces for providing resource and high-level network constraints.

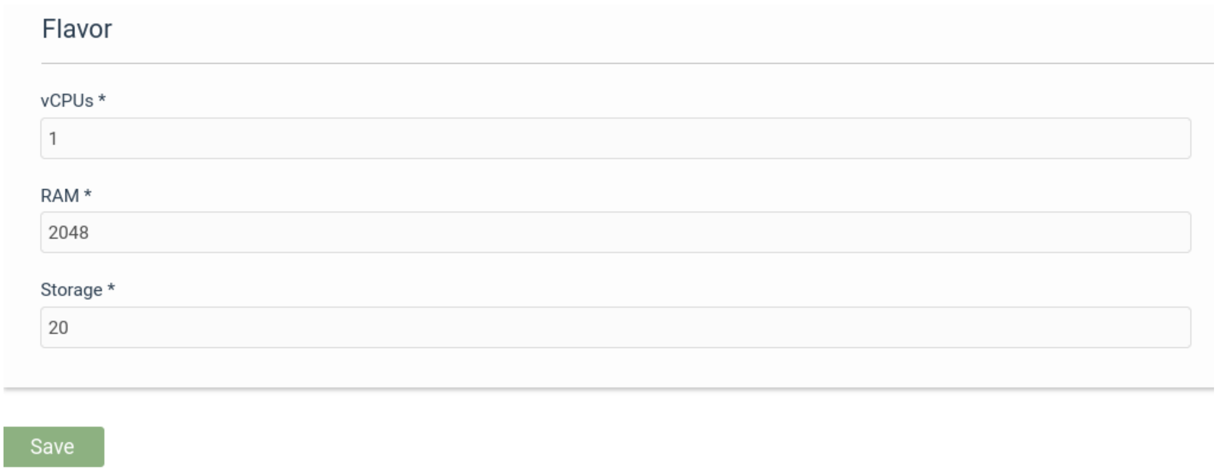

General

Select SSH Key: adminKey

Select Provider: KubernetesSp

Select Region *: gr-athens

Figure 14. Deployment constraints (VIM related)

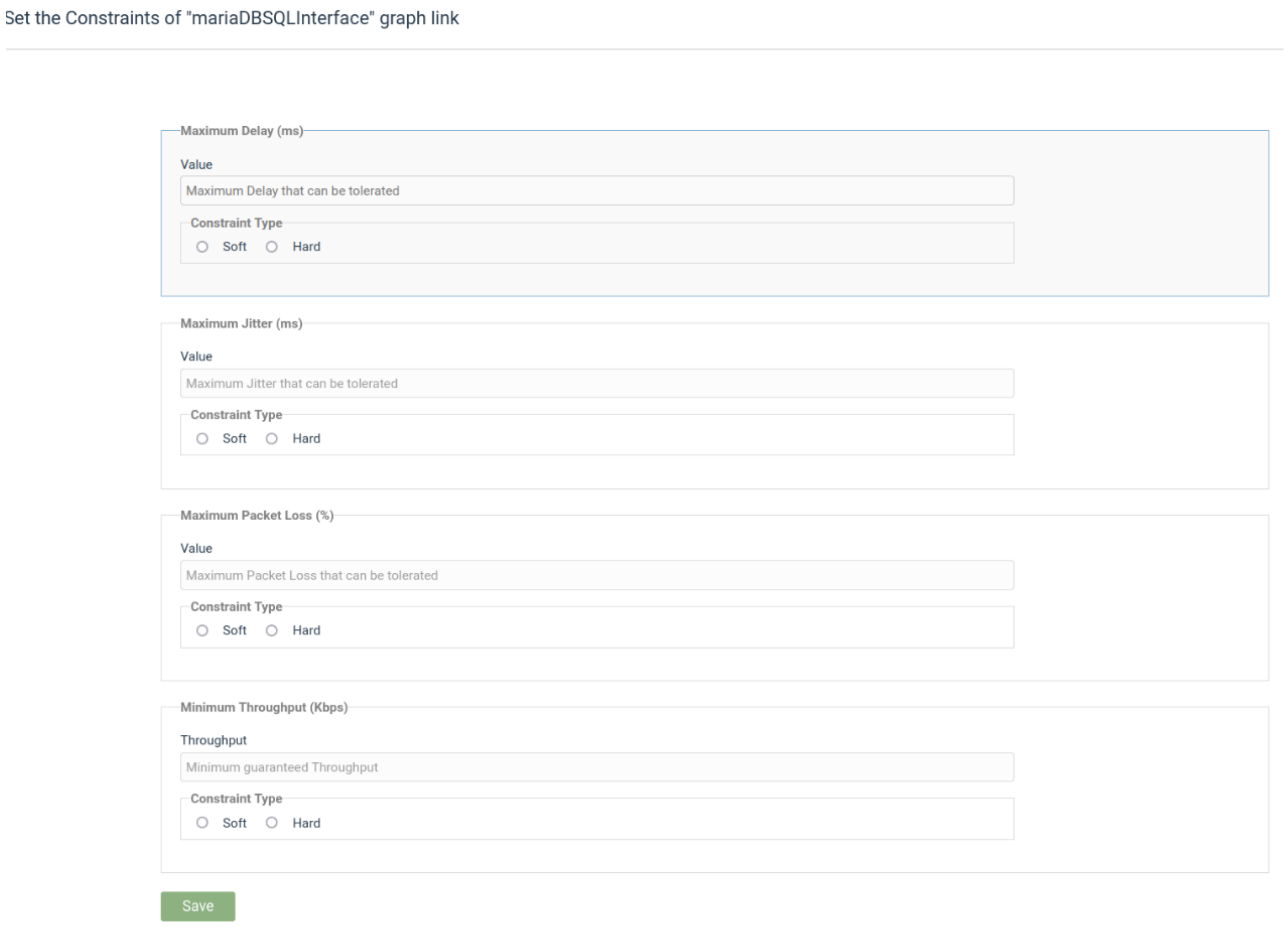

Flavor

vCPUs *
1

RAM *
2048

Storage *
20

Figure 15. Deployment constraints (resources)


Set the Constraints of "mariaDBSQLInterface" graph link

Maximum Delay (ms)
Value
Maximum Delay that can be tolerated
Constraint Type
Soft Hard

Maximum Jitter (ms)
Value
Maximum Jitter that can be tolerated
Constraint Type
Soft Hard

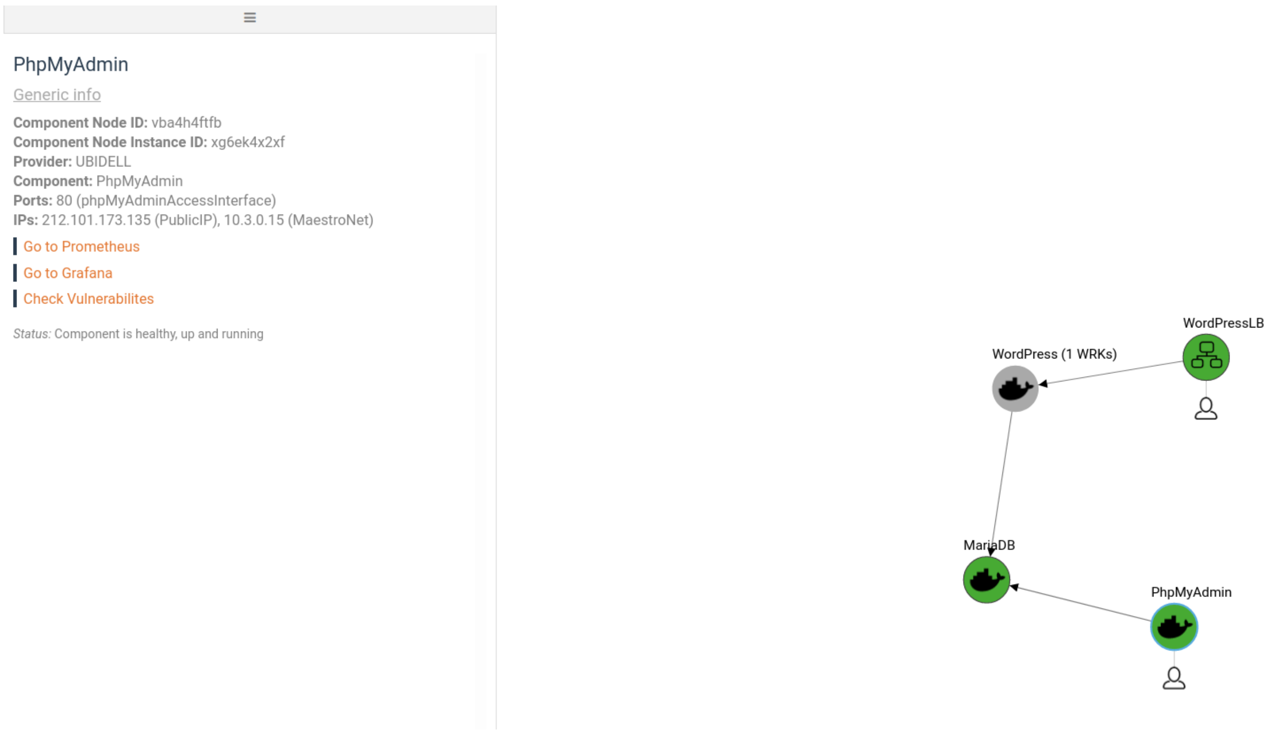
Maximum Packet Loss (%)
Value
Maximum Packet Loss that can be tolerated
Constraint Type
Soft Hard

Minimum Throughput (Kbps)
Throughput
Minimum guaranteed Throughput
Constraint Type
Soft Hard

Figure 16. Network Application constraints' declaration

3.5 Network Application deployment

Given that all the previous steps are successfully completed by the end user, the actual deployment of the Network Application takes place. Figure 17 depicts the successful deployment of a Network Application on top of the programmable resources. Logging information is provided with details regarding the status of the Network Application graph, along with some basic monitoring metrics per application component. Figure 18 shows the runtime logs for the Network Application's deployment.


PhpMyAdmin

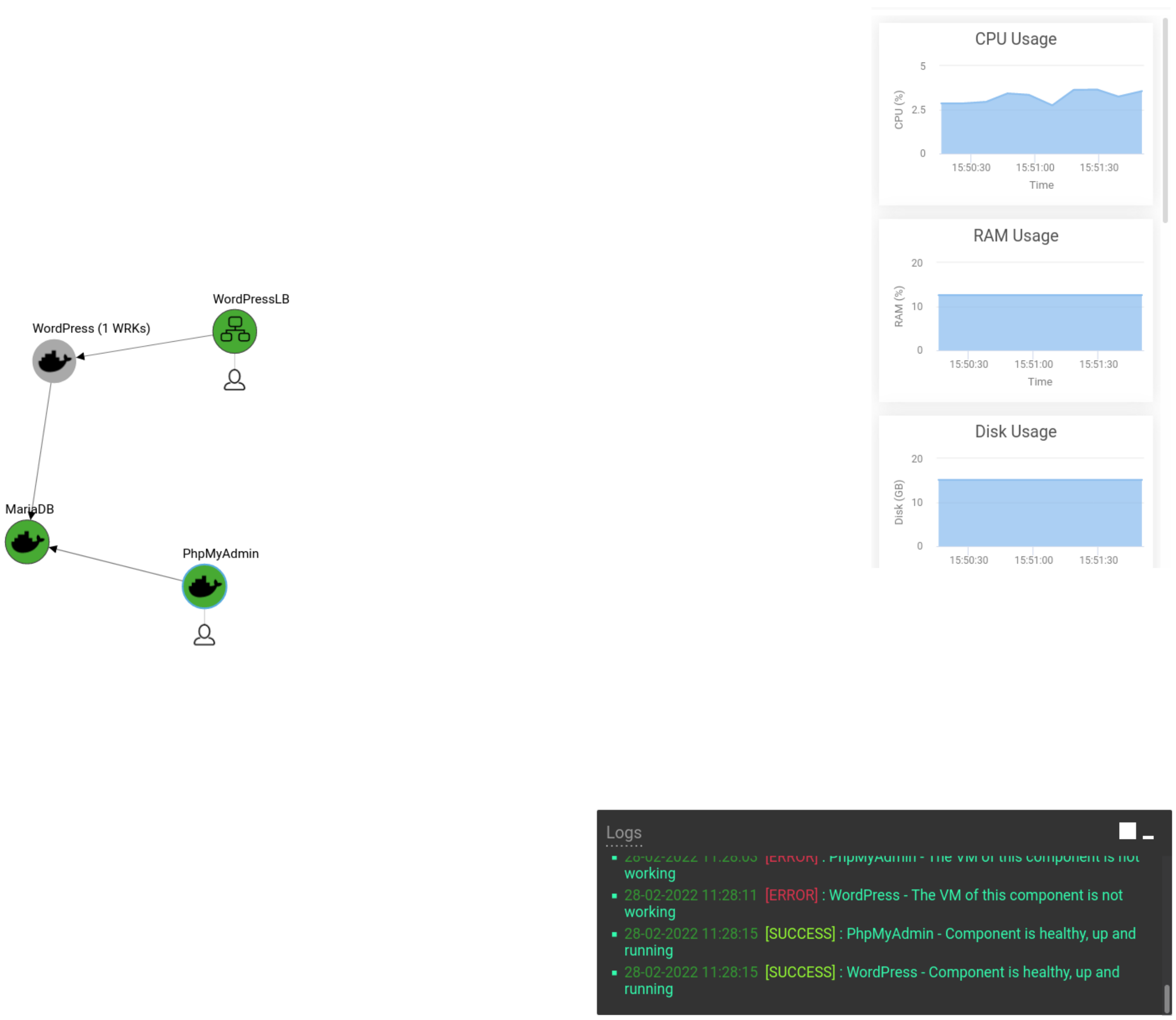
Generic info

Component Node ID: vba4h4ftfb
Component Node Instance ID: xg6ek4x2xf
Provider: UBIDELL
Component: PhpMyAdmin
Ports: 80 (phpMyAdminAccessInterface)
IPs: 212.101.173.135 (PublicIP), 10.3.0.15 (MaestroNet)

Go to Prometheus
Go to Grafana
Check Vulnerabilites

Status: Component is healthy, up and running

Figure 17. Network Application deployment just realized



```
▪ 28-02-2022 11:28:05 [ERROR] : PhpMyAdmin - The VM of this component is not
  working
▪ 28-02-2022 11:28:11 [ERROR] : WordPress - The VM of this component is not
  working
▪ 28-02-2022 11:28:15 [SUCCESS] : PhpMyAdmin - Component is healthy, up and
  running
▪ 28-02-2022 11:28:15 [SUCCESS] : WordPress - Component is healthy, up and
  running
```

Figure 18. Runtime logs for the Network Application's deployment

3.6 Runtime policies

The Policies Manager provides policies enforcement over the deployed application graphs following a continuous match-resolve-act approach. Specifically, the match phase regards the mapping of the set of applied rules that are satisfied based on the alerts coming from the monitoring infrastructure (see Fig. 15). The resolve phase regards the process of conflict resolution for different rules that may be valid and triggered at the same time. Thus, the resolve phase aims at resolution among these rules, taking into account the pre-defined salience of each rule. The act phase regards the provision of a set of suggested actions by the policy manager to the orchestration components, the Deployment Manager and the Execution Manager of the NAO, responsible for application graphs placement and management, respectively (Figure 19).

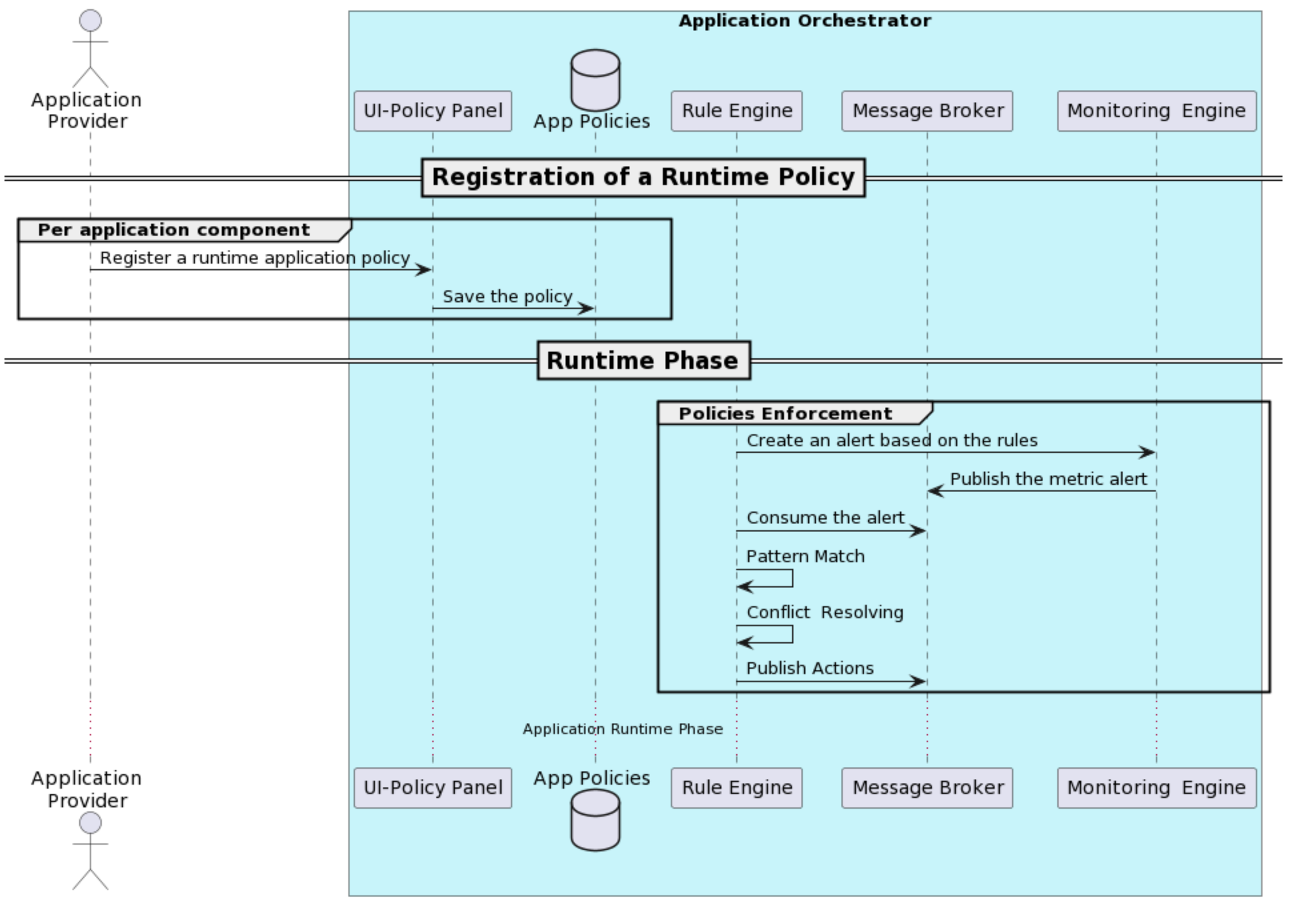
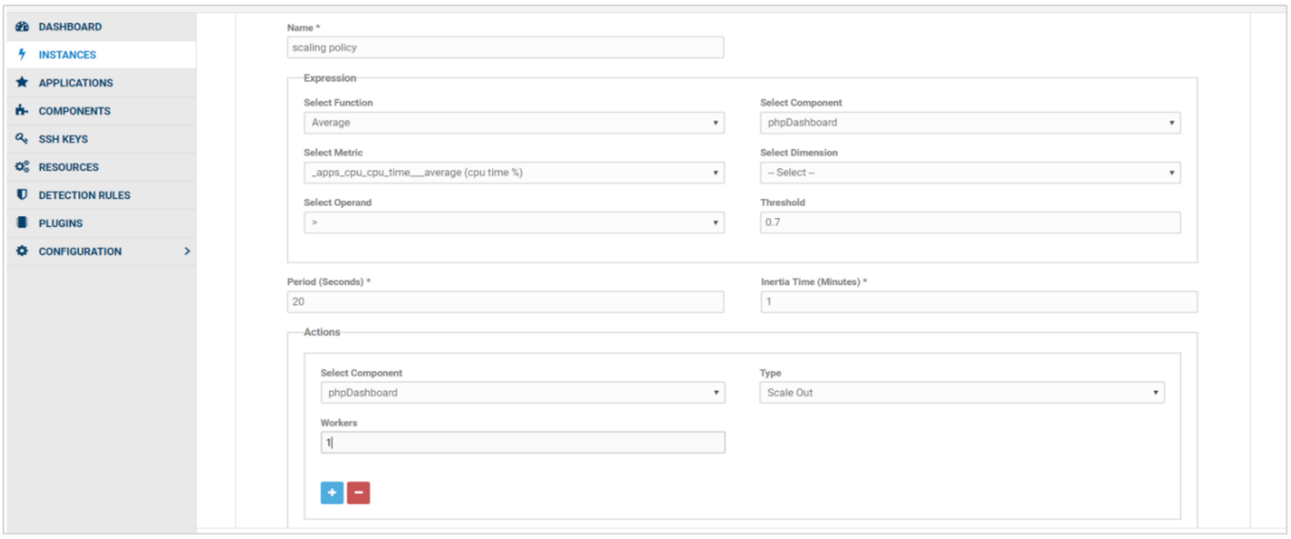


Figure 19. Runtime policies' activation

Policies enforcement is realized through a rule-based framework that attempts to derive execution instructions based on the current set of data and the active rules; rules associated with the deployed application graphs at each point of time. Specifically, we have adopted the Drools rules-based management system, an open-source solution that supports the implementation of runtime policies enforcement mechanisms.

The Policy Editor uses Drools "under the hood", but in order to overcome the cumbersome specificities it has been implemented in a user-friendly way for the declaration of runtime policies that are going to be applied during the operation of the associated application graph. The policies regard elasticity and security management policies.

The Declaration of a policy is done through the UI (Figure 20) and a set of validation mechanisms on the backend; multiple rule-based expressions can be declared following a condition, event, action approach (upon specific conditions, identification of events that lead to actions). The user interface is developed with REACT.JS, while the validation mechanisms are based on the design and development of mechanisms (based on Java) for translating the declared rules to Drools.



Name *: scaling policy

Expression

Select Function: Average	Select Component: phpDashboard
Select Metric: _apps_cpu_cpu_time__average (cpu time %)	Select Dimension: -- Select --
Select Operand: >	Threshold: 0.7

Period (Seconds) *: 20

Inertia Time (Minutes) *: 1

Actions

Select Component: phpDashboard	Type: Scale Out
Workers: 1	

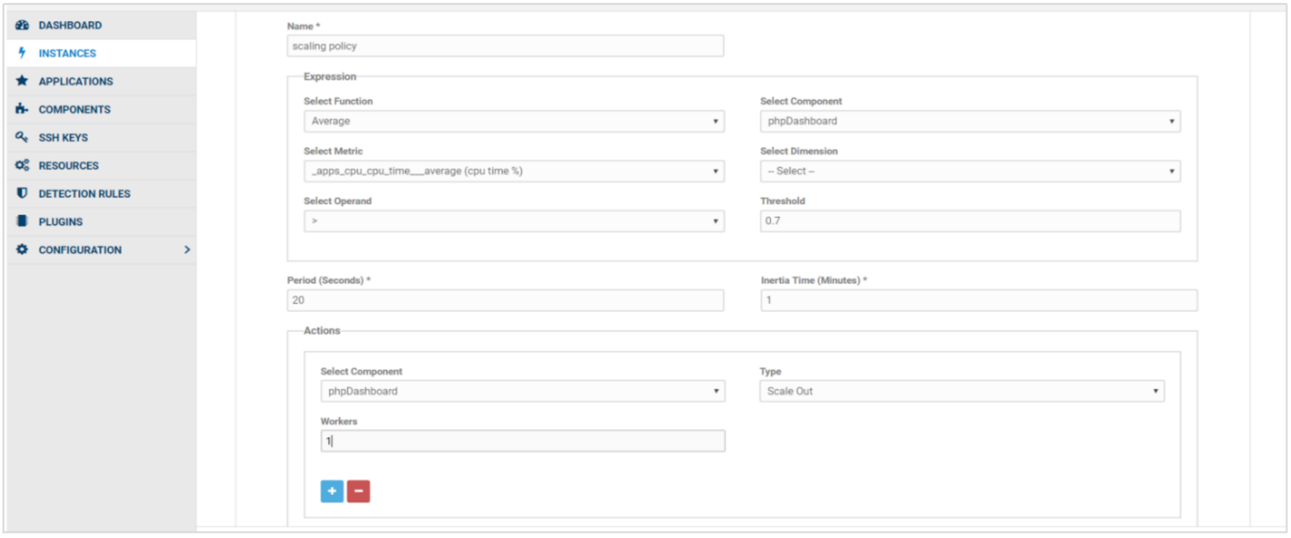
Figure 20. User interface for the policies' definition

To declare a policy, a Network Application must be alive (i.e., successfully deployed and running). The Monitoring metrics must be present inside Prometheus. After that through the GUI the composition of the policy targets a specific metric of a specific application component (or a combination of two or more metrics).

The actions that are for the moment supported regard:

- Scaling actions:
  - scale in;
  - scale out.
- Security actions.
- Block traffic.
- Forensic.

Figure 21 shows the user interface for definition of the actions the policy will trigger.



Name *: scaling policy

Expression

Select Function: Average	Select Component: phpDashboard
Select Metric: _apps_cpu_cpu_time__average (cpu time %)	Select Dimension: -- Select --
Select Operand: >	Threshold: 0.7

Period (Seconds) *: 20

Inertia Time (Minutes) *: 1

Actions

Select Component: phpDashboard	Type: Scale Out
Workers: 1	

Figure 21. User interface for definition of the actions the policy will trigger

4 Summary of Recommendations

This handbook firstly has offered an overview of the 5G-INDUCE concept, platform and experimental facilities for third party users to become aware of the 5G-INDUCE approach and available resources in helping onboard and test their network applications. Secondly, as a precondition for onboarding, the preparation stage has been presented using a concrete example network application. Consequently, A detailed onboarding methodology and procedure has been described with major steps illustrated with diagrams or screenshots to facilitate the deeper understanding. It is highly recommended that any interested parties should follow the preparation and onboarding procedures step by step for a smoother experience.

Furthermore, the following provides a summary list of recommendations in terms of best practices (or lessons learnt) gained from some of the 5G-INDUCE use cases. The purpose is to further help third party developers of similar use cases to prepare and onboard their Network Applications of same or similar types over the 5G-INDUCE platform and ExFas.

Use Case on Human Gesture Recognition

- It is recommended to launch the gesture recognition container when the video is already streaming, otherwise it will restart until it finds an active video stream.
- No specific order is required to deploy the containers, but it is recommended to launch the gesture recognition first and make sure it works correctly before transmitting data to the AGV using the AGV control

Use Case on VR Immersion and AGV Control

- 360-degrees video requires a reliable uplink channel (from camera to the Edge) with an available bandwidth of more than 20-25 Mbps.
- Watch for the CPU load to assure the availability of resources assigned to the containers to avoid video malfunctions.

Use Case on Inspection and Surveillance Services for Critical Infrastructures

- If the microservice of the Network App uses a GPU resource, it is recommended to use the same or lower Cuda version and Nvidia drivers as the one installed in the platform.
- The Network App developer should provide a visual interface to easily visualise that their services are running successfully. For instance, if the Network App deploys a Message Bus service, RabbitMQ software provides an HTTP interface to see the messages being shared.

Use Case on Indoor Vehicle Management and Collision Avoidance

- Implementing socket communication is advised for achieving high-speed communication between the server and clients in location-tracking applications. In this way, we managed to reduce the intercommunication time maintaining super-low latency and high stability.
- Employing a NoSQL database is a recommended best practice for efficiently storing and managing location coordinates. Among diverse DBs offered, we worked with MongoDB (NO SQL) combined with specific SQL features, offering the best solution for our use case.