EXPERIMENTATION AND VALIDATION OPENNESS FOR LONGTERM EVOLUTION OF VERTICAL INDUSTRIES IN 5G ERA AND BEYOND

[H2020 - Grant Agreement No.101016608]

Deliverable D4.5

Network Apps for FoF Operations (FoF)

Editor Tanel Järvet (CAFA)

Contributors (ININ), (UMA), (ZORT)

Version 1.0

Date August 31st, 2023

Distribution PUBLIC (PU)

DISCLAIMER

REVISION HISTORY

Revision	Date	Responsible	Comment
1.0	August 08, 2023	Tanel Järvet CAFA	Reviewers comments received
1.1	August 27, 2023	Tanel Järvet, CAFA	Clean version after review
1.2	August 31, 2023	Tanel Järvet, CAFA	Pdf version for submission

LIST OF AUTHORS

Partner ACRONYM	Partner FULL NAME	Name & Surname
TID	TELEFONICA INVESTIGACIÓN Y DESARROLLO	Javier Garcia
CAF	CAFA Tech	Tanel Järvet, Märten Rannu, Shvea Järvet
UMA	UNIVERSIDAD DE MALAGA	Bruno Garcia, Carlos Andreo, Francisco Luque Schempp
ININ	INTERNET INSTITUT	Jaka Cijan, Rudolf Susnik, Luka Korsic
ZORT	ZORTENET	Andreas Oikonomakis

GLOSSARY

Abbreviations/Acronym	*Description*
AI	*Artificial Intelligence*
ACK	*Acknowledgement*
API	*Application Programming Interface*
AR	*Augmented Reality*
CAPIF	*Common API Framework*
CI/CD	*Continuous Integration / Continuous Development*
CLI	*Command Line Interface*
CSS	*Cascading Style Sheets*
CV	*Computer Vision*
DSCP	*Differentiated Services Code Point*
DoC	*Degree of Curing*
EC	European Commission
FDR	Frequency Domain Reflectometry
FoF	Factories of the Future
gNodeB / gNB	*Next Generation (5G) Base Station*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IIoT	*Industrial Internet of Things*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
KPI	*Key Performance Indicator*
M2M	*Machine to Machine*
MNO	*Mobile Network Operator*
NEF	*Network Exposure Function*
Network App	*Network Application*
NIC	*Network Interface Card*
NPN	*Non-Public Network*
OAuth	*Open Authorization*
OVS	*Open Virtual Switch*
PHP	*Hypertext Pre-processor*
PPE	*Personal Protective Equipment*
QoS	*Quality of Service*
REST	*Representational State Transfer*
ROS	*Robot Operating System*
RTPS	*Real Time Publish Subscribe*
SDK	*Software Development Kit*
SLA	*Service Level Agreement*
SLS	*Service Level Specification*

SME	*Small and Medium-sized Enterprises*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
UE	*User Equipment*
UI	*User Interface*
ULCCP	*Unmanned Life Central Control Platform*
vAPP	*Vertical Application*
VCL	*Visual Components Library*
VPD	*Vapour Pressure Deficit*
WiFi	*Wireless Fidelity*

EXECUTIVE SUMMARY

The objective of this deliverable is to present in detail the final prototypes and the two integration rounds that have been followed for each of the four Network Applications by the three SMEs participating in the task.

Initially, the deliverable describes in detail the final prototypes of the Network Apps developed within pillar of Efficiency in Factory of the Future Operations (FoF) in the EVOLVED-5G context, driven by Task 4.3: NetMapper Network Application by CAF, Anomaly Detection Network App by ZORT, Industrial grade 5G connectivity Network App by ININ, and finally Smart irrigation Network App by UMA/CSIC.

Next, the two cycles of integration activities and use case testing that have been followed for each of the three Network Applications, are described.

The first round of integrations has been carried out with the aim of ensuring seamless and reliable communication between various components within the system, including Network Apps, Vertical Apps (vApp), NEF, CAPIF and 5G network connectivity, on top of the cloud infrastructure provided by the Athens and Malaga platforms. The connectivity of 5G with the cloud infrastructure has been verified, specifically the connection between vApps and the 5G network.

The purpose of the second integration round was to validate the use-cases utilizing the final components of EVOLVED-5G. On the one hand, NEF, CAPIF and the SDK had been enriched with additional features. On the other hand, SMEs finalized their Network Apps by enhancing the 3.0 version and using the last versions of NEF, CAPIF and SDK. This version 4.1 of the Network Apps also exploited the validation pipeline before the integration test.

Finally, the Networks Apps were deployed in Kubernetes clusters in Athens NCSRD and Malaga UMA premises instead of using Docker containers running locally.

With the second round of integration tests, the Networks Apps of the pillar have reached their final stage, interacting with the last versions of NEF and CAPIF through the SDK and communicating with their respective vApp(s).

The three SME use-cases have also been validated and such results highlight the fact that the Network Apps reached a mature enough state to be used by other SMEs through the Evolved-5G Marketplace.

TABLE OF CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

The current report complements the final prototype of the three Network Apps, that have been developed within the Factory of the Future Operations (FoF) pillar to support the innovative interaction of employees and machines and is driven by Task 4.3. The report provides details on the development of the final prototype (version 4.1) in terms of technical architecture, features and dependencies, while also utilising the final version of the tools (SDK, NEF, CAPIF) developed within the EVOLVED-5G framework. Moreover, the report contributes to the testing and evaluation of the use cases, described in the previous deliverable of WP4 (D4.2), through the integration activities that took place both in Malaga and Athens infrastructure focusing on the iterative validation of 5G connectivity and communication between components (5G network <–> Network Applications <-> Vertical Applications).

While the term NetApp was used in the D4.2 document, the consortium decided in 2022 that the term Network App will be used in the future in the EVOLVED 5G project.

1.2 STRUCTURE OF THE DOCUMENT

- Section 2 "**CONTEXT OF THE PILAR**" echoes D4.2 and presents a summary of the FoF pillar goals, challenges and specificities.
- Section 3 "**FINAL PROTOTYPE OF NETWORK APPLICATIONS**" describes the finalized version of the FoF Network Apps (version 4.1). After a reminder of the Network App use-case(s), it describes the technical architecture, features and dependencies of each Network App.
- Section 4 "**INTEGRATION ACTIVITIES AND USE-CASE TESTING**" presents the two integration rounds for use-case testing performed during the project. The first reported test was performed with intermediate versions of Network Apps and components (NEF, CAPIF and SDK) while the second test was performed with final versions of Network Apps and all EVOLVED-5G components.

1.3 TARGET AUDIENCE

The release of the deliverable is public, intending to expose the overall EVOLVED-5G ecosystem and Network Apps progress to a wide variety of research individuals and communities.

From specific to broader, different target audiences for D4.4 are identified as detailed below:

- **Project Consortium**: To validate the fact that all FoF pillar Network Apps have reached their final state. One of the main goals is to document the technical evolution of these Network Apps with respect to the initial vision and use-case.
- **Industry 4.0 and FoF (factories of the future) vertical groups:** To crystallise a common understanding of technologies, and tools that were used for the development of the Network Apps. Besides, it also demonstrates the final architecture and features a Network App can reach. A non-exhaustive list of Industry 4.0-related groups is as follows:

  - Manufacturing industries (including both large and SMEs) and IIoT (Industrial Internet of Things) technology providers.
  - European, national, and regional manufacturing initiatives, including funding programs, 5G-related research projects, public bodies and policy makers.
  - Technology transfer organizations and market-uptake experts, researchers, and individuals.
  - Standardisation Bodies and Open-Source Communities.
  - Industry 4.0 professionals and researchers with technical knowledge and expertise, who have an industrial professional background and work on industry 4.0-related areas.
  - Industry 4.0 Investors and business angels.
- **Other vertical industries and groups**: To seek impact on other 5G-enabled vertical industries and groups in the long run. Indeed, all the architectural components of the facility are designed to secure interoperability beyond vendor specific implementation and across multiple domains. The same categorization as the above but beyond Industry 4.0 can be of application.
- **The scientific audience, general public and the funding EC Organisation:** To document the work performed and justify the effort reported for the relevant activities. The scientific audience can also get an insight of finalized Network Apps' processes, tools and features.

2 CONTEXT OF THE FoF OPERATIONS PILLAR

In the context of Factories of the Future, one of the most critical aspects where 5G technology can bring significant enhancements is the convergence of Operations Technology (OT) and Information Technology (IT). This convergence aims to achieve a higher level of efficiency in the overall FoF operations capacity. Some of the key technologies involved in this transformation are big data and analytics, simulation, Internet-of-Things (IoT), cloud computing, computer vision, AI and Machine Learning. These technologies are seamlessly integrated with cutting-edge production processes, such as hybrid manufacturing, adaptive manufacturing, and smart manufacturing equipment and systems. The combination of these advanced technologies and production methodologies empowers factories to operate at unprecedented levels of productivity, flexibility, and intelligence, introducing a new era of innovation and competitiveness.

However, alongside these promising advancements, there are also challenges to tackle. One such challenge relates to IoT monitoring within the factory. As the number of interconnected devices increases, ensuring smooth and secure communication between them becomes crucial. Safety is another concern that demands attention. With advanced technologies and automation, it's essential to maintain a safe working environment for both humans and machines. Additionally, anomaly detection in the vast amount of data generated by these smart systems is crucial. Identifying irregular patterns or potential issues is essential to prevent downtime and optimize production efficiency.

In this prism, the EVOLVED-5G project tackles the aforementioned challenges with the development of specific Network Apps by the partners within the FoF pillar as follows:

- IoT and M2M based monitoring and digital twinning, with a platform allowing monitoring information from manufacturing machinery, sensors and actuators to be collected and maintained modelled as digital replicas of the physical machinery (ININ).
- In addition, the issue of anomaly detection in both OT and IT network segments, for the detection and mitigation of detected anomalies, exploiting sensory and monitoring information coming from the 5G system as well as the machinery control and management (ZORT).
- Safety of operations within the factory environment tackled by an AI based video analyzer for factory workers safety (CAFA).
- Smart irrigation, the components of which are controlled using 5G communication. (UMA)

Moreover FoF pillar partners target the automation through vertical applications (vApps) utilized by end-users, along with Network Apps designed to facilitate the deployment of 5G communication functionality.

The numerous applications of IIoT (Industrial IoT), is the IoT/M2M remote monitoring platform. That platform is provided by ININ and is a cloud-based management system for remote gateway and sensor control. IoT/M2M-based remote monitoring platform benefits from the 5G integration will be the following: extending capabilities of the IoT/M2M system components to support automated deployment, components scaling and lifecycle management, reduced IoT/M2M system service deployment time, extending capabilities of the IoT/M2M Gateway to support operation with 5G NSA/SA capabilities and also extending the network performance monitoring capabilities of the system to include Industry 4.0 network and applications metrics.

One different application of IIoT is the CAFA AI based video safety analyzer for factory workers safety. CAFA`s Safety VideoLyzer detects whether or not Personal Protective Equipment (PPE) is being worn by factory workers and provides near real time a warning signal directly to the control room when any element of PPE equipment is not being detected. 5G integration in conjunction with distributing the image analysis to local processing near the user can reduce the load on the network as only results from the image analysis are transmitted further in the network.

Another equally important IoT's application with the two mentioned above has been coined by ZORTENET. That is the ML-driven anomaly system for industrial processes. This system provides the means of detection and classification of network anomalies that influence industrial performance, contribute to information leakage, and potentially breech multitenant isolation.
Challenges of FoF are mainly related to the aspects that Network Apps deployment requires logical and well-functioning interoperable functionality by the 5G network provider. At the same time, the Network Apps deployment technology inevitably has several shortcomings in the initial period of the development of such functionalities, because the development is done in an agile way.

It is important for customers (companies) that the number of work hours required for Network Apps and vApps deployment is of a reasonable size, so that the applications are not deployed due to excessive costs. The prototypes and deployment processes of four FoF applications are described below, which have been carried out with the aim of making them as time-efficient as possible.

3 FINAL PROTOTYPE OF NETWORK APPLICATIONS

The following sections present the final prototypes for the Network Apps within the FoF pillar, describing both a use case description and an architecture overview for each of them.

3.1 CAFA NETMAPPER NETWORK APPLICATION

3.1.1 Use case description

CAFA NetMapper Network App employs computer vision software to detect whether or not Personal Protective Equipment (PPE) such as safety helmet, safety glasses are being worn by employees and provides a near real time warning signal directly to the control room safety officer when any element of PPE equipment is not being detected. The video from the factory is collected using a CAFA AMR robot, a wheeled platform that carries stereo cameras that cover a 360-degree field of view around the robot. The robot has a 5G communication modem that transmits the videos to the 5G MEC-based CAFA Safety VideoLyzer vApp, which is used to analyze whether workers are wearing PPE as described in Figure 1 below.

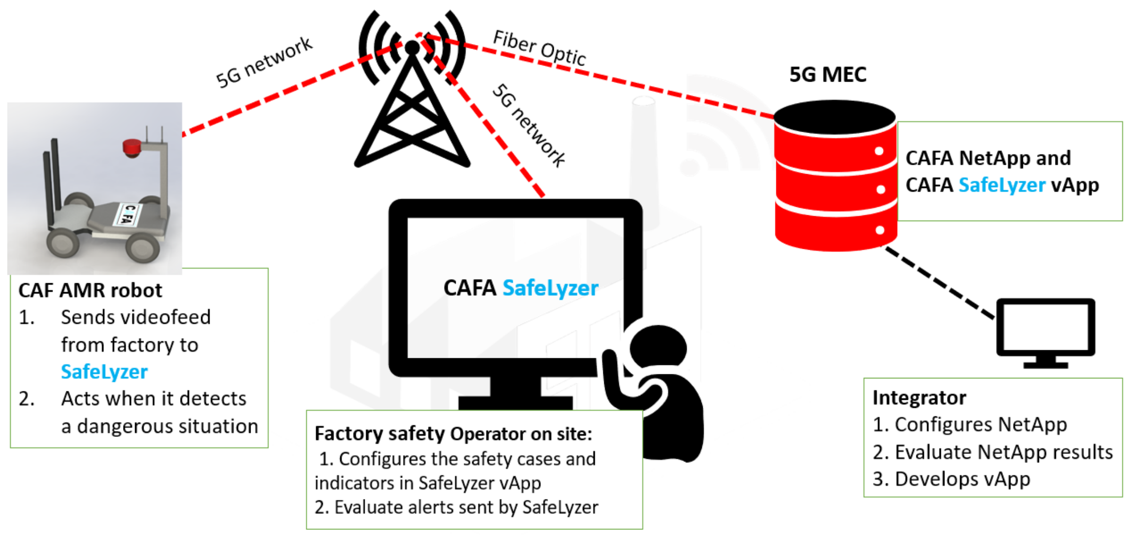

5G network
Fiber Optic
5G network
5G MEC
CAFA NetApp and
CAFA SafeLyzer vApp
CAFA SafeLyzer
CAF AMR robot
1. Sends videofeed from factory to SafeLyzer
2. Acts when it detects a dangerous situation
Factory safety Operator on site:
1. Configures the safety cases and indicators in SafeLyzer vApp
2. Evaluate alerts sent by SafeLyzer
Integrator
1. Configures NetApp
2. Evaluate NetApp results
3. Develops vApp

Figure 1 High level Architecture of CAFA NetMapper Network App and vApp

The network speed and low latency are the crucial key elements of real-time CV applications. Indeed, this real-time computer vision applications need to have a powerful computer near the camera or in the local network to provide sufficient speed and latency. This would make the whole system more expensive and more difficult to maintain its hardware and software, especially if there are more than one camera points (for example, moving robots) in multiple locations.

The high network speed and low latency of 5G networks allows running real-time computer vision applications close to the performance as if they run in local computer. The low latency is crucial in directing moving robot in confined indoor environments to avoid its collision with people, building structures and other equipment. The notification about the QoS loss allows the vApp to stop the computer vision and send the signal to the robot to stop its movement. Access to 5G capabilities will allow using one central computer with fast connections to multiple camera points.

3.1.2 Detailed Architecture

CAFA Network Application "NetMapper" creates a GBR (Guaranteed Bit Rate) discrete automation subscription to the NEF Emulator in order to receive notifications about QoS (Quality of Service) to NetApp's Flask server's endpoint at port 5555. Then the Flask server forwards the message to CAFA CV Flask server at port 5000 which changes the qos_state.txt file. The vApp's Safety VideoLyzer conducts its computer vision task when (according to the qos state file) QOS_GUARANTEED or quits the computer vision to prevent the unreliable results if QOS_NOT_GUARANTEED. The Python files – one for starting the Flask server and writing the QoS State message to file, and the VideoLyzer application are located in the same directory. The VideoLyzer computer vision application detects whether workers are wearing PPE (e.g. safety helmet). The detailed architecture of the NetMapper Network App is presented in Figure 2 below.

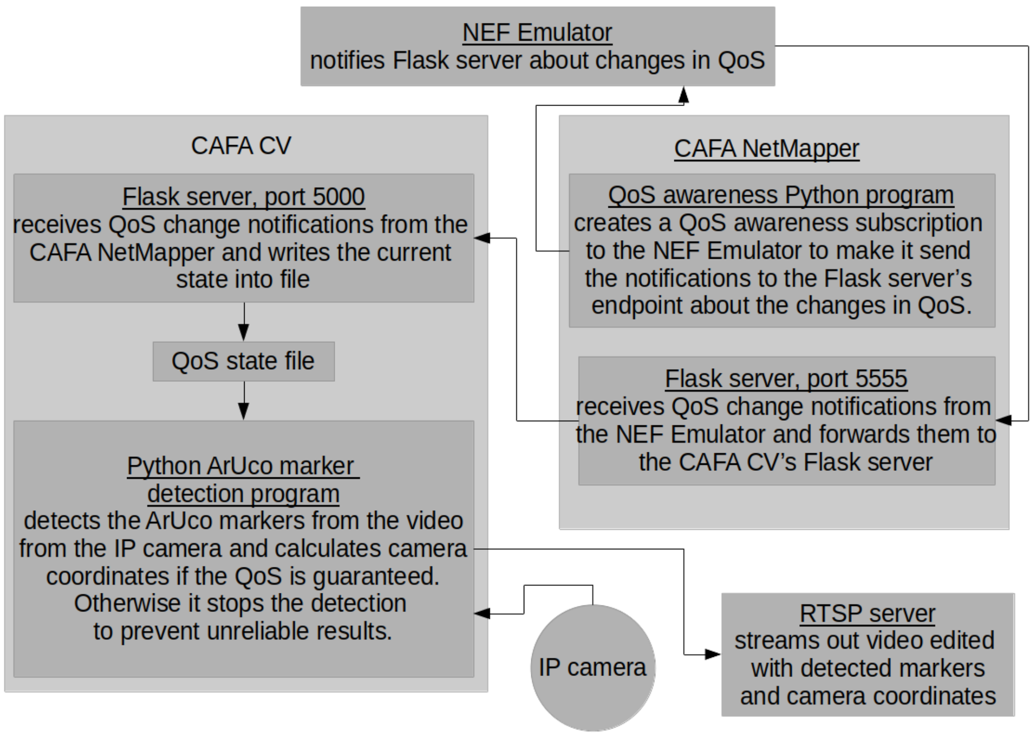

NEF Emulator
notifies Flask server about changes in QoS
CAFA CV
Flask server, port 5000
receives QoS change notifications from the CAFA NetMapper and writes the current state into file
QoS state file
Python ArUco marker detection program
detects the ArUco markers from the video from the IP camera and calculates camera coordinates if the QoS is guaranteed. Otherwise it stops the detection to prevent unreliable results.
CAFA NetMapper
QoS awareness Python program
creates a QoS awareness subscription to the NEF Emulator to make it send the notifications to the Flask server's endpoint about the changes in QoS.
Flask server, port 5555
receives QoS change notifications from the NEF Emulator and forwards them to the CAFA CV's Flask server
IP camera
RTSP server
streams out video edited with detected markers and camera coordinates

Figure 2 Detailed architecture of the early versions of the CAFA NetMapper Network App from D4.2

vApp CAFA Safety VideoLyzer

The vApp monitors 5G related Service Level Agreement (SLA) and notifies NetMapper Network App and end-user i.e., Factory safety officer if SLA is not passing. It provides user interface (UI) for the occupational safety analysis results (if Personal Protective Equipment (PPE) helmet, glasses etc. are missing). The vApp displays (UI) 5G related KPIs (throughput, latency, MEC resources). The vApp adapts the workload if updated resources are insufficient. CamControl

(Client software for robot platform control) displays QoS info and allows remote operator to reconfigure vApp parameters such as temporarily pausing video stream among other options.

Network App CAFA NetMapper

The Network App receives 5G network QoS information from the NEF Emulator and sends instructions to vApp to switch to another mode of operation if necessary (e.g., if 5G QoS is too poor to transmit 4K video).

The use case can be comprised of three steps:

1. Network App receives information about 5G network conditions by setting up subscriptions with certain data throughput levels to be notified automatically as soon as those criteria cannot be met.
2. Network App detects that required data throughput is not passing (based on the automatic notification from NEF Emulator) and notifies end-user i.e., Factory safety officer and robot's remote operator as well as sends instructions to vApp to switch operating modes to less demanding.
   After some time, Network conditions improve. Network App automatically receives feedback from 5G infrastructure manager about restored data throughput conditions and sends complementary instructions to vApp to resume operation at former levels. In order to achieve the interaction with the NEF Emulator the Network App opens a port and listens for incoming asynchronous messages about 5G network conditions (guaranteed data throughput related messages) utilizing the AsSessionWithQoS API. This subscription-based communication prevents unnecessary back and forth messaging in most use cases. Also, subscriptions can be canceled if network conditions are poor, and robot cannot perform its basic tasks anyway.
3. Finally, the Network App now registers to CAPIF to access NEF. The whole Network App is containerized through Docker and was deployed on UMA's premises.

Figure 3 below shows the architecture of the last version of the CAFA NetMapper Network App and Safety VideoLyzer vApp.

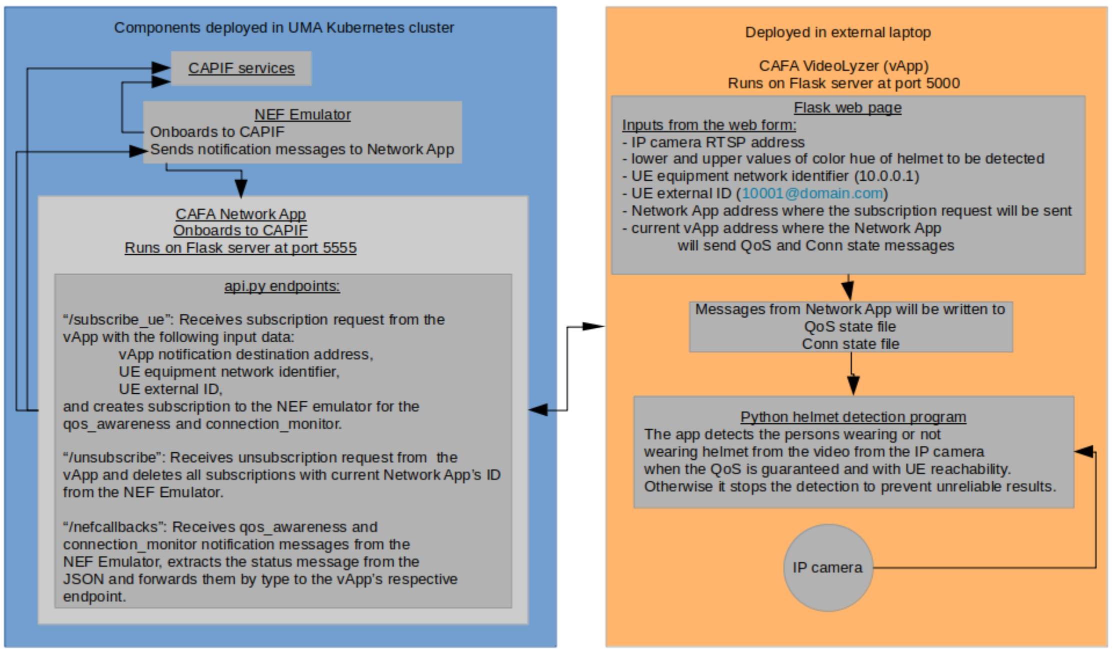

Components deployed in UMA Kubernetes cluster
CAPIF services
NEF Emulator
Onboards to CAPIF
Sends notification messages to Network App
CAFA Network App
Onboards to CAPIF
Runs on Flask server at port 5555
api.py endpoints:
"/subscribe_ue": Receives subscription request from the vApp with the following input data:
vApp notification destination address,
UE equipment network identifier,
UE external ID,
and creates subscription to the NEF emulator for the qos_awareness and connection_monitor.
"/unsubscribe": Receives unsubscription request from the vApp and deletes all subscriptions with current Network App's ID from the NEF Emulator.
"/nefcallbacks": Receives qos_awareness and connection_monitor notification messages from the NEF Emulator, extracts the status message from the JSON and forwards them by type to the vApp's respective endpoint.
Deployed in external laptop
CAFA VideoLyzer (vApp)
Runs on Flask server at port 5000
Flask web page
Inputs from the web form:
- IP camera RTSP address
- lower and upper values of color hue of helmet to be detected
- UE equipment network identifier (10.0.0.1)
- UE external ID (10001@domain.com)
- Network App address where the subscription request will be sent
- current vApp address where the Network App will send QoS and Conn state messages
Messages from Network App will be written to
QoS state file
Conn state file
Python helmet detection program
The app detects the persons wearing or not wearing helmet from the video from the IP camera when the QoS is guaranteed and with UE reachability. Otherwise it stops the detection to prevent unreliable results.
IP camera

Figure 3 Final architecture of the CAFA NetMapper Network App and Safety VideoLyzer vApp

3.2 Industrial Grade 5G Connectivity Network Application

ININ's Network App (further Network App) enables KPIs provided by NEF to be collected by the 5G IoT System and saved for further analytics. It interacts with NEF APIs through CAPIF while providing API endpoint for the vApp IoT Management. Network App's responsibility is to proactively monitor and configure the 5G data connection parameters to meet the SLA requirements for critical sensors used in FoF.

The final version of Network App is marked as v4.1 and includes all dependencies for proper communication with CAPIF v3.1.2, Evolved5G SDK v1.1 and NEF v2.2.2.

The Network App is implemented as an integral part of the solution called "5G IoT System" (Figure 4), which, apart from the Network App itself and "IoT Gateway" (which serves as a UE device forwarding critical and best-effort data, and providing certain 5G KPIs through its 5G KPI Monitoring functionality), consists of the following building blocks:

- IoT Management (provides centralized management of IoT Gateways),
- IoT Collector (central storage for processing and storing data collected by IoT Gateways),
- IoT Reporter (KPIs visualization).

The vApp is mainly focusing on two key functions. Firstly, it is primarily responsible for remotely managing multiple IoT Gateways, overseeing IoT Management. Secondly, the vApp collects industrial process-related data from sensors connected to these IoT Gateways, functioning as an IoT Collector. Additionally, the vApp also gathers specific 5G-related data/KPIs through the measurement engine running on each of the connected IoT Gateways. .This is further extended by using the Network App that enables KPIs provided by NEF to be collected by the IoT System and saved in the common database for further analytics. Common database is part of the IoT Collector as depicted in Figure 4. Thus, the Network App interacts with NEF APIs while providing API endpoint for the vApp (IoT Management).

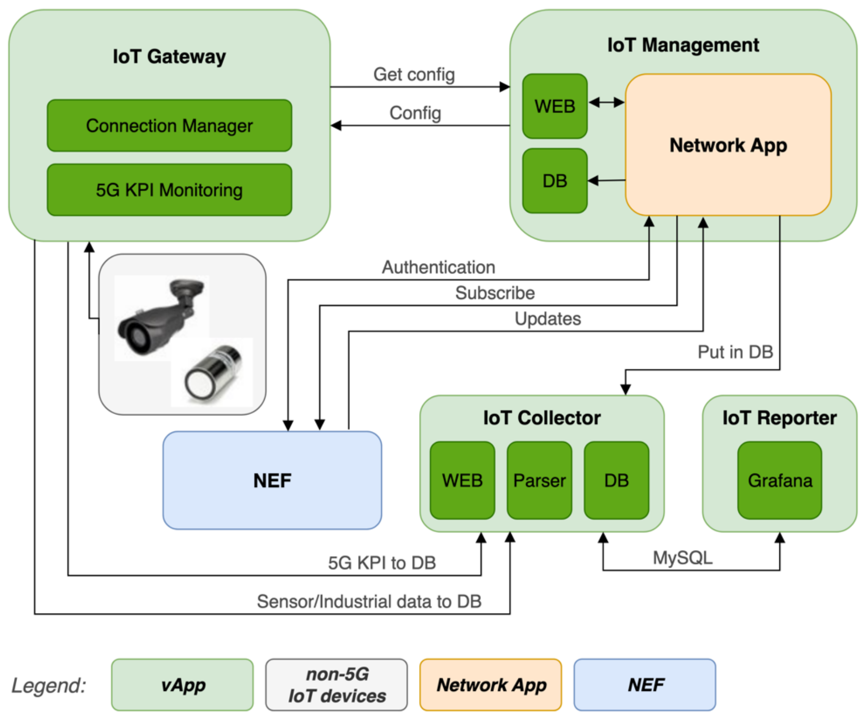


Figure 4 5G IoT System overview

The final prototype of the Network App supports two available subscription-based NEF APIs as provided by the NEF:

- Monitoring Event API for location related KPIs collection and
- Session with QoS for the subscription to the slice with specific QoS characteristics.

The 5G IoT System benefits out of the EVOLVED-5G ecosystem by utilizing the following:

- Current 5G network status provided by NEF and integrated in the 5G IoT System environment.
- Additional KPIs collected from NEF for deep down analytics and anomalies detection.
- Influencing traffic patterns on the UE (5G IoT Gateway) by subscribing to QoS assured sessions.

3.2.1 Use case description

Network App serving an "Industrial grade 5G connectivity with assured QoS and integrated SLA/SLS monitoring capabilities" use case is presented in this section. The use case assumes specific IoT and M2M devices used in Factories of the Future (FoF) applications that require a stable communication environment described by certain QoS parameters such as bandwidth, latency, error rate, etc. It is assumed the above mentioned IoT and M2M devices have no 5G capabilities and are therefore physically connected to the IoT Gateway (i.e., part of 5G IoT System – see Fig. 4) which provides 5G connectivity for them. It is also assumed that FoF network, to which IoT Gateway is connected to, is a Non-Public 5G Network (NPN).

A possible solution to the challenge described above is the Network App which enhances the IoT monitoring process by enabling end-to-end network performance monitoring based on automated collection of various radio, network, and cloud related performance metrics. Through the Network App, basic network and service level agreements (SLA/SLS) and service monitoring features of the 5G IoT System can be enhanced by utilising NEF APIs to collect additional 5G network related Key Performance Indicators (KPIs). Some of the KPIs, gathered either by 5G IoT System or through NEF, can be further used as metrics to trigger slice reconfiguration if the application requirements are not met. The presented solution is optimized for FoF environments with strict requirements for collecting, analyzing, and representing/visualizing data captured by various sensors playing essential role for the industry process.

In a real-world environment, including FoF NPN 5G, there are typically various heterogeneous sensors and other devices connected to the network. As previously mentioned, certain (critical) sensors require specific QoS conditions (e.g., bandwidth, latency, etc. as defined by SLA/SLS) to operate properly, while for other sensors and devices, best-effort conditions are sufficient. Based on the latter premise, the Network App's responsibility is to proactively monitor and configure the 5G data connection parameters so as to meet the SLA requirements for critical sensors. Therefore, the Network App retrieves available monitoring service parameters collected by 5G IoT System and by NEF, and, whenever necessary, executes re-configuration of the system with new optimal parameters to try to achieve the expected SLA.

A simple example of the above would be a scenario with two non-5G IoT devices (a sensor collecting industrial process related data – critical data, and a surveillance video camera - best effort data) connected to an IoT Gateway which provides them with 5G connection. As long as there is no excessive background traffic and SLA requirements are met, no action is required. Suddenly, background traffic increases significantly, causing QoS parameters of the 5G data connection established by the IoT Gateway to deteriorate. Based on the comparison of the SLA requirements and the actual status of the data connection (i.e., SLA not achievable based on the 5G QoS monitoring data received from both NEF and IoT system), the Network App requests the IoT Gateway to suspend the forwarding of the best effort traffic generated by the surveillance video camera, thus forwarding sensors' traffic only. In case the situation in the network recovers, the Network App requests the IoT Gateway to resume forwarding best effort traffic.

3.2.2 Detailed Architecture

The Network App, including vApp building blocks, has been initially deployed within the non-public 5G network of Internet Institute. As in the time of writing the deliverable, the Network App is in the prototype phase, the verification and testing of the end-to-end scenario has been partially performed manually.

Figure 5 below depicts the sequence diagram for the building blocks comprising the overall system. In the first step (1), the Network App starts and obtains validated NEF token, which is further used for other communication purposes, e.g., registration and de-registration to NEF API. In the second step (2), the user is required to configure NEF API parameters for the IoT Gateway (UE) selected. IoT Management UI (part of vApp) is used for configuration purposes. In the third step (3), the NetApp registers to the UE and subscribes to the NEF API. Then (4), IoT Gateway is ready to forward both traffic generated by critical sensor (critical traffic) and traffic generated by the surveillance video camera (best-effort traffic) to the IoT Collector. In the fifth step (5), increased background traffic is simulated, resulting in "QoS NOT GUARANTEED" message received by the Network App, which, in-turn, passes the info to the IoT Management in order to request IoT Gateway (UE) to suspend forwarding best-effort traffic, while keeping forwarding critical traffic. After background traffic flow being decreased (6), the “QoS GUARANTEED” message is received by the Network App, followed by the info passed to the IoT management and finally IoT Gateway (UE) is requested to resume forwarding best-effort traffic. By this stage, the functional test cycle is over.

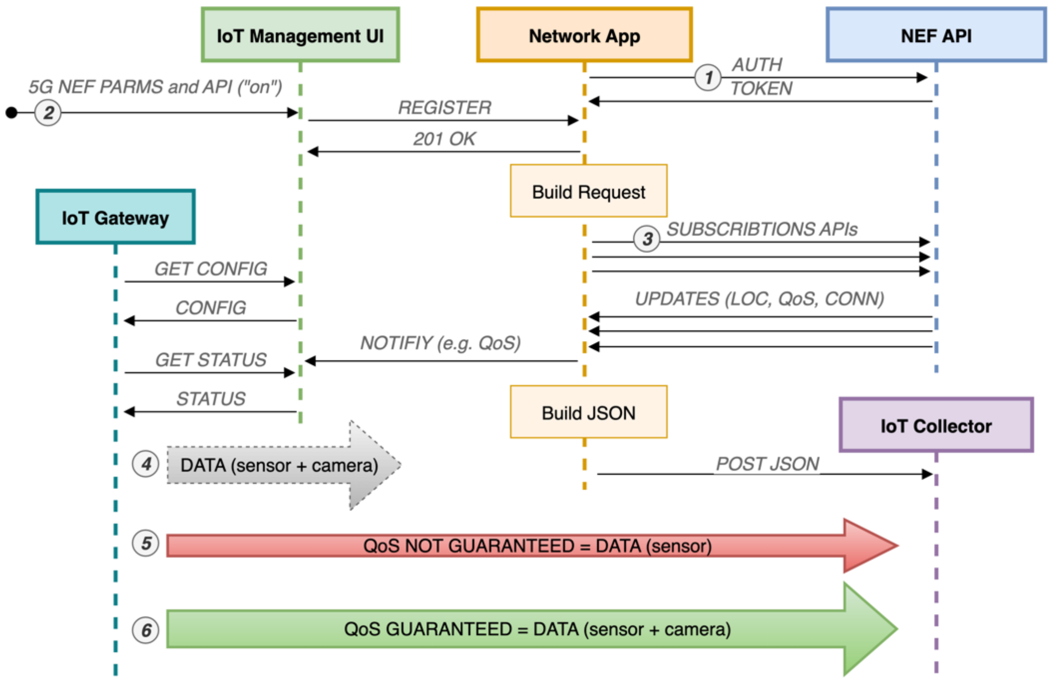


Figure 5 Sequence diagram of ININ Network App test and validation cycle

3.2.3 Additional dependencies

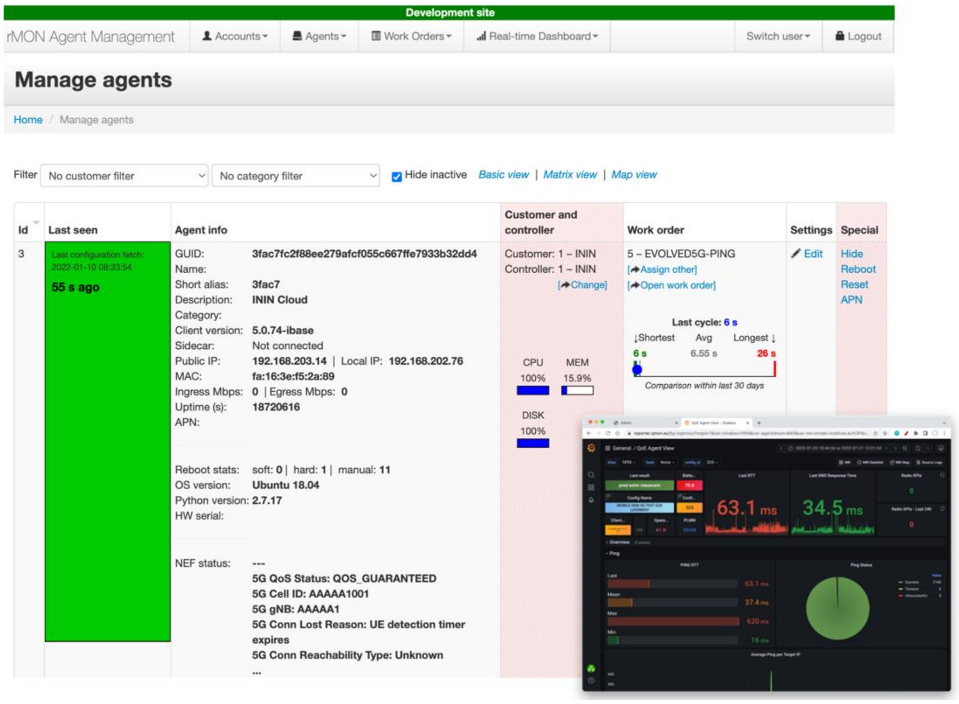
For Evolved-5G adoption and successful Network App deployment, some extensions for the 5G IoT System have been done. IoT management, which is primarily responsible for centralized management of IoT Gateways now supports extensions for NEF:

- NEF registration,
- NEF QoS subscriptions,
- database adjustments and analytics improvement.

Database adjustments and extensions for NEF have been done also at IoT Collector, which primarily collects data from sensor and KPIs provided by the engine running on itself.

Network App can be deployed anywhere as a container image, it requires connectivity to the NEF and must have bidirectional communication allowed with vApp over HTTPS, which temporarily runs in ININ private cloud, but can be otherwise deployed anywhere.

Figure 6 below describes Grafana based IoT Management and IoT Reporter.



Manage agents

Home / Manage agents

Filter: No customer filter | No category filter | ☑ Hide inactive | Basic view | Matrix view | Map view

Id	Last seen	Agent info	Customer and controller	Work order	Settings	Special
3	Last configuration fetch: 2022-01-10 08:33:54 **55 s ago**	GUID: **3fac7fc2f88ee279afcf055c667ffe7933b32dd4** Name: Short alias: **3fac7** Description: **ININ Cloud** Category: Client version: **5.0.74-ibase** Sidecar: Not connected Public IP: **192.168.203.14** | Local IP: **192.168.202.76** MAC: **fa:16:3e:f5:2a:89** Ingress Mbps: **0** | Egress Mbps: **0** Uptime (s): **18720616** APN: Reboot stats: soft: **0** | hard: **1** | manual: **11** OS version: **Ubuntu 18.04** Python version: **2.7.17** HW serial: NEF status: --- **5G QoS Status: QOS_GUARANTEED 5G Cell ID: AAAAA1001 5G gNB: AAAAA1 5G Conn Lost Reason: UE detection timer expires 5G Conn Reachability Type: Unknown** ...	Customer: 1 – ININ Controller: 1 – ININ [Change] CPU 100% MEM 15.9% DISK 100%	5 – EVOLVED5G-PING [Assign other] [Open work order] **Last cycle: 6 s** ↓Shortest 6 s Avg 6.55 s Longest ↓ 26 s *Comparison within last 30 days*	Edit	Hide Reboot Reset APN

Figure 6 IoT Management and IoT Reporter (Grafana)

Figure 7 below depicts 5G IoT Gateway is X86 Industrial Based computer with Sierra Wireless EM9191 modem (based on Qualcomm SD55). It requires 200V power supply and provisioned SIM card and APN settings.

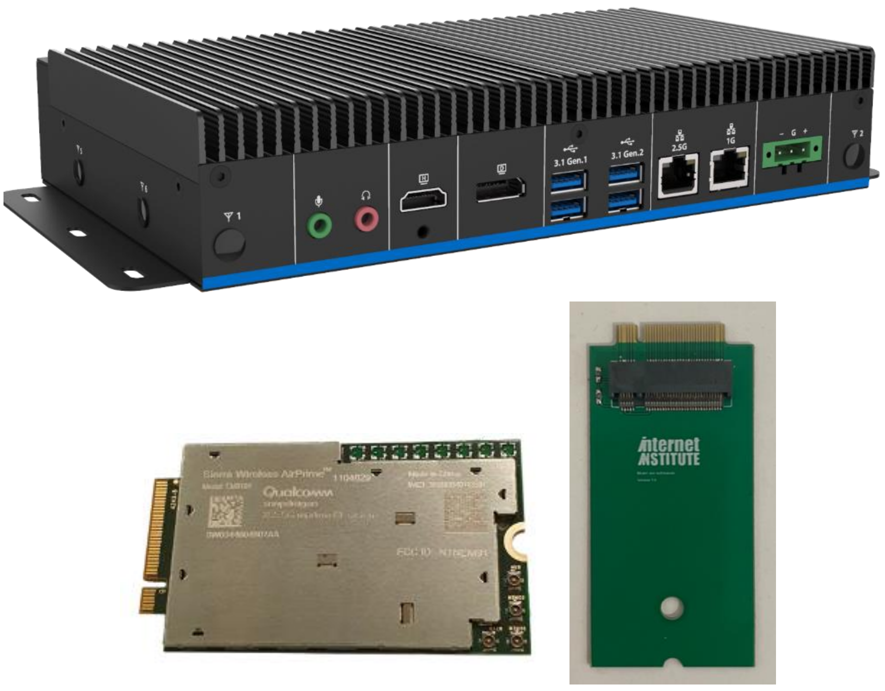


Figure 7 ININ's 5G IoT Gateway based on x86 industrial-grade PC, integrated Sierra Wireless EM9191 modem and ININ's in-house developed M2 board

3.3 UMA Smart Irrigation 5G Agriculture Network Application

3.3.1 Use case description

The smart irrigation use case consists of the deployment within the fields of a large number of sensors capable of measuring several parameters of interest, such as Vapor Pressure Deficit (VPD) or Frequency Domain Reflectometry (FDR), being quite useful to monitor the state of the field. These and additional parameters can be used for the creation of optimized irrigation schedules, in order to waste the least amount of water resources. At a later stage, this use case can be extended with the inclusion of a multispectral camera to provide an estimate of the yield that can be generated.

The use case has some limitations that hinder its actual development. The main ones are access to electricity and coverage. This implies the study of advanced wireless solutions that make efficient use of energy resources so that deployment is sustainable over a long period of time without human intervention.

In this stage of the use case implementation, we can already identify two main benefits derived from the use of 5G connectivity:

- Sensors do not need to be pre-configured with information about the location where they will be deployed. This greatly reduces the complexity when preparing the terrain, and also allows immediate reuse of the sensors in another location when needed.
- Since the location information is taken from the network, sensors do not need to make use of GPS signals, which reduces both the energy consumption and the cost per sensor.

3.3.2 Detailed Architecture

The architecture currently deployed in the field is shown in Figure 8. Starting from left to right, there are sensors that obtain measurements from the terrain and send them to the Network App using a 5G connection. The Network App then leverages the capabilities of the NEF emulator to obtain information about the specific cell through which the data was transmitted. This data is used to supplement the information received with details about the location where the measurements were taken and is stored in a centralized database. The vApp is responsible for requesting data from the Network App and representing them graphically, facilitating decision making for optimized irrigation.

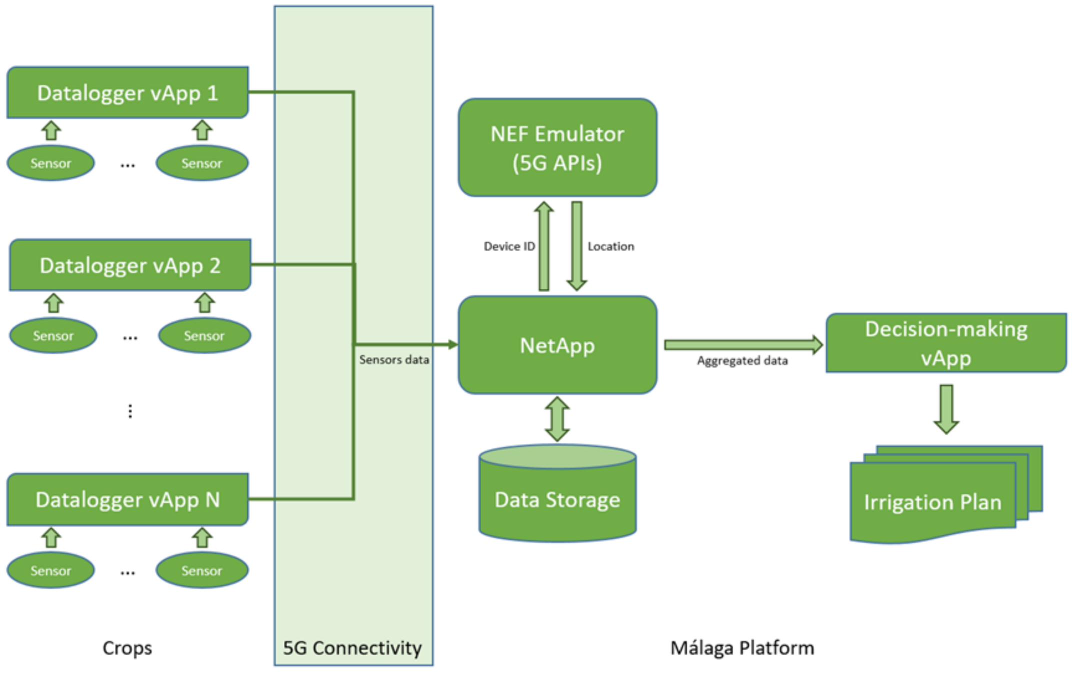


Figure 8 Smart Irrigation use case architecture

3.4 Anomaly Detection Network Application

3.4.1 Use case description

Anomaly Detection in the network context is vital to the management of the network infrastructure and services health during operations. Moreover, in the industrial network context and with the emergence of 5G for Non-Public Networks, it is expected to be an indispensable component of future deployments. However, the current integration with the specific to 5G core events and monitoring is in its infancy adopting in most of the applications proprietary or intrusive methods for retrieving 5G network information to detect anomalies. The EVOLVED-5G network anomaly detection application is based on concepts originally applied to wireless and wired networks that demand the deployment of the application within the protected/monitored network domain. The use case of such an application involves the following basic functional components:

- **Monitoring components**: Components that allow ingestion of monitoring information from various locations of the infrastructure
- **Policy component**: The component that allows the introduction of user and network attributes as well as associated policies
- **Detection and mitigation components**: Components that would be able to detect an anomaly that would disrupt normal operations of the infrastructure and perform mitigation actions.
- **Visualization and Management components**: Components that offer dashboard capabilities for management of the operation of the application as well as immediate monitoring of the current situation.

By approaching the anomaly detection for 5G deployments in Industry 4.0 setting via the EVOLVED -5G framework, a network application that integrates all the basic functional components mentioned above and at the same time providing a complete solution for safe industrial operations, has been developed.

To showcase the functionalities of such a network application, a small-scale factory production line was created, programmed to perform a sequence that simulates a factory operation which requires the coordination of multiple devices to perform a task. The small-scale production line consists of the following devices:

- **Robot Arm**: A Raspberry Pi with an embedded robot arm as shown in Figure 9 below.

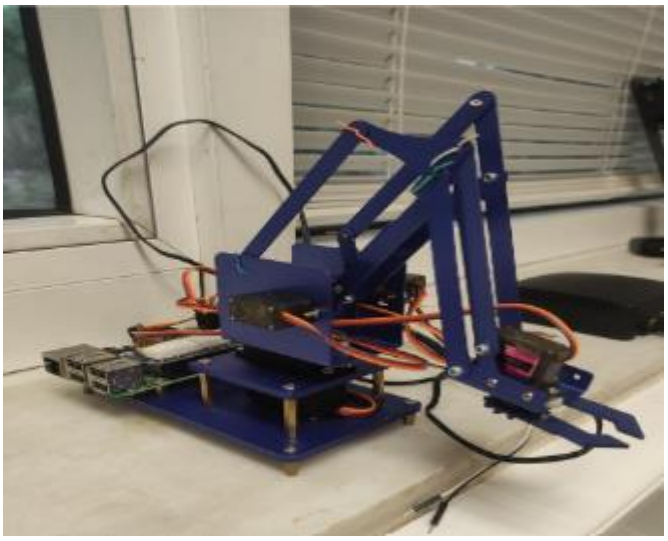


Figure 9 Robot Arm Raspberry Pi

- **Industrial Belt:** LEGO ev3 robot is modified as a small-scale industrial belt shown in Figure 10 below.

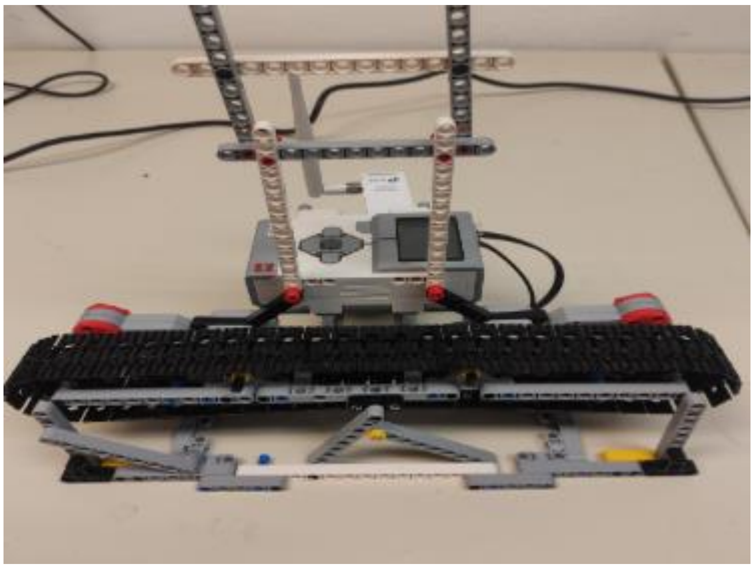


Figure 10 Lego Ev3 Industrial Belt

- **Camera Controller:** A Raspberry Pi with a camera module (shown in Figure 11 below).

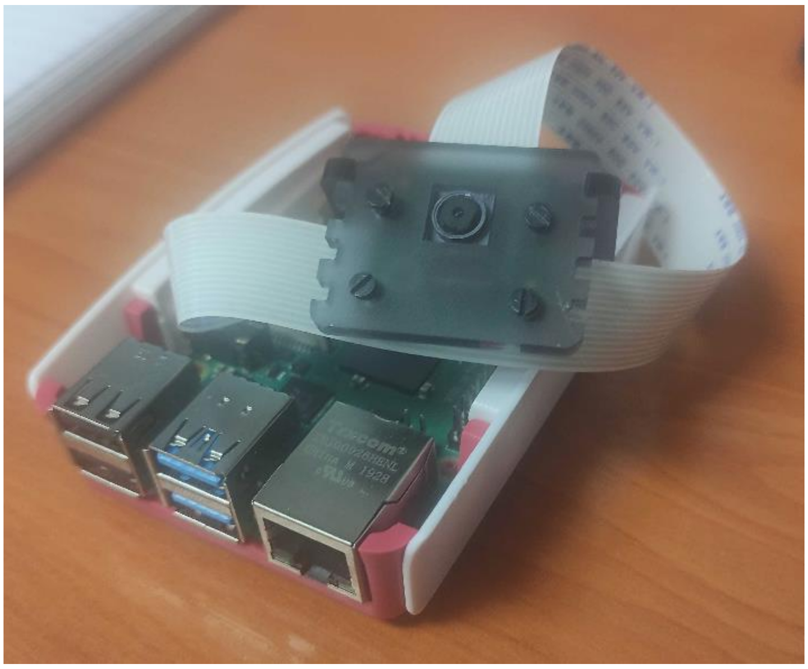


Figure 11 Raspberry Pi-Camera module

All components are running custom socket servers to achieve their coordination via the network. The whole setup is depicted bellow in Figure 12.

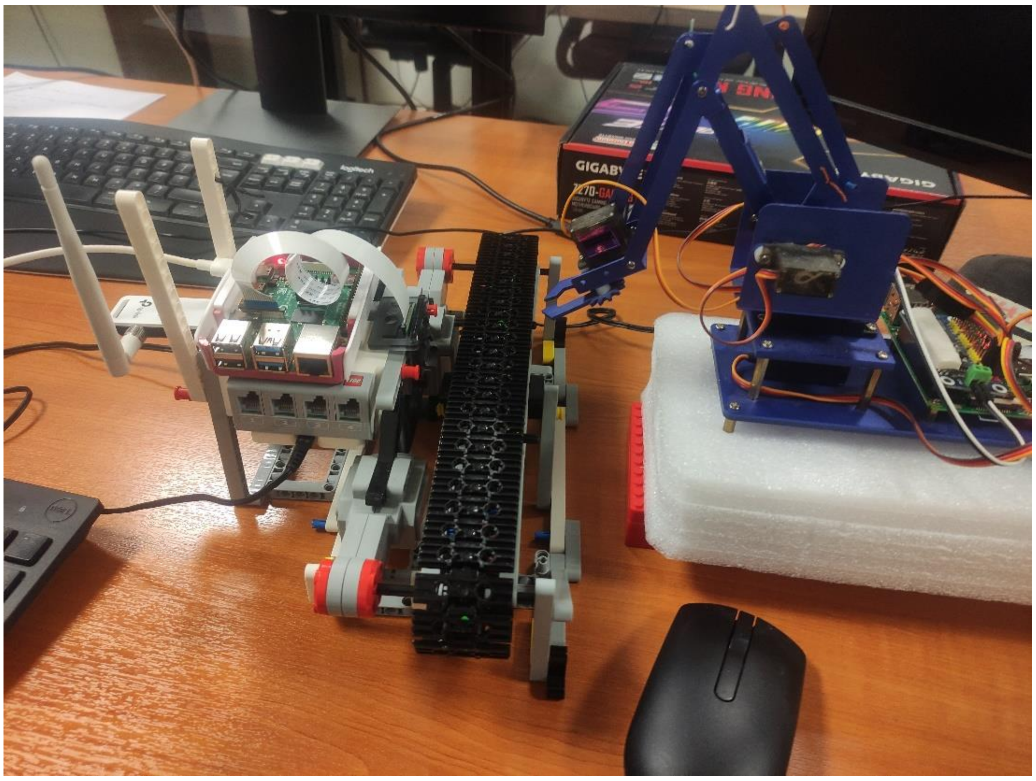


Figure 12 Footage of the small-scale production line simulation

The small-scale production line performs the sequence described by the diagram in Figure 13 bellow.

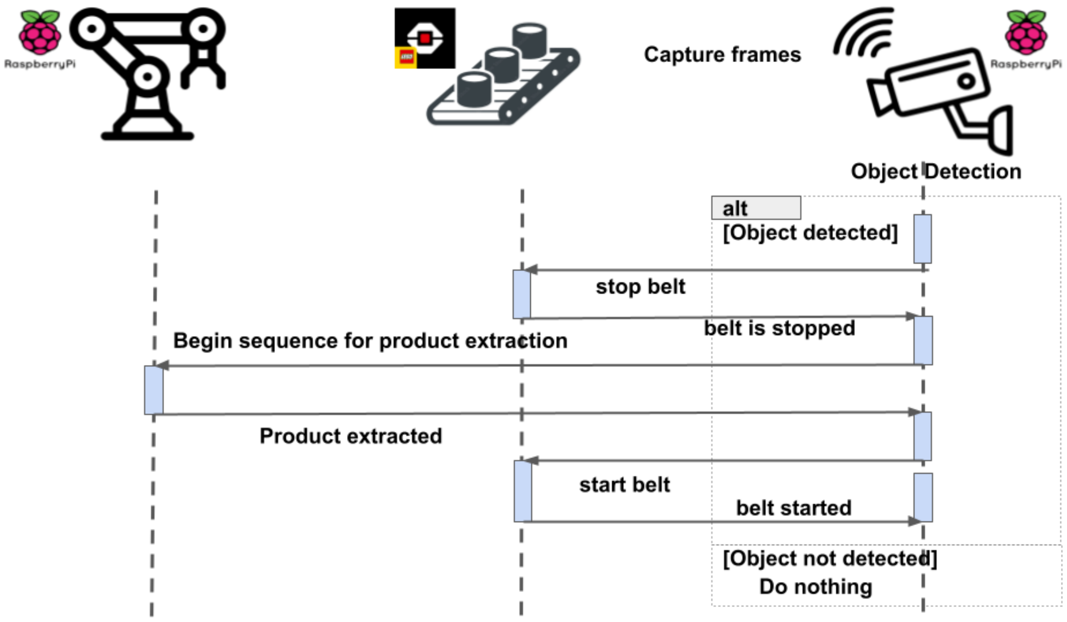


Figure 13 Flow Diagram of Production Line sequence

The operation starts with the belt at running state and the camera controller capturing video frames on a specific area of the belt. When an object is detected by the camera the controller instructs the following:

- Stop the belt and wait confirmation that the belt is stopped.
- Instruct the robot arm to extract the object and wait for confirmation that the object is extracted.
- Start the belt until another object is detected by the camera.

The described sequence can be seen in action at the consequent frames below. The final frame shows the object extraction from the robot arm:

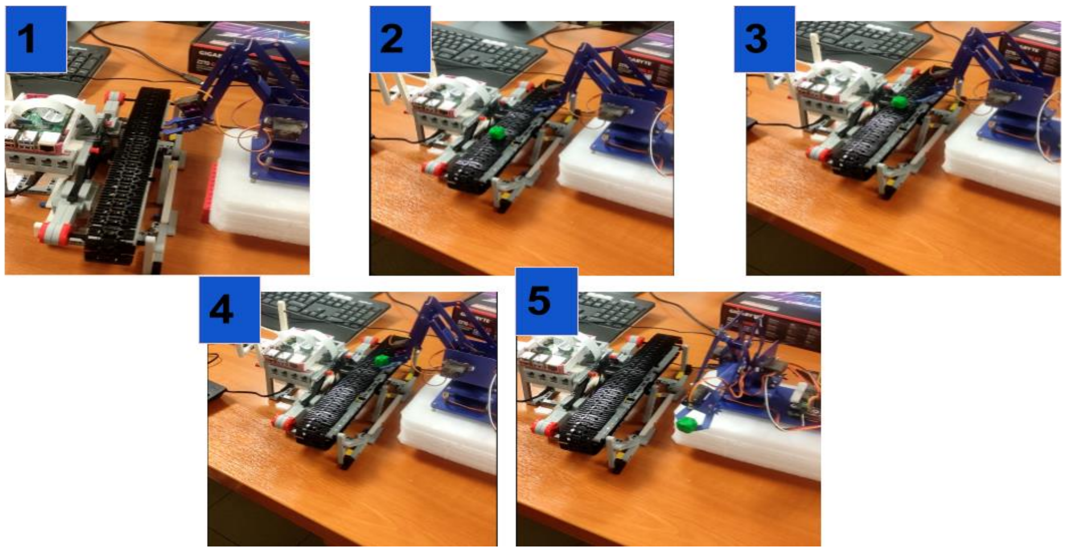
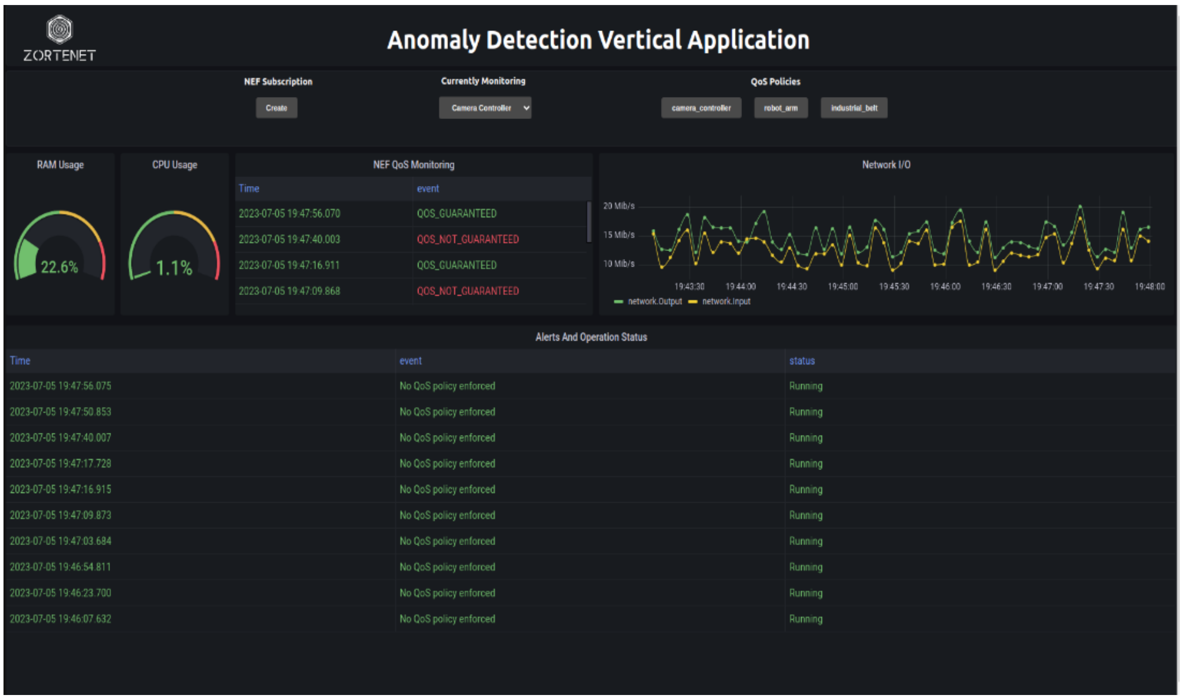


Figure 14 Frames of sequence

Zortenet’s Network App can provide to the end-user a holistic view of its monitored infrastructure and ensures the safety of the operation by using the policy enforcement and auto mitigation functionality. The user can create policies regarding which device is considered critical for the operation in terms of QoS. If one of the critical devices has not guaranteed QoS the mitigation action is triggered to halt the operation util the QoS is guaranteed. After the QoS of the critical devices is guaranteed, the operation is resumed. The end-user's view is provided below Figure 15 via a screenshot of the Vertical Application.



NEF QoS Monitoring

Time	event
2023-07-05 19:47:56.070	QOS_GUARANTEED
2023-07-05 19:47:40.003	QOS_NOT_GUARANTEED
2023-07-05 19:47:16.911	QOS_GUARANTEED
2023-07-05 19:47:09.868	QOS_NOT_GUARANTEED

Alerts And Operation Status

Time	event	status
2023-07-05 19:47:56.075	No QoS policy enforced	Running
2023-07-05 19:47:50.853	No QoS policy enforced	Running
2023-07-05 19:47:40.007	No QoS policy enforced	Running
2023-07-05 19:47:17.728	No QoS policy enforced	Running
2023-07-05 19:47:16.915	No QoS policy enforced	Running
2023-07-05 19:47:09.873	No QoS policy enforced	Running
2023-07-05 19:47:03.684	No QoS policy enforced	Running
2023-07-05 19:46:54.811	No QoS policy enforced	Running
2023-07-05 19:46:23.700	No QoS policy enforced	Running
2023-07-05 19:46:07.632	No QoS policy enforced	Running

Figure 15 Vertical Application Overview

Each component of the dashboard is enumerated and presented in Figure 16 below.

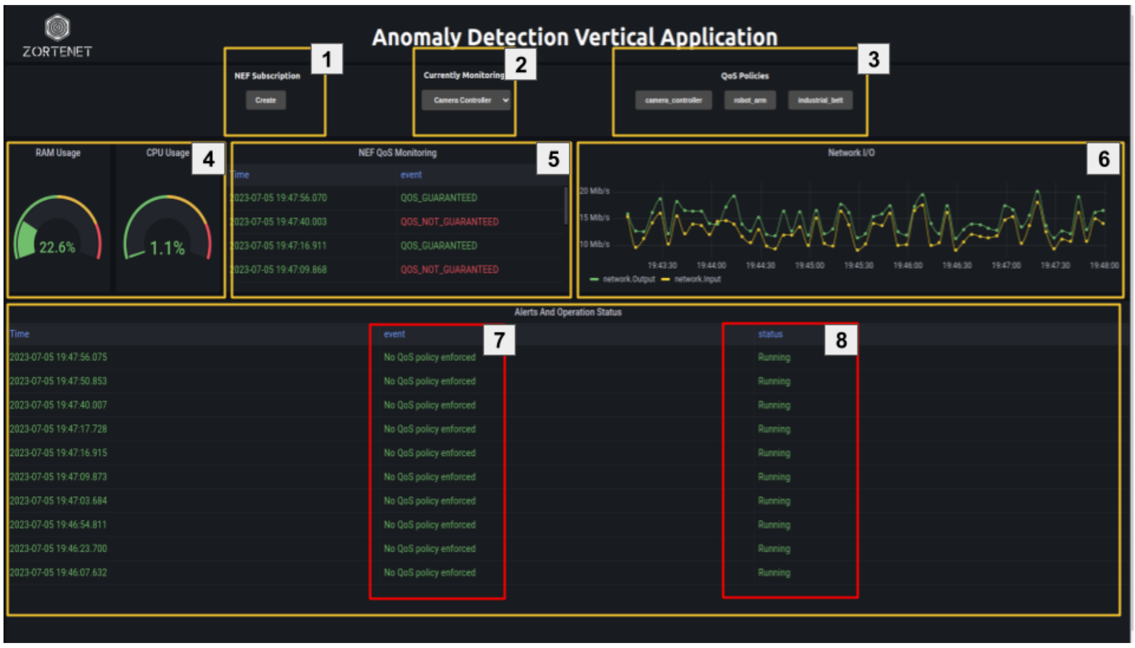
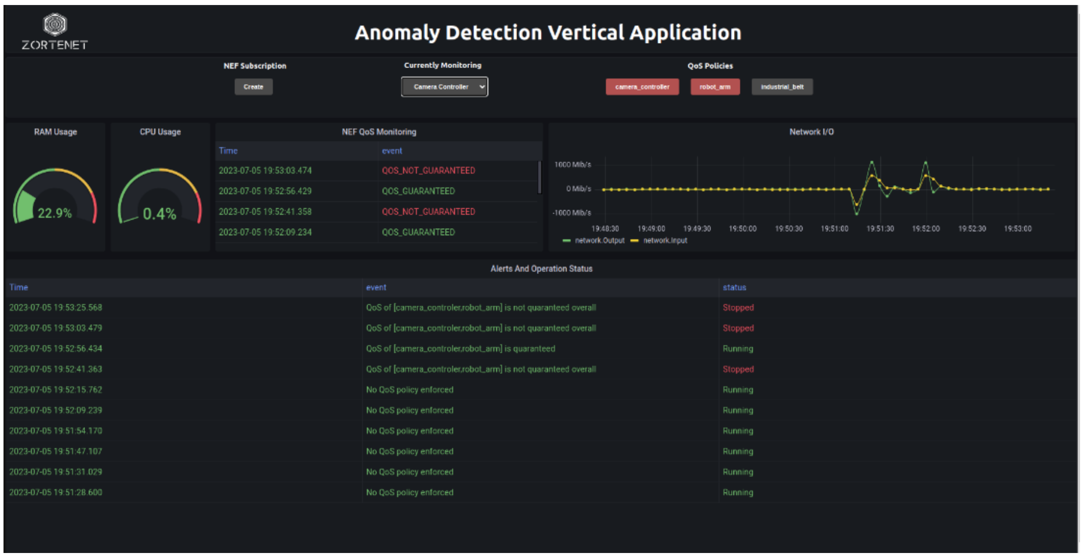


Figure 16 Dashboard Components

Zortenet's Network App deployment sequence is following:

- **1. NEF Subscription:** The user can create a subscription request to NEF by pressing this button
- **2. Currently Monitoring:** The user can choose which device to monitor on demand by selecting this dropdown menu. Views **no4**, **no5** and **no6** will change accordingly.
- **3. QoS Policies:** The user can specify which devices are considered critical for the operation.
- **4. RAM & CPU usages**: Gauges of the ram and CPU usages of the currently monitored device.
- **5. NEF QoS Monitoring**: NEF callbacks for the currently monitored device
- **6. Network I/O:** Network I/O metrics for the currently monitored device
- **7. Alerts and Operation status – event:** displays the QoS of the critical devices defined by the policies.
- **8. Alerts and Operation status – status:** displays the status of the operation (Stopped or Running)

Finally, an example of a view with enforced policies and alerts is provided in Figure 17 below. The status of the operation is stopped because the critical devices do not have a guaranteed QoS.



NEF QoS Monitoring

Time	event
2023-07-05 19:53:03.474	QOS_NOT_GUARANTEED
2023-07-05 19:52:56.429	QOS_GUARANTEED
2023-07-05 19:52:41.358	QOS_NOT_GUARANTEED
2023-07-05 19:52:09.234	QOS_GUARANTEED

Alerts And Operation Status

Time	event	status
2023-07-05 19:53:25.568	QoS of [camera_controler,robot_arm] is not guaranteed overall	Stopped
2023-07-05 19:53:03.479	QoS of [camera_controler,robot_arm] is not guaranteed overall	Stopped
2023-07-05 19:52:56.434	QoS of [camera_controler,robot_arm] is guaranteed	Running
2023-07-05 19:52:41.363	QoS of [camera_controler,robot_arm] is not guaranteed overall	Stopped
2023-07-05 19:52:15.762	No QoS policy enforced	Running
2023-07-05 19:52:09.239	No QoS policy enforced	Running
2023-07-05 19:51:54.170	No QoS policy enforced	Running
2023-07-05 19:51:47.107	No QoS policy enforced	Running
2023-07-05 19:51:31.029	No QoS policy enforced	Running
2023-07-05 19:51:28.600	No QoS policy enforced	Running

Figure 17 A view with enforced policies and alerts

3.4.2 Detailed Architecture

Zortenet's Anomaly Detection Network App follows the standalone approach, and it resides between the 5G Network (CAPIF and NEF) and the Vertical Application which is the interface that the end-user interacts with. The overall architecture of the Network App and the interactions among the is depicted in the Figure 18 below.

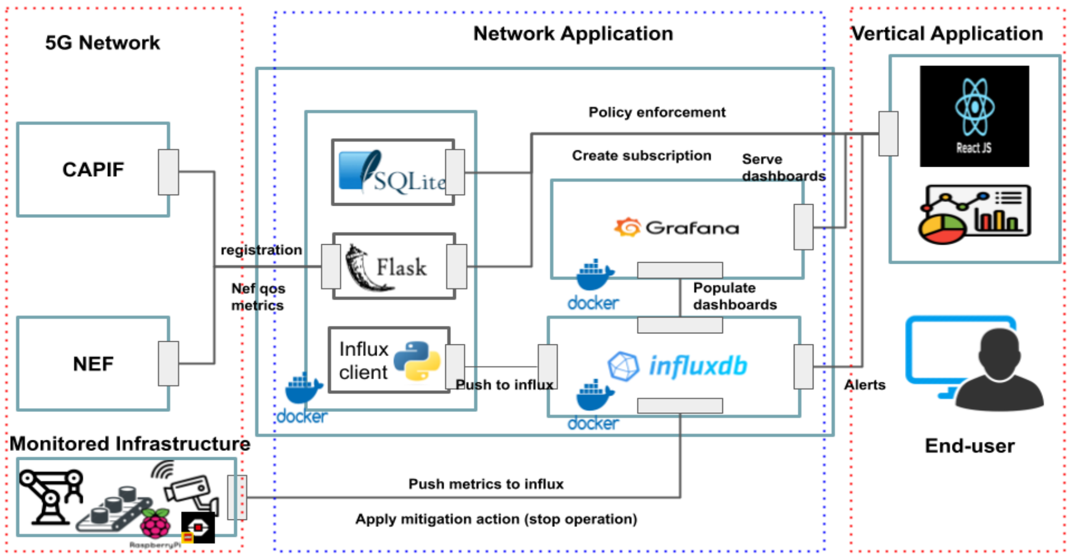


Figure 18 Detailed Architecture Diagram of Zortenet's Anomaly Detection Network App

The network application consists of three Docker containers:

- **Grafana**: This is a dockerized version of Grafana that provides the capability to serve dedicated dashboards. Grafana service has 2 interfaces as depicted:
    1. **Iface - Serve dashboards**: this interface is used by the vApp to access the dashboards.
    2. **Iface – Populate dashboards:** grafana communicates via this interface with the influxdb container to fetch real-time data and populate the visualization dashboards.
- **Influxdb:** Infuxdb as a time-series database allows the network application to collect and store real time metrics from the monitored infrastructure. Influxdb has 4 interfaces as described below:
    1. **Iface - Populate dashboards:** as described above the communication with grafana is achieved via this interface
    2. **Iface – Push metrics to influx & Apply mitigation action:** this interface is dedicated for the communication between the network's application influxdb service and the monitored infrastructure. The devices use this interface to push real time metrics such as CPU, RAM usages and Network I/O. Finally, this interface is also used to notify the monitored infrastructure for a mitigation action, in our use case the mitigation action is to halt the operation.
    3. **Iface – Alerts:** The vApp receives alerts through this interface.
    4. **Iface – Push to influx:** This interface is used from the network application back-end to push NEF Emulator's QoS metrics to influxdb.
- **App-backend:** This container is a basic python image container which embeds three different functionalities and their communication interfaces. The core component of this container is the python flask server which takes care the communication between NEF and the vApp:
    1. **Sqlite / Iface-Policy enforcement**: This functionality and its interface is used by the vApp where the end-user can enforce a QoS related policy about the monitored infrastructure.

2. **Flask server**: this functionality is responsible for the registration of the network application with CAPIF using the evolved5G SDK as depicted with the **iface-registration.** The end-user can create a NEF subscription from the vApp using the **iface-Create subscription.** Finally, the **iface-Nef qos metrics** is used by NEF to send its QoS callback after the subscription is created.
3. **Influx-client:** this is used to push NEF QoS metrics callbacks to influxdb to be visualized from the vApp dashboard.

3.4.3 Additional dependencies

Zortenet's Network App as a containerized application manages to integrate influxdb, Sqlite, Grafana and Python's flask and create an easily deployable stack as shown in Figure 19 below.

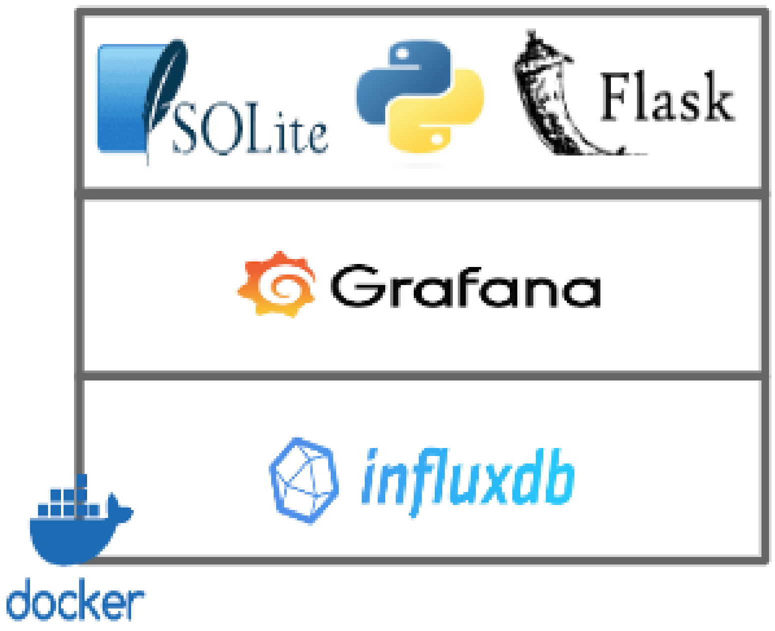


Figure 19 Zortenet Network Application Technologies Stack

4 INTEGRATION ACTIVITIES AND USE CASES TESTING

4.1 PURPOSE OF THE INTEGRATION TESTS (1ST ROUND)

The first round of integrations carried out by FoF partners spanned the period from April 2022 to April 2023 with the aim of connecting Network App, vApp, NEF, CAPIF and 5G network and cloud infrastructure.

In the initial phase of integration, the following components were utilized:

- Network Applications v1 and v2
- NEF releases up to v1.6.2
- CAPIF releases up to v2
- SDK releases up to v0.8.7.

The primary objective of these integrations was to verify the connectivity between specific Network Application, vApp and local 5G network and either Demokritos' or UMA's 5G network.

Among the Network Apps that are involved in the specific task, CAFA Tech’s and UMA’s cases have been tested on top of UMA’s infrastructure, while ININ and ZORTNET have been tested using Athens NCSRD infrastructure.

4.2 TOPOLOGY AND SETUP

4.2.1 CAFA NetMapper Network Application

The architecture reflecting the CAFA Network App initial tests in April 2022 at UMA is described in Figure 20 below.

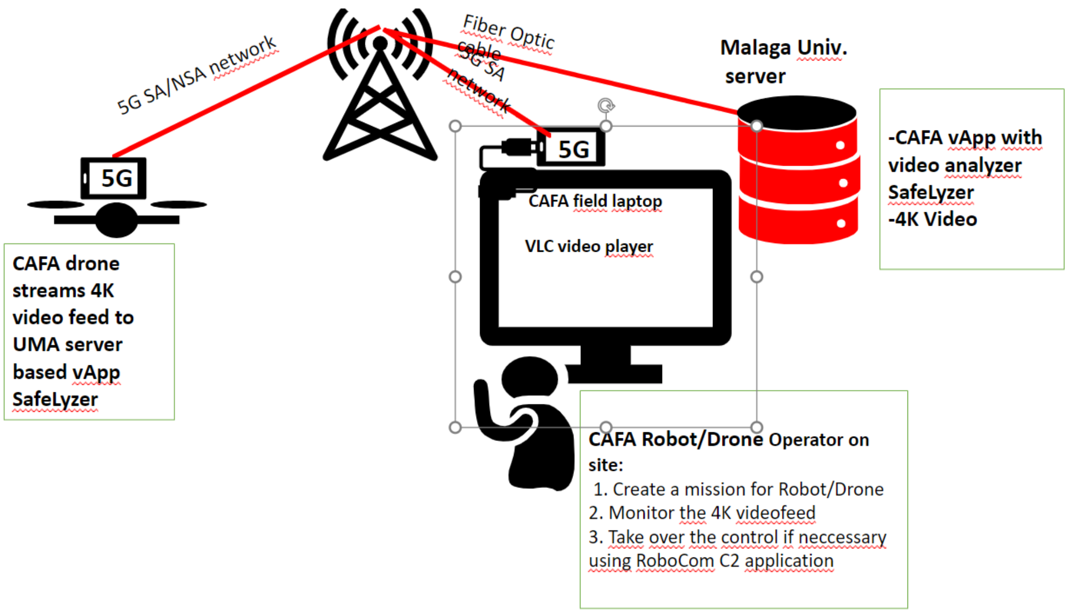

5G SA/NSA network
Fiber Optic cable
5G SA network
Malaga Univ. server
5G
5G
CAFA field laptop
VLC video player
-CAFA vApp with video analyzer SafeLyzer
-4K Video
CAFA drone streams 4K video feed to UMA server based vApp SafeLyzer
CAFA Robot/Drone Operator on site:
1. Create a mission for Robot/Drone
2. Monitor the 4K videofeed
3. Take over the control if neccessary using RoboCom C2 application

Figure 20 Architecture of the CAFA Network App tests in 2022 at UMA

CAFA Network Application "NetMapper" creates a GBR (Guaranteed Bit Rate) discrete automation subscription to the NEF Emulator in order to receive notifications about QoS (Quality of Service) to NetApp's Flask server's endpoint at port 5555.

Then the Flask server forwards the message to CAFA CV Flask server at port 5000 which changes the qos_state.txt file.

The vApp's Safety VideoLyzer conducts its computer vision task when (according to the qos state file) QOS_GUARANTEED or quits the computer vision to prevent unreliable results if QOS_NOT_GUARANTEED.

The Python files – one for starting the Flask server and writing the QoS State message to file, and the VideoLyzer application are in the same directory. The VideoLyzer computer vision application detects whether workers are wearing PPE (e.g., safety helmet). The detailed architecture of the NetMapper Network App is presented in subchapter 3.1.

4.2.2 Industrial Grade 5G Connectivity Network Application

The topology of the overall setup consists of multiple interconnected components as listed on Table 1 and the interaction among these components is represented in Figure 21

Table 1 below depicts list of building blocks/components involved in the integration, including respective components' owner/developer name and its deployment location according to the topology.

Table 1 List of building blocks/components involved in the ININ 1st Round integration

Building block	**Owner/developer**	**Location of SF or HW**
IoT Gateway	ININ	NCSR Demokritos testbed
IoT Management	ININ	ININ private cloud
IoT Collector	ININ	ININ private cloud
Network App	ININ	NCSR Demokritos private cloud
5G SA Network	NCSR Demokritos	NCSR Demokritos testbed
NEF	NCSR Demokritos	NCSR Demokritos private cloud
CAPIF	NCSR Demokritos	NCSR Demokritos private cloud

Figure 21 below depicts data flow during ININ Network App 1st round of deployment.

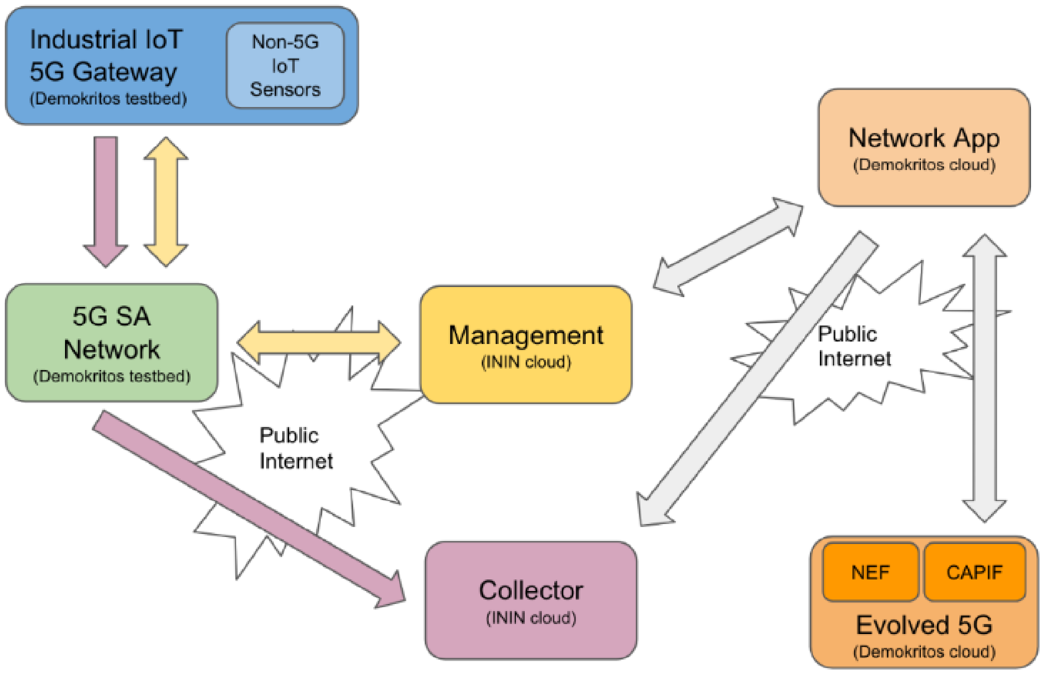
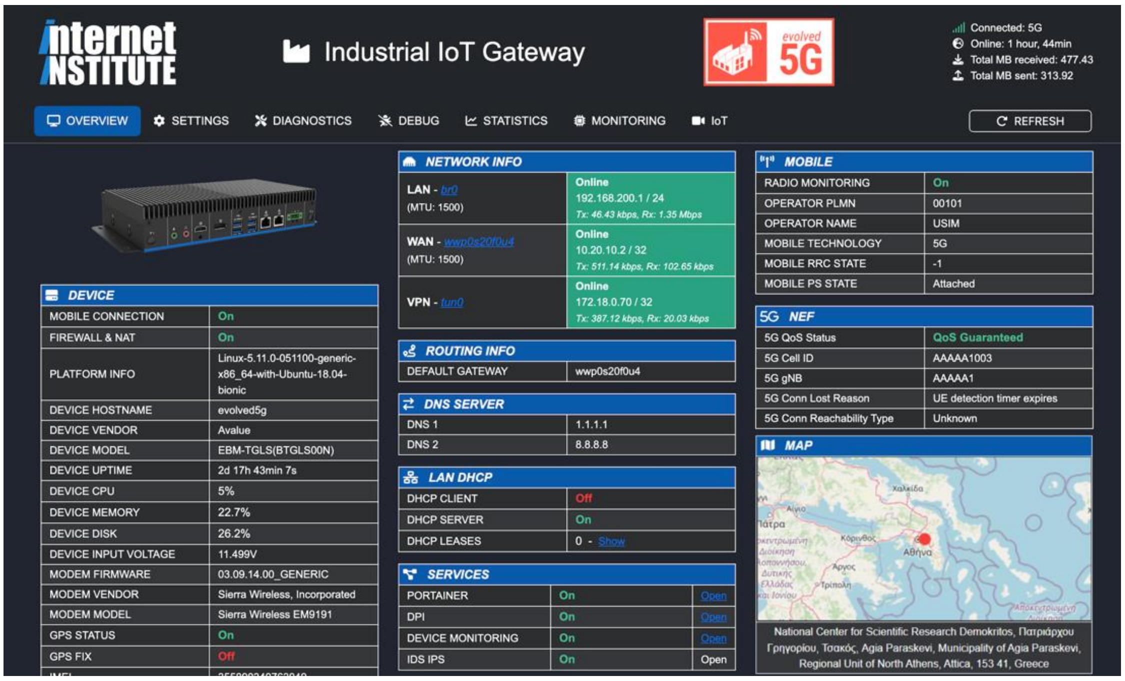

Industrial IoT 5G Gateway (Demokritos testbed)
Non-5G IoT Sensors
5G SA Network (Demokritos testbed)
Management (ININ cloud)
Public Internet
Collector (ININ cloud)
Network App (Demokritos cloud)
Public Internet
NEF
CAPIF
Evolved 5G (Demokritos cloud)

Figure 21 Data flow diagram during ININ Network App 1st round of deployment

The complete process of integration testing can be described and confirmed in the following sequence steps:

1. IoT GW is connected to 5G network at NCSR Demokritos testbed. Provisioned SIM card with appropriate APN settings was used to established 5G connectivity. This step is depicted in Figure 22, Figure 23 and Figure 24 below.



Connected: 5G
Online: 1 hour, 44min
Total MB received: 477.43
Total MB sent: 313.92

DEVICE	
MOBILE CONNECTION	On
FIREWALL & NAT	On
PLATFORM INFO	Linux-5.11.0-051100-generic-x86_64-with-Ubuntu-18.04-bionic
DEVICE HOSTNAME	evolved5g
DEVICE VENDOR	Avalue
DEVICE MODEL	EBM-TGLS(BTGLS00N)
DEVICE UPTIME	2d 17h 43min 7s
DEVICE CPU	5%
DEVICE MEMORY	22.7%
DEVICE DISK	26.2%
DEVICE INPUT VOLTAGE	11.499V
MODEM FIRMWARE	03.09.14.00_GENERIC
MODEM VENDOR	Sierra Wireless, Incorporated
MODEM MODEL	Sierra Wireless EM9191
GPS STATUS	On
GPS FIX	Off

NETWORK INFO	
LAN - br0 (MTU: 1500)	Online 192.168.200.1 / 24 *Tx: 46.43 kbps, Rx: 1.35 Mbps*
WAN - wwp0s20f0u4 (MTU: 1500)	Online 10.20.10.2 / 32 *Tx: 511.14 kbps, Rx: 102.65 kbps*
VPN - tun0	Online 172.18.0.70 / 32 *Tx: 387.12 kbps, Rx: 20.03 kbps*

ROUTING INFO	
DEFAULT GATEWAY	wwp0s20f0u4

DNS SERVER	
DNS 1	1.1.1.1
DNS 2	8.8.8.8

LAN DHCP	
DHCP CLIENT	Off
DHCP SERVER	On
DHCP LEASES	0 - Show

SERVICES		
PORTAINER	On	Open
DPI	On	Open
DEVICE MONITORING	On	Open
IDS IPS	On	Open

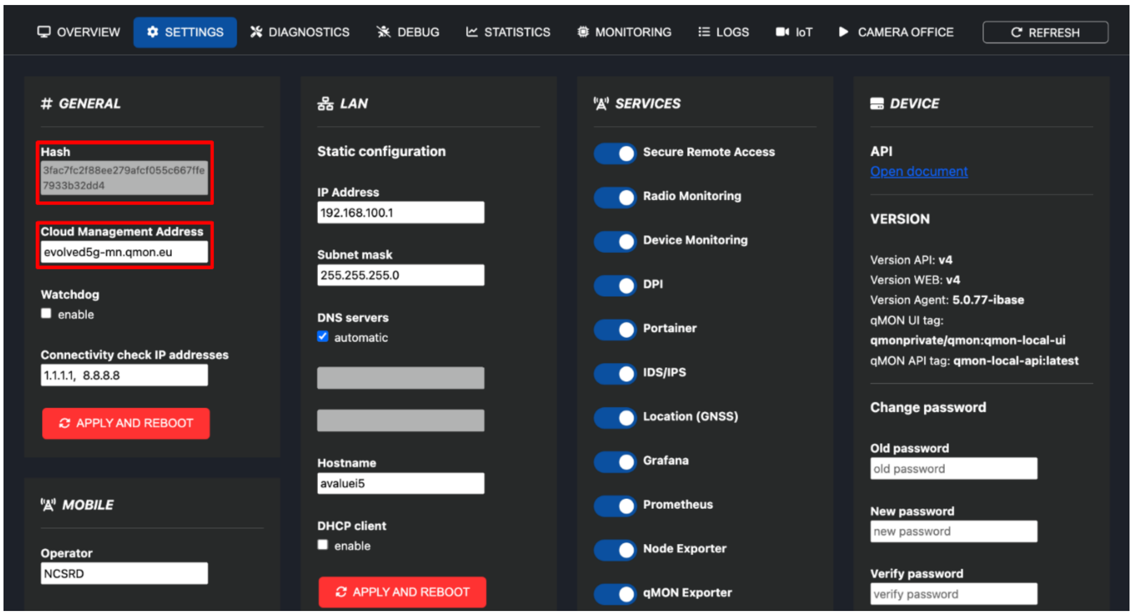
MOBILE	
RADIO MONITORING	On
OPERATOR PLMN	00101
OPERATOR NAME	USIM
MOBILE TECHNOLOGY	5G
MOBILE RRC STATE	-1
MOBILE PS STATE	Attached

5G NEF	
5G QoS Status	QoS Guaranteed
5G Cell ID	AAAAA1003
5G gNB	AAAAA1
5G Conn Lost Reason	UE detection timer expires
5G Conn Reachability Type	Unknown

MAP

National Center for Scientific Research Demokritos, Πατριάρχου Γρηγορίου, Τσακός, Agia Paraskevi, Municipality of Agia Paraskevi, Regional Unit of North Athens, Attica, 153 41, Greece

Figure 22 UE (IoT GW) main dashboard with 5G NEF support



# GENERAL

Hash
3fac7fc2f88ee279afcf055c667ffe7933b32dd4

Cloud Management Address
evolved5g-mn.qmon.eu

Watchdog
☐ enable

Connectivity check IP addresses
1.1.1.1, 8.8.8.8

APPLY AND REBOOT

MOBILE

Operator
NCSRD

LAN

Static configuration

IP Address
192.168.100.1

Subnet mask
255.255.255.0

DNS servers
☑ automatic

Hostname
avaluei5

DHCP client
☐ enable

APPLY AND REBOOT

SERVICES

- Secure Remote Access
- Radio Monitoring
- Device Monitoring
- DPI
- Portainer
- IDS/IPS
- Location (GNSS)
- Grafana
- Prometheus
- Node Exporter
- qMON Exporter

DEVICE

API
Open document

VERSION

Version API: v4
Version WEB: v4
Version Agent: 5.0.77-ibase
qMON UI tag:
qmonprivate/qmon:qmon-local-ui
qMON API tag: qmon-local-api:latest

Change password

Old password
old password

New password
new password

Verify password
verify password

Figure 23 UE (IoT GW) settings view with unique hash and IoT Management URL

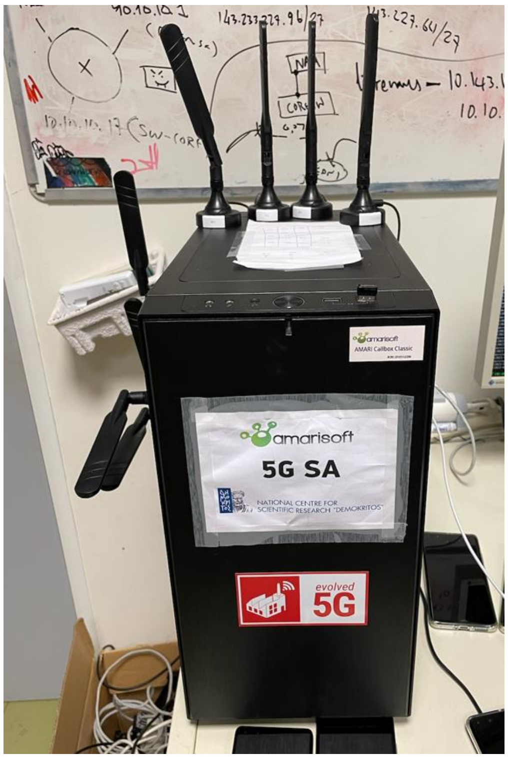
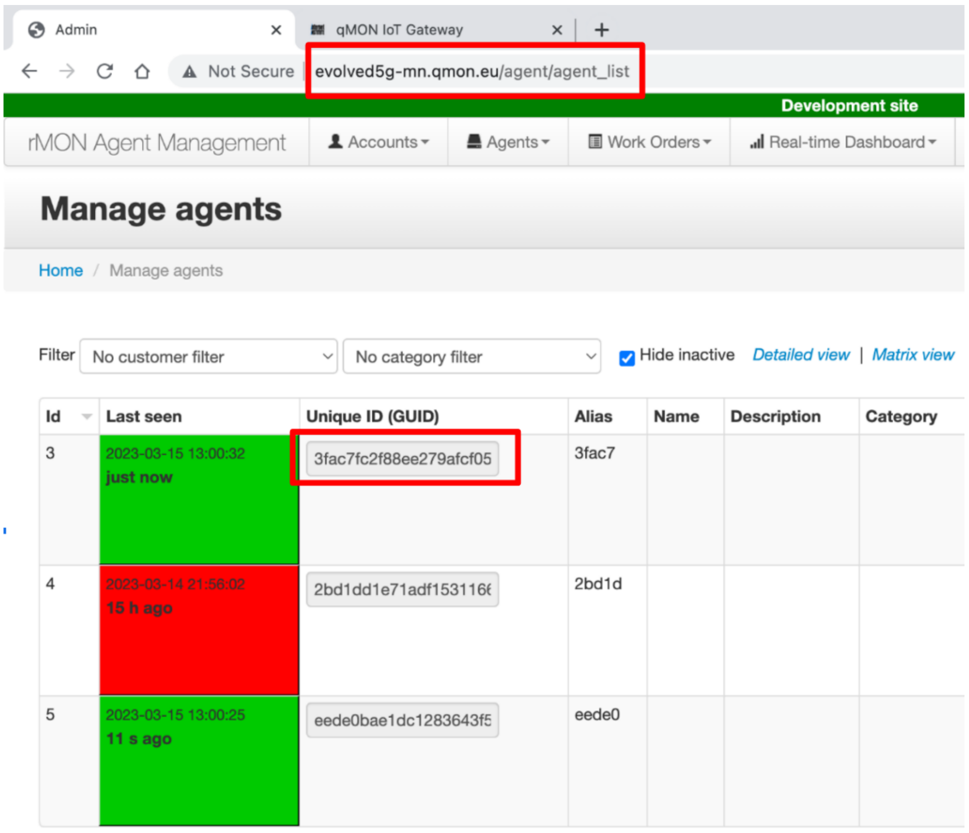


Figure 24 5G SA Network station at NCSR Demokritos

2. IoT GW has established a connection to IoT Management and IoT Collector over 5G network. A secure VPN connection was formed to provide connectivity between ININ and NSCR Demokritos clouds as shown in Figure 25 below.



Admin | qMON IoT Gateway

Not Secure | evolved5g-mn.qmon.eu/agent/agent_list

Development site

rMON Agent Management | Accounts | Agents | Work Orders | Real-time Dashboard

Manage agents

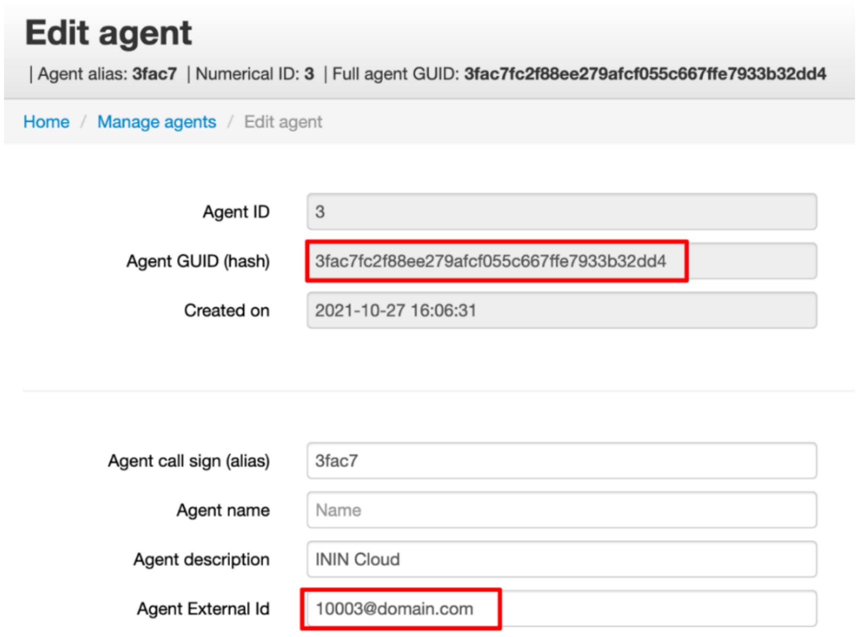
Home / Manage agents

Filter: No customer filter | No category filter | ☑ Hide inactive | Detailed view | Matrix view

Id	Last seen	Unique ID (GUID)	Alias	Name	Description	Category
3	🟩 2023-03-15 13:00:32 just now	3fac7fc2f88ee279afcf05	3fac7			
4	🟥 2023-03-14 21:56:02 15 h ago	2bd1dd1e71adf153116	2bd1d			
5	🟩 2023-03-15 13:00:25 11 s ago	eede0bae1dc1283643f5	eede0			

Figure 25 IoT Management with observed IoT Gateway (hash unique)

3. IoT Management subscribes (based on SLA/QoS requirements) IoT GW for receiving events notifications through Network App as shown in Figure 26 and Figure 27 below.



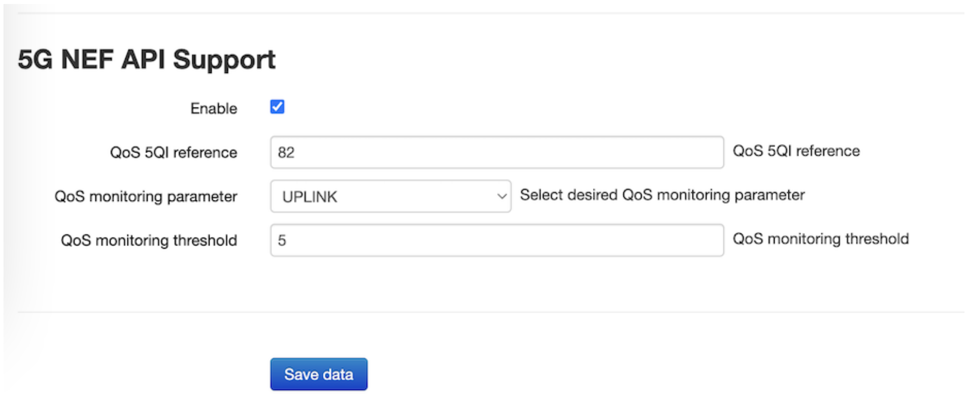
Edit agent

| Agent alias: **3fac7** | Numerical ID: **3** | Full agent GUID: **3fac7fc2f88ee279afcf055c667ffe7933b32dd4**

Home / Manage agents / Edit agent

Agent ID	3
Agent GUID (hash)	3fac7fc2f88ee279afcf055c667ffe7933b32dd4
Created on	2021-10-27 16:06:31
Agent call sign (alias)	3fac7
Agent name	Name
Agent description	ININ Cloud
Agent External Id	10003@domain.com

Figure 26 UE (IoT GW) must be preconfigured with unique identifier – External ID for NEF

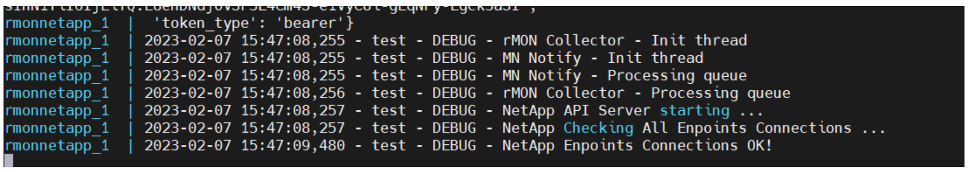


5G NEF API Support

Enable	✓	
QoS 5QI reference	82	QoS 5QI reference
QoS monitoring parameter	UPLINK	Select desired QoS monitoring parameter
QoS monitoring threshold	5	QoS monitoring threshold

Figure 27 Enabling 5G NEF events per UE through IoT Management

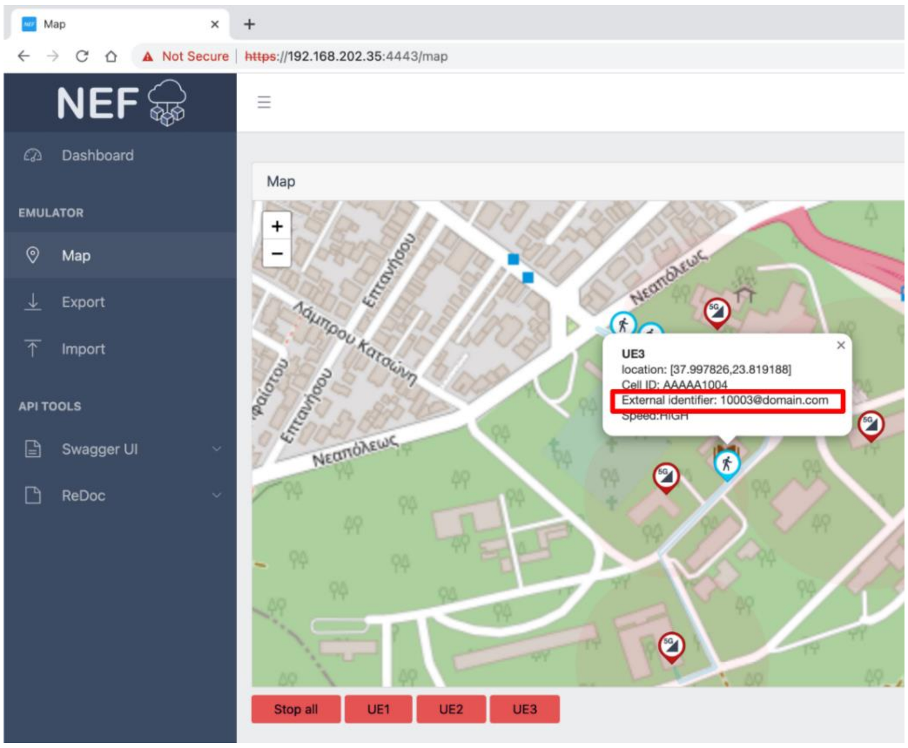
4. Network App was successfully deployed. It authenticates and immediately starts receiving notifications from NEF as shown in Figure 28 below.



```
rmonnetapp_1  |   'token_type': 'bearer'}
rmonnetapp_1  | 2023-02-07 15:47:08,255 - test - DEBUG - rMON Collector - Init thread
rmonnetapp_1  | 2023-02-07 15:47:08,255 - test - DEBUG - MN Notify - Init thread
rmonnetapp_1  | 2023-02-07 15:47:08,255 - test - DEBUG - MN Notify - Processing queue
rmonnetapp_1  | 2023-02-07 15:47:08,256 - test - DEBUG - rMON Collector - Processing queue
rmonnetapp_1  | 2023-02-07 15:47:08,257 - test - DEBUG - NetApp API Server starting ...
rmonnetapp_1  | 2023-02-07 15:47:08,257 - test - DEBUG - NetApp Checking All Enpoints Connections ...
rmonnetapp_1  | 2023-02-07 15:47:09,480 - test - DEBUG - NetApp Enpoints Connections OK!
```

Figure 28 Network App was launched successfully

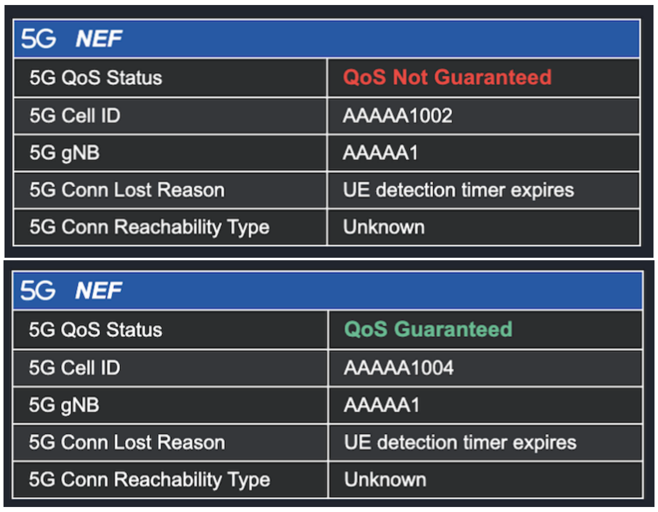
Network App notifies IoT Management on events in 5G Network – simulated by the NEF emulator (e.g., Location data, QoS data). IoT Management reports status to IoT GW as shown in Figure 29 below.



UE3
location: [37.997826,23.819188]
Cell ID: AAAAA1004
External identifier: 10003@domain.com
Speed:HIGH

Figure 29 Observed UE in NEF emulator

5. IoT GW takes action (starts or stops transmitting BE data) based on actual 5G network conditions. Non-5G based devices (e.g., camera) will be restricted from transmitting BE traffic if the QoS is not guaranteed as shown in Figure 30 below.



5G *NEF*	
5G QoS Status	QoS Not Guaranteed
5G Cell ID	AAAAA1002
5G gNB	AAAAA1
5G Conn Lost Reason	UE detection timer expires
5G Conn Reachability Type	Unknown

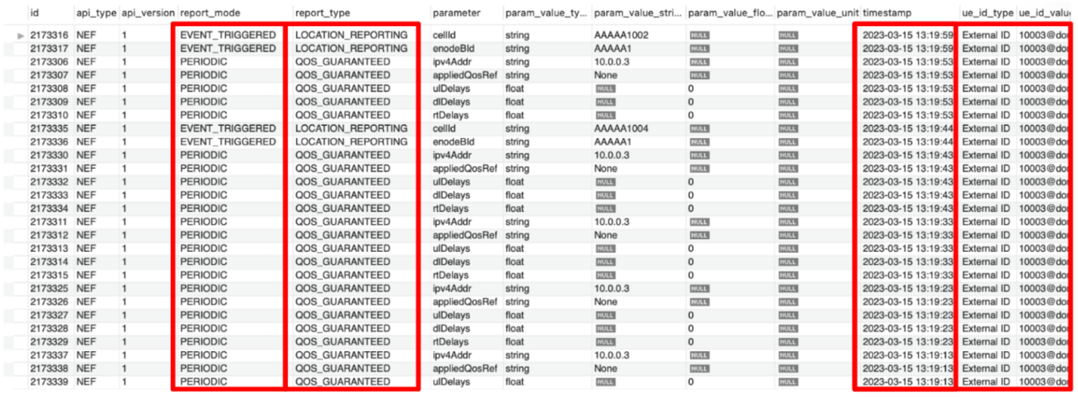
5G *NEF*	
5G QoS Status	QoS Guaranteed
5G Cell ID	AAAAA1004
5G gNB	AAAAA1
5G Conn Lost Reason	UE detection timer expires
5G Conn Reachability Type	Unknown

Figure 30 Screenshoot of the network conditions, i.e., QoS, monitoring shows both of the two options possible (QoS Guaranteed vs. QoS Not Guaranteed)

Figure 31 below depicts surveillance video-camera – a sample of best effort traffic generator used in the respective use case.

Figure 31 Surveillance video-camera

6. Network App puts results to IoT Collector database as shown in Figure 32 below.



id	api_type	api_version	report_mode	report_type	parameter	param_value_ty...	param_value_stri...	param_value_flo...	param_value_unit	timestamp	ue_id_type	ue_id_valu
2173316	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1002	NULL	NULL	2023-03-15 13:19:59	External ID	10003@do
2173317	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-03-15 13:19:59	External ID	10003@do
2173306	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-03-15 13:19:53	External ID	10003@do
2173307	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-03-15 13:19:53	External ID	10003@do
2173308	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-03-15 13:19:53	External ID	10003@do
2173309	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-03-15 13:19:53	External ID	10003@do
2173310	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-03-15 13:19:53	External ID	10003@do
2173335	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1004	NULL	NULL	2023-03-15 13:19:44	External ID	10003@do
2173336	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-03-15 13:19:44	External ID	10003@do
2173330	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-03-15 13:19:43	External ID	10003@do
2173331	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-03-15 13:19:43	External ID	10003@do
2173332	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-03-15 13:19:43	External ID	10003@do
2173333	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-03-15 13:19:43	External ID	10003@do
2173334	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-03-15 13:19:43	External ID	10003@do
2173311	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-03-15 13:19:33	External ID	10003@do
2173312	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-03-15 13:19:33	External ID	10003@do
2173313	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-03-15 13:19:33	External ID	10003@do
2173314	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-03-15 13:19:33	External ID	10003@do
2173315	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-03-15 13:19:33	External ID	10003@do
2173325	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-03-15 13:19:23	External ID	10003@do
2173326	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-03-15 13:19:23	External ID	10003@do
2173327	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-03-15 13:19:23	External ID	10003@do
2173328	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-03-15 13:19:23	External ID	10003@do
2173329	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-03-15 13:19:23	External ID	10003@do
2173337	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-03-15 13:19:13	External ID	10003@do
2173338	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-03-15 13:19:13	External ID	10003@do
2173339	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-03-15 13:19:13	External ID	10003@do

Figure 32 IoT Collector database storing Network App results

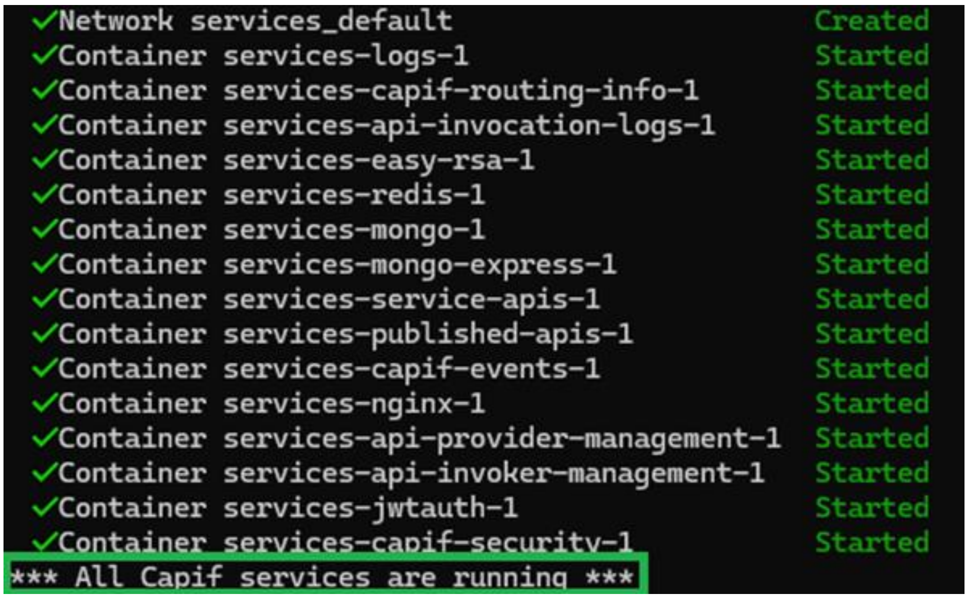
4.2.3 Smart Irrigation 5G Agriculture Network Application

The following steps have been deployed manually outside of Kubernetes in Openstack, using Docker containers on a PC locally.

The Network App is a Docker container, which uses Flask. Flask is a Python framework that allows to create a REST API. The endpoints that make up the API make it possible to obtain data from the sensors that are positioned on the ground. These data are stored in a PostgreSQL database that will be managed by the SQL Alchemy library. The data is then used in the vApp to make an optimal decision.

The code of the Network App can be seen in the following repository: https://github.com/EVOLVED-5G/UmaCsicNetApp/tree/evolved5g

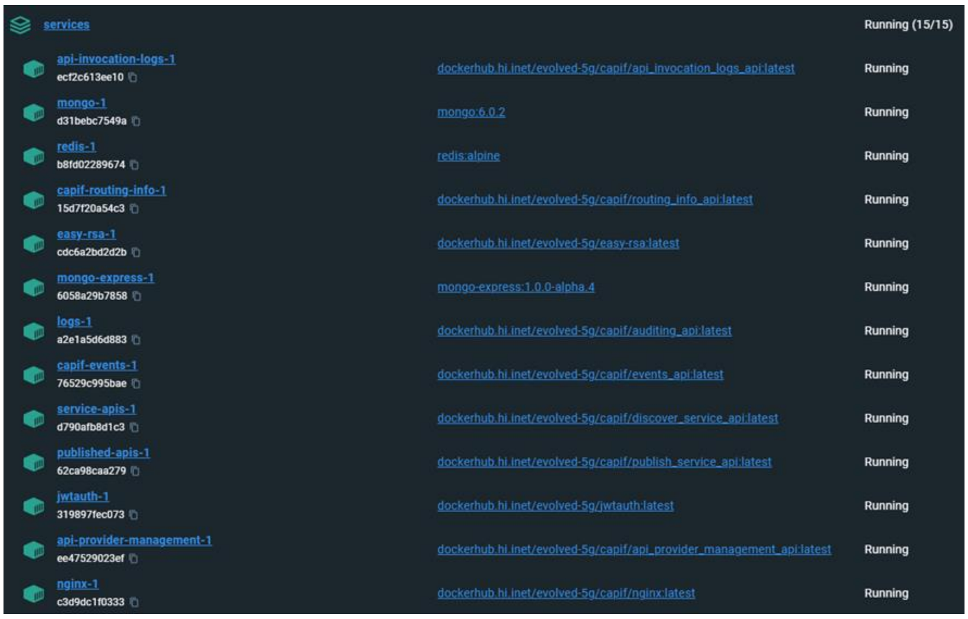
Going deeper into the technical part, the first thing to do is launch the CAPIF containers. When all CAPIF services are up as shown in Figure 33, the NEF containers are lifted. The NEF dashboard is logged in so that the scenario can be imported.



```
 ✓Network services_default                       Created
 ✓Container services-logs-1                      Started
 ✓Container services-capif-routing-info-1        Started
 ✓Container services-api-invocation-logs-1       Started
 ✓Container services-easy-rsa-1                  Started
 ✓Container services-redis-1                     Started
 ✓Container services-mongo-1                     Started
 ✓Container services-mongo-express-1             Started
 ✓Container services-service-apis-1              Started
 ✓Container services-published-apis-1            Started
 ✓Container services-capif-events-1              Started
 ✓Container services-nginx-1                     Started
 ✓Container services-api-provider-management-1   Started
 ✓Container services-api-invoker-management-1    Started
 ✓Container services-jwtauth-1                   Started
 ✓Container services-capif-security-1            Started
*** All Capif services are running ***
```

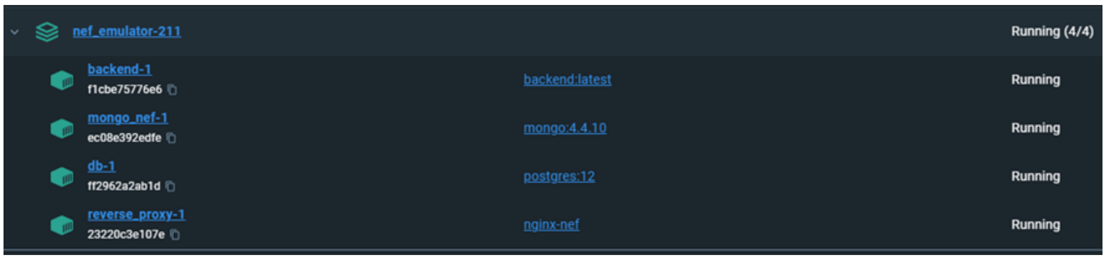
Figure 33 CAPIF services running

When the CAPIF and NEF containers are lifted as shown in Figure 34 and Figure 35, proceed to lift those of the Network App. If everything went well, the result obtained is shown in Figure 36.



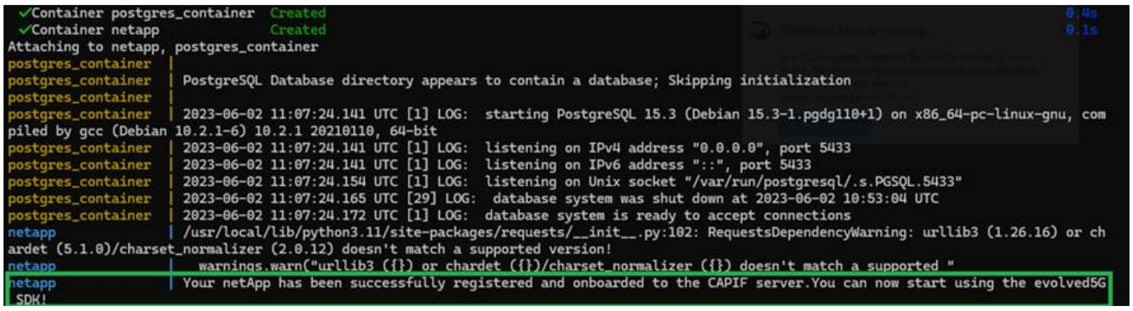
```
services                                                              Running (15/15)
api-invocation-logs-1     dockerhub.hi.inet/evolved-5g/capif/api_invocation_logs_api:latest      Running
ecf2c613ee10
mongo-1                   mongo:6.0.2                                                            Running
d31bebc7549a
redis-1                   redis:alpine                                                           Running
b8fd02289674
capif-routing-info-1      dockerhub.hi.inet/evolved-5g/capif/routing_info_api:latest             Running
15d7f20a54c3
easy-rsa-1                dockerhub.hi.inet/evolved-5g/easy-rsa:latest                           Running
cdc6a2bd2d2b
mongo-express-1           mongo-express:1.0.0-alpha.4                                            Running
6058a29b7858
logs-1                    dockerhub.hi.inet/evolved-5g/capif/auditing_api:latest                 Running
a2e1a5d6d883
capif-events-1            dockerhub.hi.inet/evolved-5g/capif/events_api:latest                   Running
76529c995bae
service-apis-1            dockerhub.hi.inet/evolved-5g/capif/discover_service_api:latest         Running
d790afb8d1c3
published-apis-1          dockerhub.hi.inet/evolved-5g/capif/publish_service_api:latest          Running
62ca98caa279
jwtauth-1                 dockerhub.hi.inet/evolved-5g/jwtauth:latest                            Running
319897fec073
api-provider-management-1 dockerhub.hi.inet/evolved-5g/capif/api_provider_management_api:latest  Running
ee47529023ef
nginx-1                   dockerhub.hi.inet/evolved-5g/capif/nginx:latest                        Running
c3d9dc1f0333
```

Figure 34 CAPIF containers up



```
nef_emulator-211                                        Running (4/4)
backend-1          backend:latest                       Running
f1cbe75776e6
mongo_nef-1        mongo:4.4.10                         Running
ec08e392edfe
db-1               postgres:12                          Running
ff2962a2ab1d
reverse_proxy-1    nginx-nef                            Running
23220c3e107e
```

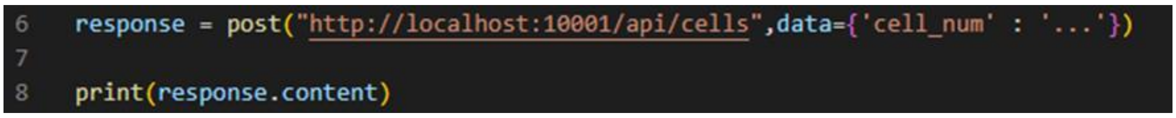
Figure 35 NEF containers up



```
 ✓Container postgres_container  Created                                                          0.4s
 ✓Container netapp              Created                                                          0.1s
Attaching to netapp, postgres_container
postgres_container  |
postgres_container  | PostgreSQL Database directory appears to contain a database; Skipping initialization
postgres_container  |
postgres_container  | 2023-06-02 11:07:24.141 UTC [1] LOG:  starting PostgreSQL 15.3 (Debian 15.3-1.pgdg110+1) on x86_64-pc-linux-gnu, com
piled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
postgres_container  | 2023-06-02 11:07:24.141 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5433
postgres_container  | 2023-06-02 11:07:24.141 UTC [1] LOG:  listening on IPv6 address "::", port 5433
postgres_container  | 2023-06-02 11:07:24.154 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5433"
postgres_container  | 2023-06-02 11:07:24.165 UTC [29] LOG:  database system was shut down at 2023-06-02 10:53:04 UTC
postgres_container  | 2023-06-02 11:07:24.172 UTC [1] LOG:  database system is ready to accept connections
netapp              | /usr/local/lib/python3.11/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or ch
ardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
netapp              |   warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
netapp              | Your netApp has been successfully registered and onboarded to the CAPIF server.You can now start using the evolved5G
 SDK!
```

Figure 36 NetApp linked to CAPIF and NEF

A cell is created in the Network App by means of a POST request as in Figure 37. The cell will receive data from sensors located in the field through a 5G network.



```python
response = post("http://localhost:10001/api/cells",data={'cell_num' : '...'})

print(response.content)
```

Figure 37 POST request to create the cell

With the idea of representing the data obtained graphically, a vApp is created. The deployment mechanism is the same as the previous ones, through containers.

4.2.4 Anomaly Detection Network Application

In the cloud infrastructure of NCSRD Openstack the depicted topology and setup was followed as shown in Figure 38 Figure 39 below.

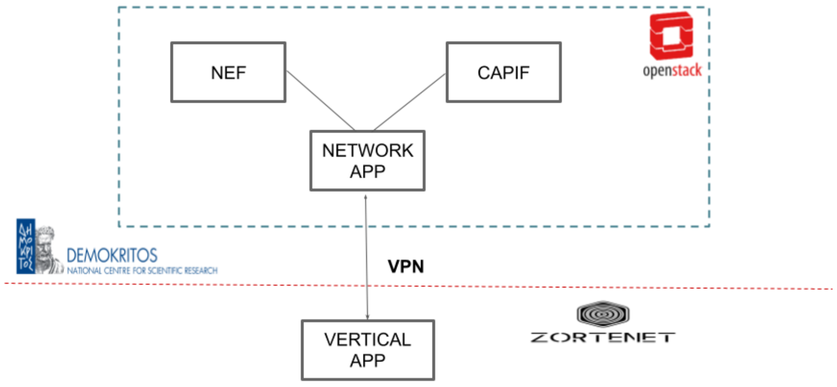
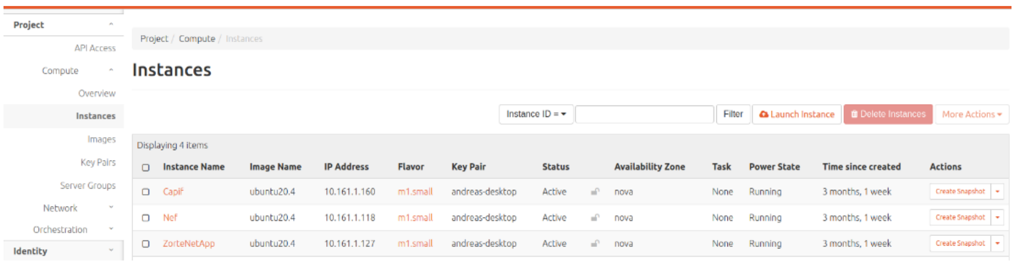


Figure 38 1st Round of Integration Topology

The creation of three separate VM was required as show in Figure 39 below.



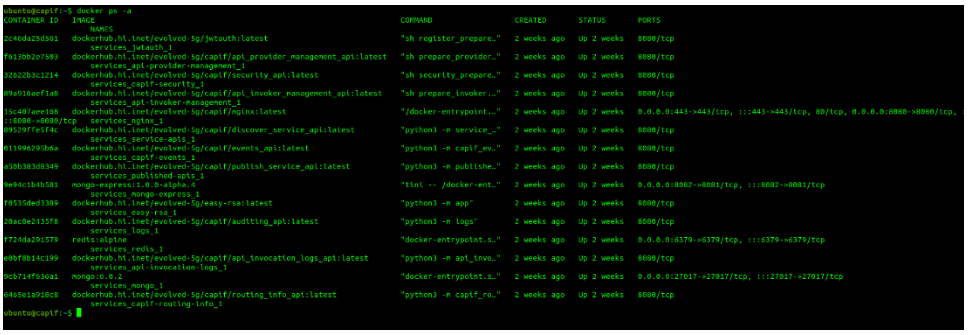
Instances

Displaying 4 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
Capif	ubuntu20.4	10.161.1.160	m1.small	andreas-desktop	Active	nova	None	Running	3 months, 1 week	Create Snapshot
Nef	ubuntu20.4	10.161.1.118	m1.small	andreas-desktop	Active	nova	None	Running	3 months, 1 week	Create Snapshot
ZorteNetApp	ubuntu20.4	10.161.1.127	m1.small	andreas-desktop	Active	nova	None	Running	3 months, 1 week	Create Snapshot

Figure 39 Deployment on NCSRD premises

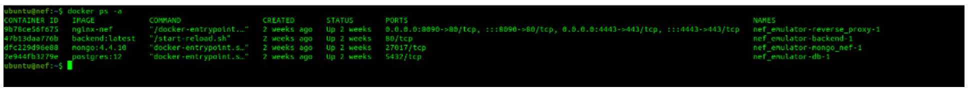
Deployment of CAPIF is shown in Figure 40 below.



```
ubuntu@capif:~$ docker ps -a
CONTAINER ID   IMAGE                                                          COMMAND                  CREATED       STATUS       PORTS
                 NAMES
2c46da25d561   dockerhub.hi.inet/evolved-5g/jwtauth:latest                    "sh register_prepare…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_jwtauth_1
f613bb2e7503   dockerhub.hi.inet/evolved-5g/capif/api_provider_management_api:latest   "sh prepare_provider…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_api-provider-management_1
32622b3c1214   dockerhub.hi.inet/evolved-5g/capif/security_api:latest         "sh security_prepare…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_capif-security_1
89a916aef1a8   dockerhub.hi.inet/evolved-5g/capif/api_invoker_management_api:latest    "sh prepare_invoker.…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_api-invoker-management_1
15c407aee168   dockerhub.hi.inet/evolved-5g/capif/nginx:latest                "/docker-entrypoint.…"   2 weeks ago   Up 2 weeks   0.0.0.0:443->443/tcp, :::443->443/tcp, 80/tcp, 0.0.0.0:8080->8080/tcp,
::8080->8080/tcp   services_nginx_1
89529ffe5f4c   dockerhub.hi.inet/evolved-5g/capif/discover_service_api:latest "python3 -m service_…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_service-apis_1
011996295b6a   dockerhub.hi.inet/evolved-5g/capif/events_api:latest           "python3 -m capif_ev…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_capif-events_1
a50b383d8349   dockerhub.hi.inet/evolved-5g/capif/publish_service_api:latest  "python3 -m publishe…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_published-apis_1
9e94c1b4b581   mongo-express:1.0.0-alpha.4                                    "tini -- /docker-ent…"   2 weeks ago   Up 2 weeks   0.0.0.0:8082->8081/tcp, :::8082->8081/tcp
                 services_mongo-express_1
f0535ded3389   dockerhub.hi.inet/evolved-5g/easy-rsa:latest                   "python3 -m app"         2 weeks ago   Up 2 weeks   8080/tcp
                 services_easy-rsa_1
20ac0e2435f8   dockerhub.hi.inet/evolved-5g/capif/auditing_api:latest         "python3 -m logs"        2 weeks ago   Up 2 weeks   8080/tcp
                 services_logs_1
f724da291579   redis:alpine                                                   "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
                 services_redis_1
e0bf8b14c199   dockerhub.hi.inet/evolved-5g/capif/api_invocation_logs_api:latest       "python3 -m api_invo…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_api-invocation-logs_1
9cb714f636a1   mongo:6.0.2                                                    "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   0.0.0.0:27017->27017/tcp, :::27017->27017/tcp
                 services_mongo_1
6465e1a918c8   dockerhub.hi.inet/evolved-5g/capif/routing_info_api:latest     "python3 -m capif_ro…"   2 weeks ago   Up 2 weeks   8080/tcp
                 services_capif-routing-info_1
ubuntu@capif:~$
```

Figure 40 Deployment of CAPIF

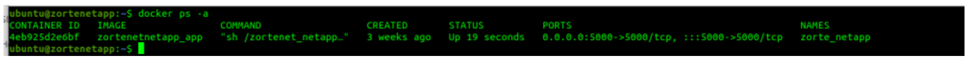
Deployment of NEF is shown in Figure 41 below.



```
ubuntu@nef:~$ docker ps -a
CONTAINER ID   IMAGE            COMMAND                  CREATED       STATUS       PORTS                                                                    NAMES
9b78ce56f675   nginx-nef        "/docker-entrypoint.…"   2 weeks ago   Up 2 weeks   0.0.0.0:8090->80/tcp, :::8090->80/tcp, 0.0.0.0:4443->443/tcp, :::4443->443/tcp   nef_emulator-reverse_proxy-1
47b13daa776b   backend:latest   "/start-reload.sh"       2 weeks ago   Up 2 weeks   80/tcp                                                                   nef_emulator-backend-1
dfc229d96e88   mongo:4.4.10     "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   27017/tcp                                                                nef_emulator-mongo_nef-1
2e944fb3279e   postgres:12      "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   5432/tcp                                                                 nef_emulator-db-1
ubuntu@nef:~$
```

Figure 41 Deployment of NEF

Deployment of Zortenet Network Application is shown in Figure 42 below.



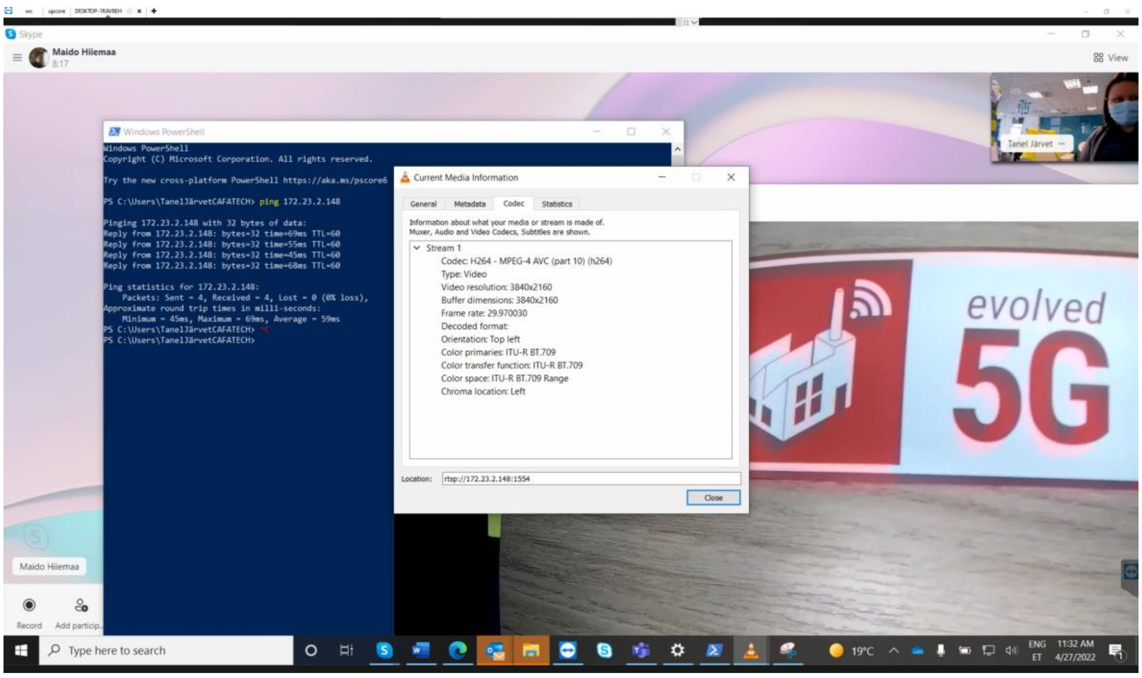
```
ubuntu@zortenetapp:~$ docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED       STATUS          PORTS                                        NAMES
4eb925d2e6bf   zortenetnetapp_app   "sh /zortenet_netapp…"   3 weeks ago   Up 19 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   zorte_netapp
ubuntu@zortenetapp:~$
```

Figure 42 Deployment of Zortenet Network Application

4.3 Results and Takeaways

4.3.1 CAFA Network Application NetMapper

CAFA Tech team set up a drone with the same equipment on board as the CAFA Robot (Quectel 5G NSA and SA modem + 4K camera). To simplify logistics, the CAFA robot was not transported to UMA this time. The SIM card was provided by UMA and added to the Quectel modem. A connection was established with UMA NSA (band n78) using a Quectel drone modem. The drone's on-board computer was accessible via the Internet (controlled using TeamViewer application). The drone is on a 5G NSA network and streams video that is visible on a CAFA field laptop connected to a UMA 5G network with an Askey USB-C modem. The VLC player was used to watch the video. The IP address of the video camera on the UMA network was: 172.23.2.148 and in VLC player rtsp://172.23.2.148:1554 as showed in Figure 43 below.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\TanelJärvetCAFATECH> ping 172.23.2.148

Pinging 172.23.2.148 with 32 bytes of data:
Reply from 172.23.2.148: bytes=32 time=69ms TTL=60
Reply from 172.23.2.148: bytes=32 time=55ms TTL=60
Reply from 172.23.2.148: bytes=32 time=45ms TTL=60
Reply from 172.23.2.148: bytes=32 time=68ms TTL=60

Ping statistics for 172.23.2.148:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 45ms, Maximum = 69ms, Average = 59ms
PS C:\Users\TanelJärvetCAFATECH> ^C
PS C:\Users\TanelJärvetCAFATECH>
```

Current Media Information

Information about what your media or stream is made of.
Muxer, Audio and Video Codecs, Subtitles are shown.

Stream 1
- Codec: H264 - MPEG-4 AVC (part 10) (h264)
- Type: Video
- Video resolution: 3840x2160
- Buffer dimensions: 3840x2160
- Frame rate: 29.970030
- Decoded format:
- Orientation: Top left
- Color primaries: ITU-R BT.709
- Color transfer function: ITU-R BT.709
- Color space: ITU-R BT.709 Range
- Chroma location: Left

Location: rtsp://172.23.2.148:1554

Figure 43 CAFA 4K camera video stream to field laptop over 5G NSA in April 2022

The CAFA drone's 4K camera streams video that is transmitted over the UMA 5G SA n78 band network and the video is displayed on the laptop screen in the VLC player. The first successful video stream will take place over the 5G SA network (both the drone and the field laptop were in the 5G SA network) as described above. Internet speed tested using Ookla Speed test application. 5G SA download speed was 138.91 Mbps and Upload speed was 11.64 Mbps.

On 28th April 2022 the UMA 5G Askey modem (connected to the drone on-board computer with an Ethernet cable) was added to the drone and the Askey modem via USB-C to the CAFA field laptop as shown in Figure 44. The CAFA drone's 4K camera streams video that is transmitted over the UMA 5G SA n78 band network and the video is displayed on the laptop screen in the VLC player.

Figure 44 CAFA and UMA team with 5G connected laptop and drone

Main conclusions of April 2022 tests:

- UMA platform also supports 5G mm wave frequencies n257 and n258. To test these frequencies, it is necessary to add mm wave antennas to the CAFA Robot.
- To get 5G SA connectivity, CAFA Robot needs to use either Quectel 5G Kit and / or Askey modem or find solutions to add AT commands to the UMA 5G network to provide access to Quectel modem.
- On-site tests provide valuable feedback on how the technical components (5G modems, network, robot camera and on-board computer, vApp on the local server, etc.) work together. These tests cannot be simulated because connectivity in practice often differs from that described in the specifications.

Next iteration tests were performed in March 2023. CAFA Tech has conducted Integration tests in cooperation with the University of Malaga. CAFA Tech team successfully established a connection between vApp and the UMA 5G network. Figure 45 below displays photos taken during the tests conducted at the UMA Platform. The aim was to assess the overall communication among CAPIF, NEF, Network Applications, and vApp, as well as the 5G connectivity with the cloud infrastructure as shown in Figure 45 below.

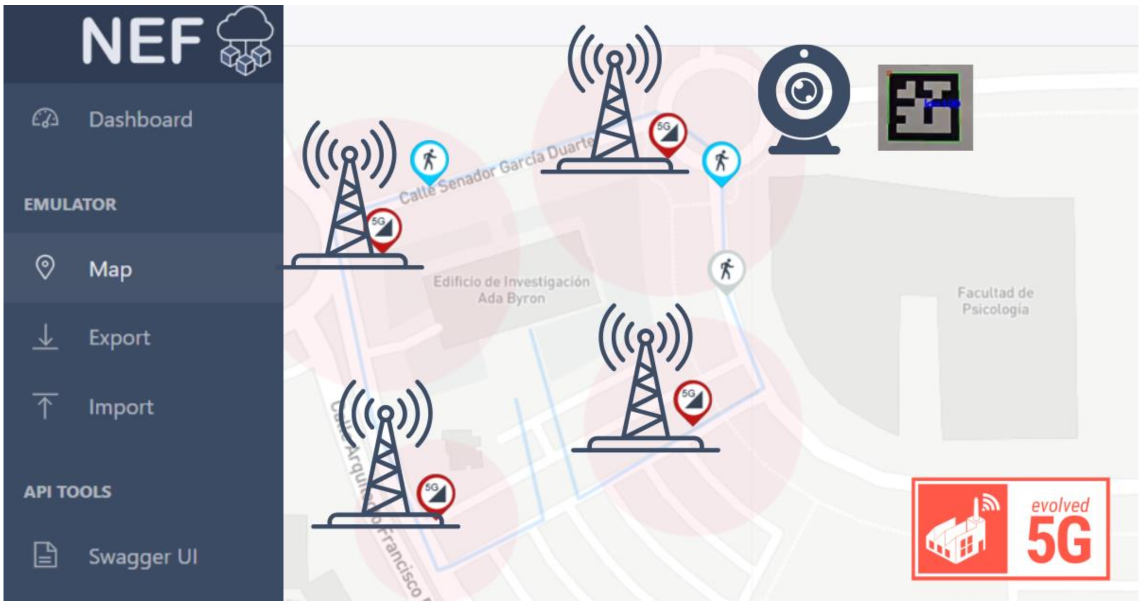

NEF
Dashboard
EMULATOR
Map
Export
Import
API TOOLS
Swagger UI
Calle Senador García Duarte
Edificio de Investigación Ada Byron
Facultad de Psicología
Calle Arquitecto Francisco
evolved 5G

Figure 45 Setup for the 1st integration round of CAFA Network App

During the tests conducted on March 6, 2023, the CAFA Tech team utilised the 5G SA (Stand-Alone) network in University of Malaga infrastructure. The camera planned for the CAFA robot connected to the 5G network, detecting markers while the CAFA Network App simultaneously checked the presence and quality of the 5G coverage area. Areas with pink circles in the Figure 45 showed virtually guaranteed 5G coverage, while the CAFA Network App immediately issued a warning when leaving the mentioned area or when the 5G network was overloaded, indicating "Quality of Service NOT_GUARANTEED."

These preliminary feasibility tests paved the way for further developments and improvements both in the vApp and the Network App side and ensured the readiness of the components for the future planned integration activities during the lifetime of the project.

4.3.2 Industrial Grade 5G Connectivity Network Application

Integration of ININ Network App was conducted in February and March 2023 in cooperation with NCSR Demokritos team. Deployment and Testing of the whole system was done twice: remotely from Ljubljana to Athens and physically with the ININ team presence at NCSR Demokritos 5G testbed.

Network App v3.0 including NEF v1.6.2, SDK v0.8.9 and CAPIF v2.1 was used for the 1st round of the integration testing. To that end, Network App, NEF and CAPIF were deployed in separate VMs within the cloud infrastructure.

A more detailed description with screenshots of the tests performed is given in subsection 4.2.2.

4.3.3 Smart Irrigation 5G Agriculture Network Application

The first round of integration activities took place on the UMA premises in 2022.
Figure 46 shows the graphical representation of the data obtained from the NetApp in real time.

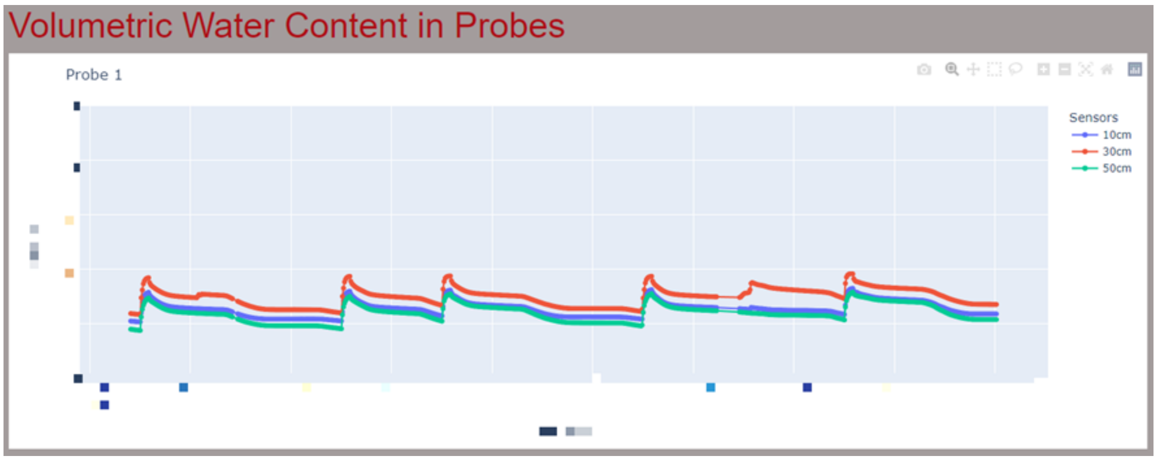


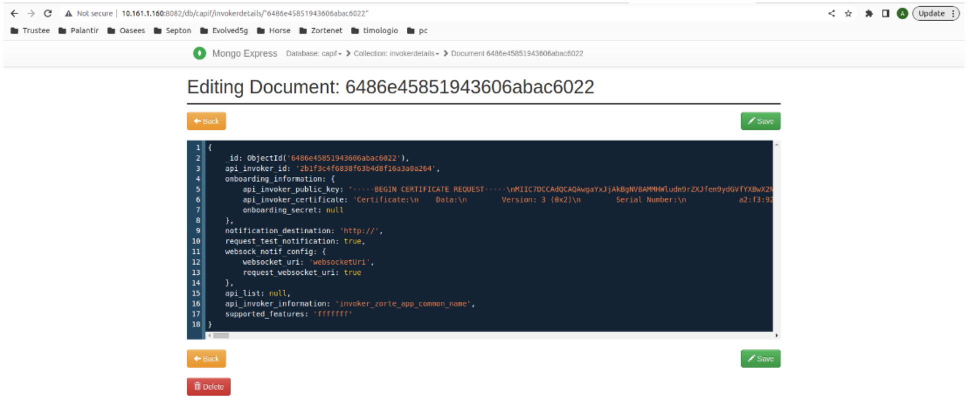
Figure 46 vApp plotting the data obtained from the NetApp graphically

Once the Network App has been manually deployed and back to the use case. The datalogger sends the data it receives from the sensors deployed in the field, over the 5G network to the Network App. The vApp consumes the data received by the NetApp and displays it in graphs. This whole process works well. In addition, the resulting graphs have been tested to show the data correctly.

4.3.4 Anomaly detection Network Application

The first round of integration activities took place on the NCSRD premises. The connection between the Vertical Application which was located at Zortenet's premises and Network application at NCSRD premises was achieved via a VPN connection.

Successful registration to CAPIF is shown in Figure 47 and Figure 48 below.

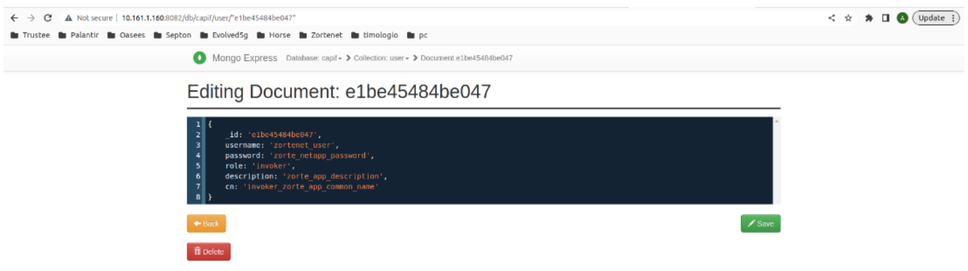


Not secure | 10.161.1.160:8082/db/capif/invokerdetails/"6486e45851943606abac6022"
Update
Trustee
Palantir
Oasees
Septon
Evolved5g
Horse
Zortenet
timologio
pc
Mongo Express
Database: capif
Collection: invokerdetails
Document 6486e45851943606abac6022
Back
Save
Back
Save
Delete

Editing Document: 6486e45851943606abac6022

```
{
    _id: ObjectId('6486e45851943606abac6022'),
    api_invoker_id: '2b1f3c4f6838f63b4d8f16a3a0a264',
    onboarding_information: {
        api_invoker_public_key: '-----BEGIN CERTIFICATE REQUEST-----\nMIIC7DCCAdQCAQAwgaYxJjAkBgNVBAMMHWludm9rZXJfem9ydGVfYXBwX2N
        api_invoker_certificate: 'Certificate:\n    Data:\n        Version: 3 (0x2)\n        Serial Number:\n            a2:f3:92
        onboarding_secret: null
    },
    notification_destination: 'http://',
    request_test_notification: true,
    websock_notif_config: {
        websocket_uri: 'websocketUri',
        request_websocket_uri: true
    },
    api_list: null,
    api_invoker_information: 'invoker_zorte_app_common_name',
    supported_features: 'fffffff'
}
```

Figure 47 Registration to CAPIF – Invoker details

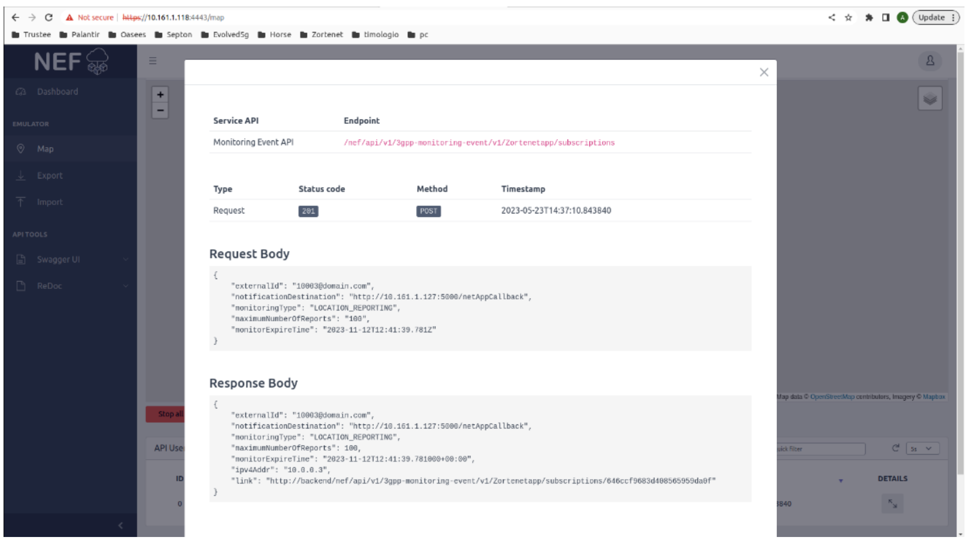

Not secure | 10.161.1.160:8082/db/capif/user/"e1be45484be047"
Trustee Palantir Oasees Septon Evolved5g Horse Zortenet timologio pc
Update
Mongo Express Database: capif > Collection: user > Document e1be45484be047

Editing Document: e1be45484be047

```
{
    _id: 'e1be45484be047',
    username: 'zortenet_user',
    password: 'zorte_netapp_password',
    role: 'invoker',
    description: 'zorte_app_description',
    cn: 'invoker_zorte_app_common_name'
}
```

Figure 48 Registration to CAPIF - Users

NEF subscription creation is shown in Figure 49 below.


Not secure | https://10.161.1.118:4443/map
Trustee Palantir Oasees Septon Evolved5g Horse Zortenet timologio pc
Update
NEF
Dashboard
EMULATOR
Map
Export
Import
API TOOLS
Swagger UI
ReDoc
Stop all
API Use
ID
DETAILS

Service API	Endpoint
Monitoring Event API	/nef/api/v1/3gpp-monitoring-event/v1/Zortenetapp/subscriptions

Type	Status code	Method	Timestamp
Request	201	POST	2023-05-23T14:37:10.843840

Request Body

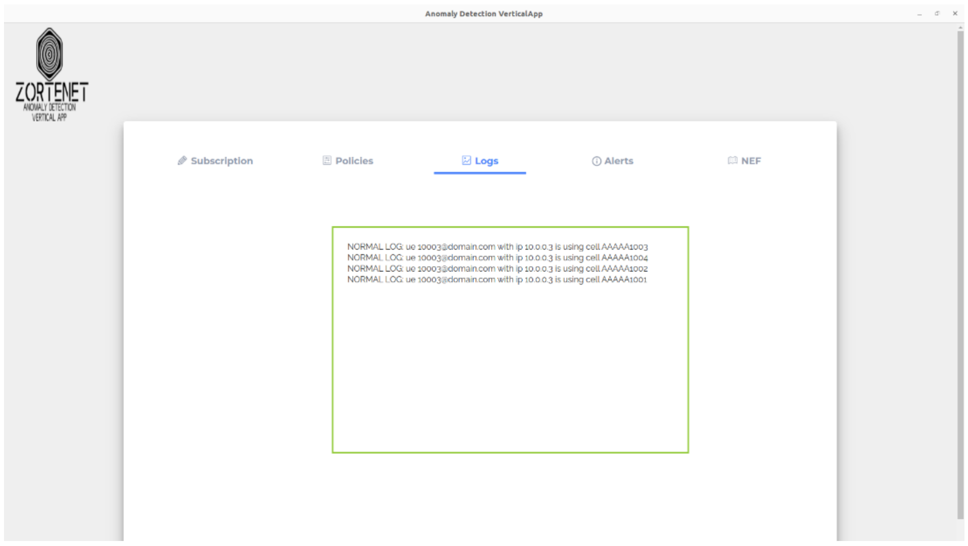
```
{
    "externalId": "10003@domain.com",
    "notificationDestination": "http://10.161.1.127:5000/netAppCallback",
    "monitoringType": "LOCATION_REPORTING",
    "maximumNumberOfReports": "100",
    "monitorExpireTime": "2023-11-12T12:41:39.781Z"
}
```

Response Body

```
{
    "externalId": "10003@domain.com",
    "notificationDestination": "http://10.161.1.127:5000/netAppCallback",
    "monitoringType": "LOCATION_REPORTING",
    "maximumNumberOfReports": 100,
    "monitorExpireTime": "2023-11-12T12:41:39.781000+00:00",
    "ipv4Addr": "10.0.0.3",
    "link": "http://backend/nef/api/v1/3gpp-monitoring-event/v1/Zortenetapp/subscriptions/646ccf9683d408565959da0f"
}
```

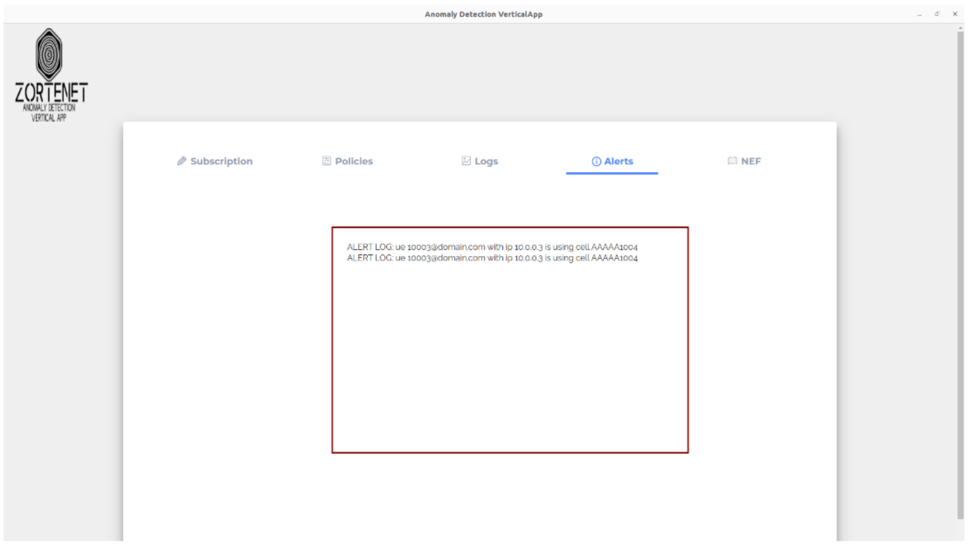
Figure 49 NEF Subscription

Receiving NEF Callbacks from the first version of Vertical Application is shown in Figure 50 and Figure 51 below.



```
NORMAL LOG: ue 10003@domain.com with ip 10.0.0.3 is using cell AAAAA1003
NORMAL LOG: ue 10003@domain.com with ip 10.0.0.3 is using cell AAAAA1004
NORMAL LOG: ue 10003@domain.com with ip 10.0.0.3 is using cell AAAAA1002
NORMAL LOG: ue 10003@domain.com with ip 10.0.0.3 is using cell AAAAA1001
```

Figure 50 Receiving Logs from NEF



```
ALERT LOG: ue 10003@domain.com with ip 10.0.0.3 is using cell AAAAA1004
ALERT LOG: ue 10003@domain.com with ip 10.0.0.3 is using cell AAAAA1004
```

Figure 51 Receiving Alerts

4.4 PURPOSE OF THE INTEGRATIONS TESTS (2ND ROUND)

The purpose of the second integration round was to validate the use-cases with the final components of EVOLVED--5G. On the one hand, NEF, CAPIF and the SDK had been enriched with additional features. On the other hand, SMEs had also finalized their Network Apps by expanding the 3.0 version and using the last versions of NEF, CAPIF and SDK. The final prototype (v4.1) of the Network Apps also completed and used the validation pipeline before the integration test. Finally, the Networks Apps were deployed in Kubernetes clusters in either Athens or UMA's premises instead of using Docker containers running locally and the Openstack environment as in the first round of integration testing.

It's worth noting that until the end of WP3, it was deemed necessary for the SDK to undergo some minor improvements, primarily aimed at enhancing functionality and addressing specific bugs. During this second integration round, the final version of components was utilized:

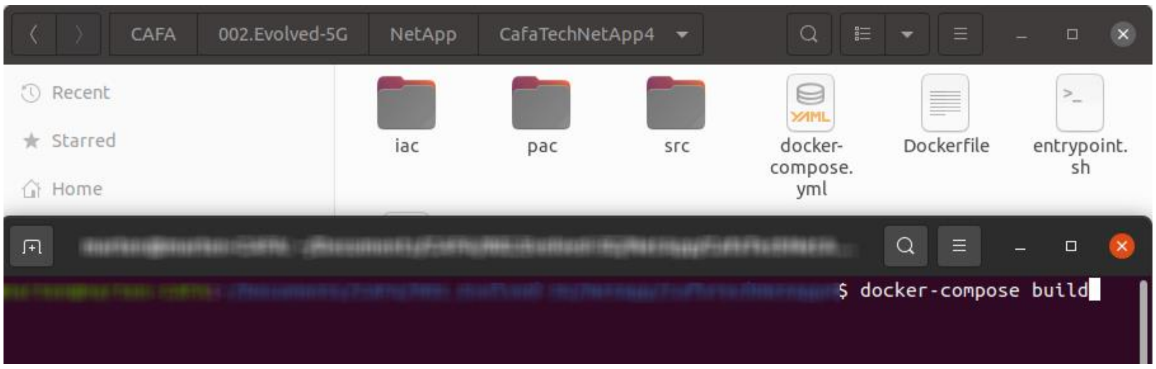
- Network Applications v4.1
- NEF v2.2.2
- CAPIF v3.1.2
- SDK v1.0.8

4.5 Topology and Setup

4.5.1 CAFA Network App NetMapper

Deployment and test sequence of CAFA NetApp in the UMA Kubernetes cluster:

1) Create image of NetApp (uses SDK evolved5g==1.0.8) in local Docker using docker-compose build as shown in Figure 52 below.



```
$ docker-compose build
```

Figure 52 Starting the Network App image build in local computer

2) Tag and push the created image to Dockerhub with following commands:

```
docker tag cafatech-netapp martenrannu/cafatech-netapp-4:v2
docker login
docker push martenrannu/cafatech-netapp-4:v2
```

As a result, the NetApp image is visible in the Dockerhub as shown in Figure 53 below.

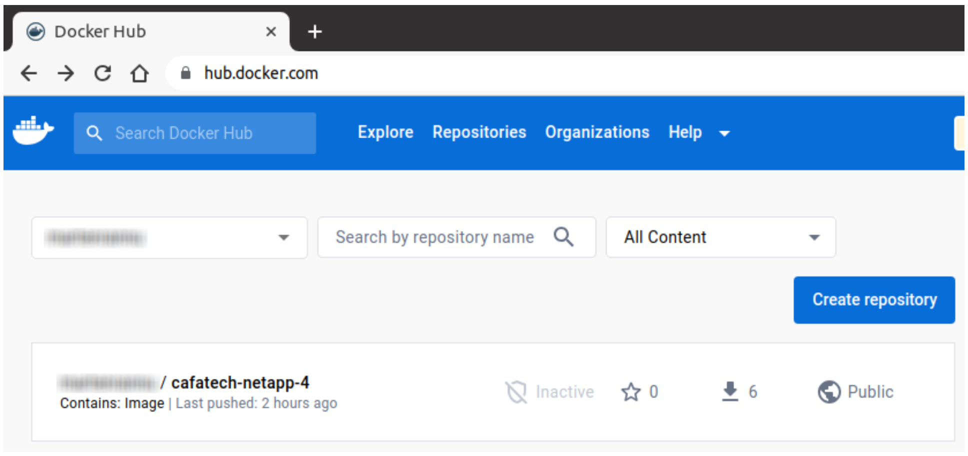


Figure 53 CAFA NetMapper Network App image pushed to Dockerhub

3) Connecting to UMA server, using VPN profile files given by UMA.

4) Setting up the cluster in the Malaga Kubernetes with the following commands:

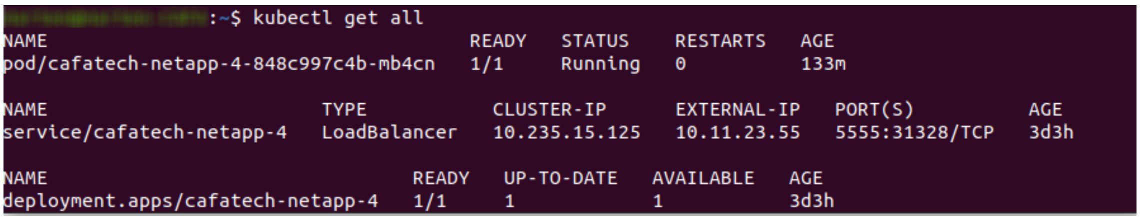
--kubectl config set-cluster CAFA --server=https://10.173.0.117:8383 --insecure-skip-tls-verify=true
--kubectl config set-credentials cafa-developer --token=<TOKEN>
--kubectl config set-context cafa_context --cluster=CAFA --user=cafa-developer --namespace cafatech
--kubectl config use-context cafa_context

5) Deploying the NetApp to the Malaga Kubernetes and configuring its service, using following commands:

----kubectl apply -f "deployment.yaml"

----kubectl apply -f "service.yaml"

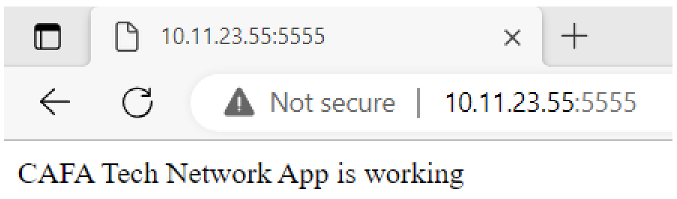
The environment variables were included in deployment.yaml (no separate environment.yaml). The IP address of Network App was published, using LoadBalancer in the service.yaml.
The deployed Network App and its service can be verified with this command as it can be seen from the terminal's output in Figure 54 below:
---kubectl get all



```
[redacted]:~$ kubectl get all
NAME                                  READY   STATUS    RESTARTS   AGE
pod/cafatech-netapp-4-848c997c4b-mb4cn   1/1     Running   0          133m

NAME                        TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)          AGE
service/cafatech-netapp-4   LoadBalancer   10.235.15.125   10.11.23.55    5555:31328/TCP   3d3h

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/cafatech-netapp-4   1/1     1            1           3d3h
```

Figure 54 CAFA NetMapper Network App deployed in UMA Kubernetes cluster

The CAFA Network App's test page is accessible (when VPN is connected) at its external IP as shown in Figure 55 below:

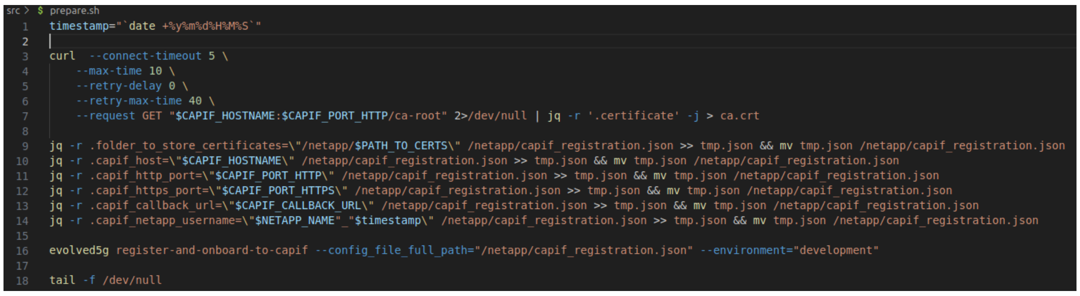


CAFA Tech Network App is working

Figure 55 The CAFA Network App's test page

6) The Network App onboarding to CAPIF (deployed in UMA Kubernetes cluster):

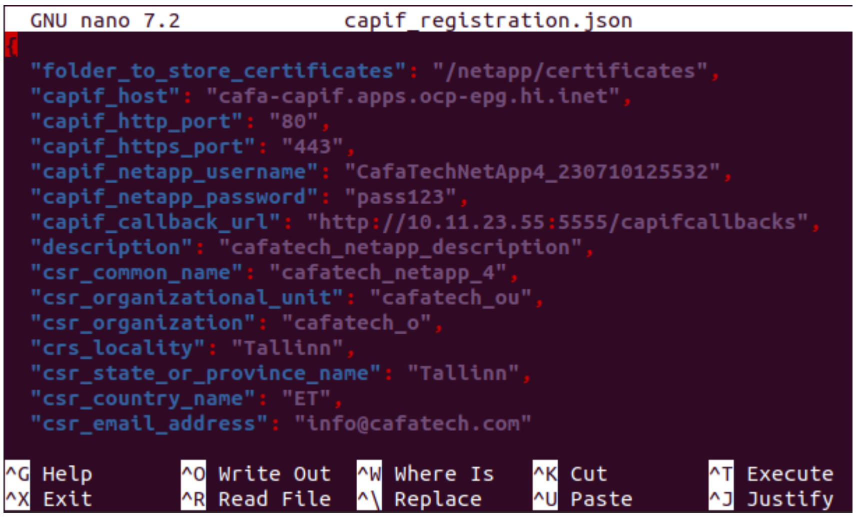
The Network App prepare.sh script used the env variables (originated from the deployment.yaml file) and current timestamp and updated the capif_registration.json on Network App's startup as shown in Figure 56 below.



```
src > $ prepare.sh
timestamp="`date +%y%m%d%H%M%S`"

curl  --connect-timeout 5 \
    --max-time 10 \
    --retry-delay 0 \
    --retry-max-time 40 \
    --request GET "$CAPIF_HOSTNAME:$CAPIF_PORT_HTTP/ca-root" 2>/dev/null | jq -r '.certificate' -j > ca.crt

jq -r .folder_to_store_certificates=\"/netapp/$PATH_TO_CERTS\" /netapp/capif_registration.json >> tmp.json && mv tmp.json /netapp/capif_registration.json
jq -r .capif_host=\"$CAPIF_HOSTNAME\" /netapp/capif_registration.json >> tmp.json && mv tmp.json /netapp/capif_registration.json
jq -r .capif_http_port=\"$CAPIF_PORT_HTTP\" /netapp/capif_registration.json >> tmp.json && mv tmp.json /netapp/capif_registration.json
jq -r .capif_https_port=\"$CAPIF_PORT_HTTPS\" /netapp/capif_registration.json >> tmp.json && mv tmp.json /netapp/capif_registration.json
jq -r .capif_callback_url=\"$CAPIF_CALLBACK_URL\" /netapp/capif_registration.json >> tmp.json && mv tmp.json /netapp/capif_registration.json
jq -r .capif_netapp_username=\"$NETAPP_NAME"_"$timestamp\" /netapp/capif_registration.json >> tmp.json && mv tmp.json /netapp/capif_registration.json

evolved5g register-and-onboard-to-capif --config_file_full_path="/netapp/capif_registration.json" --environment="development"

tail -f /dev/null
```

Figure 56 CAFA NetMapper Network App's startup

As shown in Figure 57 below, the current timestamp was added to the end of the capif_netapp_username so it is different at every subsequent deployment and therefore doesn't create a conflict in the CAPIF when there could be an entry with the same username.

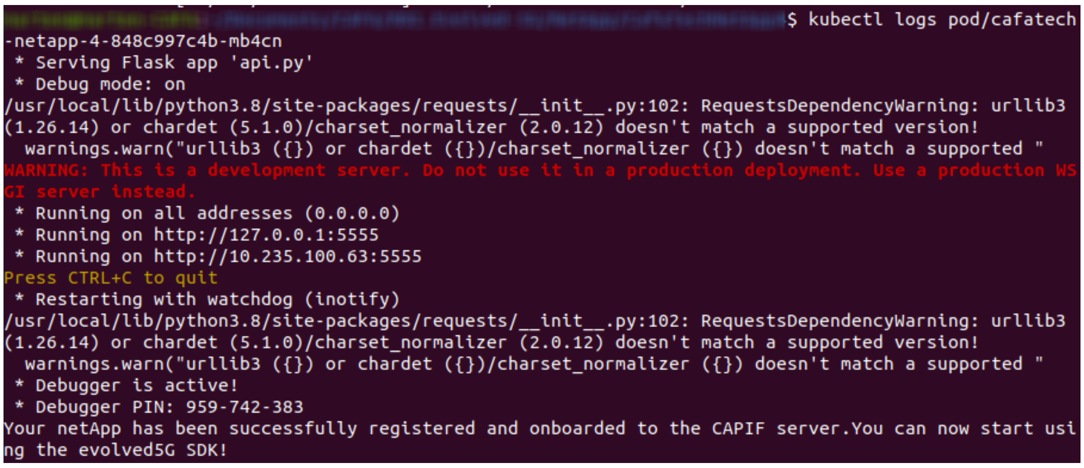


```
  GNU nano 7.2                 capif_registration.json
{
  "folder_to_store_certificates": "/netapp/certificates",
  "capif_host": "cafa-capif.apps.ocp-epg.hi.inet",
  "capif_http_port": "80",
  "capif_https_port": "443",
  "capif_netapp_username": "CafaTechNetApp4_230710125532",
  "capif_netapp_password": "pass123",
  "capif_callback_url": "http://10.11.23.55:5555/capifcallbacks",
  "description": "cafatech_netapp_description",
  "csr_common_name": "cafatech_netapp_4",
  "csr_organizational_unit": "cafatech_ou",
  "csr_organization": "cafatech_o",
  "crs_locality": "Tallinn",
  "csr_state_or_province_name": "Tallinn",
  "csr_country_name": "ET",
  "csr_email_address": "info@cafatech.com"

^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify
```

Figure 57 capif_registration.json file with values updated from .env variables

It was then used by evolved5g register-and-onboard-to-capif function.

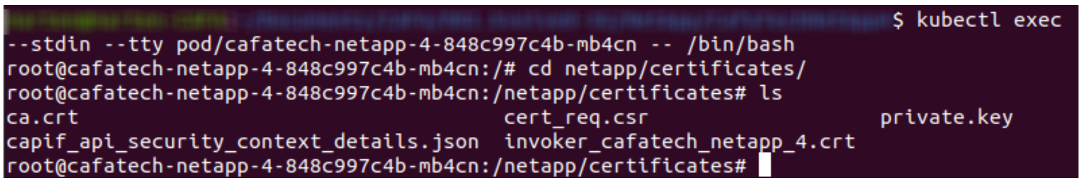
The CAFA NetMapper Network App has onboarded to CAPIF as shown in the last row of Figure 58 below.



```
[redacted]$ kubectl logs pod/cafatech
-netapp-4-848c997c4b-mb4cn
 * Serving Flask app 'api.py'
 * Debug mode: on
/usr/local/lib/python3.8/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3
(1.26.14) or chardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
WARNING: This is a development server. Do not use it in a production deployment. Use a production WS
GI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5555
 * Running on http://10.235.100.63:5555
Press CTRL+C to quit
 * Restarting with watchdog (inotify)
/usr/local/lib/python3.8/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3
(1.26.14) or chardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
 * Debugger is active!
 * Debugger PIN: 959-742-383
Your netApp has been successfully registered and onboarded to the CAPIF server.You can now start usi
ng the evolved5G SDK!
```

Figure 58 CAFA NetMapper Network App has onboarded to CAPIF at its startup

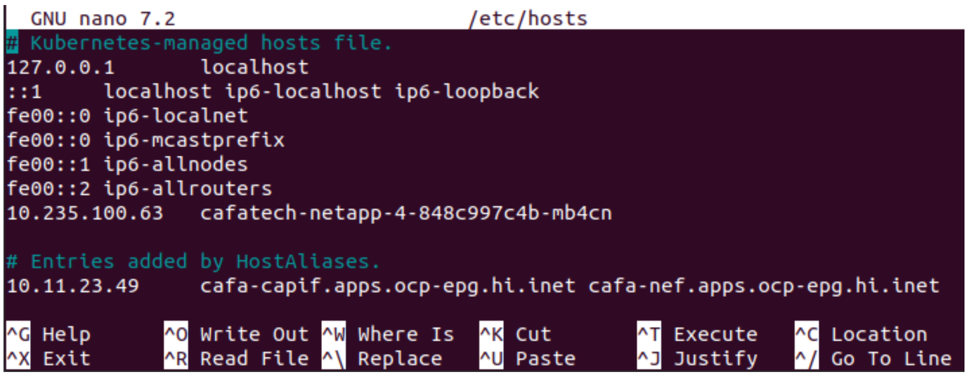
The onboarding has also created necessary certificates in the Network App's predefined directory (shell into the Network App's pod) as shown in Figure 59 below.



```
[redacted] $ kubectl exec
--stdin --tty pod/cafatech-netapp-4-848c997c4b-mb4cn -- /bin/bash
root@cafatech-netapp-4-848c997c4b-mb4cn:/# cd netapp/certificates/
root@cafatech-netapp-4-848c997c4b-mb4cn:/netapp/certificates# ls
ca.crt                               cert_req.csr                private.key
capif_api_security_context_details.json  invoker_cafatech_netapp_4.crt
root@cafatech-netapp-4-848c997c4b-mb4cn:/netapp/certificates#
```

Figure 59 CAPIF registration files in the CAFA Network App certificates directory

The Network App's start also updated its */etc/*hosts file (see the last row starting 10.11.23.49) as shown in Figure 60 below.



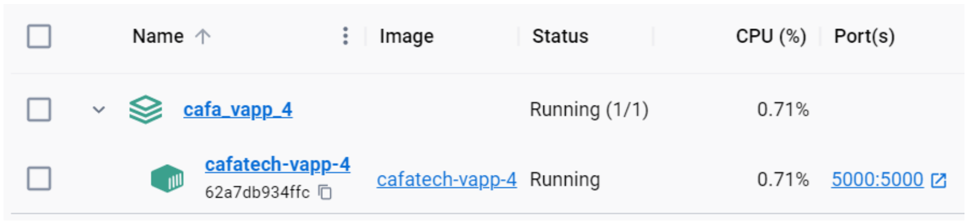
```
  GNU nano 7.2                          /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
10.235.100.63   cafatech-netapp-4-848c997c4b-mb4cn

# Entries added by HostAliases.
10.11.23.49     cafa-capif.apps.ocp-epg.hi.inet cafa-nef.apps.ocp-epg.hi.inet

^G Help       ^O Write Out ^W Where Is ^K Cut        ^T Execute   ^C Location
^X Exit       ^R Read File ^\ Replace  ^U Paste      ^J Justify   ^/ Go To Line
```

Figure 60 CAPIF and NEF addresses added to /etc/hosts file

7) Connecting CAFA NetMapper Network App from vApp:

CAFA Safety VideoLyzer vApp takes IP camera's RTSP address as an input and uses Computer Vision (CV) on its video feed to detect missing safety helmets of the workers in the factory. When the helmet with correct color (e.g. blue safety helmet for specific factory) is missing, it outputs a warning.

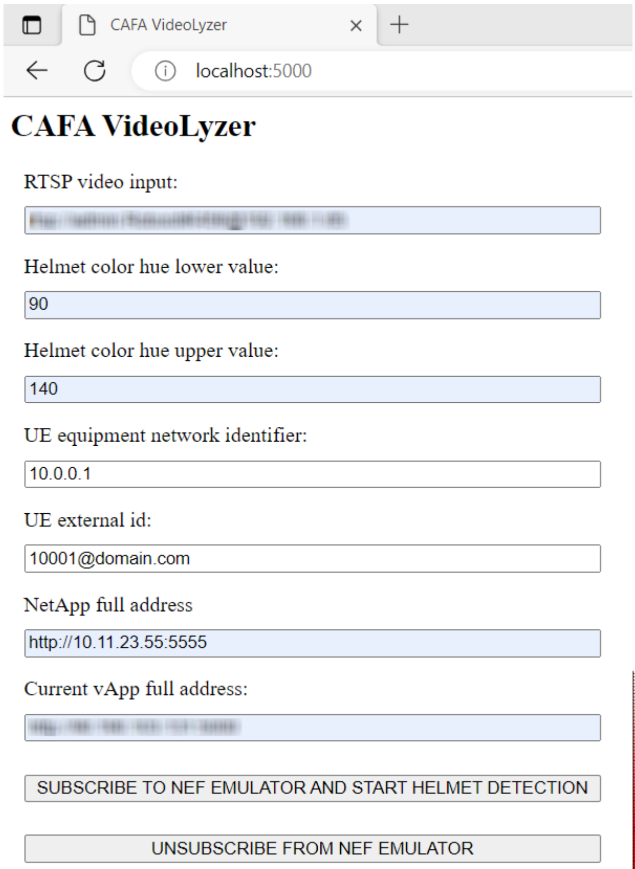
Run the vApp container in the local computer's Docker is shown Figure 61 below.



Name	Image	Status	CPU (%)	Port(s)
cafa_vapp_4		Running (1/1)	0.71%	
cafatech-vapp-4 62a7db934ffc	cafatech-vapp-4	Running	0.71%	5000:5000

Figure 61 CAFA vApp Docker container running in the external laptop

CAFA Safety VideoLyzer vApp's web UI with the hue chart to choose the safety helmet color to be detected is shown in Figure 62 below.



CAFA VideoLyzer

localhost:5000

CAFA VideoLyzer

RTSP video input:

[redacted]

Helmet color hue lower value:

90

Helmet color hue upper value:

140

UE equipment network identifier:

10.0.0.1

UE external id:

10001@domain.com

NetApp full address

http://10.11.23.55:5555

Current vApp full address:

[redacted]

SUBSCRIBE TO NEF EMULATOR AND START HELMET DETECTION

UNSUBSCRIBE FROM NEF EMULATOR

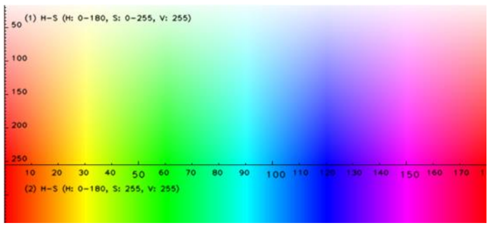
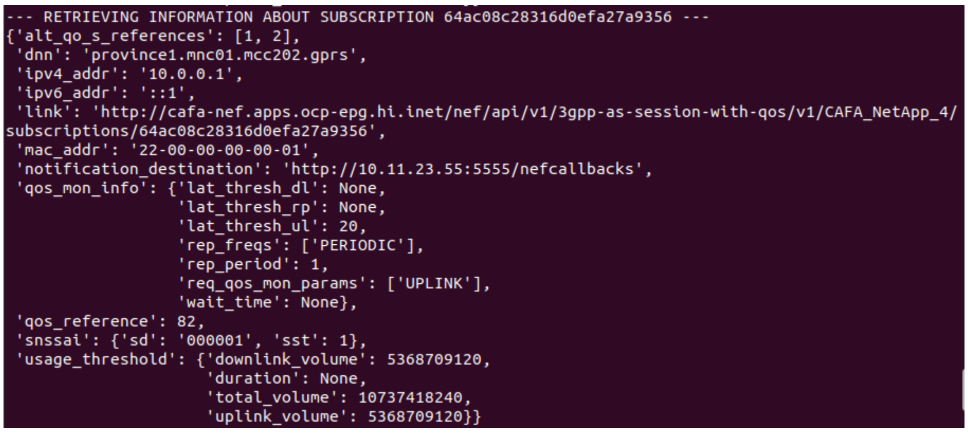


Figure 62 CAFA Safety VideoLyzer vApp web user interface with input fields and safety helmet hue chart

The button press sends a subscription request to Network App's address with UE info and the address of the current vApp to send the messages back to. When the Network App receives the request, it creates a QoS, Loss of Connectivity and UE Reachability subscriptions to NEF (deployed in the UMA Kubernetes cluster). The logs from the Network App pod can be obtained with command:

`kubectl logs pod/cafatech-netapp-4-848c997c4b-mb4cn`

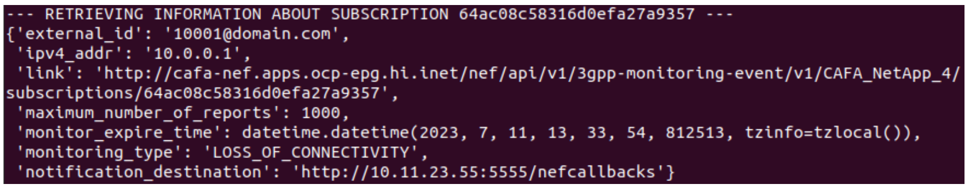
QoS subscription created as depicted in Figure 63 below.



```
--- RETRIEVING INFORMATION ABOUT SUBSCRIPTION 64ac08c28316d0efa27a9356 ---
{'alt_qo_s_references': [1, 2],
 'dnn': 'province1.mnc01.mcc202.gprs',
 'ipv4_addr': '10.0.0.1',
 'ipv6_addr': '::1',
 'link': 'http://cafa-nef.apps.ocp-epg.hi.inet/nef/api/v1/3gpp-as-session-with-qos/v1/CAFA_NetApp_4/
subscriptions/64ac08c28316d0efa27a9356',
 'mac_addr': '22-00-00-00-00-01',
 'notification_destination': 'http://10.11.23.55:5555/nefcallbacks',
 'qos_mon_info': {'lat_thresh_dl': None,
                  'lat_thresh_rp': None,
                  'lat_thresh_ul': 20,
                  'rep_freqs': ['PERIODIC'],
                  'rep_period': 1,
                  'req_qos_mon_params': ['UPLINK'],
                  'wait_time': None},
 'qos_reference': 82,
 'snssai': {'sd': '000001', 'sst': 1},
 'usage_threshold': {'downlink_volume': 5368709120,
                     'duration': None,
                     'total_volume': 10737418240,
                     'uplink_volume': 5368709120}}
```

Figure 63 Quality of Service (QoS) subscription created to NEF emulator

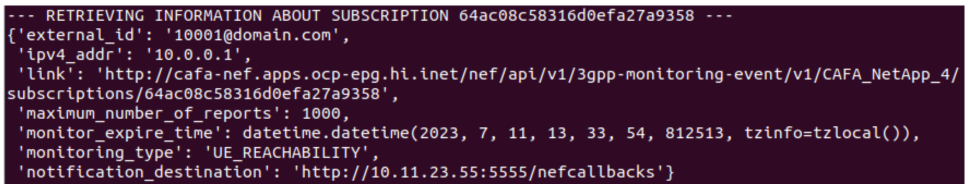
Loss of Connectivity subscription created as shown in Figure 64 below.



```
--- RETRIEVING INFORMATION ABOUT SUBSCRIPTION 64ac08c58316d0efa27a9357 ---
{'external_id': '10001@domain.com',
 'ipv4_addr': '10.0.0.1',
 'link': 'http://cafa-nef.apps.ocp-epg.hi.inet/nef/api/v1/3gpp-monitoring-event/v1/CAFA_NetApp_4/
subscriptions/64ac08c58316d0efa27a9357',
 'maximum_number_of_reports': 1000,
 'monitor_expire_time': datetime.datetime(2023, 7, 11, 13, 33, 54, 812513, tzinfo=tzlocal()),
 'monitoring_type': 'LOSS_OF_CONNECTIVITY',
 'notification_destination': 'http://10.11.23.55:5555/nefcallbacks'}
```

Figure 64 Loss of Connectivity subscription created to NEF emulator

UE Reachability subscription created as shown in Figure 65 below.



```
--- RETRIEVING INFORMATION ABOUT SUBSCRIPTION 64ac08c58316d0efa27a9358 ---
{'external_id': '10001@domain.com',
 'ipv4_addr': '10.0.0.1',
 'link': 'http://cafa-nef.apps.ocp-epg.hi.inet/nef/api/v1/3gpp-monitoring-event/v1/CAFA_NetApp_4/
subscriptions/64ac08c58316d0efa27a9358',
 'maximum_number_of_reports': 1000,
 'monitor_expire_time': datetime.datetime(2023, 7, 11, 13, 33, 54, 812513, tzinfo=tzlocal()),
 'monitoring_type': 'UE_REACHABILITY',
 'notification_destination': 'http://10.11.23.55:5555/nefcallbacks'}
```

Figure 65 UE Reachability subscription created to NEF emulator

NEF Emulator can be accessed as shown in Figure 66 below.

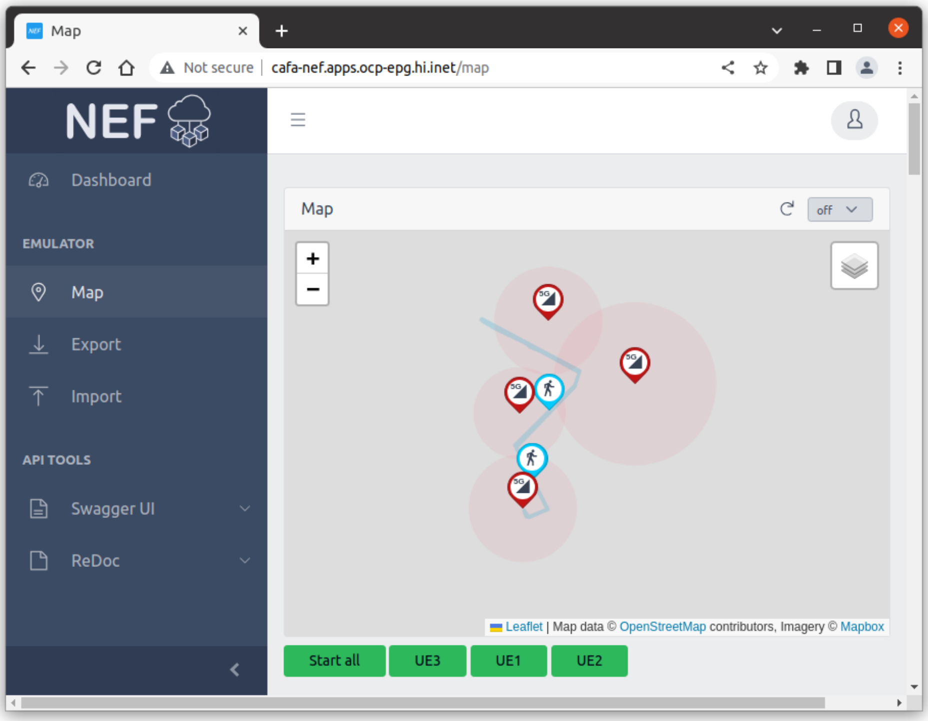

Map
Not secure
cafa-nef.apps.ocp-epg.hi.inet/map
NEF
Dashboard
EMULATOR
Map
Export
Import
API TOOLS
Swagger UI
ReDoc
Map
off
Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox
Start all
UE3
UE1
UE2

Figure 66 NEF Emulator accessed

8) Working principle of vApp with Network App:

NEF Emulator sends QoS and Conn messages to Network App's /nefcallbacks endpoint, which extracts QoS and Conn state info from it and sends it to vApp's corresponding endpoints which modifies its helmet detection behavior accordingly.

4.5.2 Industrial Grade 5G Connectivity Network Application

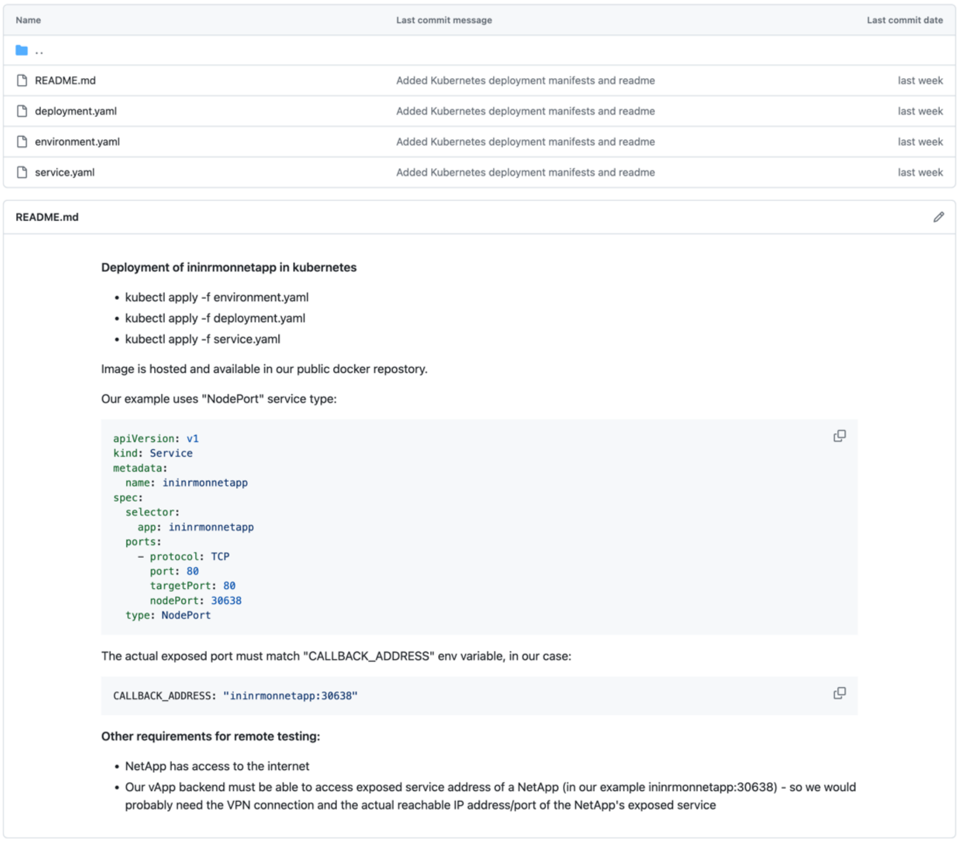
Integration topology consists of multiple components interconnected as listed in the Table 2 below.

Table 2 ININ IoT Network App topology

Building block	**Owner/developer**	**Location of SF or HW**
IoT Gateway	ININ	ININ testbed
IoT Management	ININ	ININ private cloud
IoT Collector	ININ	ININ private cloud
Network App	ININ	NCSR Demokritos Kubernetes Cluster
5G SA Network	ININ	ININ testbed
NEF	NCSR Demokritos	NCSR Demokritos Kubernetes Cluster
CAPIF	NCSR Demokritos	NCSR Demokritos Kubernetes Cluster

For successful Network App deployment inside NCSR Demokritos Cluster three Kubernetes manifests were prepared:

- deployment.yaml
- environment.yaml
- service.yaml



Name	Last commit message	Last commit date
..		
README.md	Added Kubernetes deployment manifests and readme	last week
deployment.yaml	Added Kubernetes deployment manifests and readme	last week
environment.yaml	Added Kubernetes deployment manifests and readme	last week
service.yaml	Added Kubernetes deployment manifests and readme	last week

README.md

Deployment of ininrmonnetapp in kubernetes

- kubectl apply -f environment.yaml
- kubectl apply -f deployment.yaml
- kubectl apply -f service.yaml

Image is hosted and available in our public docker repostory.

Our example uses "NodePort" service type:

```
apiVersion: v1
kind: Service
metadata:
  name: ininrmonnetapp
spec:
  selector:
    app: ininrmonnetapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30638
  type: NodePort
```

The actual exposed port must match "CALLBACK_ADDRESS" env variable, in our case:

```
CALLBACK_ADDRESS: "ininrmonnetapp:30638"
```

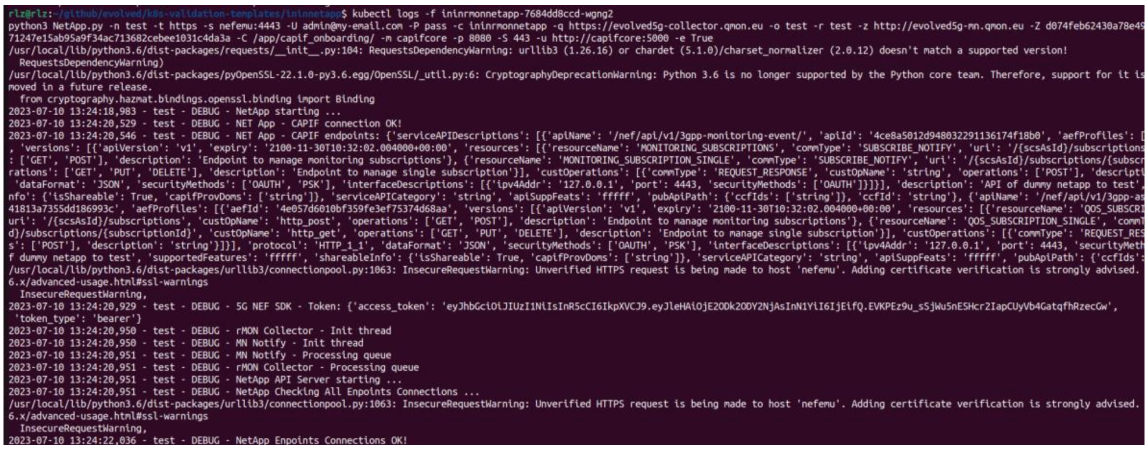
Other requirements for remote testing:

- NetApp has access to the internet
- Our vApp backend must be able to access exposed service address of a NetApp (in our example ininrmonnetapp:30638) - so we would probably need the VPN connection and the actual reachable IP address/port of the NetApp's exposed service

Figure 67 Kubernetes manifests files on Github ININ's repo

The complete process of integration testing can be described and confirmed in sequence steps:

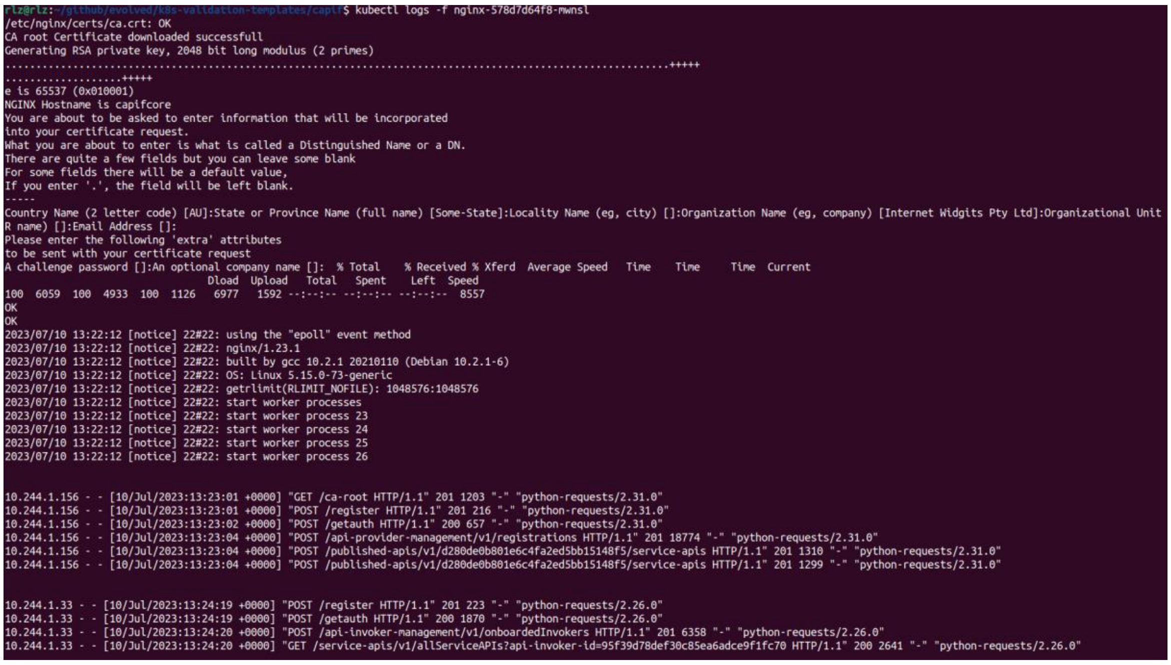
1. ININ's Network App successful deployment was performed on NCSRD's Kubernetes as shown in Figure 68 below.



```
rlz@rlz:~/github/evolved/k8s-validation-templates/ininnetapp$ kubectl logs -f ininrmonnetapp-7684dd8ccd-wgng2
python3 NetApp.py -n test -t https -s nefemu:4443 -U admin@my-email.com -P pass -c ininrmonnetapp -q https://evolved5g-collector.qmon.eu -o test -r test -z http://evolved5g-mn.qmon.eu -Z d074feb62430a78e49
71247e15ab95a9f34ac713682cebee1031c4da3a -C /app/capif_onboarding/ -m capifcore -p 8080 -S 443 -u http://capifcore:5000 -e True
/usr/local/lib/python3.6/dist-packages/requests/__init__.py:104: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  RequestsDependencyWarning)
/usr/local/lib/python3.6/dist-packages/pyOpenSSL-22.1.0-py3.6.egg/OpenSSL/_util.py:6: CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support for it is
moved in a future release.
  from cryptography.hazmat.bindings.openssl.binding import Binding
2023-07-10 13:24:18,983 - test - DEBUG - NetApp starting ...
2023-07-10 13:24:20,529 - test - DEBUG - NET App - CAPIF connection OK!
2023-07-10 13:24:20,546 - test - DEBUG - NET App - CAPIF endpoints: {'serviceAPIDescriptions': [{'apiName': '/nef/api/v1/3gpp-monitoring-event/', 'apiId': '4ce8a5012d948032291136174f18b0', 'aefProfiles': [
, 'versions': [{'apiVersion': 'v1', 'expiry': '2100-11-30T10:32:02.004000+00:00', 'resources': [{'resourceName': 'MONITORING_SUBSCRIPTIONS', 'commType': 'SUBSCRIBE_NOTIFY', 'uri': '/{scsAsId}/subscriptions
: ['GET', 'POST'], 'description': 'Endpoint to manage monitoring subscriptions'}, {'resourceName': 'MONITORING_SUBSCRIPTION_SINGLE', 'commType': 'SUBSCRIBE_NOTIFY', 'uri': '/{scsAsId}/subscriptions/{subscr
rations': ['GET', 'PUT', 'DELETE'], 'description': 'Endpoint to manage single subscription'}], 'custOperations': [{'commType': 'REQUEST_RESPONSE', 'custOpName': 'string', 'operations': ['POST'], 'descripti
 'dataFormat': 'JSON', 'securityMethods': ['OAUTH', 'PSK'], 'interfaceDescriptions': [{'ipv4Addr': '127.0.0.1', 'port': 4443, 'securityMethods': ['OAUTH']}]}], 'description': 'API of dummy netapp to test',
nfo': {'isShareable': True, 'capifProvDoms': ['string']}, 'serviceAPICategory': 'string', 'apiSuppFeats': 'fffff', 'pubApiPath': {'ccfIds': ['string']}, 'ccfId': 'string'}, {'apiName': '/nef/api/v1/3gpp-as
41813a7355dd186993c', 'aefProfiles': [{'aefId': '4e057d6010bf359fe3ef75374d68aa', 'versions': [{'apiVersion': 'v1', 'expiry': '2100-11-30T10:32:02.004000+00:00', 'resources': [{'resourceName': 'QOS_SUBSCRI
uri': '/{scsAsId}/subscriptions', 'custOpName': 'http_post', 'operations': ['GET', 'POST'], 'description': 'Endpoint to manage monitoring subscriptions'}, {'resourceName': 'QOS_SUBSCRIPTION_SINGLE', 'commT
d}/subscriptions/{subscriptionId}', 'custOpName': 'http_get', 'operations': ['GET', 'PUT', 'DELETE'], 'description': 'Endpoint to manage single subscription'}], 'custOperations': [{'commType': 'REQUEST_RES
s': ['POST'], 'description': 'string'}]}], 'protocol': 'HTTP_1_1', 'dataFormat': 'JSON', 'securityMethods': ['OAUTH', 'PSK'], 'interfaceDescriptions': [{'ipv4Addr': '127.0.0.1', 'port': 4443, 'securityMeth
f dummy netapp to test', 'supportedFeatures': 'fffff', 'shareableInfo': {'isShareable': True, 'capifProvDoms': ['string']}, 'serviceAPICategory': 'string', 'apiSuppFeats': 'fffff', 'pubApiPath': {'ccfIds':
/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py:1063: InsecureRequestWarning: Unverified HTTPS request is being made to host 'nefemu'. Adding certificate verification is strongly advised.
6.x/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
2023-07-10 13:24:20,929 - test - DEBUG - 5G NEF SDK - Token: {'access_token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODk2ODY2NjAsInN1YiI6IjEifQ.EVKPEz9u_sSjWu5nESHcr2IapCUyVb4GatqfhRzecGw',
 'token_type': 'bearer'}
2023-07-10 13:24:20,950 - test - DEBUG - rMON Collector - Init thread
2023-07-10 13:24:20,950 - test - DEBUG - MN Notify - Init thread
2023-07-10 13:24:20,951 - test - DEBUG - MN Notify - Processing queue
2023-07-10 13:24:20,951 - test - DEBUG - rMON Collector - Processing queue
2023-07-10 13:24:20,951 - test - DEBUG - NetApp API Server starting ...
2023-07-10 13:24:20,951 - test - DEBUG - NetApp Checking All Enpoints Connections ...
/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py:1063: InsecureRequestWarning: Unverified HTTPS request is being made to host 'nefemu'. Adding certificate verification is strongly advised.
6.x/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
2023-07-10 13:24:22,036 - test - DEBUG - NetApp Enpoints Connections OK!
```

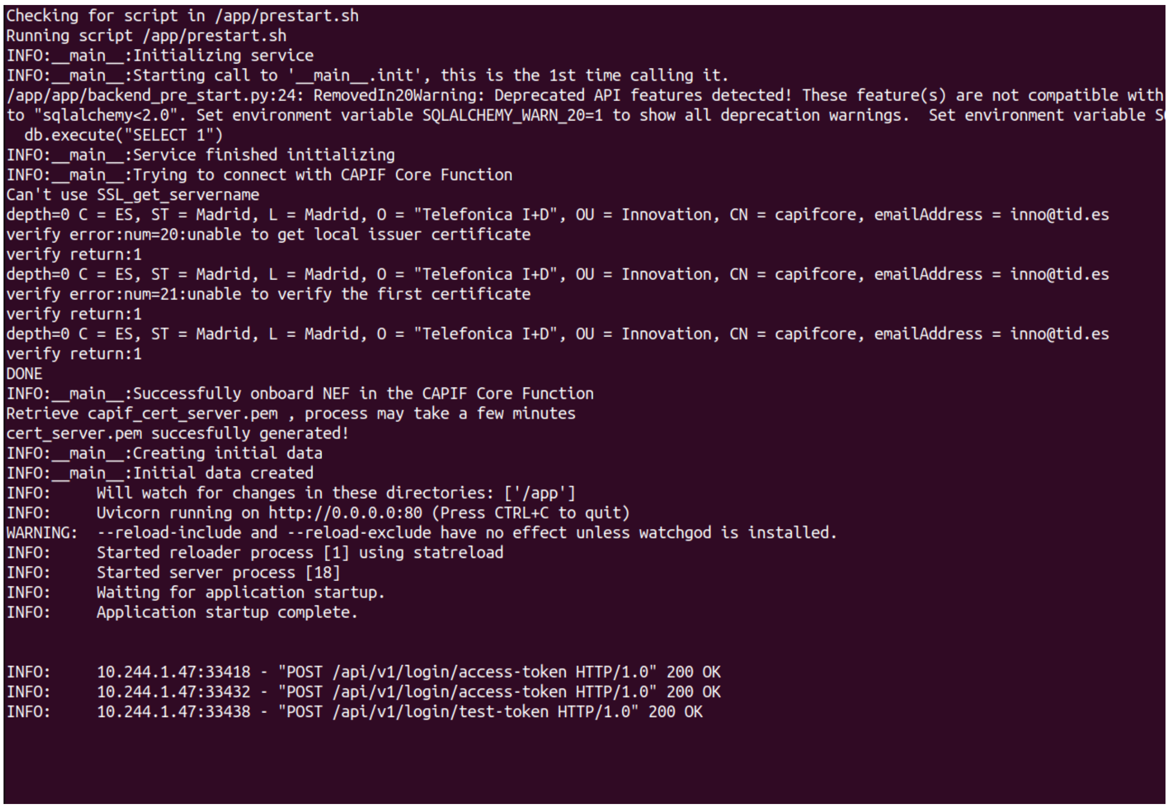
Figure 68 Network App deployed on Kubernetes

2. CAPIF and NEF were also deployed on NCSRD's Kubernetes as shown in Figure 69 and Figure 70 below.



```
rlz@rlz:~/github/evolved/k8s-validation-templates/capif$ kubectl logs -f nginx-578d7d64f8-mwnsl
/etc/nginx/certs/ca.crt: OK
CA root Certificate downloaded successfull
Generating RSA private key, 2048 bit long modulus (2 primes)
.......................................................................................................+++++
...................+++++
e is 65537 (0x010001)
NGINX Hostname is capifcore
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:State or Province Name (full name) [Some-State]:Locality Name (eg, city) []:Organization Name (eg, company) [Internet Widgits Pty Ltd]:Organizational Unit
R name) []:Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:An optional company name []:  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6059  100  4933  100  1126   6977   1592 --:--:-- --:--:-- --:--:--  8557
OK
OK
2023/07/10 13:22:12 [notice] 22#22: using the "epoll" event method
2023/07/10 13:22:12 [notice] 22#22: nginx/1.23.1
2023/07/10 13:22:12 [notice] 22#22: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/07/10 13:22:12 [notice] 22#22: OS: Linux 5.15.0-73-generic
2023/07/10 13:22:12 [notice] 22#22: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/07/10 13:22:12 [notice] 22#22: start worker processes
2023/07/10 13:22:12 [notice] 22#22: start worker process 23
2023/07/10 13:22:12 [notice] 22#22: start worker process 24
2023/07/10 13:22:12 [notice] 22#22: start worker process 25
2023/07/10 13:22:12 [notice] 22#22: start worker process 26


10.244.1.156 - - [10/Jul/2023:13:23:01 +0000] "GET /ca-root HTTP/1.1" 201 1203 "-" "python-requests/2.31.0"
10.244.1.156 - - [10/Jul/2023:13:23:01 +0000] "POST /register HTTP/1.1" 201 216 "-" "python-requests/2.31.0"
10.244.1.156 - - [10/Jul/2023:13:23:02 +0000] "POST /getauth HTTP/1.1" 200 657 "-" "python-requests/2.31.0"
10.244.1.156 - - [10/Jul/2023:13:23:04 +0000] "POST /api-provider-management/v1/registrations HTTP/1.1" 201 18774 "-" "python-requests/2.31.0"
10.244.1.156 - - [10/Jul/2023:13:23:04 +0000] "POST /published-apis/v1/d280de0b801e6c4fa2ed5bb15148f5/service-apis HTTP/1.1" 201 1310 "-" "python-requests/2.31.0"
10.244.1.156 - - [10/Jul/2023:13:23:04 +0000] "POST /published-apis/v1/d280de0b801e6c4fa2ed5bb15148f5/service-apis HTTP/1.1" 201 1299 "-" "python-requests/2.31.0"


10.244.1.33 - - [10/Jul/2023:13:24:19 +0000] "POST /register HTTP/1.1" 201 223 "-" "python-requests/2.26.0"
10.244.1.33 - - [10/Jul/2023:13:24:19 +0000] "POST /getauth HTTP/1.1" 200 1870 "-" "python-requests/2.26.0"
10.244.1.33 - - [10/Jul/2023:13:24:20 +0000] "POST /api-invoker-management/v1/onboardedInvokers HTTP/1.1" 201 6358 "-" "python-requests/2.26.0"
10.244.1.33 - - [10/Jul/2023:13:24:20 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=95f39d78def30c85ea6adce9f1fc70 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
```

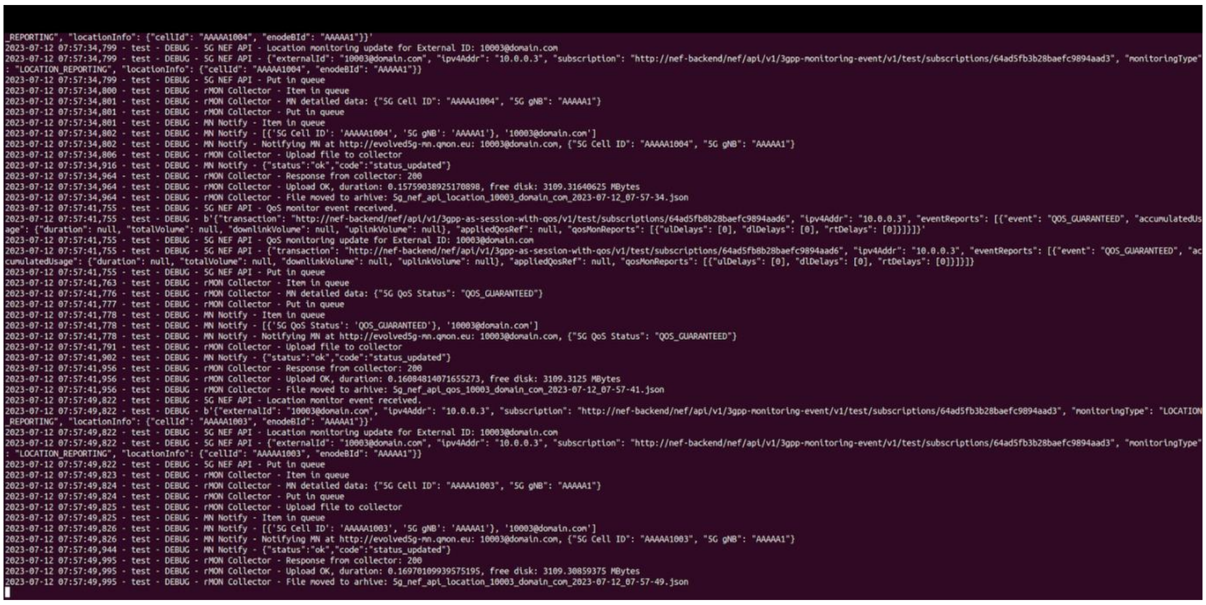
Figure 69 CAPIF deployed on Kubernetes



```
Checking for script in /app/prestart.sh
Running script /app/prestart.sh
INFO:__main__:Initializing service
INFO:__main__:Starting call to '__main__.init', this is the 1st time calling it.
/app/app/backend_pre_start.py:24: RemovedIn20Warning: Deprecated API features detected! These feature(s) are not compatible with
to "sqlalchemy<2.0". Set environment variable SQLALCHEMY_WARN_20=1 to show all deprecation warnings.  Set environment variable S
  db.execute("SELECT 1")
INFO:__main__:Service finished initializing
INFO:__main__:Trying to connect with CAPIF Core Function
Can't use SSL_get_servername
depth=0 C = ES, ST = Madrid, L = Madrid, O = "Telefonica I+D", OU = Innovation, CN = capifcore, emailAddress = inno@tid.es
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = ES, ST = Madrid, L = Madrid, O = "Telefonica I+D", OU = Innovation, CN = capifcore, emailAddress = inno@tid.es
verify error:num=21:unable to verify the first certificate
verify return:1
depth=0 C = ES, ST = Madrid, L = Madrid, O = "Telefonica I+D", OU = Innovation, CN = capifcore, emailAddress = inno@tid.es
verify return:1
DONE
INFO:__main__:Successfully onboard NEF in the CAPIF Core Function
Retrieve capif_cert_server.pem , process may take a few minutes
cert_server.pem succesfully generated!
INFO:__main__:Creating initial data
INFO:__main__:Initial data created
INFO:     Will watch for changes in these directories: ['/app']
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
WARNING:  --reload-include and --reload-exclude have no effect unless watchgod is installed.
INFO:     Started reloader process [1] using statreload
INFO:     Started server process [18]
INFO:     Waiting for application startup.
INFO:     Application startup complete.


INFO:     10.244.1.47:33418 - "POST /api/v1/login/access-token HTTP/1.0" 200 OK
INFO:     10.244.1.47:33432 - "POST /api/v1/login/access-token HTTP/1.0" 200 OK
INFO:     10.244.1.47:33438 - "POST /api/v1/login/test-token HTTP/1.0" 200 OK
```

Figure 70 NEF deployed on Kubernetes

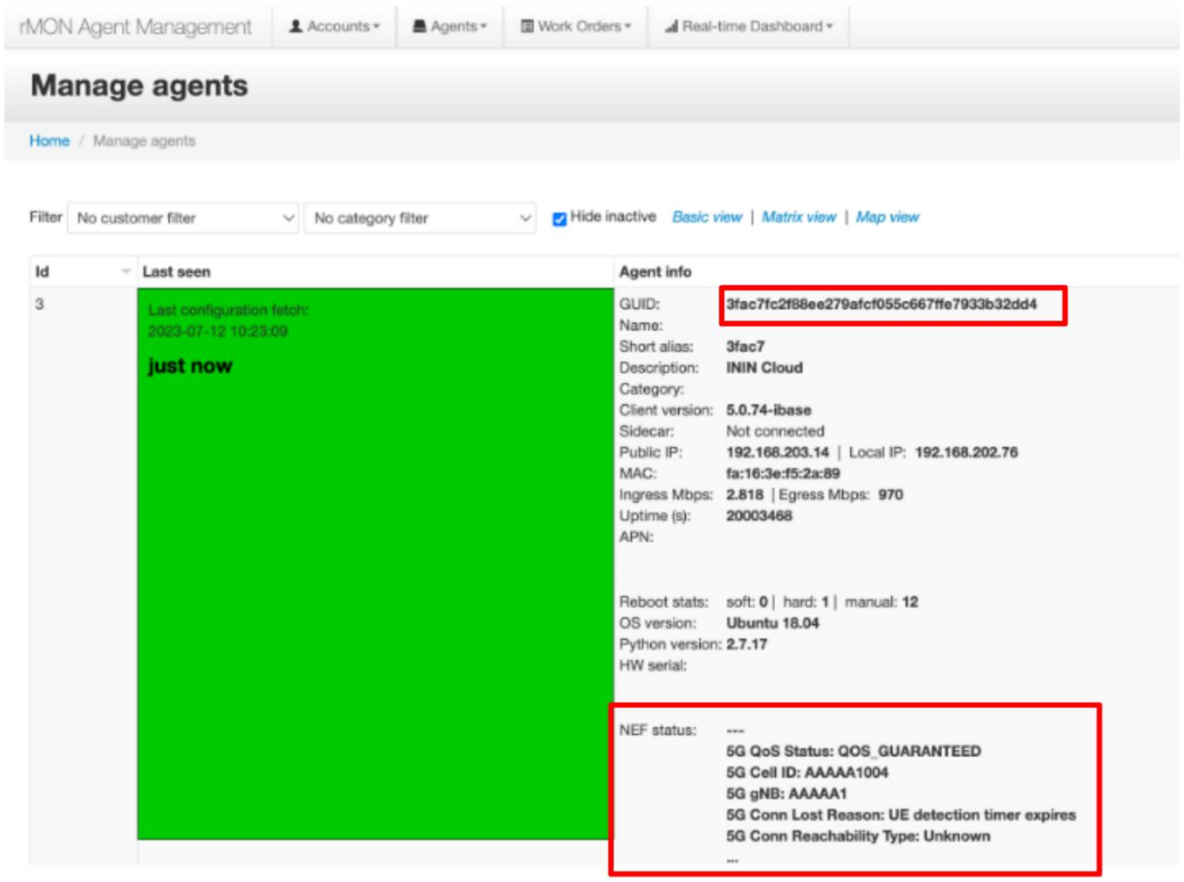
3. After successful deployment of all necessary components end-to-end tests were performed as depicted in Figure 71 below.



```
_REPORTING", "locationInfo": {"cellId": "AAAAA1004", "enodeBId": "AAAAA1"}}'
2023-07-12 07:57:34,799 - test - DEBUG - 5G NEF API - Location monitoring update for External ID: 10003@domain.com
2023-07-12 07:57:34,799 - test - DEBUG - 5G NEF API - {"externalId": "10003@domain.com", "ipv4Addr": "10.0.0.3", "subscription": "http://nef-backend/nef/api/v1/3gpp-monitoring-event/v1/test/subscriptions/64ad5fb3b28baefc9894aad3", "monitoringType"
: "LOCATION_REPORTING", "locationInfo": {"cellId": "AAAAA1004", "enodeBId": "AAAAA1"}}
2023-07-12 07:57:34,799 - test - DEBUG - 5G NEF API - Put in queue
2023-07-12 07:57:34,800 - test - DEBUG - rMON Collector - Item in queue
2023-07-12 07:57:34,801 - test - DEBUG - rMON Collector - MN detailed data: {"5G Cell ID": "AAAAA1004", "5G gNB": "AAAAA1"}
2023-07-12 07:57:34,801 - test - DEBUG - rMON Collector - Put in queue
2023-07-12 07:57:34,801 - test - DEBUG - MN Notify - Item in queue
2023-07-12 07:57:34,802 - test - DEBUG - MN Notify - [{'5G Cell ID': 'AAAAA1004', '5G gNB': 'AAAAA1'}, '10003@domain.com']
2023-07-12 07:57:34,802 - test - DEBUG - MN Notify - Notifying MN at http://evolved5g-mn.qmon.eu: 10003@domain.com, {"5G Cell ID": "AAAAA1004", "5G gNB": "AAAAA1"}
2023-07-12 07:57:34,806 - test - DEBUG - rMON Collector - Upload file to collector
2023-07-12 07:57:34,916 - test - DEBUG - MN Notify - {"status":"ok","code":"status_updated"}
2023-07-12 07:57:34,964 - test - DEBUG - rMON Collector - Response from collector: 200
2023-07-12 07:57:34,964 - test - DEBUG - rMON Collector - Upload OK, duration: 0.15759038925170898, free disk: 3109.31640625 MBytes
2023-07-12 07:57:34,964 - test - DEBUG - rMON Collector - File moved to arhive: 5g_nef_api_location_10003_domain_com_2023-07-12_07-57-34.json
2023-07-12 07:57:41,755 - test - DEBUG - 5G NEF API - QoS monitor event received.
2023-07-12 07:57:41,755 - test - DEBUG - b'{"transaction": "http://nef-backend/nef/api/v1/3gpp-as-session-with-qos/v1/test/subscriptions/64ad5fb8b28baefc9894aad6", "ipv4Addr": "10.0.0.3", "eventReports": [{"event": "QOS_GUARANTEED", "accumulatedUs
age": {"duration": null, "totalVolume": null, "downlinkVolume": null, "uplinkVolume": null}, "appliedQosRef": null, "qosMonReports": [{"ulDelays": [0], "dlDelays": [0], "rtDelays": [0]}]}]}'
2023-07-12 07:57:41,755 - test - DEBUG - 5G NEF API - QoS monitoring update for External ID: 10003@domain.com
2023-07-12 07:57:41,755 - test - DEBUG - 5G NEF API - {"transaction": "http://nef-backend/nef/api/v1/3gpp-as-session-with-qos/v1/test/subscriptions/64ad5fb8b28baefc9894aad6", "ipv4Addr": "10.0.0.3", "eventReports": [{"event": "QOS_GUARANTEED", "ac
cumulatedUsage": {"duration": null, "totalVolume": null, "downlinkVolume": null, "uplinkVolume": null}, "appliedQosRef": null, "qosMonReports": [{"ulDelays": [0], "dlDelays": [0], "rtDelays": [0]}]}]}
2023-07-12 07:57:41,755 - test - DEBUG - 5G NEF API - Put in queue
2023-07-12 07:57:41,763 - test - DEBUG - rMON Collector - Item in queue
2023-07-12 07:57:41,776 - test - DEBUG - rMON Collector - MN detailed data: {"5G QoS Status": "QOS_GUARANTEED"}
2023-07-12 07:57:41,777 - test - DEBUG - rMON Collector - Put in queue
2023-07-12 07:57:41,778 - test - DEBUG - MN Notify - Item in queue
2023-07-12 07:57:41,778 - test - DEBUG - MN Notify - [{'5G QoS Status': 'QOS_GUARANTEED'}, '10003@domain.com']
2023-07-12 07:57:41,778 - test - DEBUG - MN Notify - Notifying MN at http://evolved5g-mn.qmon.eu: 10003@domain.com, {"5G QoS Status": "QOS_GUARANTEED"}
2023-07-12 07:57:41,791 - test - DEBUG - rMON Collector - Upload file to collector
2023-07-12 07:57:41,902 - test - DEBUG - MN Notify - {"status":"ok","code":"status_updated"}
2023-07-12 07:57:41,956 - test - DEBUG - rMON Collector - Response from collector: 200
2023-07-12 07:57:41,956 - test - DEBUG - rMON Collector - Upload OK, duration: 0.16084814071655273, free disk: 3109.3125 MBytes
2023-07-12 07:57:41,956 - test - DEBUG - rMON Collector - File moved to arhive: 5g_nef_api_qos_10003_domain_com_2023-07-12_07-57-41.json
2023-07-12 07:57:49,822 - test - DEBUG - 5G NEF API - Location monitor event received.
2023-07-12 07:57:49,822 - test - DEBUG - b'{"externalId": "10003@domain.com", "ipv4Addr": "10.0.0.3", "subscription": "http://nef-backend/nef/api/v1/3gpp-monitoring-event/v1/test/subscriptions/64ad5fb3b28baefc9894aad3", "monitoringType": "LOCATION
_REPORTING", "locationInfo": {"cellId": "AAAAA1003", "enodeBId": "AAAAA1"}}'
2023-07-12 07:57:49,822 - test - DEBUG - 5G NEF API - Location monitoring update for External ID: 10003@domain.com
2023-07-12 07:57:49,822 - test - DEBUG - 5G NEF API - {"externalId": "10003@domain.com", "ipv4Addr": "10.0.0.3", "subscription": "http://nef-backend/nef/api/v1/3gpp-monitoring-event/v1/test/subscriptions/64ad5fb3b28baefc9894aad3", "monitoringType"
: "LOCATION_REPORTING", "locationInfo": {"cellId": "AAAAA1003", "enodeBId": "AAAAA1"}}
2023-07-12 07:57:49,822 - test - DEBUG - 5G NEF API - Put in queue
2023-07-12 07:57:49,823 - test - DEBUG - rMON Collector - Item in queue
2023-07-12 07:57:49,824 - test - DEBUG - rMON Collector - MN detailed data: {"5G Cell ID": "AAAAA1003", "5G gNB": "AAAAA1"}
2023-07-12 07:57:49,824 - test - DEBUG - rMON Collector - Put in queue
2023-07-12 07:57:49,825 - test - DEBUG - rMON Collector - Upload file to collector
2023-07-12 07:57:49,825 - test - DEBUG - MN Notify - Item in queue
2023-07-12 07:57:49,826 - test - DEBUG - MN Notify - [{'5G Cell ID': 'AAAAA1003', '5G gNB': 'AAAAA1'}, '10003@domain.com']
2023-07-12 07:57:49,826 - test - DEBUG - MN Notify - Notifying MN at http://evolved5g-mn.qmon.eu: 10003@domain.com, {"5G Cell ID": "AAAAA1003", "5G gNB": "AAAAA1"}
2023-07-12 07:57:49,944 - test - DEBUG - MN Notify - {"status":"ok","code":"status_updated"}
2023-07-12 07:57:49,995 - test - DEBUG - rMON Collector - Response from collector: 200
2023-07-12 07:57:49,995 - test - DEBUG - rMON Collector - Upload OK, duration: 0.16970109939575195, free disk: 3109.30859375 MBytes
2023-07-12 07:57:49,995 - test - DEBUG - rMON Collector - File moved to arhive: 5g_nef_api_location_10003_domain_com_2023-07-12_07-57-49.json
```

Figure 71 ININ’s Network App logs

Status of NEF regarding UE (IoT GW) is visible also on IoT Management dashboard as shown in Figure 72 below.



Id	Last seen	Agent info
3	Last configuration fetch: 2023-07-12 10:23:09 **just now**	GUID: 3fac7fc2f88ee279afcf055c667ffe7933b32dd4 Name: Short alias: 3fac7 Description: ININ Cloud Category: Client version: 5.0.74-ibase Sidecar: Not connected Public IP: 192.168.203.14 | Local IP: 192.168.202.76 MAC: fa:16:3e:f5:2a:89 Ingress Mbps: 2.818 | Egress Mbps: 970 Uptime (s): 20003468 APN: Reboot stats: soft: 0 | hard: 1 | manual: 12 OS version: Ubuntu 18.04 Python version: 2.7.17 HW serial: NEF status: --- 5G QoS Status: QOS_GUARANTEED 5G Cell ID: AAAAA1004 5G gNB: AAAAA1 5G Conn Lost Reason: UE detection timer expires 5G Conn Reachability Type: Unknown ---

Figure 72 Current status of NEF regarding UE (IoT GW)

4. NEF events are collected in DB and exposed to Grafana for deep-down analytics and correlation steps as shown in Figure 73 below.

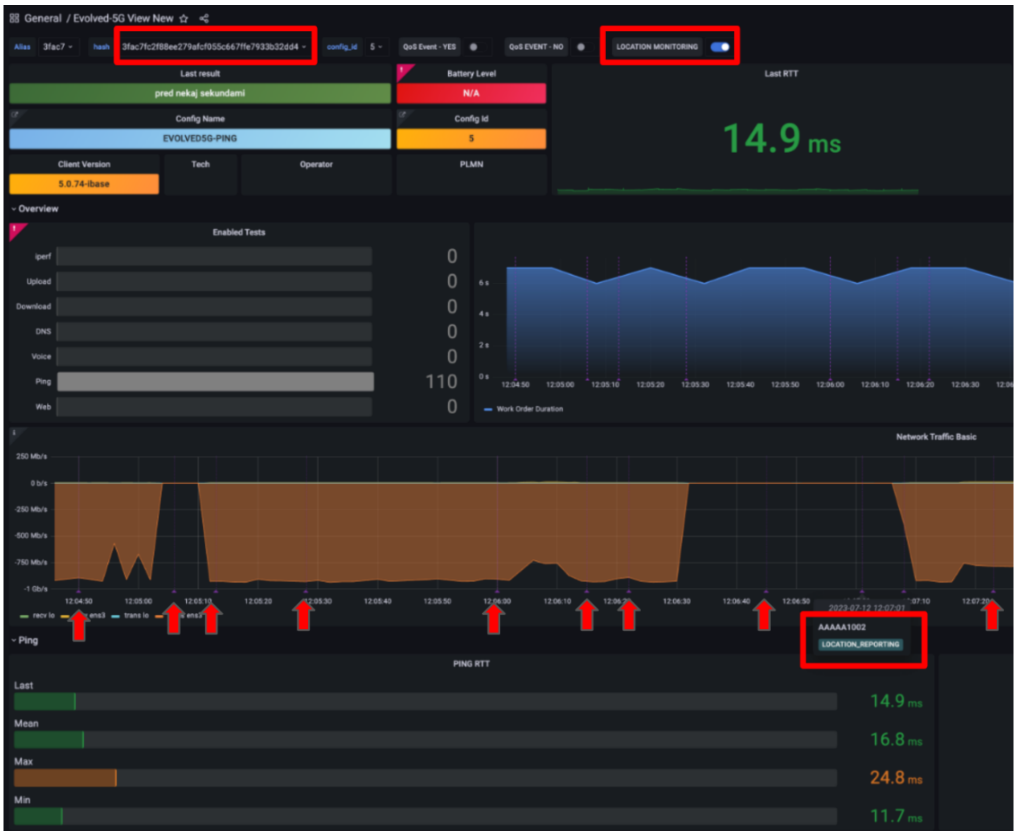


Figure 73 LOCATION_REPORTING triggered events on Grafana

5. Dynamic traffic steering based QoS subscriptions events are represented on Grafana dashboard and collected on IoT Collector database as shown in Figure 74 below.

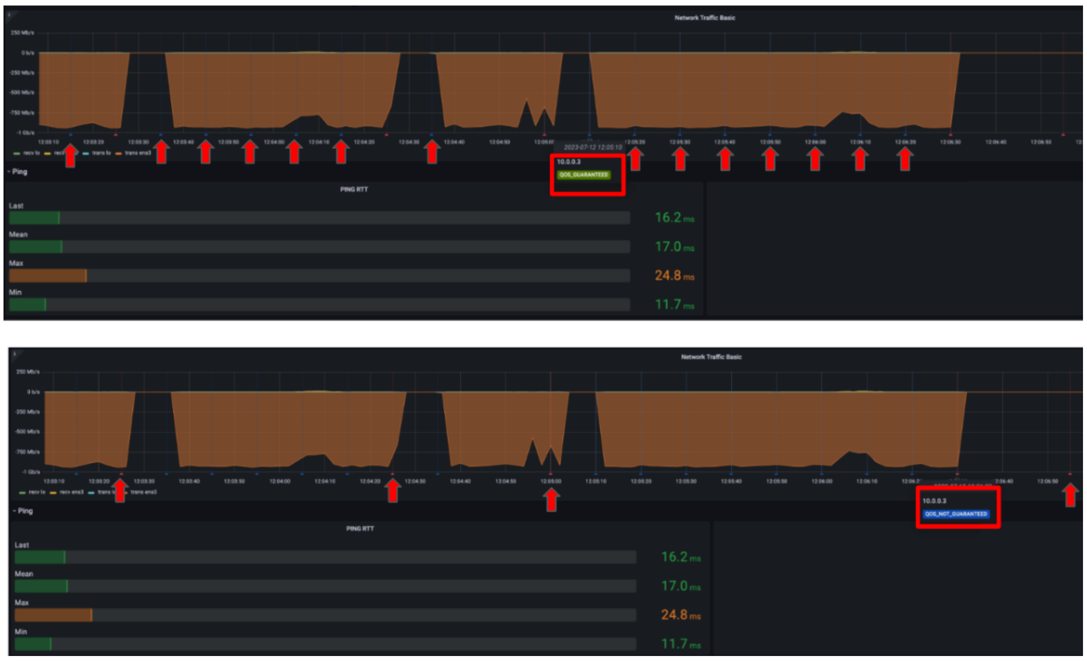


Figure 74 QOS_GUARANTEED and QOS_NOT_GUARANTEED periodic events on Grafana

4.5.3 Smart Irrigation 5G Agriculture Network Application

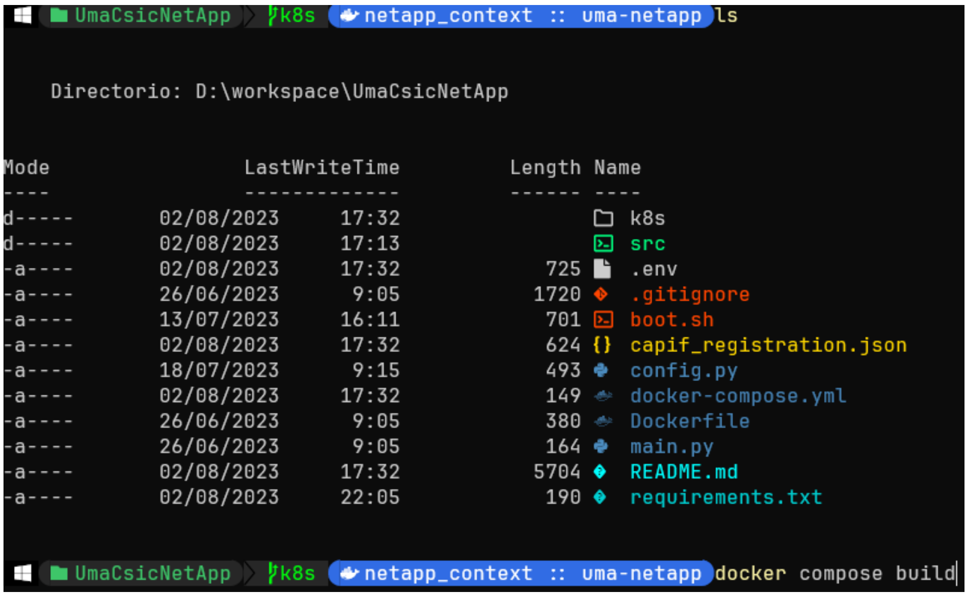
Deployment of UMA Network App has been done into UMA Kubernetes cluster. The version of the applications used are:

- NEF: 2.2.2
- CAPIF: 3.1.2
- SDK: 1.0.8

The code of the NetApp can be seen in the following repository: https://github.com/EVOLVED-5G/UmaCsicNetApp/tree/k8s

Starting with deployment:

1. Create the Network App image. To do this, run the command docker compose build as in Figure 75 below.



```
 UmaCsicNetApp  k8s  netapp_context :: uma-netapp ls


    Directorio: D:\workspace\UmaCsicNetApp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        02/08/2023     17:32                k8s
d-----        02/08/2023     17:13                src
-a----        02/08/2023     17:32            725 .env
-a----        26/06/2023      9:05           1720 .gitignore
-a----        13/07/2023     16:11            701 boot.sh
-a----        02/08/2023     17:32            624 capif_registration.json
-a----        18/07/2023      9:15            493 config.py
-a----        02/08/2023     17:32            149 docker-compose.yml
-a----        26/06/2023      9:05            380 Dockerfile
-a----        26/06/2023      9:05            164 main.py
-a----        02/08/2023     17:32           5704 README.md
-a----        02/08/2023     22:05            190 requirements.txt

 UmaCsicNetApp  k8s  netapp_context :: uma-netapp docker compose build
```

Figure 75 Creation of the Network App image

2. Push image to insert it into the docker hub repository (Figure 76). You need to execute the following commands:

docker tag uma-netapp:latest carlosandreo/uma-netapp

docker login

docker push carlosandreo/uma-netapp

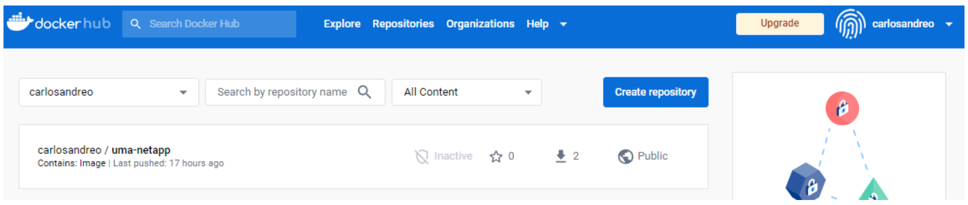


Figure 76 Network App Image into Docker hub

3. Enable VPN profile to connect to the UMA Kubernetes cluster.
4. Install kubectl.

5. Configure the access token to be able to deploy the Network App on the UMA Kubernetes cluster. To do this, you need to request the UMA access profile and execute the following commands:

```
kubectl config set-cluster <CLUSTER_NAME> --server=<API_SERVER:PORT> --insecure-skip-tls-verify=true

kubectl config set-credentials <USER_NAME> --token=<TOKEN>

kubectl config set-context <CONTEXT_NAME> --cluster=<CLUSTER_NAME> --user=<USER_NAME> --namespace <NAMESPACE>

kubectl config use-context <CONTEXT_NAME>
```

6. Telefonica deploys NEF and CAPIF on the UMA Kubernetes cluster for each SME. Then, it provides us with the endpoints to be able to do the onboarding of the Network App. Add the endpoints in the /etc/hosts file of the NEF and CAPIF. To do this step it is necessary to run Notepad++ with administrator permissions.
7. Check NEF access and import scenario. Then start all (Figure 77).

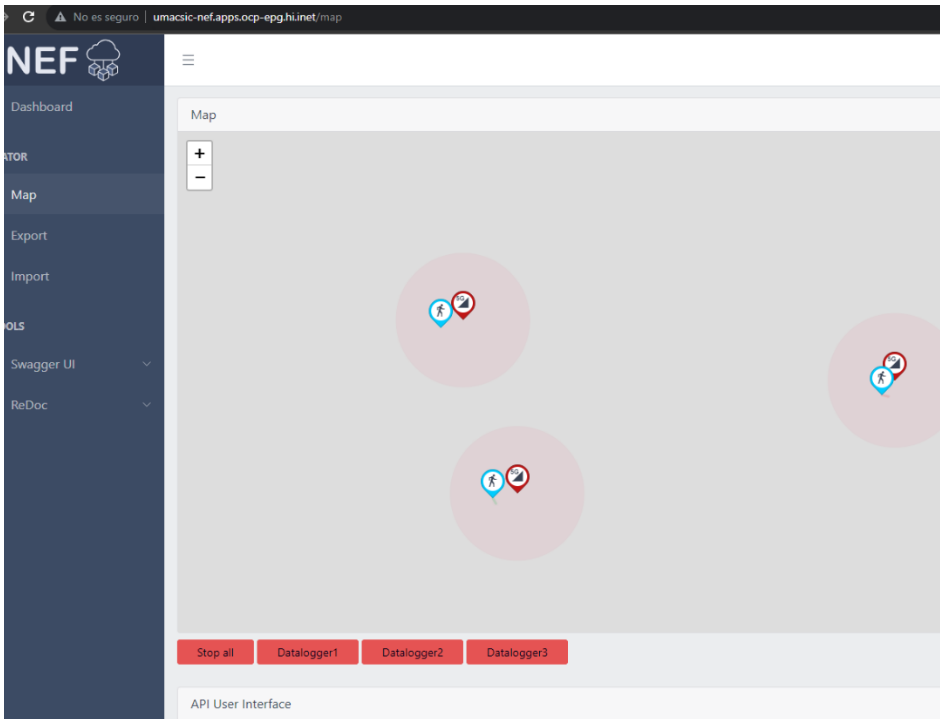


Figure 77 NEF with right scenario

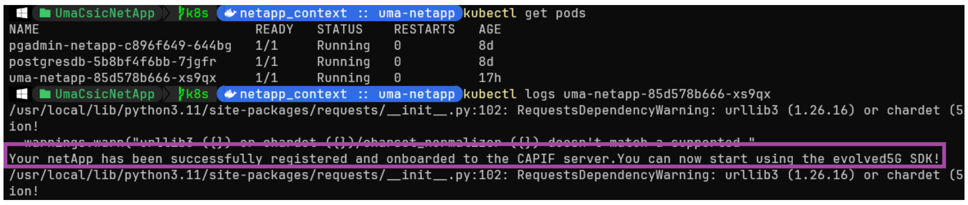
8. Before starting with the NetApp deployment, it is necessary to deploy PostgreSQL to be able to store the data received from the sensors deployed in the field. Then to have control over the database, pgAdmin is deployed. Finally, we proceed to deploy the Network App by connecting it to the previously deployed PostgreSQL. The commands to execute for the three deployments are:

```
Kubectl apply –f environment.yaml

Kubectl apply –f deployment.yaml

Kubectl apply –f service.yaml
```

All applied files are in the UmaCsicNetApp repository: https://github.com/EVOLVED-5G/UmaCsicNetApp/tree/k8s/k8s

9. When all the pods in the deployment are running you can check if the NetApp has done the onboarding with CAPIF by accessing the NetApp pod logs (Figure 78). The following command is executed where namePodNetApp is the name assigned to the NetApp pod by the cluster:

Kubectl logs namePodNetApp



```
UmaCsicNetApp  k8s  netapp_context :: uma-netapp kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
pgadmin-netapp-c896f649-644bg    1/1     Running   0          8d
postgresdb-5b8bf4f6bb-7jgfr      1/1     Running   0          8d
uma-netapp-85d578b666-xs9qx      1/1     Running   0          17h
UmaCsicNetApp  k8s  netapp_context :: uma-netapp kubectl logs uma-netapp-85d578b666-xs9qx
/usr/local/lib/python3.11/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5
ion!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
Your netApp has been successfully registered and onboarded to the CAPIF server.You can now start using the evolved5G SDK!
/usr/local/lib/python3.11/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5
ion!
```

Figure 78 NetApp onboarding CAPIF

10. As shown in Figure 79 the NetApp is already running in the UMA Kubernetes cluster because to access it through the browser, it is necessary to put an IP address that the cluster provides when you apply the services.yaml file.

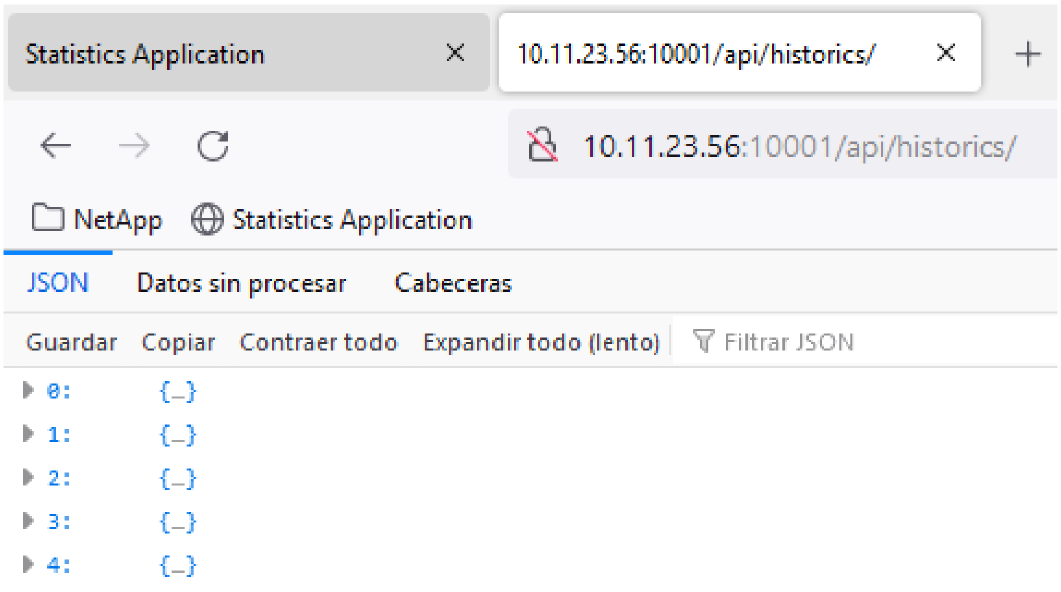
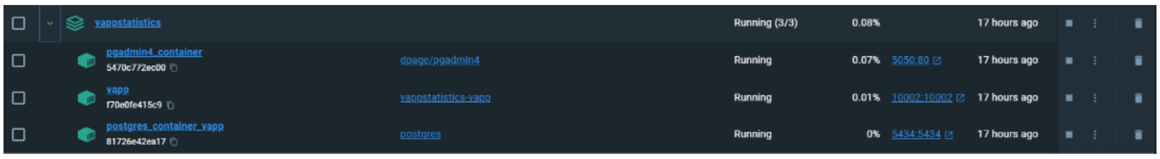


Figure 79 NetApp running on the UMA Kubernetes cluster

11. Connect the vApp to the NetApp. To connect the vApp with the NetApp the vApp has been deployed in Docker on a local PC (Figure 80).



vappstatistics		Running (3/3)	0.08%		17 hours ago
pgadmin4_container 5470c772ec00	dpage/pgadmin4	Running	0.07%	5050:80	17 hours ago
vapp f70e0fe415c9	vappstatistics-vapp	Running	0.01%	10002:10002	17 hours ago
postgres_container_vapp 81726e42ea17	postgres	Running	0%	5434:5434	17 hours ago

Figure 80 Containers deployed to run the vApp

12. As mentioned above in section 4.1.3 the vApp represents in graphs the data obtained by the Network App (Figure 81).

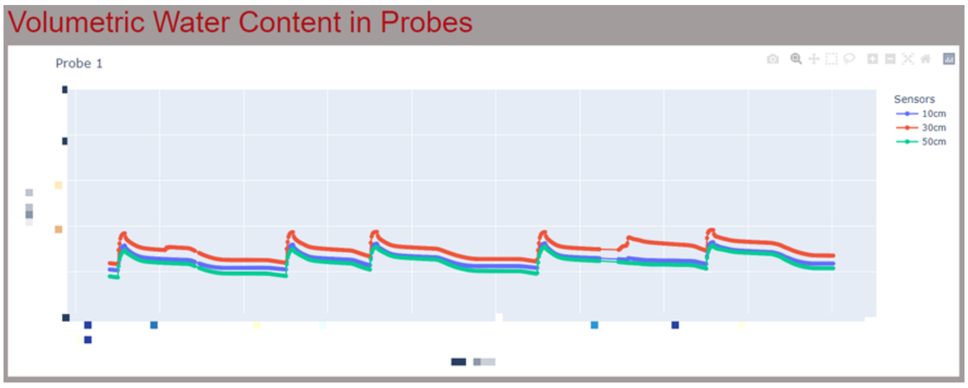


Figure 81 vApp plotting the data obtained from the NetApp graphically

4.5.4 Anomaly Detection Network Application

The final version of Network App v4.1 alongside NEF v2.2.2, SDK v1.1 and CAPIF v3.1.2 took place at NCSRD premises. Network App, NEF and CAPIF were deployed within the Kubernetes cluster. The devices that simulate a small production line are Wi-Fi enabled so a 5G connectivity was possible by using the 5G CPE Box and Amarisoft mini as depicted below in Figure 82.

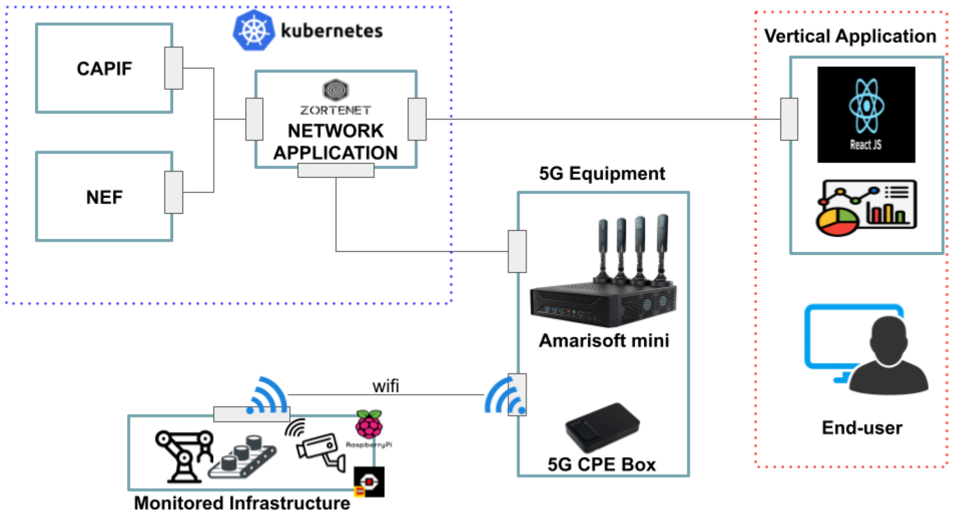


Figure 82 Topology of Zortenet Network App round 2

As mentioned, the devices have a 5G connection by using the Waveshare 5G CPE Box which is capable of providing 5G connectivity via Wi-Fi. Finally, the 5G connection is provided by Amarisoft mini as shown in Figure 83 below.

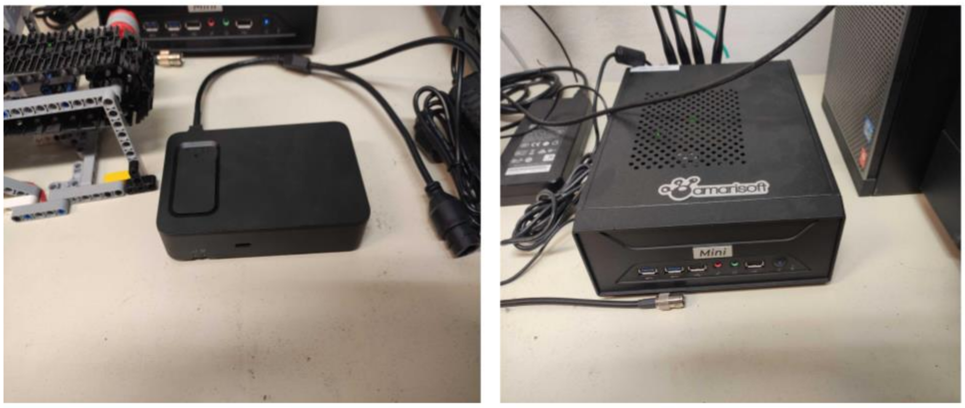


Figure 83 5G CPE Box and Amarisoft mini

4.6 Results and Takeaways

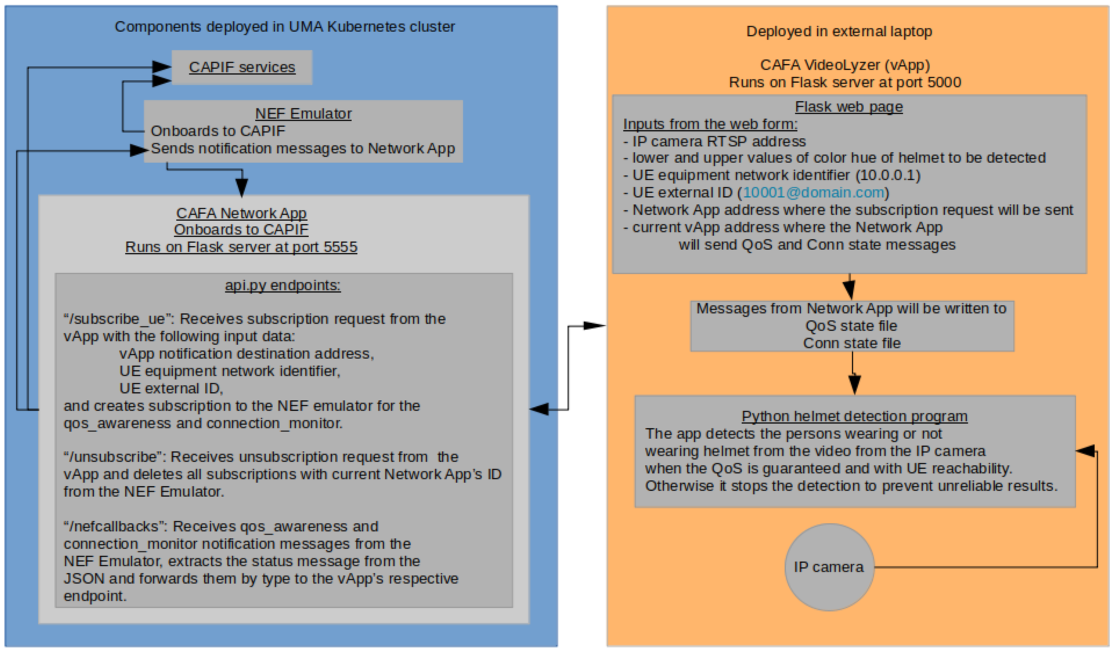
4.6.1 CAFA Network Application NetMapper

Deployment of CAFA Network App NetMapper into UMA Kubernetes cluster was done on 5th-10th July 2023 with the cooperation with Telefonica and UMA team. Deployment and verification of the whole system was done remotely from Tallinn (CAFA premises) to Malaga (UMA 5G testbed).

The components involved were:

- CAFA Network App NetMapper 4.1
- SDK v1.1.0 (1.0.8 when installed through pipeline)
- CAPIF 3.1.2
- NEF emulator 2.2.2

CAPIF and NEF emulator were also deployed in UMA Kubernetes cluster as depicted in Figure 84 below.


Components deployed in UMA Kubernetes cluster

CAPIF services

NEF Emulator
Onboards to CAPIF
Sends notification messages to Network App

CAFA Network App
Onboards to CAPIF
Runs on Flask server at port 5555

api.py endpoints:

"/subscribe_ue": Receives subscription request from the vApp with the following input data:
vApp notification destination address,
UE equipment network identifier,
UE external ID,
and creates subscription to the NEF emulator for the qos_awareness and connection_monitor.

"/unsubscribe": Receives unsubscription request from the vApp and deletes all subscriptions with current Network App's ID from the NEF Emulator.

"/nefcallbacks": Receives qos_awareness and connection_monitor notification messages from the NEF Emulator, extracts the status message from the JSON and forwards them by type to the vApp's respective endpoint.

Deployed in external laptop

CAFA VideoLyzer (vApp)
Runs on Flask server at port 5000

Flask web page
Inputs from the web form:
- IP camera RTSP address
- lower and upper values of color hue of helmet to be detected
- UE equipment network identifier (10.0.0.1)
- UE external ID (10001@domain.com)
- Network App address where the subscription request will be sent
- current vApp address where the Network App will send QoS and Conn state messages

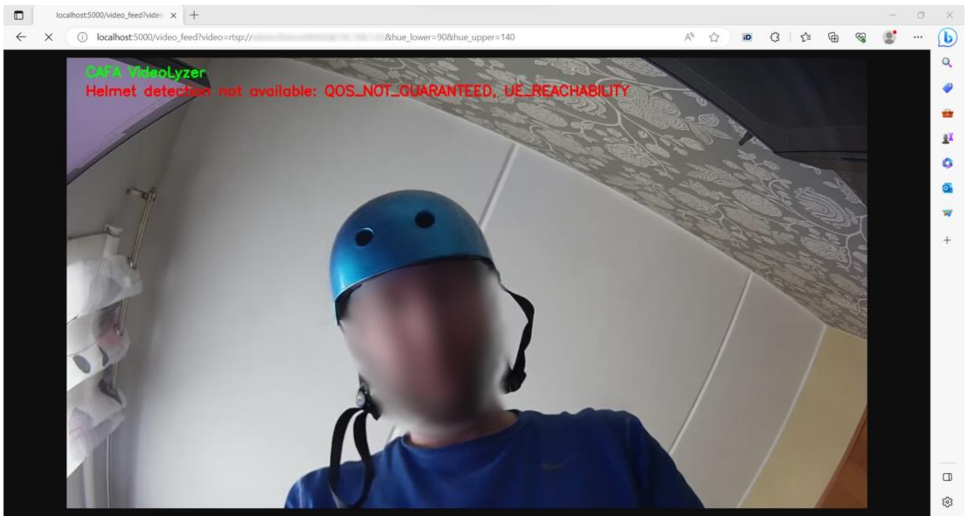
Messages from Network App will be written to
QoS state file
Conn state file

Python helmet detection program
The app detects the persons wearing or not wearing helmet from the video from the IP camera when the QoS is guaranteed and with UE reachability. Otherwise it stops the detection to prevent unreliable results.

Figure 84 Updated CAFA NetMapper Network App – SafeLyzer vApp – CAPIF – NEF architecture

CAFA Tech was successfully deployed CAFA NetMapper Network App and validated the Safety VideoLyzer use case with the final components of EVOLVED 5G. On the one hand, NEF, CAPIF and the SDK had been enriched with additional features. On the other hand, CAFA Tech had also finalized our NetMapper Network App by expanding the 3.0 version and using the last versions of NEF, CAPIF and SDK. The final prototype (v4.1) of the Network Apps also completed and used the validation pipeline before the integration test. Finally, the Networks Apps were deployed in Kubernetes clusters in UMA's premises instead of using Docker containers running locally and the Openstack environment as in the first round of integration testing.

When the UE in the map was outside the cell area or it was in the cell where there was also another UE, then the helmet detection was not available (see the red warning in the Figure 85 below). This was because in the circumstances of loss of connectivity and/or no quality of service the computer vision behavior was not reliable and could lead to false alarms and/or no true alerts.


localhost:5000/video_feed?video=rtsp://[redacted]&hue_lower=90&hue_upper=140

CAFA VideoLyzer
Helmet detection not available: QOS_NOT_GUARANTEED, UE_REACHABILITY

Figure 85 The Personal Protective Equipment - helmet - detection is not available when QoS or connectivity is not guaranteed

When the UE was inside the cell and there were no other UEs in that cell, the helmet detection was available. When a person had no safety helmet, the warning was shown with person in the red rectangle as shown in Figure 86 below.

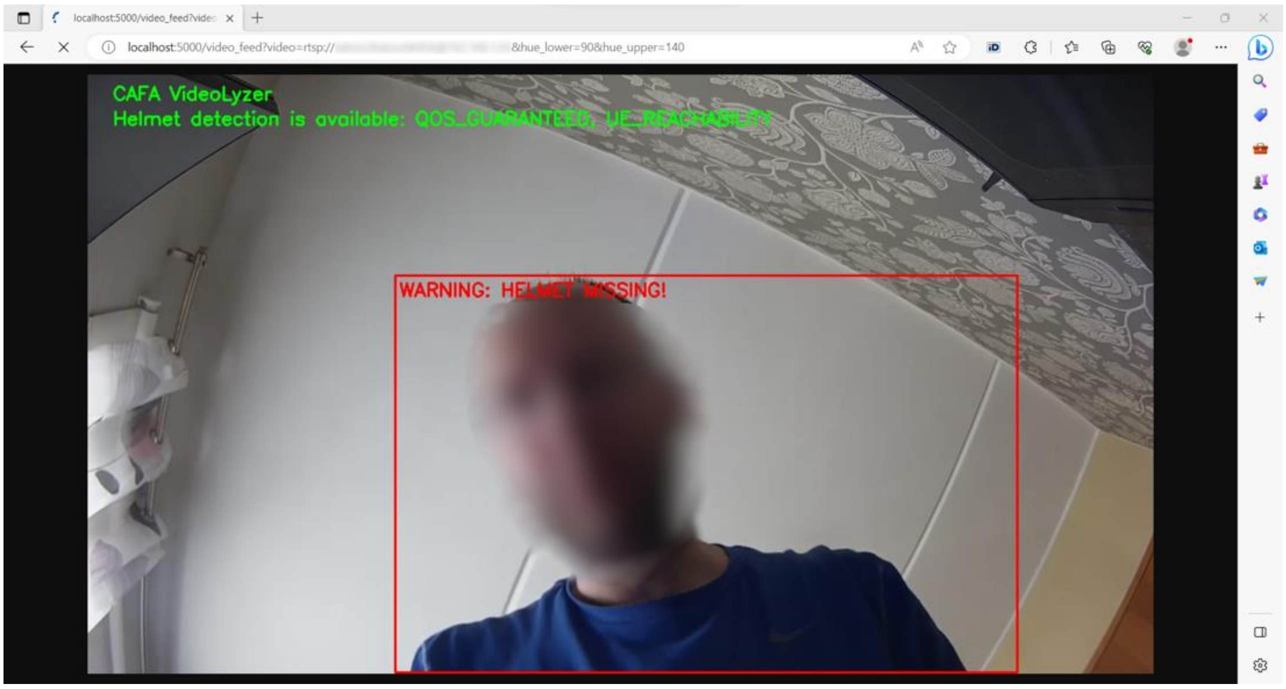

localhost:5000/video_feed?video=rtsp://
&hue_lower=90&hue_upper=140
CAFA VideoLyzer
Helmet detection is available: QOS_GUARANTEED, UE_REACHABILITY
WARNING: HELMET MISSING!

Figure 86 Personal Protective Equipment safety helmet detection is available when QoS and connectivity is guaranteed. Missing safety helmet detected

When the correct helmet was detected, the green rectangle was around the helmet to indicate it as shown in Figure 87 below.

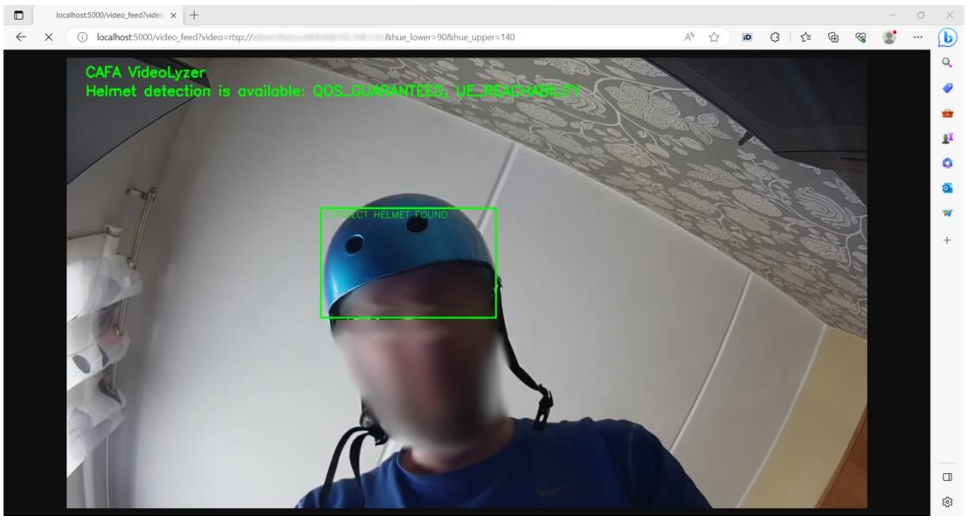

localhost:5000/video_feed?video=rtsp://
&hue_lower=90&hue_upper=140
CAFA VideoLyzer
Helmet detection is available: QOS_GUARANTEED, UE_REACHABILITY
CORRECT HELMET FOUND

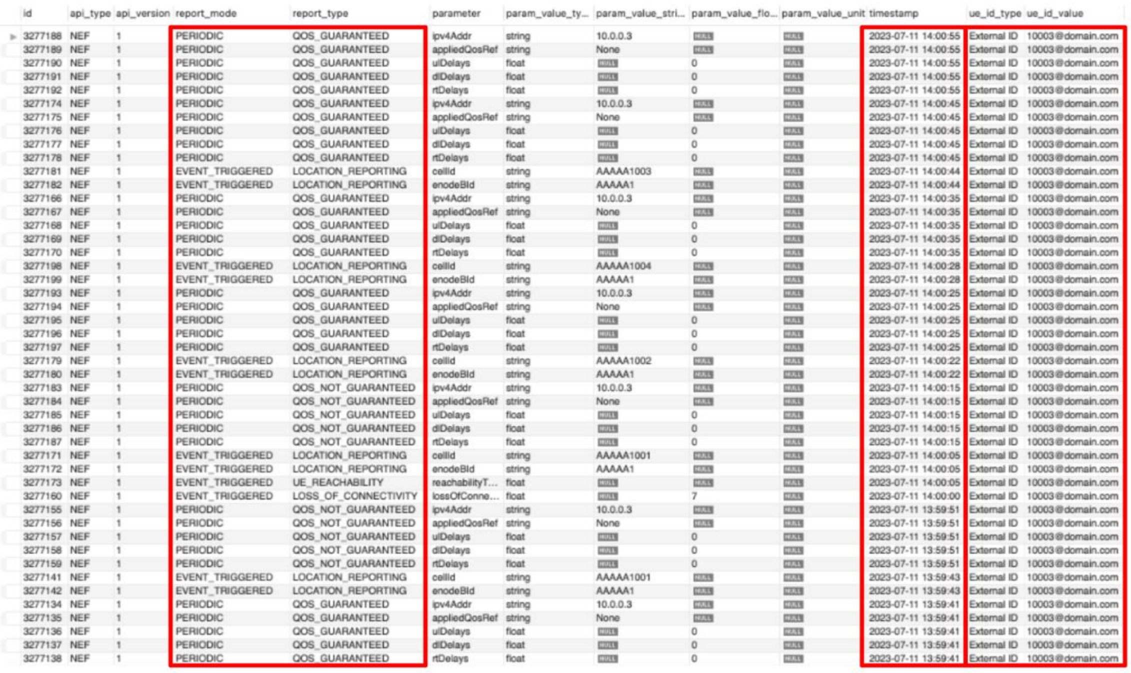
Figure 87 Personal Protective Equipment - correct safety helmet - detected

4.6.2 Industrial Grade 5G Connectivity Network Application

Integration of ININ Network App was done in July 2023 with the cooperation of NCSR Demokritos team. Testing and verification of the whole system was done remotely from Ljubljana (ININ premises) to Athens (NCSR Demokritos 5G testbed).

Final version Network App v4.1 including NEF v2.2.2, SDK v1.1 and CAPIF v3.1.2 was used for the 2nd round of the integration. Network App, NEF and CAPIF were deployed within the Kubernetes cluster.

Network App was successfully deployed and acts as a middle component between vApp and NEF. It starts receiving events from NEF and notifies IoT Management on events in 5G Network. IoT Management reports status to IoT GW. IoT GW starts or stops transmitting BE data based on actual 5G network conditions. All events and data are stored into IoT Collector database for further analytics.



id	api_type	api_version	report_mode	report_type	parameter	param_value_ty...	param_value_stri...	param_value_flo...	param_value_unit	timestamp	ue_id_type	ue_id_value
3277188	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 14:00:55	External ID	10003@domain.com
3277189	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 14:00:55	External ID	10003@domain.com
3277190	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 14:00:55	External ID	10003@domain.com
3277191	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 14:00:55	External ID	10003@domain.com
3277192	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 14:00:55	External ID	10003@domain.com
3277174	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 14:00:45	External ID	10003@domain.com
3277175	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 14:00:45	External ID	10003@domain.com
3277176	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 14:00:45	External ID	10003@domain.com
3277177	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 14:00:45	External ID	10003@domain.com
3277178	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 14:00:45	External ID	10003@domain.com
3277181	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1003	NULL	NULL	2023-07-11 14:00:44	External ID	10003@domain.com
3277182	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-07-11 14:00:44	External ID	10003@domain.com
3277166	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 14:00:35	External ID	10003@domain.com
3277167	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 14:00:35	External ID	10003@domain.com
3277168	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 14:00:35	External ID	10003@domain.com
3277169	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 14:00:35	External ID	10003@domain.com
3277170	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 14:00:35	External ID	10003@domain.com
3277198	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1004	NULL	NULL	2023-07-11 14:00:28	External ID	10003@domain.com
3277199	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-07-11 14:00:28	External ID	10003@domain.com
3277193	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 14:00:25	External ID	10003@domain.com
3277194	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 14:00:25	External ID	10003@domain.com
3277195	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 14:00:25	External ID	10003@domain.com
3277196	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 14:00:25	External ID	10003@domain.com
3277197	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 14:00:25	External ID	10003@domain.com
3277179	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1002	NULL	NULL	2023-07-11 14:00:22	External ID	10003@domain.com
3277180	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-07-11 14:00:22	External ID	10003@domain.com
3277183	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 14:00:15	External ID	10003@domain.com
3277184	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 14:00:15	External ID	10003@domain.com
3277185	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 14:00:15	External ID	10003@domain.com
3277186	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 14:00:15	External ID	10003@domain.com
3277187	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 14:00:15	External ID	10003@domain.com
3277171	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1001	NULL	NULL	2023-07-11 14:00:05	External ID	10003@domain.com
3277172	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-07-11 14:00:05	External ID	10003@domain.com
3277173	NEF	1	EVENT_TRIGGERED	UE_REACHABILITY	reachabilityT...	float	NULL	NULL	NULL	2023-07-11 14:00:05	External ID	10003@domain.com
3277160	NEF	1	EVENT_TRIGGERED	LOSS_OF_CONNECTIVITY	lossOfConne...	float	NULL	7	NULL	2023-07-11 14:00:00	External ID	10003@domain.com
3277155	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 13:59:51	External ID	10003@domain.com
3277156	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 13:59:51	External ID	10003@domain.com
3277157	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 13:59:51	External ID	10003@domain.com
3277158	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 13:59:51	External ID	10003@domain.com
3277159	NEF	1	PERIODIC	QOS_NOT_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 13:59:51	External ID	10003@domain.com
3277141	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	cellId	string	AAAAA1001	NULL	NULL	2023-07-11 13:59:43	External ID	10003@domain.com
3277142	NEF	1	EVENT_TRIGGERED	LOCATION_REPORTING	enodeBId	string	AAAAA1	NULL	NULL	2023-07-11 13:59:43	External ID	10003@domain.com
3277134	NEF	1	PERIODIC	QOS_GUARANTEED	ipv4Addr	string	10.0.0.3	NULL	NULL	2023-07-11 13:59:41	External ID	10003@domain.com
3277135	NEF	1	PERIODIC	QOS_GUARANTEED	appliedQosRef	string	None	NULL	NULL	2023-07-11 13:59:41	External ID	10003@domain.com
3277136	NEF	1	PERIODIC	QOS_GUARANTEED	ulDelays	float	NULL	0	NULL	2023-07-11 13:59:41	External ID	10003@domain.com
3277137	NEF	1	PERIODIC	QOS_GUARANTEED	dlDelays	float	NULL	0	NULL	2023-07-11 13:59:41	External ID	10003@domain.com
3277138	NEF	1	PERIODIC	QOS_GUARANTEED	rtDelays	float	NULL	0	NULL	2023-07-11 13:59:41	External ID	10003@domain.com

Figure 88 All NEF events received by Network App are stored on IoT Collector

All NEF events received by Network App are stored on IoT Collector is described in the Figure 88 above.

End-to-end connectivity and verification test was performed successfully in both rounds of integration.

As a result of successful deployment, a YouTube movie was produced and published on Evovled-5G YT channel: Internet Institute Network App deployment tests at the NCSRD Athens 5G platform

4.6.3 Smart Irrigation 5G Agriculture Network Application

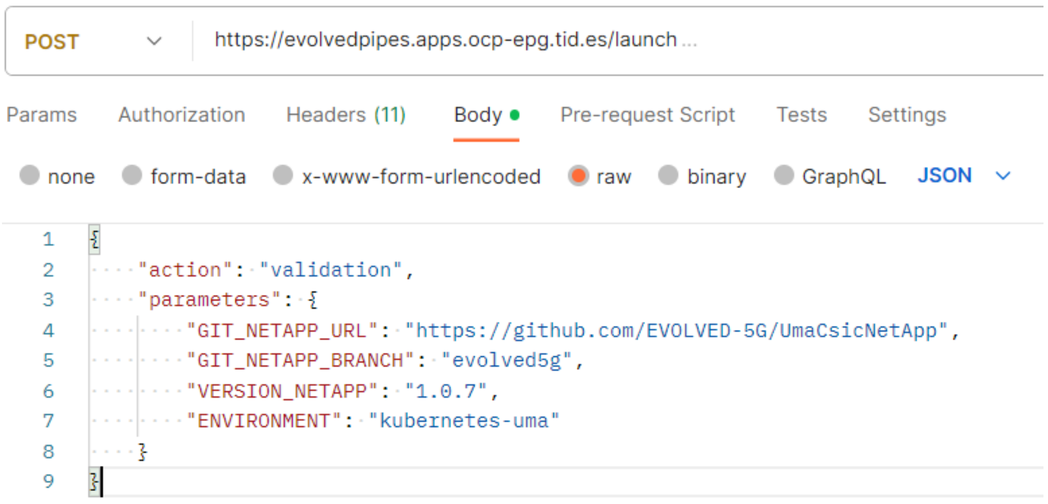
Deployment of UMA Network App has been done into UMA Kubernetes cluster. The version of the applications used were:

- NEF: 2.2.2

- CAPIF: 3.1.2
- SDK: 1.0.8

The code of the NetApp can be seen in the following repository: https://github.com/EVOLVED-5G/UmaCsicNetApp/tree/k8s

The NetApp with the first use case was working. To check that it is correct, the validation pipeline is applied to it. This pipeline consists of making a post request indicating the arguments GIT_NETAPP_URL, GIT_NETAPP_BRANCH, VERSION_NETAPP, ENVIRONMENT as shown in Figure 89 below. The result of the pipeline with SDK version 1.0.7 in the UMA and Athens environments is SUCCESS.

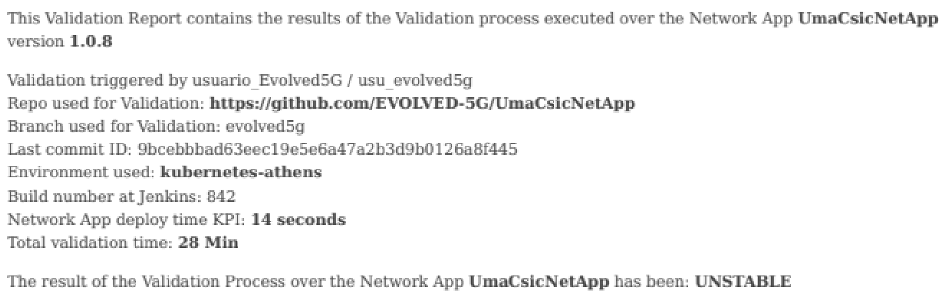


```
POST https://evolvedpipes.apps.ocp-epg.tid.es/launch ...
```

```
{
    "action": "validation",
    "parameters": {
        "GIT_NETAPP_URL": "https://github.com/EVOLVED-5G/UmaCsicNetApp",
        "GIT_NETAPP_BRANCH": "evolved5g",
        "VERSION_NETAPP": "1.0.7",
        "ENVIRONMENT": "kubernetes-uma"
    }
}
```

Figure 89 Validation pipeline in the Malaga and Athens clusters with SDK version 1.0.7

With SDK version 1.0.8 we get UNSTABLE status in the report which we received in the validation email as shown in Figure 90. However, in SonarQube (as depicted in Figure 91) the result was successful. UMA team contacted with Telefonica team but has not received a response. With version 1.0.7 the same thing happened but the Telefonica team fixed it. UMA team realized that there may be files that have not been deleted from previous validation attempts.

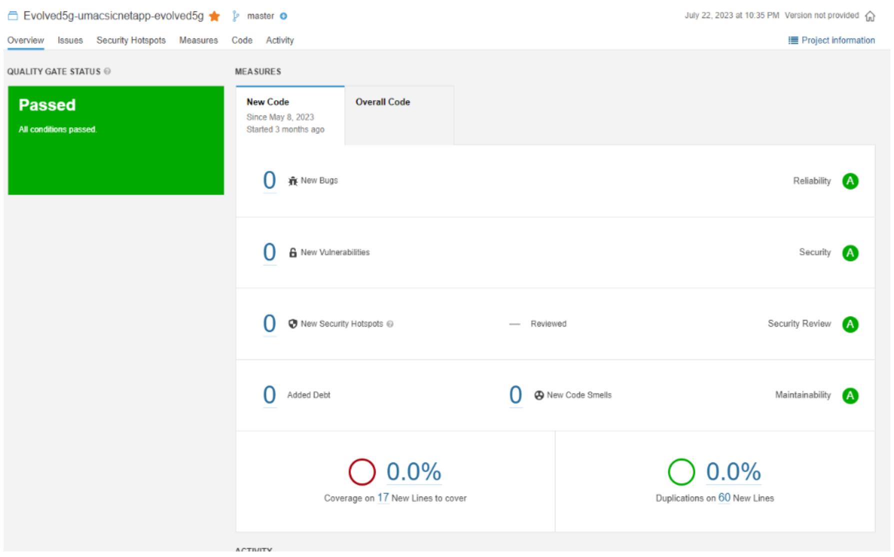


This Validation Report contains the results of the Validation process executed over the Network App **UmaCsicNetApp** version **1.0.8**

Validation triggered by usuario_Evolved5G / usu_evolved5g
Repo used for Validation: **https://github.com/EVOLVED-5G/UmaCsicNetApp**
Branch used for Validation: evolved5g
Last commit ID: 9bcebbbad63eec19e5e6a47a2b3d9b0126a8f445
Environment used: **kubernetes-athens**
Build number at Jenkins: 842
Network App deploy time KPI: **14 seconds**
Total validation time: **28 Min**

The result of the Validation Process over the Network App **UmaCsicNetApp** has been: **UNSTABLE**

Figure 90 Validation pipeline on Athens cluster and I get UNSTABLE status with SDK version 1.0.8.



Evolved5g-umacsicnetapp-evolved5g master July 22, 2023 at 10:35 PM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project information

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code — Since May 8, 2023 — Started 3 months ago | Overall Code

0 New Bugs		Reliability A
0 New Vulnerabilities		Security A
0 New Security Hotspots	— Reviewed	Security Review A
0 Added Debt	0 New Code Smells	Maintainability A

0.0% Coverage on 17 New Lines to cover

0.0% Duplications on 60 New Lines

Figure 91 SonarQube validation NetApp with SDK version 1.0.8.

In the future, an attempt will be made to add to the NetApp a camera that captures images of the tree containing the sensors, so that it will be possible to estimate the yield that a tree can achieve. The camera to be used is a multispectral camera containing different bands as shown in Figure 92 below. Each band captures different perspectives of the tree. It will be located on a pole that will be higher than the tree. It will be focused on the top of the tree to visualize as much space as possible. The image obtained will be stored in a webcam image server and may be subjected to different filters, facilitating the possibility of recognizing the amount of fruit produced.

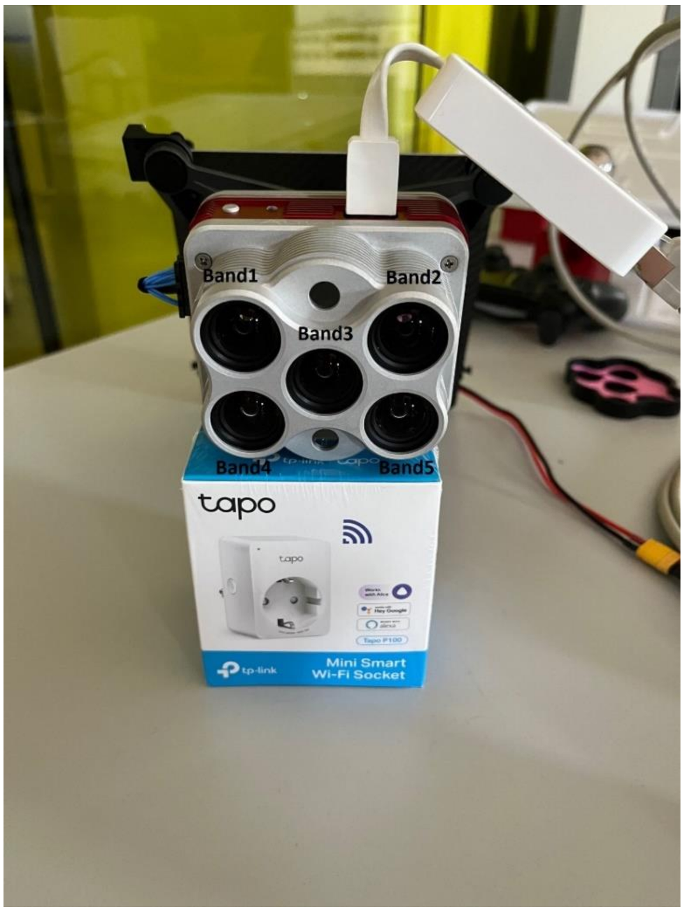


Figure 92 Micasense multispectral camera and smart plug

By adding the camera to the architecture, it can be seen in Figure 93 that the base is the same as the one we discussed before. The only part that changes is the right side. The NetApp makes a request for the camera to take a picture and store it on the webcam image server. Images can be stored locally or in the clouds. Once the photo is stored, it will be accessed from the vApp to display the result obtained.

Given the bad conditions to which the camera must be subjected, in the middle of the field at high temperatures proceeding to its heating, a Smart plug will be used as shown in Figure 92. Thus, the start and shutdown of the camera can be programmed.

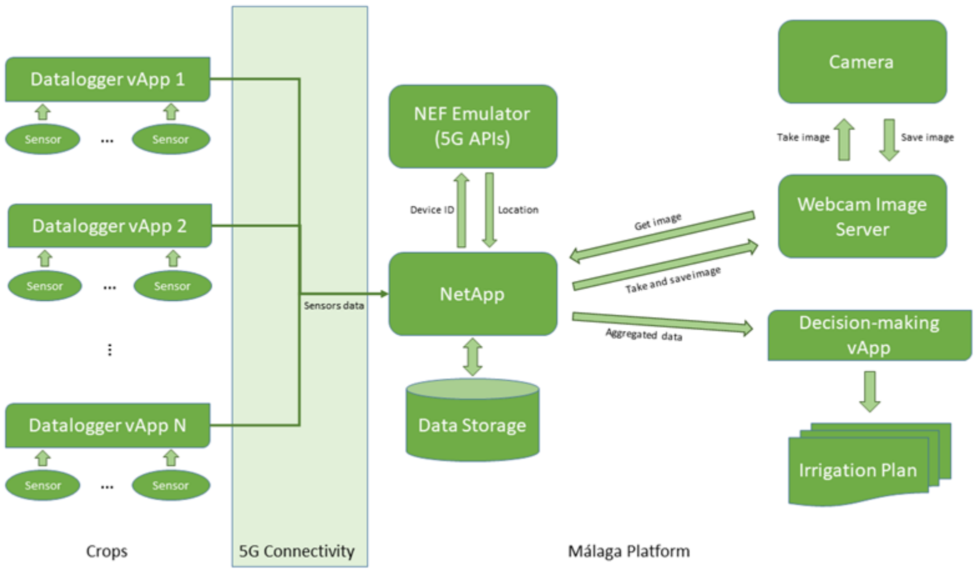
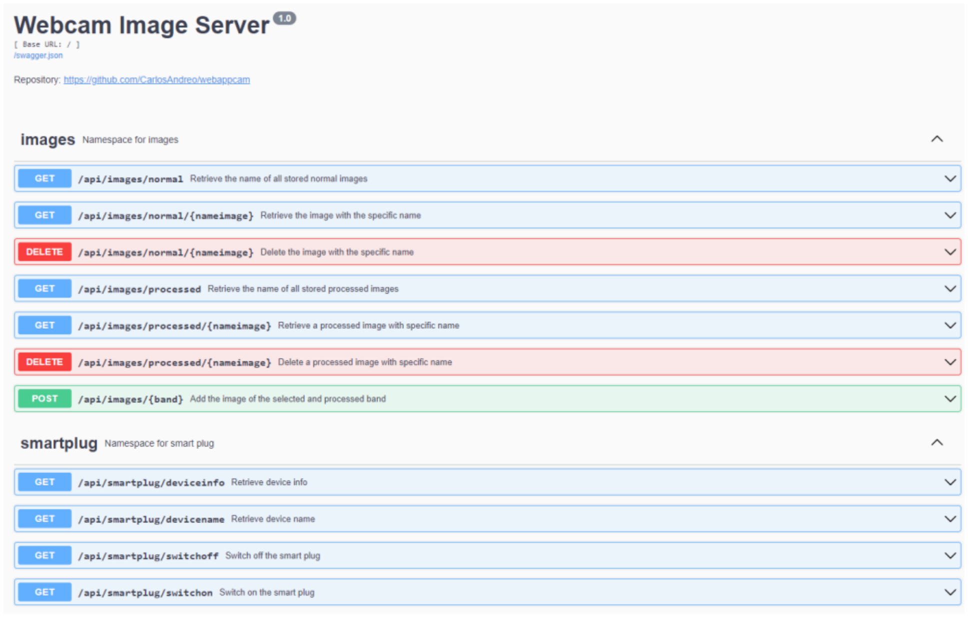


Figure 93 Adding a camera to the UMA NetApp

The webcam image server is a Docker container that deploys a REST API with Flask. The API includes endpoints for storing images and for controlling the smart plug as shown in Figure 94.


Webcam Image Server 1.0

[Base URL: /]
/swagger.json

Repository: https://github.com/CarlosAndreo/webappcam

images Namespace for images

```
GET     /api/images/normal  Retrieve the name of all stored normal images
GET     /api/images/normal/{nameimage}  Retrieve the image with the specific name
DELETE  /api/images/normal/{nameimage}  Delete the image with the specific name
GET     /api/images/processed  Retrieve the name of all stored processed images
GET     /api/images/processed/{nameimage}  Retrieve a processed image with specific name
DELETE  /api/images/processed/{nameimage}  Delete a processed image with specific name
POST    /api/images/{band}  Add the image of the selected and processed band
```

smartplug Namespace for smart plug

```
GET     /api/smartplug/deviceinfo  Retrieve device info
GET     /api/smartplug/devicename  Retrieve device name
GET     /api/smartplug/switchoff  Switch off the smart plug
GET     /api/smartplug/switchon  Switch on the smart plug
```

Figure 94 Endpoints Webcam Image Server

The multispectral camera and smart plug have been tested but have not yet been put into production along with the Smart Irrigation use case.

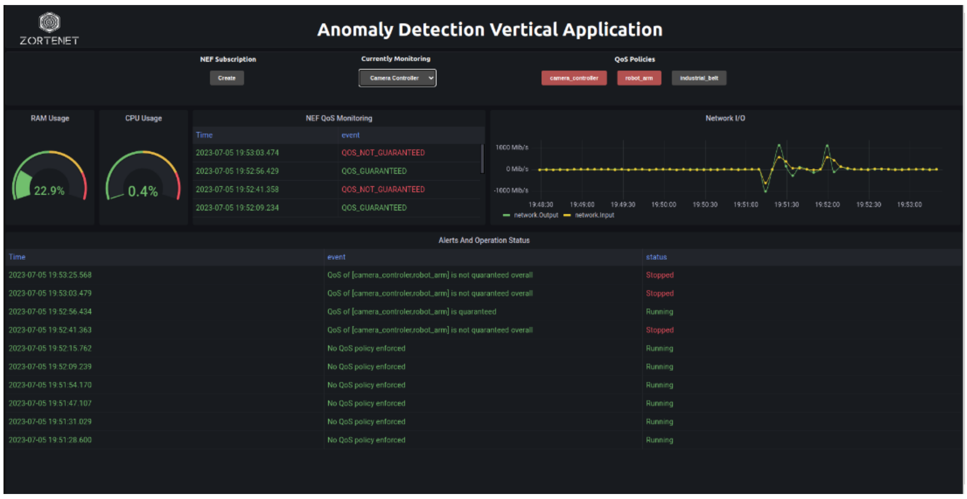
All these applications have been developed using Python 3.10 and depending on the developer's environment, Visual Studio Code or PyCharm as IDE. Also, the version of Docker used is >=24.0.2.

4.6.4 Anomaly Detection Network Application

In this second round of integration activities, many new features from the Network Application side were introduced and there was a major update in the Vertical Application in terms of User

Interface and visualization technologies. The challenge was to test this new setup to a non-local deployment and ensure that each new functionality works properly.

The latest version of Zortenet's Network Application was successfully deployed on Kubernetes cluster as shown in Figure 95 and Figure 96. The user can create subscriptions by accessing the rich dashboard of the latest vertical application and have a holistic view in real time of this monitored infrastructure. In addition, the QoS notification callbacks are successfully received and by combining them with the QoS policies of the critical devices the system can decide if the operation should continue or not.



NEF QoS Monitoring

Time	event
2023-07-05 19:53:03.474	QOS_NOT_GUARANTEED
2023-07-05 19:52:56.429	QOS_GUARANTEED
2023-07-05 19:52:41.358	QOS_NOT_GUARANTEED
2023-07-05 19:52:09.234	QOS_GUARANTEED

Alerts And Operation Status

Time	event	status
2023-07-05 19:53:25.568	QoS of [camera_controler,robot_arm] is not guaranteed overall	Stopped
2023-07-05 19:53:03.479	QoS of [camera_controler,robot_arm] is not guaranteed overall	Stopped
2023-07-05 19:52:56.434	QoS of [camera_controler,robot_arm] is guaranteed	Running
2023-07-05 19:52:41.363	QoS of [camera_controler,robot_arm] is not guaranteed overall	Stopped
2023-07-05 19:52:15.762	No QoS policy enforced	Running
2023-07-05 19:52:09.239	No QoS policy enforced	Running
2023-07-05 19:51:54.170	No QoS policy enforced	Running
2023-07-05 19:51:47.107	No QoS policy enforced	Running
2023-07-05 19:51:31.029	No QoS policy enforced	Running
2023-07-05 19:51:28.600	No QoS policy enforced	Running

Figure 95 View of the Zortenet Vertical Application

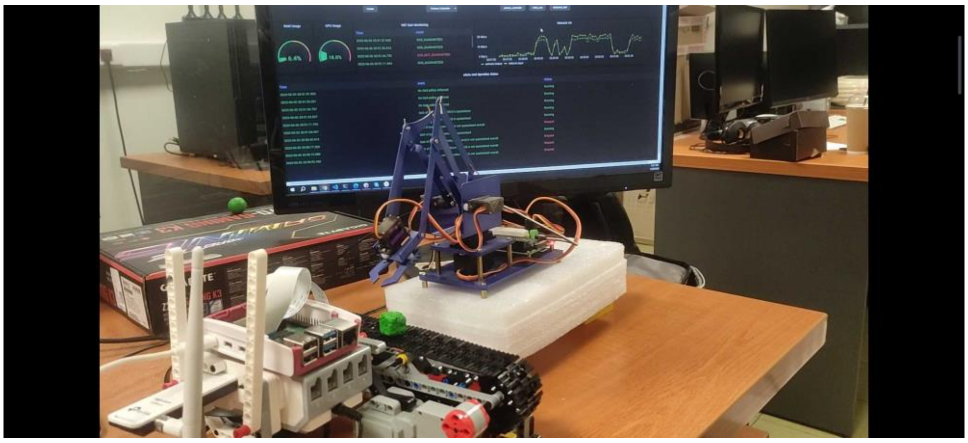
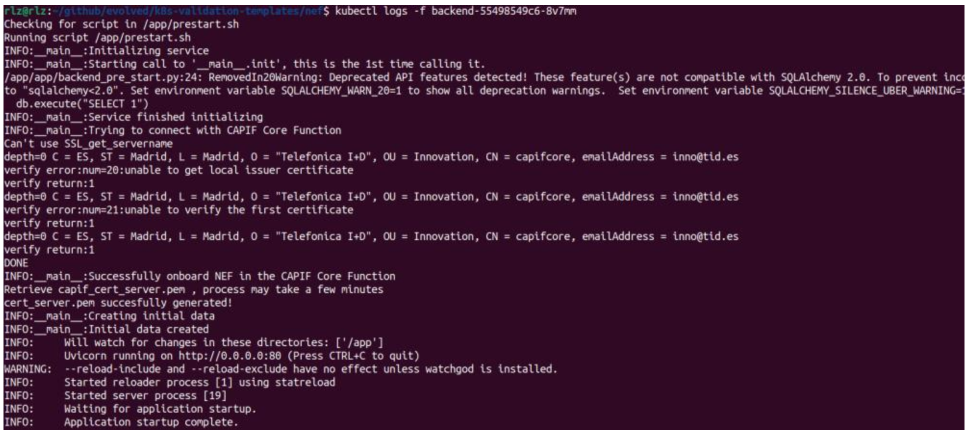


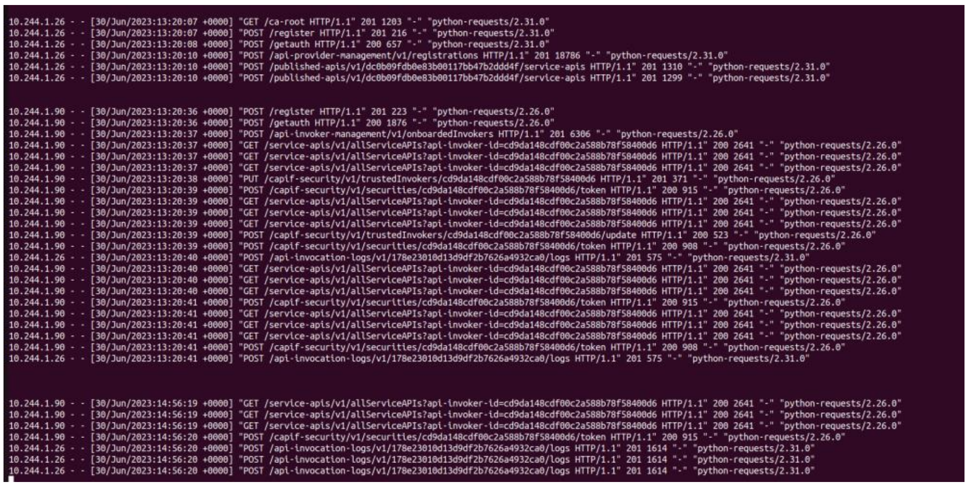
Figure 96 Footage from the 2nd round of integration activities

The 2nd round of integration activities at NCSRD premises was a success starting by the successful deployment of latest versions of NEF, CAPIF and Zortenet Network Application in the Kubernetes cluster as shown in Figure 97, Figure 98, Figure 99, Figure 100, Figure 101 below.



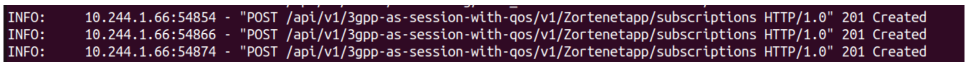
```
rlz@rlz:~/github/evolved/k8s-validation-templates/nef$ kubectl logs -f backend-55498549c6-8v7mm
Checking for script in /app/prestart.sh
Running script /app/prestart.sh
INFO:__main__:Initializing service
INFO:__main__:Starting call to '__main__.init', this is the 1st time calling it.
/app/app/backend_pre_start.py:24: RemovedIn20Warning: Deprecated API features detected! These feature(s) are not compatible with SQLAlchemy 2.0. To prevent inc
to "sqlalchemy<2.0". Set environment variable SQLALCHEMY_WARN_20=1 to show all deprecation warnings.  Set environment variable SQLALCHEMY_SILENCE_UBER_WARNING=
  db.execute("SELECT 1")
INFO:__main__:Service finished initializing
INFO:__main__:Trying to connect with CAPIF Core Function
Can't use SSL_get_servername
depth=0 C = ES, ST = Madrid, L = Madrid, O = "Telefonica I+D", OU = Innovation, CN = capifcore, emailAddress = inno@tid.es
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = ES, ST = Madrid, L = Madrid, O = "Telefonica I+D", OU = Innovation, CN = capifcore, emailAddress = inno@tid.es
verify error:num=21:unable to verify the first certificate
verify return:1
depth=0 C = ES, ST = Madrid, L = Madrid, O = "Telefonica I+D", OU = Innovation, CN = capifcore, emailAddress = inno@tid.es
verify return:1
DONE
INFO:__main__:Successfully onboard NEF in the CAPIF Core Function
Retrieve capif_cert_server.pem , process may take a few minutes
cert_server.pem succesfully generated!
INFO:__main__:Creating initial data
INFO:__main__:Initial data created
INFO:     Will watch for changes in these directories: ['/app']
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
WARNING:  --reload-include and --reload-exclude have no effect unless watchgod is installed.
INFO:     Started reloader process [1] using statreload
INFO:     Started server process [19]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Figure 97 NEF and CAPIF deployment



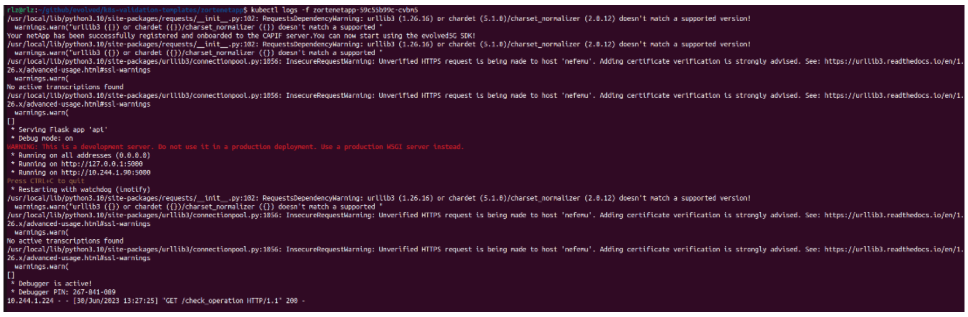
```
10.244.1.26 - - [30/Jun/2023:13:20:07 +0000] "GET /ca-root HTTP/1.1" 201 1203 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:13:20:07 +0000] "POST /register HTTP/1.1" 201 216 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:13:20:08 +0000] "POST /getauth HTTP/1.1" 200 657 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:13:20:10 +0000] "POST /api-provider-management/v1/registrations HTTP/1.1" 201 18786 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:13:20:10 +0000] "POST /published-apis/v1/dc0b09fdb0e83b00117bb47b2ddd4f/service-apis HTTP/1.1" 201 1310 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:13:20:10 +0000] "POST /published-apis/v1/dc0b09fdb0e83b00117bb47b2ddd4f/service-apis HTTP/1.1" 201 1299 "-" "python-requests/2.31.0"

10.244.1.90 - - [30/Jun/2023:13:20:36 +0000] "POST /register HTTP/1.1" 201 223 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:36 +0000] "POST /getauth HTTP/1.1" 200 1876 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:37 +0000] "POST /api-invoker-management/v1/onboardedInvokers HTTP/1.1" 201 6306 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:37 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:37 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:37 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:38 +0000] "PUT /capif-security/v1/trustedInvokers/cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 201 371 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:39 +0000] "POST /capif-security/v1/securities/cd9da148cdf00c2a588b78f58400d6/token HTTP/1.1" 200 915 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:39 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:39 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:39 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:39 +0000] "POST /capif-security/v1/trustedInvokers/cd9da148cdf00c2a588b78f58400d6/update HTTP/1.1" 200 523 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:39 +0000] "POST /capif-security/v1/securities/cd9da148cdf00c2a588b78f58400d6/token HTTP/1.1" 200 908 "-" "python-requests/2.26.0"
10.244.1.26 - - [30/Jun/2023:13:20:40 +0000] "POST /api-invocation-logs/v1/178e23010d13d9df2b7626a4932ca0/logs HTTP/1.1" 201 575 "-" "python-requests/2.31.0"
10.244.1.90 - - [30/Jun/2023:13:20:40 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:40 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:40 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:41 +0000] "POST /capif-security/v1/securities/cd9da148cdf00c2a588b78f58400d6/token HTTP/1.1" 200 915 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:41 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:41 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:41 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:13:20:41 +0000] "POST /capif-security/v1/securities/cd9da148cdf00c2a588b78f58400d6/token HTTP/1.1" 200 908 "-" "python-requests/2.26.0"
10.244.1.26 - - [30/Jun/2023:13:20:41 +0000] "POST /api-invocation-logs/v1/178e23010d13d9df2b7626a4932ca0/logs HTTP/1.1" 201 575 "-" "python-requests/2.31.0"


10.244.1.90 - - [30/Jun/2023:14:56:19 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:14:56:19 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:14:56:19 +0000] "GET /service-apis/v1/allServiceAPIs?api-invoker-id=cd9da148cdf00c2a588b78f58400d6 HTTP/1.1" 200 2641 "-" "python-requests/2.26.0"
10.244.1.90 - - [30/Jun/2023:14:56:20 +0000] "POST /capif-security/v1/securities/cd9da148cdf00c2a588b78f58400d6/token HTTP/1.1" 200 915 "-" "python-requests/2.26.0"
10.244.1.26 - - [30/Jun/2023:14:56:20 +0000] "POST /api-invocation-logs/v1/178e23010d13d9df2b7626a4932ca0/logs HTTP/1.1" 201 1614 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:14:56:20 +0000] "POST /api-invocation-logs/v1/178e23010d13d9df2b7626a4932ca0/logs HTTP/1.1" 201 1614 "-" "python-requests/2.31.0"
10.244.1.26 - - [30/Jun/2023:14:56:20 +0000] "POST /api-invocation-logs/v1/178e23010d13d9df2b7626a4932ca0/logs HTTP/1.1" 201 1614 "-" "python-requests/2.31.0"
```

Figure 98 CAPIF logs



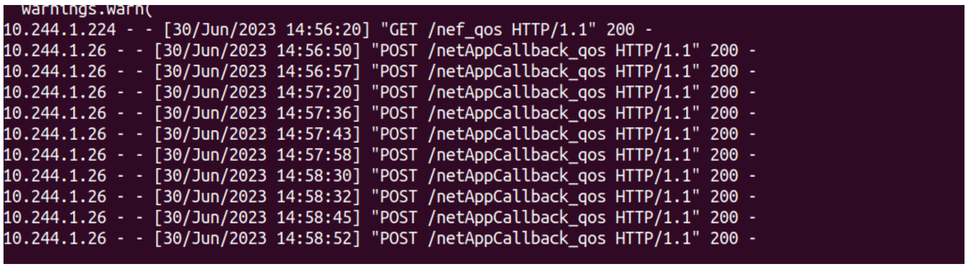
```
INFO:     10.244.1.66:54854 - "POST /api/v1/3gpp-as-session-with-qos/v1/Zortenetapp/subscriptions HTTP/1.0" 201 Created
INFO:     10.244.1.66:54866 - "POST /api/v1/3gpp-as-session-with-qos/v1/Zortenetapp/subscriptions HTTP/1.0" 201 Created
INFO:     10.244.1.66:54874 - "POST /api/v1/3gpp-as-session-with-qos/v1/Zortenetapp/subscriptions HTTP/1.0" 201 Created
```

Figure 99 Network Application Subscription to NEF



```
rlz@rlz:~/github/evolved/k8s-validation-templates/zortenetapp$ kubectl logs -f zortenetapp-59c55b99c-cvbm5
/usr/local/lib/python3.10/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
Your netApp has been successfully registered and onboarded to the CAPIF server.You can now start using the evolved5G SDK!
/usr/local/lib/python3.10/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
/usr/local/lib/python3.10/site-packages/urllib3/connectionpool.py:1056: InsecureRequestWarning: Unverified HTTPS request is being made to host 'nefemu'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.
26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
No active transcriptions found
/usr/local/lib/python3.10/site-packages/urllib3/connectionpool.py:1056: InsecureRequestWarning: Unverified HTTPS request is being made to host 'nefemu'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.
26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
[]
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.244.1.90:5000
Press CTRL+C to quit
 * Restarting with watchdog (inotify)
/usr/local/lib/python3.10/site-packages/requests/__init__.py:102: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (5.1.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a supported "
/usr/local/lib/python3.10/site-packages/urllib3/connectionpool.py:1056: InsecureRequestWarning: Unverified HTTPS request is being made to host 'nefemu'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.
26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
No active transcriptions found
/usr/local/lib/python3.10/site-packages/urllib3/connectionpool.py:1056: InsecureRequestWarning: Unverified HTTPS request is being made to host 'nefemu'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.
26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
[]
 * Debugger is active!
 * Debugger PIN: 267-841-089
10.244.1.224 - - [30/Jun/2023 13:27:25] "GET /check_operation HTTP/1.1" 200 -
```

Figure 100 Network Application onboard to CAPIF



```
 warnings.warn(
10.244.1.224 - - [30/Jun/2023 14:56:20] "GET /nef_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:56:50] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:56:57] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:57:20] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:57:36] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:57:43] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:57:58] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:58:30] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:58:32] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:58:45] "POST /netAppCallback_qos HTTP/1.1" 200 -
10.244.1.26 - - [30/Jun/2023 14:58:52] "POST /netAppCallback_qos HTTP/1.1" 200 -
```

Figure 101 QoS notifications from NEF

5 CONCLUSIONS AND NEXT STEPS

As with 5G mm wave frequencies (24-27 GHz) networks are rolling out and to be deployed in EU countries in 2023-2025, this project is crucial for Factory of the Future solutions that require data to be uploaded at speeds faster than 100Mbps. With 5G mm wave frequencies (24-27 GHz) providing coverage of up to 250m around base stations, understanding the locations from high-speed data from robots. For example: LIDAR or high-resolution cameras data streams can be transmitted is essential for commercial operations. The EVOLVED 5G project is developing a Common API ((Application Programming Interface) Framework (CAPIF) for communicating with 5G NEF (Network Exposure Function) applications, enabling industries and robotics companies to retrieve 5G network quality of service and coverage data in real-time.

The work presented in this deliverable describes in detail the final prototype of the Network Apps developed within the Factory of the Future Operations pillar in the EVOLVED-5G context, driven by Task 4.3. Moreover, detailed descriptions of the two iterations of integration tests that the Network Apps have undergone on top of the EVOLVED-5G infrastructure, both in Athens and UMA platforms, were provided. The integration activities aimed to test the functionality of the Network app when seamlessly connected with the vApp as well as evaluating the overall use case for each Network App. With the second round of integration tests, the Networks Apps of the FoF pillar have reached their final stage, interacting with the last versions of NEF and CAPIF through the SDK and communicating with their respective vApp(s). The use-cases have been validated on a real 5G network provided by the two respective infrastructures: at UMA and at NCSRD. The successful outcome of the integration testing also highlights the fact that the Network Apps reached a mature enough state that can be used by other SMEs through the EVOLVED-5G Marketplace.

Conclusions by CAFA

The CAFA Team, in cooperation with the UMA team, carried out CAFA NetMapper NetworkApp and Safety VideoLyzer vApp deployment on the UMA Kubernetes platform. Integrations and tests were carried out in several iterative rounds, which helped to start first with deployments and technical tests of applications with simpler functionality.

For CAFA Tech, as a robotics company, it was very important to develop a specific Network App NetMapper, which helps to plan and adjust the transmission of data streams from the robots based on the existing 5G communication quality. During development work and integrations, important experience was gained, which can be used to install robotics-specific applications (e.g., safety analyses applications etc.) on Kubernetes platforms managed by MNOs.

Conclusions by ININ

The integration of the ININ Network App, in collaboration with the NCSR Demokritos team, marked a significant milestone achieved in 2023. This endeavor encompassed two rounds of testing – one conducted remotely from Ljubljana to Athens, and the other physically with the presence of the ININ team at the NCSR Demokritos 5G testbed. The deployment featured multiple interconnected components, each playing a crucial role in the overall use case deployment and showcasing. Key components included the IoT Gateway, IoT Management, IoT Collector, Network App, 5G SA Network, NEF, and CAPIF.

The integration of the ININ Network App with the 5G NEF represents a pivotal step in enhancing the capabilities and reach of ININ's IoT system. As demonstrated, NEF, with its functionalities, serves as a gateway to securely expose and interact with 5G network capabilities, such as the used base stations by ININ's IoT Gateway. By facilitating the retrieval of 5G base station and core network statuses through standardized interfaces, NEF streamlines the exchange of information within the ININ Network App. The advantages of incorporating NEF into ININ's network application are manifold. It expedites service innovation by providing simplified access to evolving 5G core features, catalyzing the creation, launch, and operation of new value-added functionalities in ININ's IoT System.

In summary, as clearly demonstrated in ININ's use case, NEF's standardized approach to 5GC function significantly simplifies third-party access, enabling secure network exposure to external development teams. This, in turn, expedites controlled and automated service launches, fostering a dynamic ecosystem of innovation.

Conclusions by UMA

Both the smart irrigation use case and the new use case to be deployed based on the use of a multispectral camera have great potential in industrial agriculture. It has advantages and disadvantages. Regarding the disadvantages, it requires the use of a complex infrastructure, expensive and deployed in an environment where daytime temperatures are quite influential. Among the advantages we can highlight the water savings produced by the measurements obtained from the sensors, the detection of leaf diseases as soon as possible, an estimation of the amount of production that a tree can generate per year. The advantages that these use cases entail make it worthwhile to implement this type of complex infrastructure.

The next step is to deploy the multispectral camera use case in production. It is currently being tested and acceptable results are being obtained.

Conclusions by Zortenet

Zortenet’s Network Application aimed to provide a complete solution for monitoring and securing industrial infrastructure from anomalies. By integrating key monitoring and visualization technologies such as Influx dB and Grafana as a deployable stack a user can monitor the infrastructure as a whole and allow the system to protect itself by exploiting the Evolved5G capabilities.