H2020 5GASP Project

Grant No. 101016448

D5.2 Integration guide and API reference manual

Abstract

The purpose of this deliverable is to bring together the 5G facility owners and the NetApp developers of the consortium by using a practical, two-pronged methodology.
On the first hand, in this document facility owners provide the status of the APIs that they are currently exposed to NetApp developers so the latter shall be able to onboard, validate and certify their NetApps.
On the other hand, in this document NetApp developers provide the necessary technical details of their NetApps and also pre-installation information from their NetApps, such as network service descriptors and any other related NetApp-specific requirements. The goal is to use the information from the NetApp developers of the 5GASP consortium in order to steer facility owners on how they should refine and/or extend their offered APIs.

Document properties

Document number	D5.2
Document title	Integration guide and API reference manual
Document responsible	Lamda Networks
Document editor	Lamda Networks
Editorial team	ITAV, UOP, UNIBRIS, ODINS, ORO, YOKOGO, BLB, DRIVEU, NEO
Target dissemination level	PU
Status of the document	Final version
Version	1.0

Document history

Revision	Date	Issued by	Description
Final ToC, v0.1	24/11/2021	Lamda Networks	Final ToC agreed with all partners after revisions on the scope and purpose of the document.
v0.2	7/12/2021	Lamda Networks	Contribution to section 2 of testbeds by ORO, OdinS and UNIBRIS. Contribution to section 3 of NetApps by Lamda Networks and UNIBRIS.
v0.3	30/12/2012	Lamda Networks	Contribution to section 2 by ITAV, UoP and ININ. Contributions to section 3 by Odins, Yokogo, Neobility, BLB/DriveU, ININ. Edited Abstract and Section 1. Request for clarifications to partner's contributions to complete the draft for internal review.
v0.4	16/01/2022	Lamda Networks	Revised version after two WP5 calls where partners' inputs were further analyzed and document matching the common template for NetApps. Sent to internal reviewers: BLB and EANTC.
v1.0	19/01/2022	Lamda Networks	Final version addressing the comments of the internal reviewers.

Disclaimer

This document has been produced in the context of the 5GASP Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement number 101016448.

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
CD	Continuous Deployment
CI	Continuous Integration
C-ITS	Cooperative Intelligent Transport systems
CNF	Cloud-Native Network Function
ETSI	European Telecommunications Standards Institute
ITS	Intelligent Transport Systems
KPI	Key Performance Indicator
MAAS	Metal as a Service
NEST	Network Slice Template
NFV	Network Function Virtualization
NFVO	Network Function Virtualization Orchestrator
NODS	NetApp Onboarding and Deployment Services
NS	Network Service
NSD	Network Service Descriptor
NSaaS	Network Slice as a Service
OSM	Open Source MANO
RSU	Road Side Unit
UE	User Equipment
TMC	Traffic Management Centers
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
vOBU	Virtual On-Board Unit

1. Introduction

5GASP's goal is to cater for the needs of NetApp developers. Specifically, the project aims to offer easy-to-use APIs and tools that shall help NetApp developers (often not familiar with the 5G technology) to onboard, validate and certify their software over a state-of-the-art ecosystem of 5G facilities, that is the facilities constituting the 5GASP fabric.

To realize the above goal, 5GASP will implement a unified portal where all NetApp developers shall upload descriptions of the requirements of their NetApps. These descriptions shall include network service descriptions, descriptions of slice requirements and also descriptions of tests. The latter shall be used - amongst other reasons - to validate that specific NetApp-related APIs are met and in what degree they are met. For implementing the 5GASP portal as well as for exposing services to NetApps, the testbeds that constitute the 5GASP fabric (namely the ItAV, UoP, UNIBRIS, ORO, ININ and OdinS testbeds) must expose specific functionalities through well-defined APIs. These APIs must be commonly defined amongst the 5GASP testbeds and be ready to be offered to the NetApp developers. To that end, this deliverable describes in detail: a) the APIs that are currently offered by each one of the 5GASP testbeds; and, b) the APIs that are under development and which shall be offered initially to the NetApp developers who are members of the consortium. All such information is presented in section 2 of the document, which has been compiled by the partners that offer experimentation facilities to the NetApp developers.

In order that the currently offered and the under-development testbed APIs meet the actual requirements of all the NetApps of the consortium and equally importantly common requirements of the 3rd party NetApps which shall become members of the 5GASP community, section 3 of this deliverable has been written by the NetApp developers of the consortium. Using a common template, each NetApp developer describes the high-level architecture of the NetApp, its associated lower-level component technical descriptions and the network service descriptors of the NetApps. Using all previous descriptions, each NetApp defines which are the requested services in terms of APIs from the 5GASP portal. Finally, each developer presents the timeframe for onboarding her his/her NetApp on the 5GASP platform.

The goal is facilities be aligned with the requirements stemming from the NetApp developers at least for the use cases that will be implemented and demonstrated within the scope of the 5GASP project. To that end, section 4 concludes the deliverable with the current status API findings and a summary of ongoing work whose completion is needed in order that NetApp developers can obtain results from experimenting on the 5GASP integrated platform, using the project's tools for onboarding, validation and certification of NetApps.

2. 5GASP Testbeds' Integration APIs

This section describes the APIs that currently each testbed offers to NetApp developers. It also provides the roadmap of the ongoing and planned APIs of each testbed. These integration APIs shall be exposed to NetApp developers so that the latter can onboard, test and certify their NetApps over the 5GASP platform.

2.1 Integration APIs of ITAV testbed

ITAv's testbed provides a private network for hosting 5G Apps. Although being a private network, it is possible to make some resources (such as NetApps) publicly available which is beneficial to the 5GASP Project.

On the other hand, the ITAv testbed is not shared with any network operator or with other organizations. This caters for ITAv to complete control the resources available and to perform every change needed in the scope 5GASP.

Regarding ITAv's 5G infrastructure, it is distributed across 3 different locations, which support different use cases (see Figure 1). These locations are the following: **a)** The on-campus (University of Aveiro) site supports an energy use case, and this site is where our 5G Core is deployed. Besides, this location is also equipped with several CPEs, 5G Indoor Radios and SDRs. The secondary site is equipped with several dedicated CPEs, dedicated 5G Indoor Radios that communicates with the main site, **b)** The industrial site resulted from a partnership with an external organization and currently supports Industry 4.0 use cases. In this site we have deployed a 5G MEC infrastructure and several 5G CPEs are being used, such as an indoor 5G Radio; and, **c)** The last outdoor site can be denominated as an "outdoor location" and is also supporting an external organization, mainly for transport use cases. This site is enabled with a 5G outdoor radio that communicates with the 5G Core deployed in our main site i.e., the on-campus site.

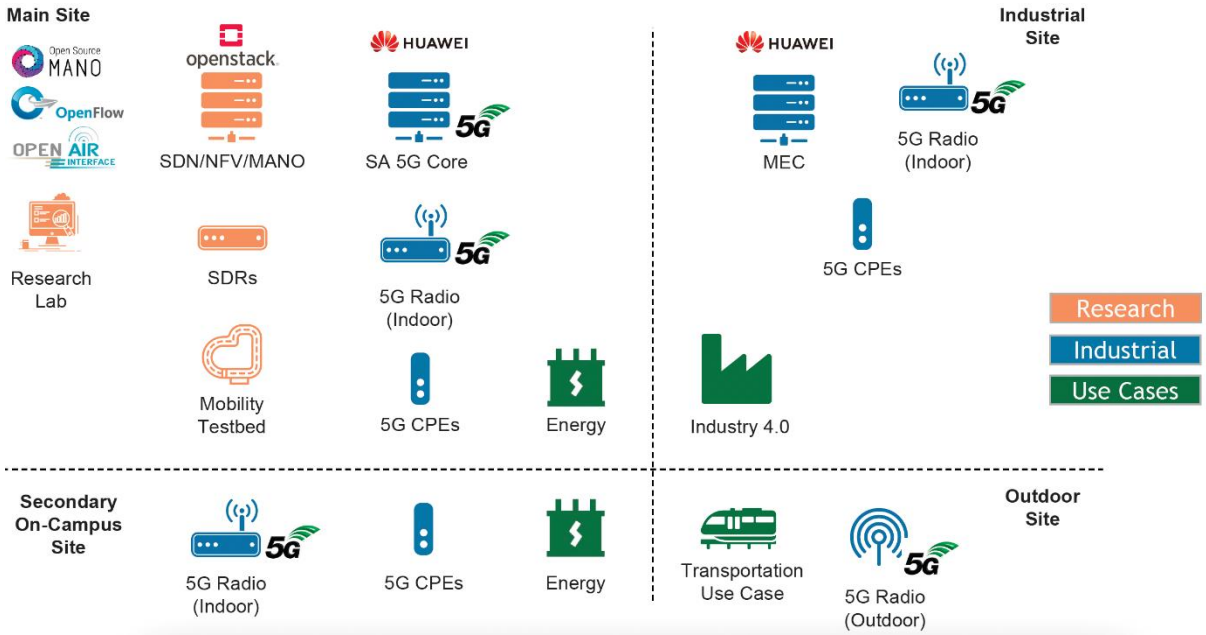

Figure 1 ITAv's locations and use cases

ITAv's testbed soon shall be able to support MEC and NFV deployments. These deployments will be supported by 5GAIner's (ITAv's 5G and AI laboratory) 5G SA Core Commercial solution and 5G MEC Commercial equipment. While slicing support in the current 5G deployment is limited to the 5G Core, it is expected that the slicing functionalities will be expanded until the second quarter of 2022.

To this date, the deployment consists of a dedicated OpenStack node that provides the hosting of VNFs for 5GASP

Currently, the testbed has no support for CNFs' hosting, although this is one of the aspects that is included in ITAv's testbed roadmap. Thus, this testbed will deploy and maintain a Kubernetes cluster to support containerized network functions which shall be demanded by some NetApps of the 5GASP project, as we will discuss in section 3 of this document.

The OpenStack node enables an Infrastructure-as-a-Service (IaaS) scenario and is used by OSM orchestrator as a VIM. ITAv's infrastructure also encompasses a VMware hyper-virtualization stack and MAAS. The latter can be adapted and made available to the 5GASP project.

Furthermore, since OpenStack is the most used tool to provide IaaS in 5G scenarios, ITAv will upgrade its resources, making available a second OpenStack node, with the same specifications with the node mentioned above. This shall enable the ITAv testbed to host more 5GASP NetApps during the use cases' implementation and demonstrations.

ITAv's testbed is also composed of several User Equipment (UE), available for the NetApp developers to test their applications. These are essentially laptops, running Kubernetes clusters, with 5G modems attached. To interact with these UE we will make available a Nova API [NOVA-API], which will be achieved using a libvirt and KVM wrapper and a VNC daemon. The endpoints to this API can be found in OpenStack's documentation [OPENSTACK-DOC]. Through this API, the NetApp developers will be able to deploy several cloud images, in the form of VMs, and receive a VNC endpoint to control these instances. It is worth considering that this API will be made available outside our 5G network, using a parallel network. The API will (i) communicate with the Kubernetes Cluster and (ii) perform configurations over the 5G modem. Thus, an additional wrapper may have to be developed to enable the communication between the API and the modem's API/CLI.

Regarding the facility's NFV Orchestrator, ITAv offers 2 different releases of OSM: (i) REL10 and (ii) REL11. Currently REL10 is being used as 5GASP's production NFVO, although, if needed, we can configure the remaining OSM instances (from different releases) to be available to the 5GASP NODS, thus enabling them to be used to orchestrate the deployment of NetApps. The OSMs expose a SOL005 interface (NBI) that is being used by the NODS to orchestrate the deployment of VNFs and NSs.

Besides all the aforementioned testbed offerings, ITAv's testbed also incorporates a CI/CD Agent and a CI/CD Manager. The latter is the entity responsible for coordinating the validation processes and it operates to serve the purposes of 5GASP's CI/CD service. This entity communicates with the CI/CD Agents deployed in the other consortium's testbeds and forwards them test suites to be performed on the NetApps deployed in these remote testbeds. Besides this, a "Test Results Visualization Dashboard" is deployed in ITAv's facility. This dashboard enables NetApp developers to gather the outputs and results of the validation of their NetApps.

Figure 2 presents the architecture of ITAv's facility.

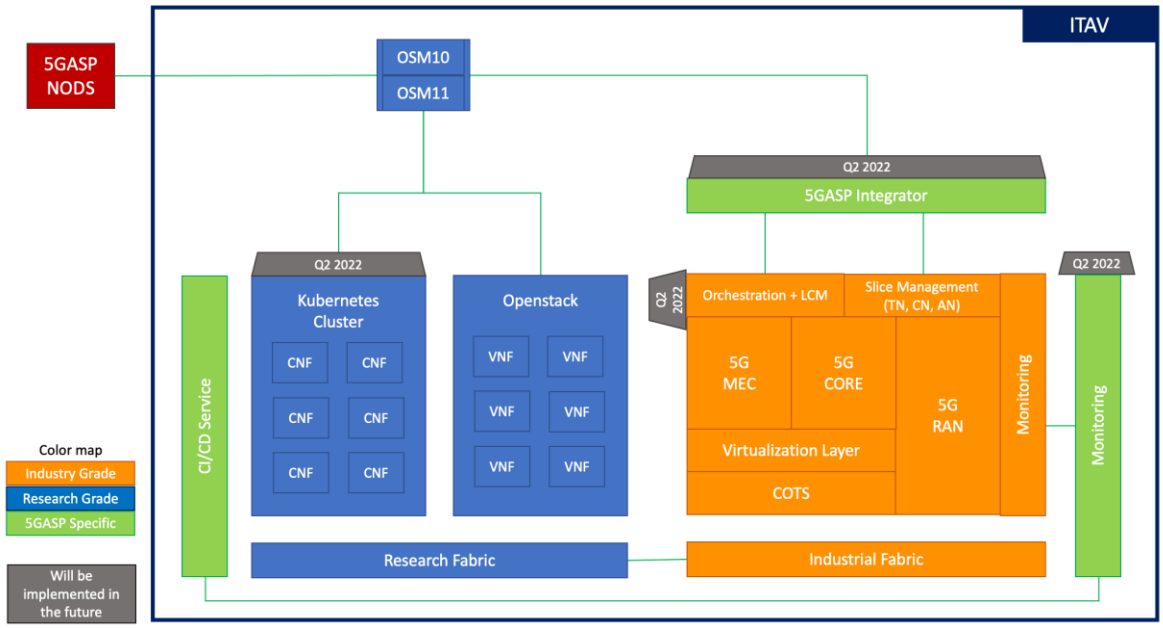

Figure 2 ITAv's Facility Architecture

Given that the testbed doesn't currently employ a monitoring stack, one of the most important goals to achieve during the first two quarters of 2022 is the design and implementation of such a monitoring mechanism. This stack will be composed of i) a Prometheus instance to store the VNF and network related metrics; and ii) Grafana, to visualize the metrics.

Figure 3 presents the current status and roadmap for the API integration in ITAv's testbed.

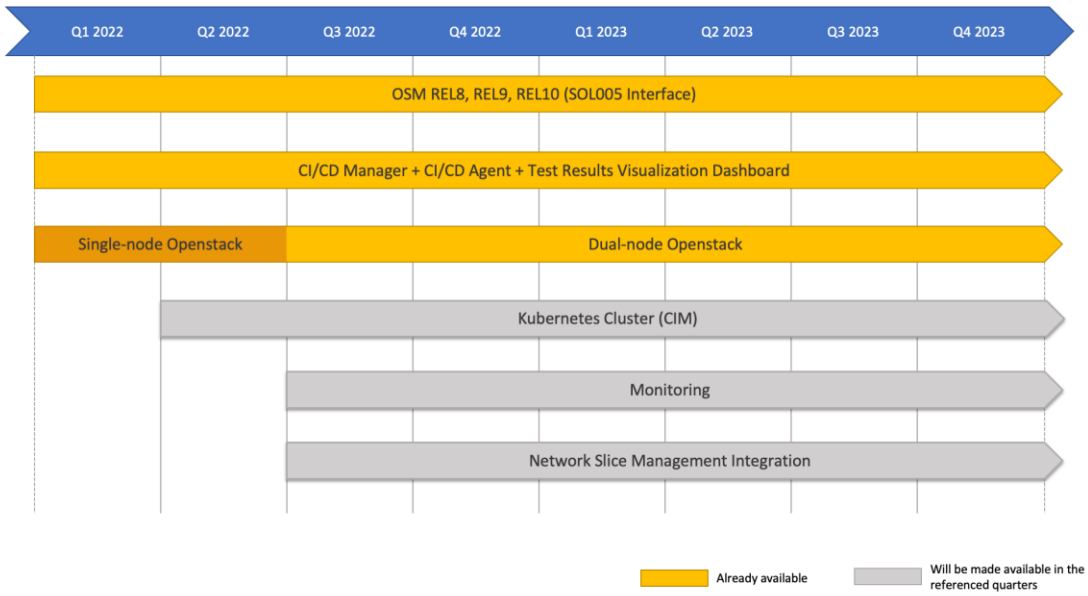

Figure 3 ITAv's facility API current status and roadmap

2.2 Integration APIs of UoP testbed

Patras 5G facility is an isolated non-public network offered for hosting 5G and IoT applications. It provides its own 5G radio and core resources, not shared with any public operator, facilitating any kind of experimentation, both indoor and outdoor (under appropriate licensing in the case of outdoor). Its overall architecture is depicted in Figure 4, along with the infrastructure elements and software tools it incorporates. The testbed is able to host core network components, as well as NFV and MEC deployments, on top of its 5G core and 5G RAN integration. Also, it maintains both an OpenStack tenant and a Kubernetes cluster for hosting NetApps consisting of VM-based NFs and Container-based NFs, respectively.

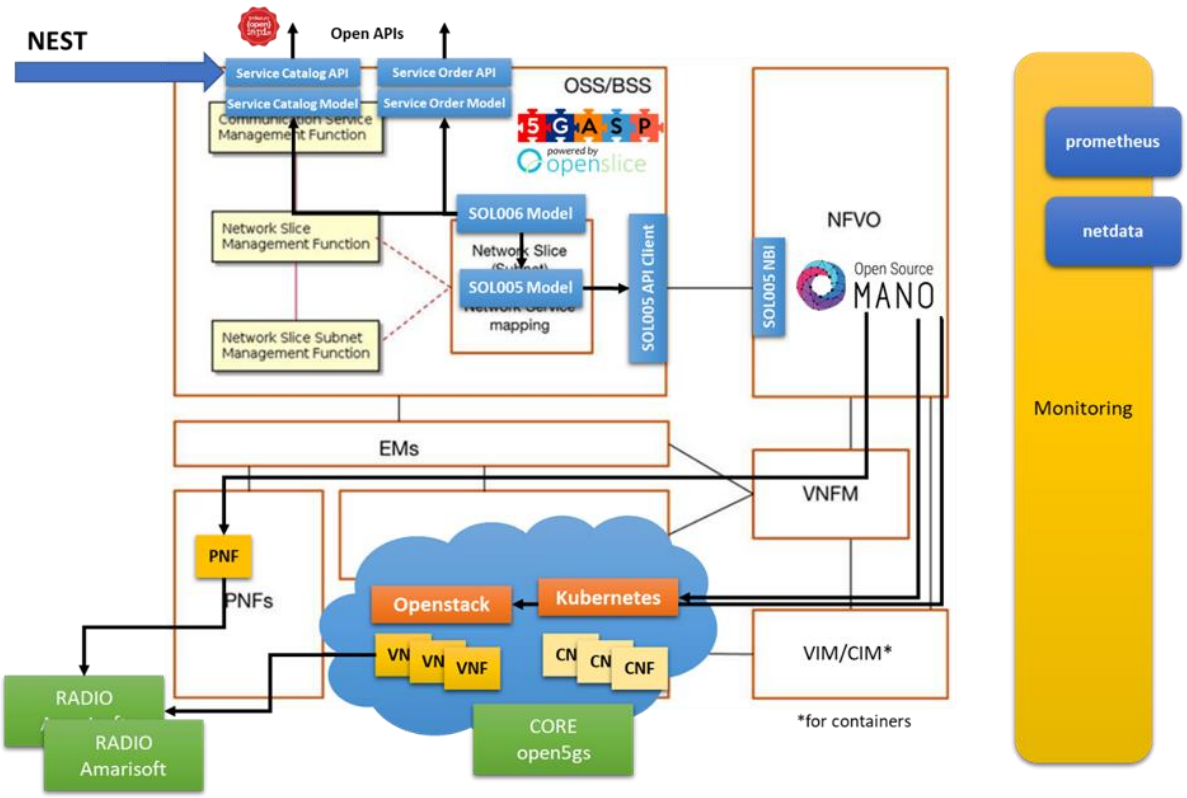

Figure 4 Patras 5G facility architecture

Patras 5G facility incorporates the following components:

- Orchestrator, based on OSM release 10, exposing its NBI via the standardized ETSI SOL005 API.
- Virtual infrastructure, managed by OpenStack and Kubernetes to deliver Infrastructure-as-a-Service (IaaS) and Containers-as-a-Service (CaaS) respectively.
- Monitoring infrastructure, providing data and metrics related to cloud infrastructure itself as well as for the deployed network functions.
- CI/CD Agent, responsible for executing the provided test against the deployed network functions.

Specifically, testbed's management and orchestration layers are supervised by OSM release 10, which exposes its NBI. The integration with 5GASP portal, as well as with other 3rd party (Operations Support System) OSS, is achieved through the ETSI SOL005 API. This interface covers VNF/NSD management, life cycle management (LCM) and FCAPS management. All the above are defined with a RESTful API performing CRUD operations.

Currently, the testbed incorporates an OSS solution. This solution is the open source project Openslice [OPENSLICE] charged to deliver Network Slices as Services (NSaas), employing TMF Opens APIs for that reason and exposing them to the developer, as seen in Figure 4.

Concerning resource virtualization, Patras 5G testbed employs OpenStack as the VIM to deploy and manage VM-based NetApps. OpenStack resources are not directly exposed and accessible internally by the Or-Vi ETSI interface. Service instantiation and management actions are delegated to OpenStack through OSM's NBI using the Or-Vi ETSI interface. Also, a Kubernetes cluster is available and utilized as the VIM for container-based NetApps. Once more, its resources are not directly exposed, but rather the cluster accepts Helm Charts from repos registered in the facilities' OSM.

Patras 5G Testbed also incorporates several UEs, available for development and experimentation purposes. The testbed offering includes several Raspberry Pi 3's running Ubuntu 20.10. Each Raspberry will integrate a Kubernetes cluster installation. The interaction with the aforementioned UE will be achieved through the Kubernetes API. Therefore, each single Raspberry, exposing Kubernetes API, can be registered in OSM essentially acting as a standalone Cloud Infrastructure Manager (CIM), further automating the onboarding, and deploying process of NetApp developers' images in the form of containers on the UE.

Furthermore, remote access to testbed's monitoring infrastructure is also offered, as depicted in Figure 5 below. Specifically, the NetApp developer or any 5G Vertical Customer can connect to a VPN service and receive real-time data and monitoring metrics from Prometheus, concerning specific monitored VNFs/PNFs, through a REST API. Historical data are also available via the offered monitoring API.

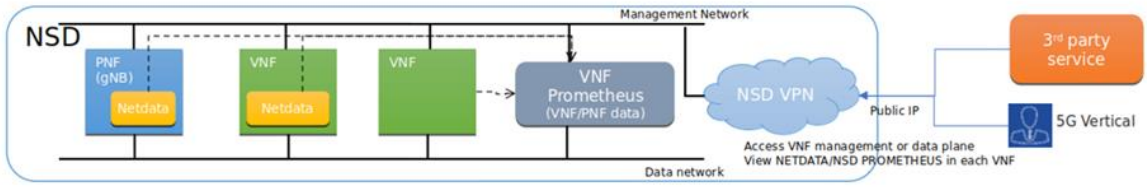

Figure 5 Patras 5G testbed Monitoring as a Service

Lastly, the testbed is scheduled to incorporate an CI/CD Agent, which is the facility's automation tool to support 5GASP's CI/CD pipeline. The CI/CD Agent setups the communication tunnel with 5GASP's CI/CD manager, receives provided test suites from the latter and executes them for the relevant NetApps being tested. Additionally, regarding network slicing management, the testbed offers slice control over 5G core domain and is currently working at extending it over RAN domain.

Patras 5G testbed's current APIs integration status and roadmap are depicted in Figure 6.

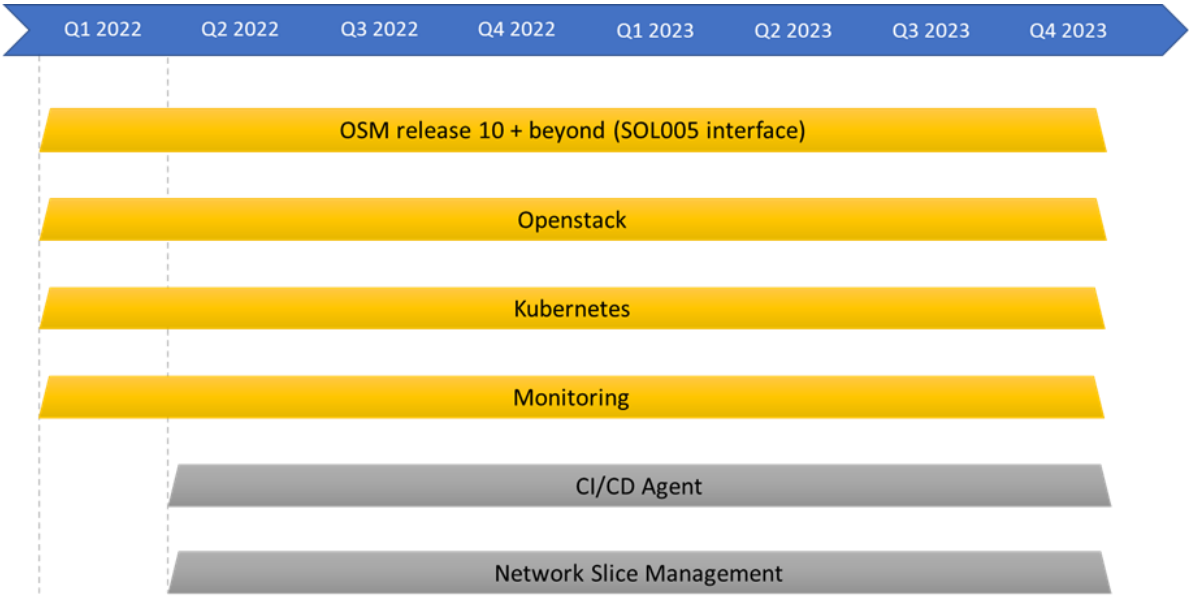

Figure 6 Patras 5G testbed's API roadmap

2.3 Integration APIs of UNIVBRIs testbed

UNIVBRIS 5GUK testbed includes different facilities, equipment and software solutions for hosting a 5GASP testbed. An overview of testbed cloud architecture is presented in Figure 7. This testbed benefits from various virtualisation platforms - VIMs (i.e., OpenStack, VMware), Kubernetes, and MAAS. These facilities and platforms are spread within the city centre of Bristol and include the Smart internet lab at the University of Bristol as a central location and two separate edges: a) WeTheCurious, in Millennium Square, and b) MShed Museum. These sites are connected using private dark fibres, and high capacity SDN-enabled switches on both core and transport networks to create an SDN fabric for site interconnection. This provides NetApp developers with the capability to develop and test their NetApps within a robust and high-end testbed while benefiting from two edge nodes separated by a few hundred meters apart and providing a MEC environment for their experimentation.

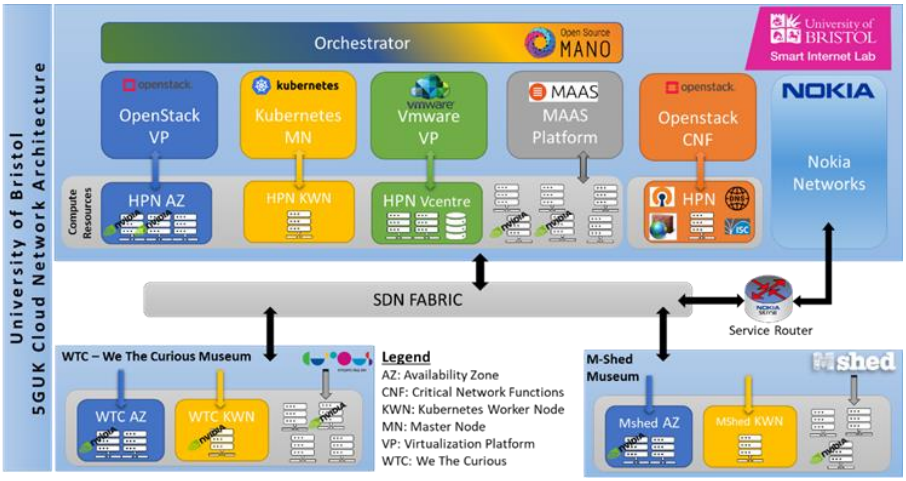

Figure 7 5GUK testbed cloud overall architecture

The testbed includes:

- **Orchestrator.** A dedicated namespace has been created for the 5GASP project at the orchestration layer and includes OSM release 10. Integration with 3rd party OSS will be realized through the northbound interface described by ETSI GS NFV-SOL005 for VNFD, NSD, LCM, Performance and Fault management interfaces. The SOL005 APIs includes the definition of GET, POST, PUT, PATCH, DELETE to support the NSD, VNFD, PNFD lifecycles and the APIs are exposed through OSM REST APIs as the NBIs.
- **Virtualisation Infrastructure.** Besides MAAS, different virtualisation platforms have been deployed to deliver IaaS (Infrastructure as a Service) and CaaS (Containers as a Service). These platforms are described in the following:
    - **OpenStack.** UNIVBRIS testbed uses OpenStack Wallaby as a VIM for ETSI MANO NVF framework to deploy VM-based NetApps. This OpenStack implementation includes three different Availability Zones (AZ), and each AZ represents one edge. Each edge includes a couple of bare-metal servers, and they are grouped as a host aggregate. In this deployment, each AZ can be added to OSM as a separate VIM. This setup uses one OpenStack controller and multiple compute nodes rather than several OpenStack implementations. The OpenStack resources for service creation are exposed to the NetApps using the OSM NBIs.
    - **Kubernetes.** This is another virtualisation platform in the UNIVBRIS testbed and may host NetApps packages as Containerized Network Functions (CNFs). The CNFs can use the NBIs exposed by the Kubernetes using the OSM South Bound Interface (SBI) to deploy the NetApps using Helm charts as described in the OSM CNF deployment procedure. Using Kubernetes API, the standard HTTP POST, PUT, DELETE, GET methods can be utilised for lifecycle operations.
    - **Other facilities.** As shown in Figure 7, other facilities are available in the UNIVBRIS testbed, Like MAAS, to implement or re-implement VIMs over the bare metals at different edges remotely and other types of VIMs like VMware to host specific types of VNFs. They may be used as a part of the UNIVBRIS 5GASP testbed, but currently, there is no integration plan for this platform within the 5GASP ecosystem.
- **Monitoring.** Suggested UNIVRBRIS testbed monitoring may be:
    - Transport network *(currently available as part of the testbed),*
    - 5G UE parameters *(availability depending on the Phone or CPE type),*
    - Regarding the support provided by NetApp 7 (discussed in section 3.7) to an enhanced NetApp, monitoring data can be sourced and exploited by the latter NetApp e.g., mobility-related monitoring such as received signal strength data, noise level, GPS, and so forth.
- **5G Network**
    - 5G SA network is another facility in the UNIVBRIS 5GASP testbed. Each edge location is equipped with dedicated gNB and RAN as well as powerful computing resources. The 5G network slicing is available based on S-NSSAI and includes SST (Service Slice Types), such as eMBB; Slice Differentiator (SD) to create multiple slices over one SST. 5QI, ARP priority

level, Downlink, and Uplink definitions for each subscriber are available as part of NSSAI and QoS. These parameters and 5G core function implementation can be exposed to the NetApps using a network creation and slicing API.

Bristol's testbed will offer a number of 5G enabled devices (2x UE & 2x CPE in SA mode) available for the NetApp users for testing and various OpenStack VMs and Kubernetes clusters to the NetApp developers. The VMs and containers can be accessed using the VNC daemon or ssh access. The UEs can communicate to the other UEs, VMs and/or containers using the radio network, internal network or using VPN in case of external network. The testbed monitoring provides open APIs to allow users to receive the testbed system and radio monitoring data (this API is currently under further development). To enable further communication, additional APIs (e.g. REST API) can be developed and required ports can be enabled by the system administrators.

Figure 8 presents the overall architecture of the 5G network slice creation and management in the UNIVBRIS 5GASP testbed. The MANO receives the required 5G slice parameters (these include slice capabilities and network slice parameters) from the 5GASP portal, through OpenSlice as we have discussed in D2.1. Based on the received slice parameters, the MANO instantiates the required VNFs using appropriate NSDs. These VNFs (including 5Gcore configuration, Transport network configuration and MEC configuration VNFs) will create end-to-end slices within the testbed to host NetApps.

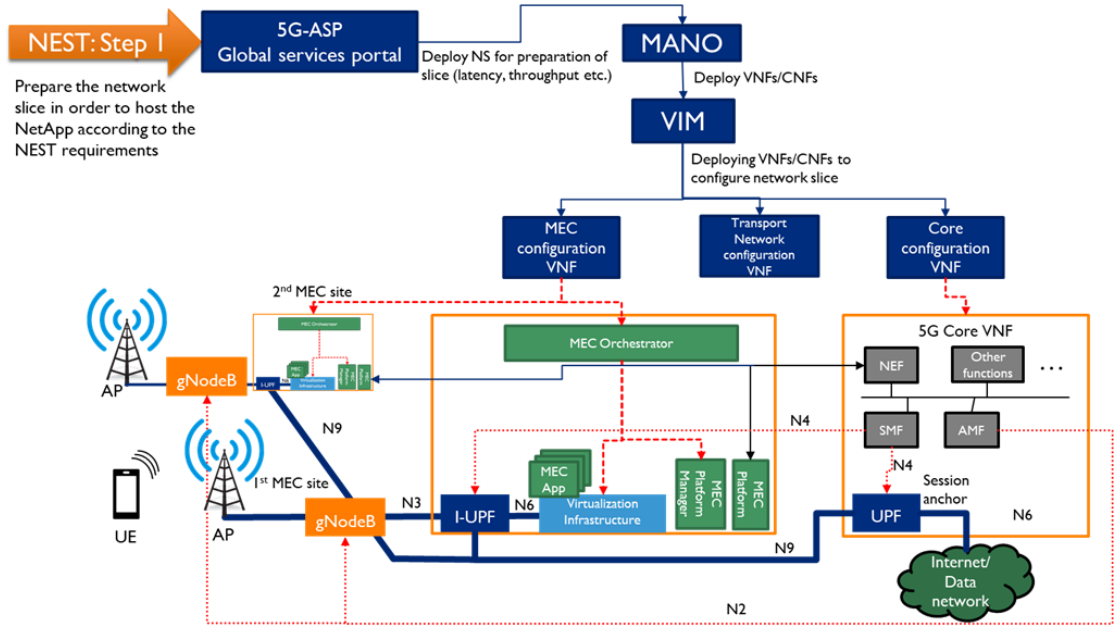

Figure 8 Overall architecture and flow of 5G network slice creation

The testbed's current APIs integration status and roadmap are depicted in Figure 9.

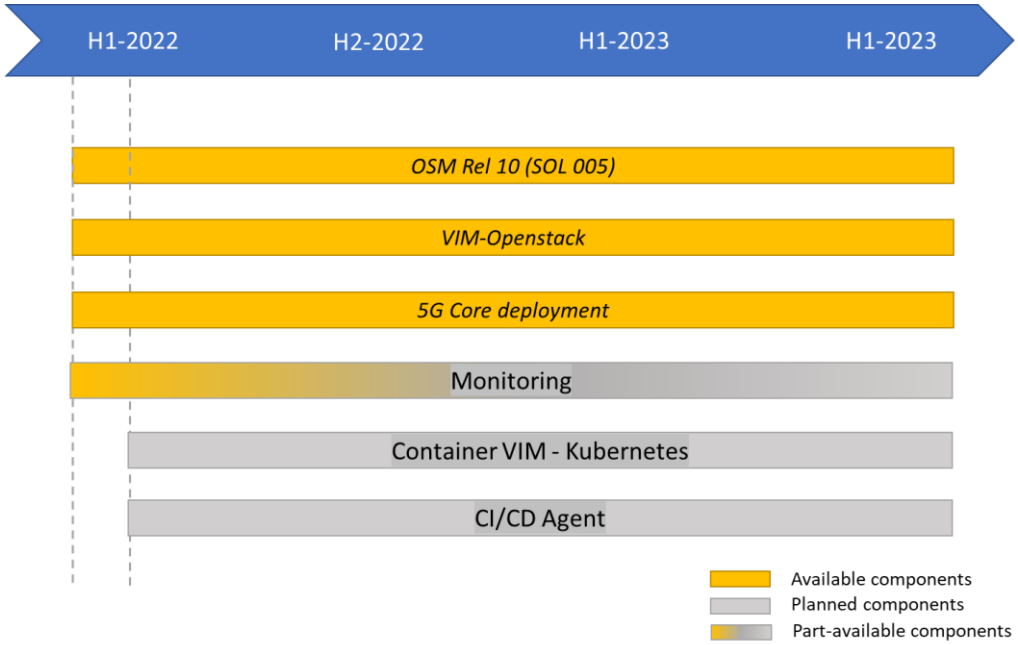

Figure 9 UNIVBRIS Testbed API roadmap

2.4 Integration APIs of ININ testbed

Testbed overview

ININ testbed is a private facility hosting a data center and a 5G mobile network in standalone (SA) mode. For the needs of 5GASP, data center capabilities will be provided on top of OpenStack-based cloud with a dedicated tenant, catering for VNF-based NetApp deployments. Additionally, a dedicated Kubernetes cluster is available to allow deployment of CNF-based components. The facility provides 5G mobile network infrastructure in SA mode using in-house built container-based images for gNB and Core Network components with the option to run additional 3rd party core components at later stages of the 5GASP project.

ININ facility includes the following components:

- Open Source MANO version 10 as the orchestrator,
- OpenStack-based cloud serving as a VIM to provide IaaS deployment model,
- Kubernetes cluster providing CaaS deployment model,
- 5G mobile network infrastructure operating in SA mode,
- Monitoring infrastructure for automated collection of data centre related metrics, NetApp component metrics (if supported by NetApp) and 5G network related KPIs,
- CI/CD Agent, responsible for automated NetApp testing,
- qMON Monitoring System providing capabilities to run end-to-end service testing of the services provided by NetApps (i.e. from the 5G UE to the 5G Core).

The architecture and integration of ININ facility into the 5GASP ecosystem is shown in Figure 10.

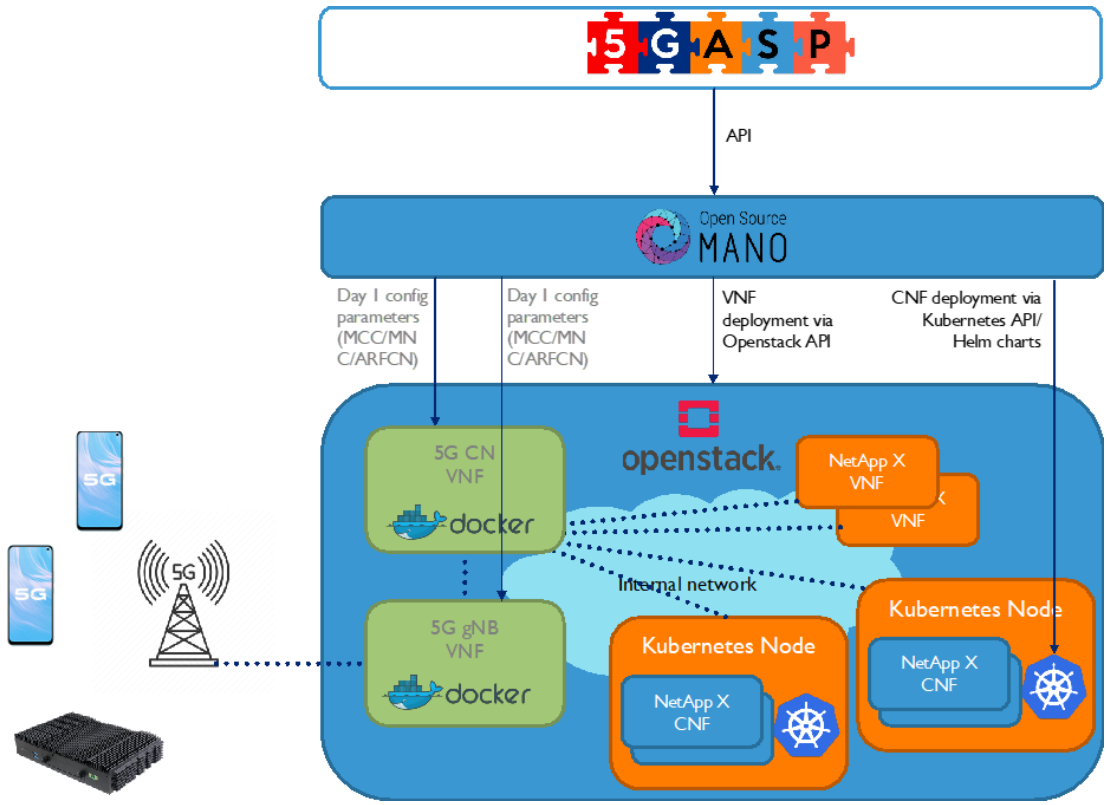

Figure 10 ININ 5GASP facility

The facility also has provisioned a 5GASP CI/CD Agent that supports executing CI/CD pipelines in the 5GASP verification process by connecting with the central 5GASP CI/CD Manager which then provides a collection of test suites and schedules the execution of the tests on the local CI/CD Agent.

qMON Monitoring System integration

For end-to-end service testing, ININ's commercial qMON monitoring solution is available to be used for the 5GASP project. qMON allows placing agents and reference servers as VNFs/CNFs. It also runs agent software as an app on Android 11+ devices. The latter will provide the capability to test true end-to-end scenarios, i.e., from the end user device and NetApp components running in the backend.

Near real-time analysis will be provided by Grafana dashboards, detailed post-analytics is also available to 5GASP project participants and can be provided based on the Tableau commercial software.

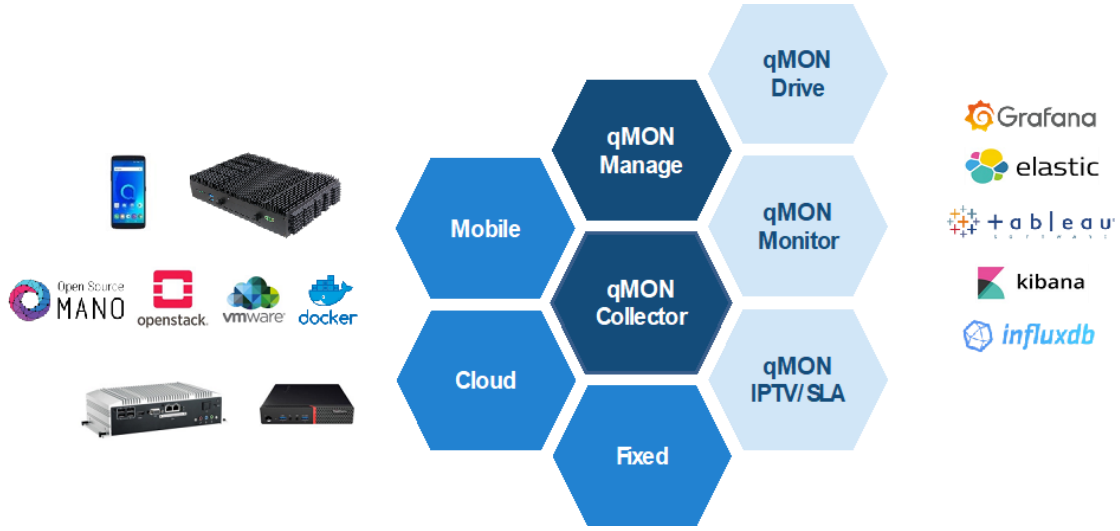

Figure 11 qMON Monitoring System

The main components of the qMON Monitoring System are the following:

- **qMON NetworkSensor:** Autonomous and distributed agents for users' and services' emulation,
- **qMON Reference Server**: Reference measurement endpoint for qMON NetworkSensor components,
- **qMON Manage:** Central cloud-based agent management and monitoring,
- **qMON Collector**: Central data collection and enrichment infrastructure,
- **qMON Insight:** Online and offline analytics for real-time status monitoring and advanced root-cause analysis.

The integration of qMON Monitoring System in the ININ's 5GASP testbed in shown in Figure 12.

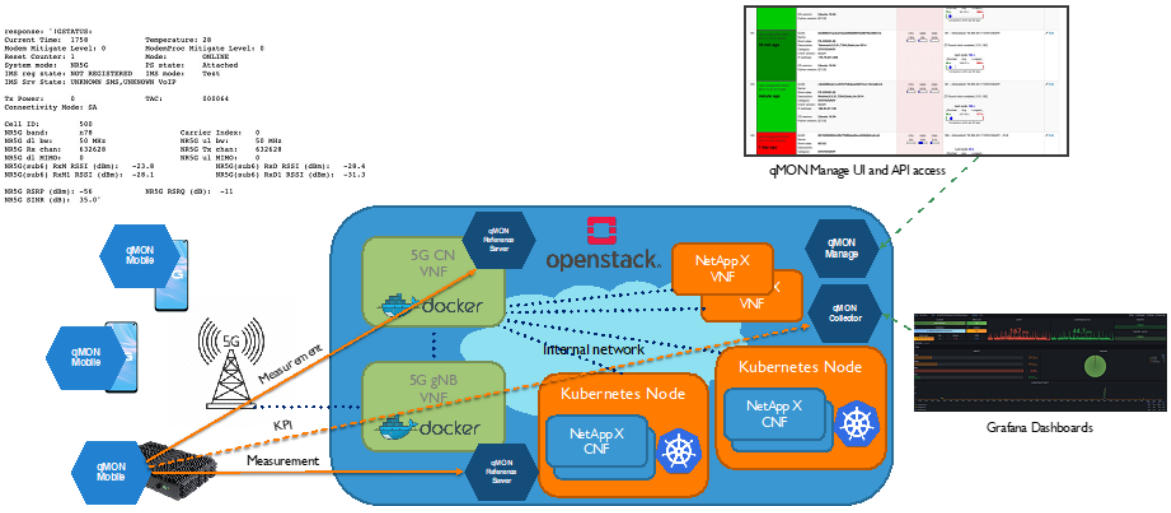

Figure 12 qMON Montitoring System integration in ININ's 5GASP facility

The qMON Monitoring System will be made available for the NetApp developers to test service and 5G radio KPIs. The measurements can be executed on-demand or they can be executed in continuous mode. qMON NetworkSensor components support the following test modules:

- Data Quality Testing (DQT) modules
  - Web
  - DNS
  - Ping
  - HTTP/FTP DL
  - FTP UL
  - Iperf

- Radio Quality Testing (RQT)
  - Full 5G KPI support for NSA/SA operation (currently supported devices are Samsung S21 and x86-based industrial PC with SieraWireless EM9191 5G modem).

Users of ININ 5GASP facility will be provided with access credentials for qMON Manage and qMON Collector components allowing them to manage the system via its UI. Additionally, some API methods can be exposed on the qMON Manage to automate 5G services testing (i.e. selection of appropriate tests' configurations from the collection of tests available). On the analytics side, the users will also get access to qMON Collector component to be able to retrieve the logs, raw KPIs and Grafana dashboards. Furthermore, all collected KPIs can be made available to users via a standard MySQL interface.

Testbed current status and timeline

Currently, data-center and orchestration capabilities are ready and deployed. Also, the facility is connected with the 5GASP portal. Based on the facility capabilities, the testbed supports the following APIs:

- ETSI SOL005 via OpenSource MANO Release 10 for NFV-based orchestration,
- OpenStack APIs (Nova, Neutron, Cinder, Glance, Keystone) for VNF-based NetApp components deployment,
- Kubernetes API for CNF-based NetApp components deployment,
- qMON Management API to support automated end-to-end network testing,
- Native MySQL DB connector to expose collected KPIs. Paragraph about exposing H/W to NetApp developers.

The planned timeline for ININ's facility API further development and integration is shown in Figure 13.

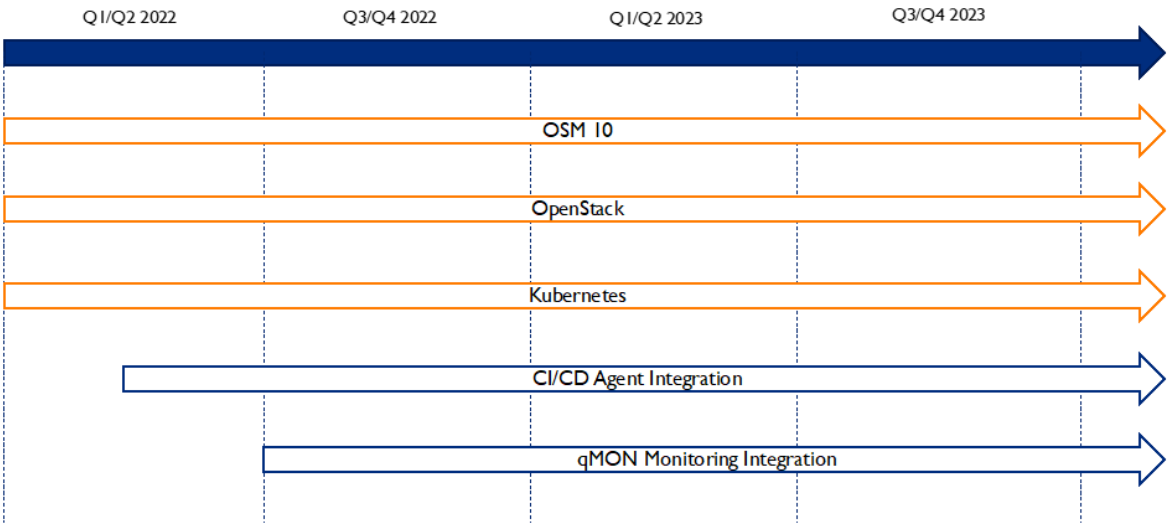

Figure 13 ININ 5GASP facility – timeline of APIs

2.5 Integration APIs of ORO testbed

ORO 5GASP testbed is described by the collection of Network Infrastructure elements and software tools to provide to each NetApp-tenant the 5G environment to develop, onboard and experiment. The NetApp experimenter benefits by the exposed APIs and Interfaces, through the 5GASP portal.

As described in Figure 14, the Bucharest facility provides an infrastructure based on:

- Orchestrator, OSM release 10 based, exposing the API Interface ETSI SOL005 standard who interprets both YAML/JSON formats. By default, this interface is working on port 9999 on the server side.
- Virtualised Infrastructure, IaaS and CaaS services, through OpenStack (VMs) and Kubernetes (CNs), the VIM being characterized by the Vi-Vnfm and Kube API Interfaces.
- Interworking framework for application integration and communication.
- Local monitoring capabilities, for network infrastructure and services, REST interface for GET parameters and KPIs.
- Network slice information and management.

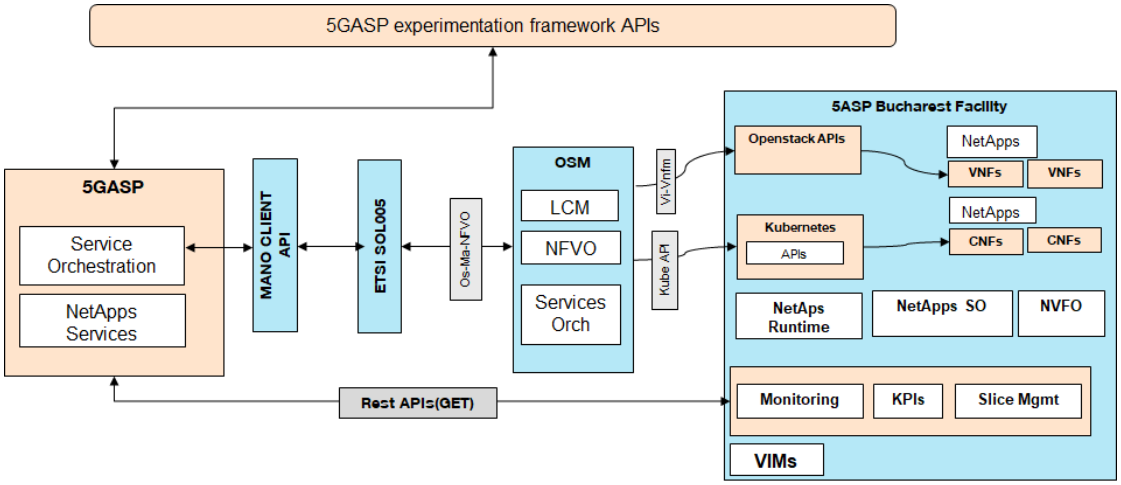

Figure 14 5GASP Bucharest API experimentation framework

The testbed is instantiated with OSM Rel10 and the OSM can be integrated through the NBI ETSI SOL005 interface standard with 3rd party OSS, providing API calls as produced by the NFVO towards the OSS, described by ETSI GS NFV-SOL005 [ETSI GS NFV-SOL005] for:

- NSD Management interface,
- LCM Management interface,
- Performance Management interface,
- Fault Management interface

In Bucharest facility, Kubernetes is the testbeds' registered VIM for container-based NetApps' deployment, managed by Helm Charts, as the core of Kubernetes' control plane is the API server. Specifically, the API server component, known as 'kube-apiserver' is exposed as a HTTP API on port 6443, to query and manipulate the state of API objects: Pods, Namespaces, ConfigMaps and Events to OSM. Through the API it is supported the retrieving, creating, updating and deleting primary resources via the standard HTTP methods: POST, PUT, DELETE, GET. The Kubernetes resources for service creation, instantiation and LCM are exposed through OSM NBIs.

Furthermore, OpenStack is the testbed VIM for ETSI MANO NFV framework for NetApps' deployment as VMs, exposed via Or-Vi interface from OpenStack through port 5000 which enables all functionalities for deployment and lifecycle management. The OpenStack resources for service creation are exposed to the NetApps through the OSM NBIs.

The facility is monitoring and collecting the network infrastructure, services and slice KPIs and it may securely expose those measurements as KPIs through a Restful API to the 5GASP NetApp developers. As the OSM is performance aware capable and fault management capable, some of the metrics can be exposed or visualised through OSM, Grafana or Prometheus and be exposed through APIs to NetApp developers, as depicted in Figure 15.

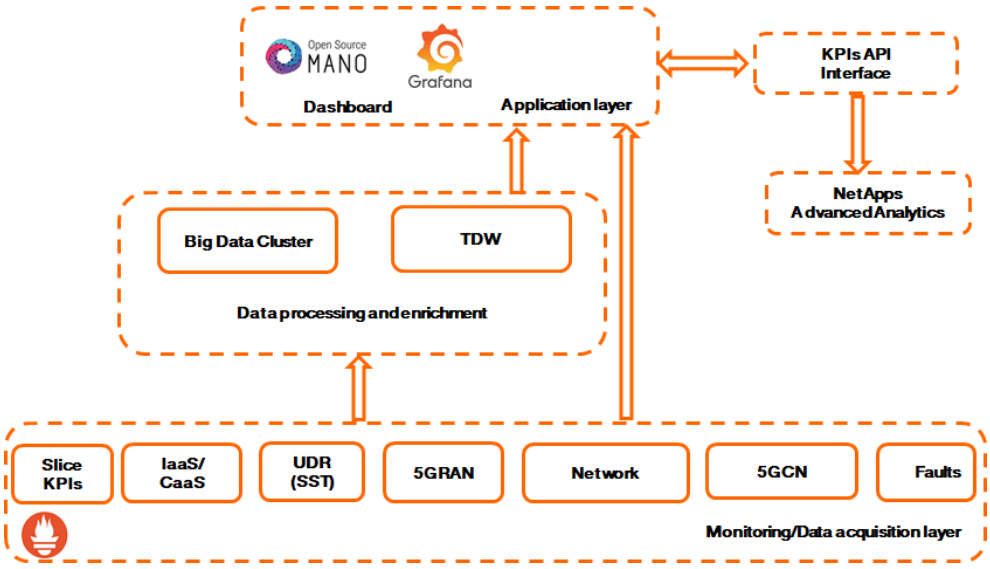

Figure 15 5GASP Bucharest testbed monitoring framework

5G network slicing is available in the Bucharest facility based on the standard Service Slice Types (SST) for eMBB and URLLC based on S-NSSAI and ensuring the 5G QoS flow. A Network Slicing API can be exposed to the 5GASP NetApps as a network management information interface related to the 5G testbed slicing capabilities (e.g. eMBB, NSSAI, SUPI provisioning) and specific network slicing parameters (e.g. DL/UL throughput).

The status of the testbed APIs and the roadmap for evolution for the needs of 5GASP project are depicted in Figure 16.

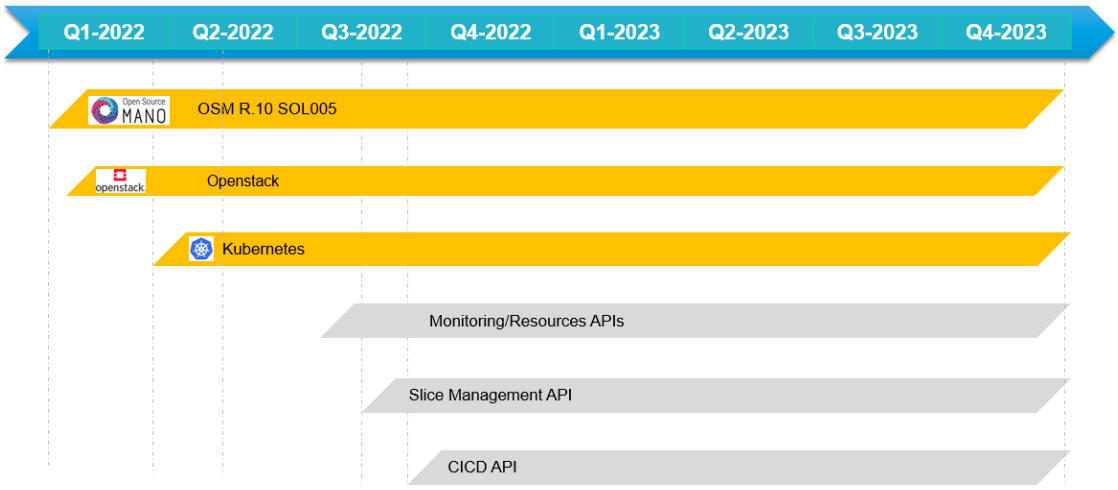

Figure 16 Bucharest facility API current status and roadmap

2.6 Integration API of OdinS testbed

Gaia-5G testbed provides the experimenters with all the infrastructure and tools required by developers to test and validate their NetApps in a real-world 5G network. All the offered tools and services shall be available through the 5GASP portal.

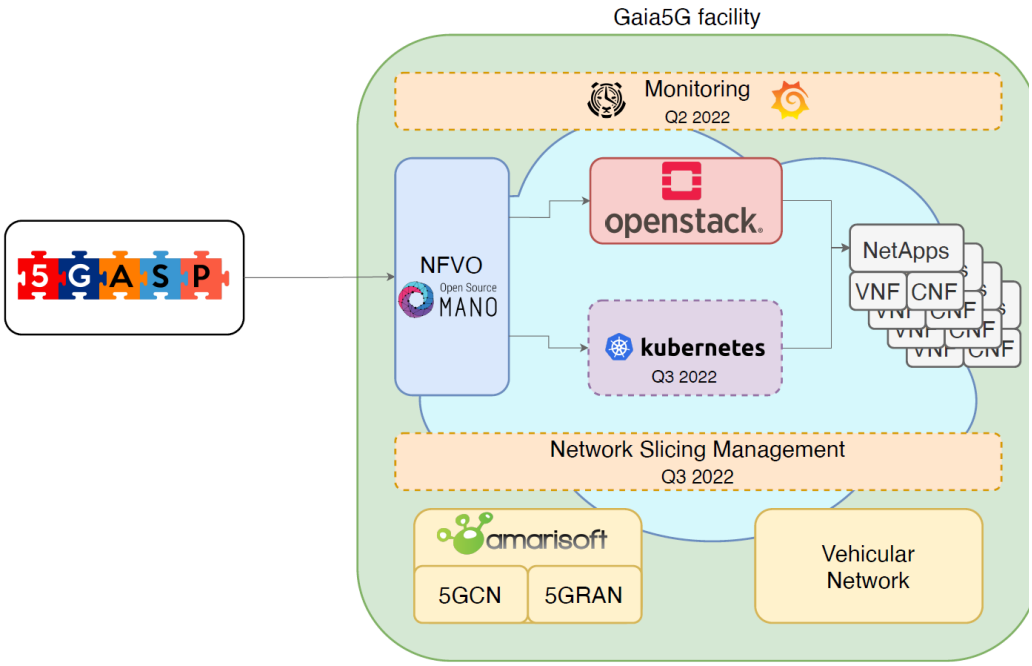

Figure 17 High level architecture of Gaia-5G facility

In the Murcia facility, OSM is the chosen NFV Orchestrator, being available in two versions: OSM version 10 and OSM version 8. Both OSM instances manage and orchestrate the deployment of VNFs and NSs. They are connected to an OpenStack substrate that serves as the VIM for the facility. Concretely, this OpenStack deployment encompasses 40 CPU cores, 125 GB of RAM and 500 GB storage.

5G provisioning is performed using the SDR technology, which is accompanied with two dedicated servers to host open source 5G cores. Besides, a commercial solution has been acquired from Amarisoft with the associated RAN hardware. Both solutions are being fully integrated for achieving a completely functional and multi-site 5G solution.

Gaia-5G also offers 802.11p networking to complete the multi-access infrastructure, providing dual role RSU/BSU capable units which can be deployed on demand alongside Espinardo campus. The API exposure of these units to the NetApp developers through the 5GASP portal is under design and once implemented, the API will be available to 5GASP NetApp developers needing such units for the NetApps.

With the above capabilities, the Murcia testbed is capable of hosting vehicular applications, which is important for the Automotive use cases of the 5GASP project. Further on in the project, 5GASP experimenters will be able to schedule time slots in which a vehicle with an OBU will be available to test their novel automotive NetApps.

In terms of APIs, Gaia-5G facility currently offers:

- NFV Orchestration, exposing a SOL005 interface based on OSM releases 8 and 10.
- Virtual infrastructure management through OpenStack, whose API is exposed to the NFVOs.
- Testing tools for NetApps, such as Jenkins and Robot Framework.

In the roadmap of the Murcia testbed, the most important APIs to be offered for 5GASP are the monitoring API and the network slicing API. The monitoring framework is currently being designed, it will leverage Grafana and the information will be stored in a dedicated time-series database to cater for efficient data retrieval. With respect to the network slicing, the facility supports a basic network slice of 10Mbps. The slice management framework is currently being designed and it will permit the establishment and control of end-to-end slices. The current status and the roadmap of the facility are depicted in Figure 18.

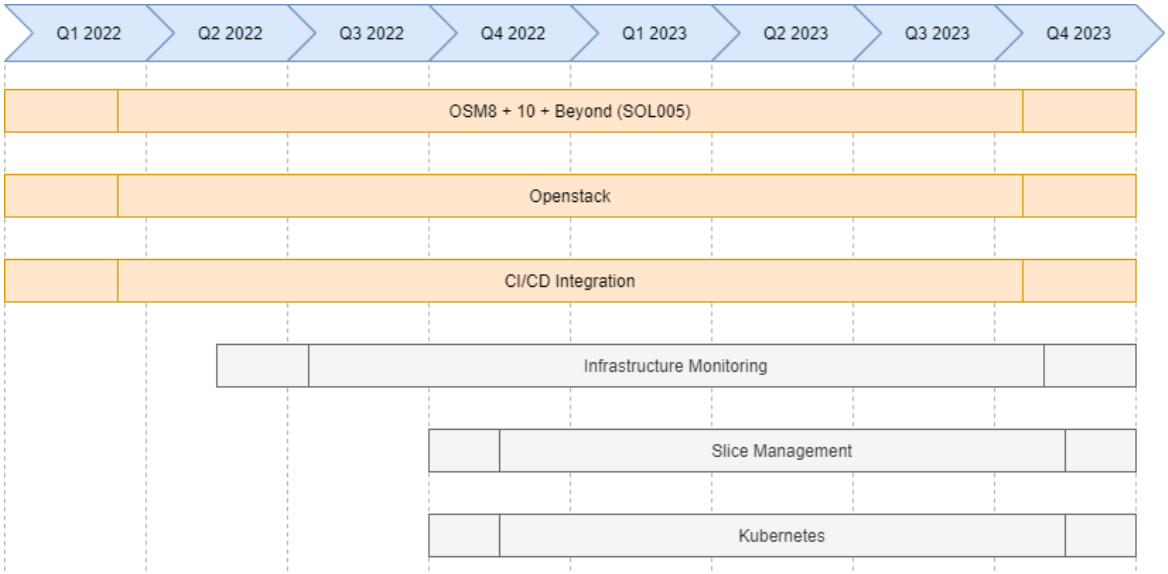

Figure 18 Gaia-5G facility API: current status and roadmap

2.7 Summary of facilities APIs for 5GASP

The following table, Table 1, summarizes the offered to NetApp developers as well as the expected common APIs of all 5GASP testbeds as discussed previously in this section.

Table 1 Summary of offered (marked with √) and expected APIs of 5GASP facilities

	ItAV	UoP	UnivBRI	ININ	ORO	OdinS
ETSI MANO SOL005	√	√	√	√	√	√
OpenStack API	√	√	√	√	√	√
Kubernetes API	expected at Q2/2022	√	expected at Q2/2022	√	expected at Q2/2022	expected at Q4/2022
CI/CD API	√	expected at Q1/2022	expected at Q2/2022	expected at Q2/2022	expected at Q4/2022	√
Monitoring API	expected at Q3/2022	√	expected at Q3/2022	expected at Q3/2022	expected at Q3/2022	expected at Q3/2022
Slicing API	expected at Q3/2022	√	√	√	expected at Q3/2022	expected at Q4/2022

Regarding security aspects of the 5GASP platform, the project has investigated the related work by the relevant WGs of ETSI. As discussed in deliverable D4.1, an analysis of the state-of-the-art in respect to security will be performed by 5GASP during the 1st half of 2022. Our current discussions target a solution based on ETSI which proposes a mechanism where each item of a VNF package has a cryptographic signature to ensure its integrity. In this way, when the onboarding process starts, then the NFV-MANO shall check the signatures and, if they're valid, then the MANO will only accept to store the packages. This way, when VNFs are instantiated, their authenticity and integrity shall be granted, as were their packages.

3. 5GASP NetApps' API requirements, current status and roadmap

This section will discuss the 5GASP NetApp's technical details and the APIs they require from the 5GASP platform. Each NetApp is described using the template defined in the form of a list in Table 2.

Table 2 Template used for NetApp analysis of section 3

A. High Level Design [HLD], explaining the architecture and of the NetApp. B. Low Level Design [LLD], explaining the design and technical details of the components of each NetApp and their interactions. C. Pre-installation information, including the network service descriptors and packaging details along with results from onboarding to laboratory environments of NetApp developers. This information shall help the profiling of NetApps that shall be used during the 5GASP onboarding and instantiation. D. Requested APIs, listing the APIs each NetApp requires from the 5GASP platform. E. Timeframe of NetApp onboarding.

3.1 NetApp 1: 'Virtual On-Board Unit provisioning'

A. High Level Design

General description

The 'Virtual On-Board Unit provisioning' (vOBU) Netapp provides virtual substitutes for physicals OBUs that aim to offload tasks and ensure low latency responses. This solution has been proven beneficial in terms of device access delay, reliability against wireless disconnections or data caches. OdinS introduces this NetApp that provides the necessary vOBUs that are instantiated at the edge of the access network with the purpose of offloading computationally-intensive tasks to the network, following the MEC approach.

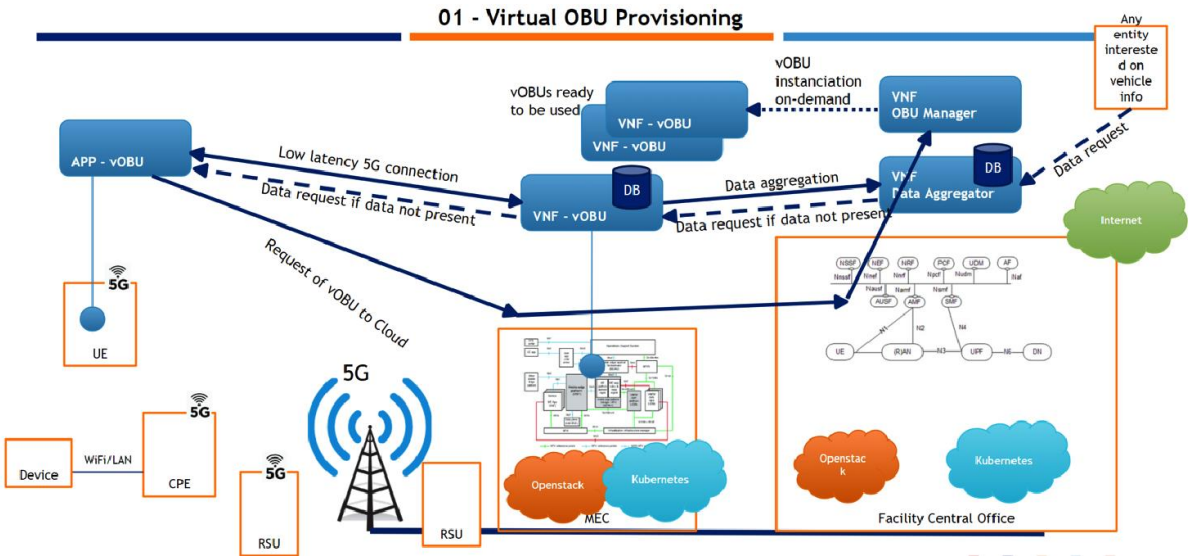

Figure 19 Virtual On-Board Unit (vOBU) provisioning NetApp infrastructure

As depicted in Figure 199, the vOBU NetApp can be used to gather logging information about
the status of the vehicle and to delegate the analytic processes to the virtual surrogate, that
could also act as a proxy for external requests of vehicle status, avoiding them to reach reach the real OBU, since the latter is the component that should be focused on processing higher priority tasks.

B. Low Level Design

The NetApp is developed in Python3, and the main packages include 'Cassandra-driver', to connect to the database, 'coapthon3' and 'http.server' to create servers that respond to the requests from other entities. CoAP is used in the communication between the OBU and the vOBU, while the other entities use HTTP for communication. The Cassandra database is divided into three different nodes that share information in real time.

The NetApp will be deployed as a Network Service using OSM. It is composed of three main components:

- vOBU: the virtual OBU itself, instantiated in the MEC to be the virtual counterpart of an already existing physical one. It will transparently answer to the requests performed to the physical OBU and it will request the OBU the information asked by external services.
- vOBU Manager: it is in charge of the instantiation of the vOBUS, and it is also in charge of their lifecycle.
- Data Aggregator: it acts as a single point of entry to the system. It will act as another level of data-caching to save network resource and answer fastly to the requests performed to the physical OBU.

C. Pre-installation information

The NetApp has been deployed locally in the Murcia testbed.

Figure 20 shows the vOBU instantiation in the OSM 8 dashboard, and

Figure 21 shows the instances running in the OpenStack VIM, note the configuration details of the IP addresses and the used flavor.

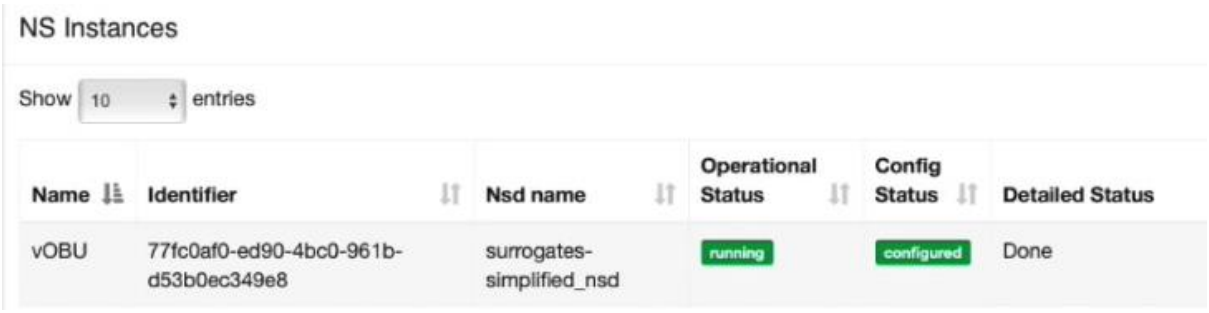

Figure 20 vOBU instantiation in OSM 8

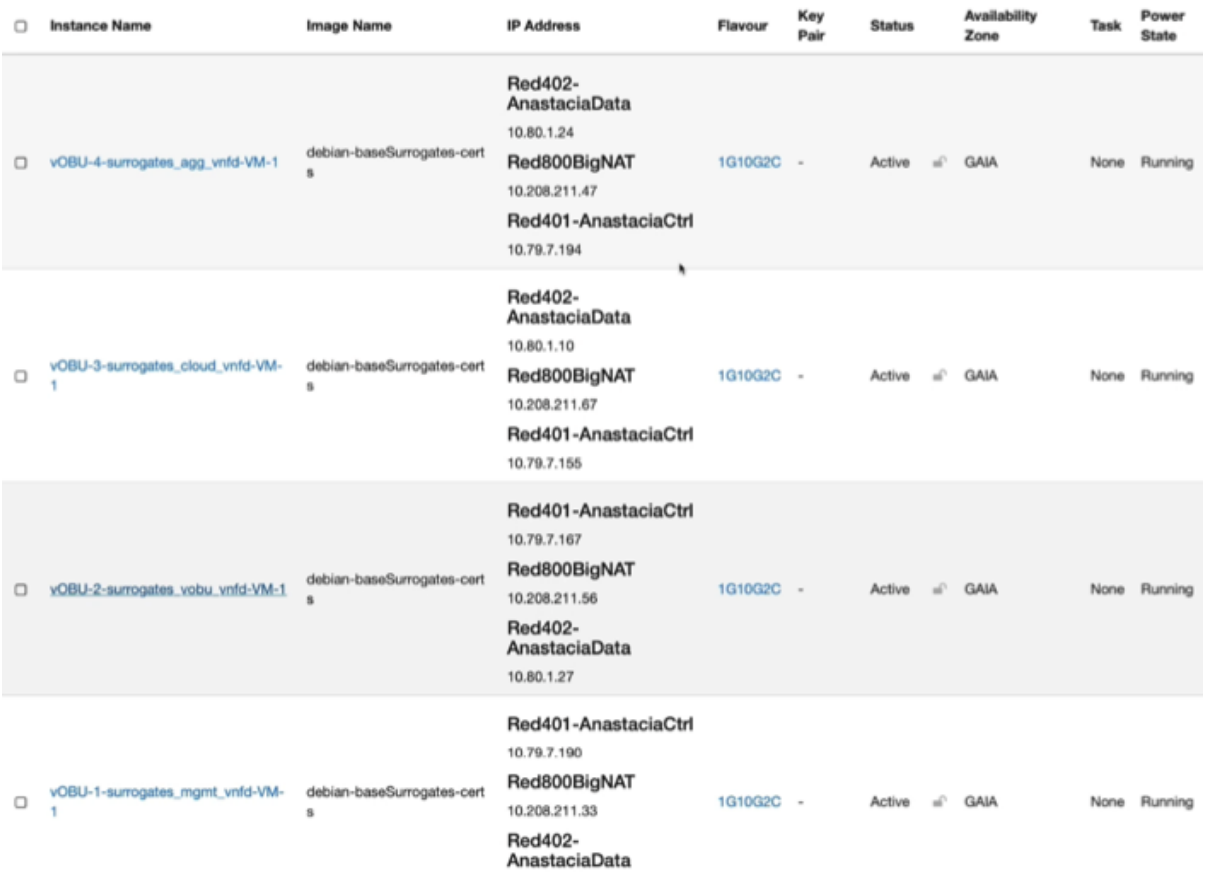

Figure 21 vOBU instances in OpenStack dashboard

D. Requested APIs

Table 3 lists the NetApp's requested APIs from the 5GASP platform.

Table 3 vOBU API requirements

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Required
Kubernetes API	Not required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

The NetApp requires a client application running in a physical OBU and the automotive communication infrastructure.

vOBU NetApp has already been deployed in the Murcia testbed, it has also been tested locally in the same testbed. Next steps consider its deployment in the Aveiro testbed also supporting the requirements of the automotive use case.

E. Timeframe

The following table describes the tasks and their timeline associated with the vOBU NetApp.

Table 4 vOBU NetApp timeframe

Task description	Task completion
NetApp initial implementation available (basic functionalities, some improvements to be implemented).	Q3 2021
NetApp VNFD/NSD implementation for OSM8 done, OSM10 version under development.	Q4 2021
GST/NEST initial version.	Q3 2021
Infrastructure tests to validate NetApp KPIs already developed. NetApp-specific tests under development.	Q1-Q2 2022
Local instance of Openslice using OSM8 and OpenStack. Testing performed manually with local Jenkins and Robot.	Q1-Q2 2022

3.2 NetApp 2 'Virtual RoadSide Unit' & NetApp3 'ITS-Station provisioning '

A. High Level Design

Both Netapps 2 and 3 have very comparable implantation similarities, where the legacy Intelligent Transport Systems (ITS) station is the development core of both NetApps. One of the differentiation keys between those two NetApps is the onboarding, deployment location, and the associated services. For NetApp2, which is edge-based ITS station

virtualization or virtualization for what is well known as a RoadSide Unit (RSU)s, which is meant to perform the localized inter-communication between RSUs and vehicles, and to perform the edge data processing (sensor data fusion, security check, etc.). Cost-wisely, it is not possible to deploy RSUs everywhere where it is still necessary to deploy the Cooperative (C-ITS) services with the same level of performance everywhere. Once latency levels are maintained for safety-based services over 5G, the virtual roadside ITS stations would allow the collection and transmission of data from both virtualized and physical ITS stations along the road with no need for wide physical deployment.

In a similar approach, and to support centralized C-ITS services, the ITS station (NetApp 3) is the deployment of a cloud-based virtualized legacy ITS station, which can also be seen as a generic ITS station implantation and dynamic C-ITS services instantiate services. In addition, NetApp 3 can also act as a Traffic Management Center (TMC) and centralized data collection of both physical or virtualized ITS stations (vehicle or roadside equipment).

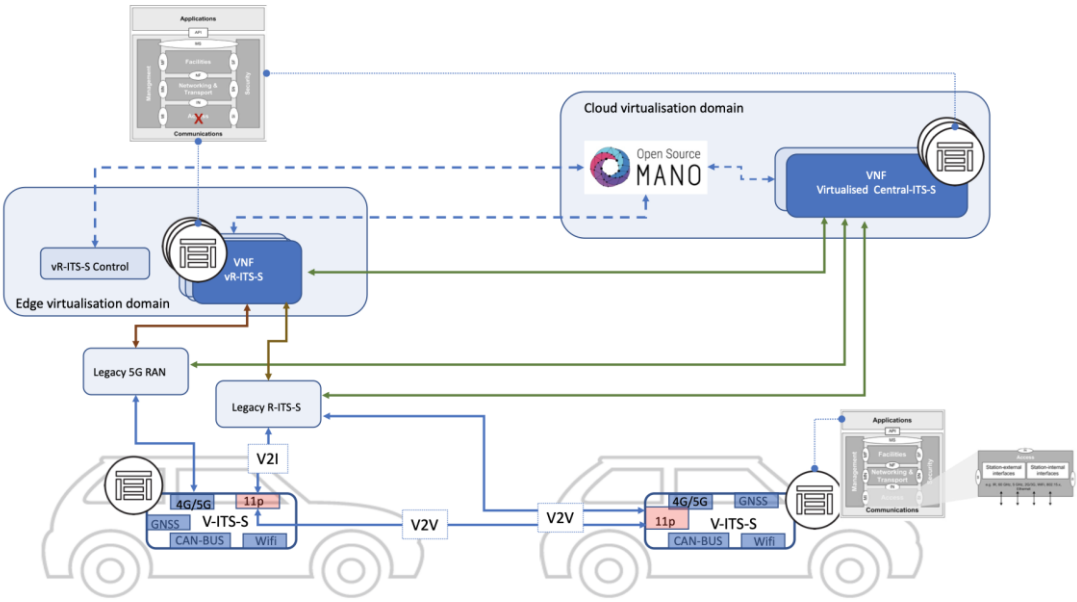

Figure 22 Virtual RoadSide Unit (vRSU) & ITS-Station provisioning NetApps infrastructure

Both NetApps promise the flexibility of C-ITS services deployment that can be overcoming many of the interoperability challenges for border-crossing scenarios and accelerate C-ITS services deployments with hybrid architectures.

B. Low Level Design

YoGoKo will use its commercially-developed middleware for the ITS station and C-ITS services. The compatible VNF implementation and packaging of the solution binary is the main challenge. OSM8 and OpenStack service account provided by Murcia site is currently in use for deployment and tests.

C. Pre-installation information

No complete deployment was achieved during 2021. The ongoing work was dedicated on building the PoC NetApp VM from scratch, providing the basic site configurations and testing scripts. Full deployment and functionalities tests are expected by Q1 of 2022 for both NetApps.

D. Requested APIs

Table 5 lists the NetApp's requested APIs from the 5GASP platform.

Table 5 NetApp 2 and NetApp 3 API requirements

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Required
Kubernetes API	Not required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

NetApp 2 and 3 are decided to be deployed at the Murcia testbed site as most of the automotive verticals. The ITAv site might be considered once the deployment is achieved at the Murcia site.

E. Timeframe

The following table describes the tasks and their timeline associated with NetApps 2 and 3.

Table 6 vRSU and ITS NetApps timeframe

Task description	Task completion
NetApp initial implementation available (basic functionalities, some improvements to be implemented)	Q3 2021
NetApp VNFD/NSD implementation for OSM8 completed. OSM10 version under development	Q1 2022
GST/NEST initial version	Q4 2021
Infrastructure tests to validate NetApp KPIs already developed. NetApp-specific tests under development.	Q2-Q3 2022
Local instance of Openslice using OSM8 and OpenStack. Testing performed manually with local Jenkins and Robot.	Q3-Q4 2022

3.4 Integration status and roadmap of NetApp 4 'Multi-domain Migration'

A. High Level Design

The Multi-domain migration NetApp bets on the dynamic instantiation of new virtual counterparts on demand in a new domain using the same configuration parameters of the former virtual surrogate, transferring state to the new instantiation and finally updating data paths using Software-Defined Networking (SDN) functions.

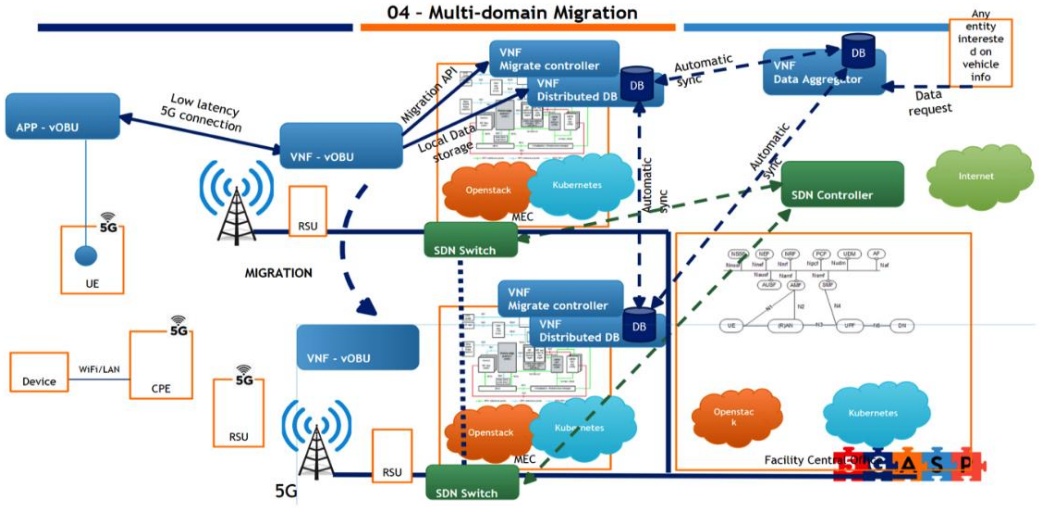

Figure 23 Multi-domain Migration NetApp

This NetApp provides interdomain mobility capabilities to the vOBUs introduced before, and in a more generic view, to any NetApp that requires it. This NetApp whose architecture is depicted in Figure 23, enables the vOBUs to be migrated to the closest MEC to the vehicle, reaching the low latency requirements that characterize vehicular applications. This migration procedure ensures that the OBU will maintain connectivity with its virtual counterpart in the former domain while the new virtual instantiation is getting ready with the same configuration and state. Once it is ready, the data paths are updated to start using the new one without any packet loss.

B. Low Level Design

The NetApp is developed in Python3, and the main packages include 'Cassandra-driver', to connect to the database, 'coapthon3' and 'http.server' to create servers that respond to the requests from other entities. CoAP is used in the communication between the OBU and vOBU while the other entities use HTTP. The Cassandra database is divided into three different nodes that share information in real time.

The NetApp will be deployed as a Network Service using OSM. It directly collaborates with the vOBU NetApp. The main components are the following:

- VNF Migrate Controller: manages the vOBUs and it is in charge of the migration process.

- VNF Distributed Database: the data of the vOBUs is stored in a distributed database in order to have the information ready in case of a migration.

C. Pre-installation information

The NetApp has been deployed locally in the Murcia testbed in its initial version. However, adjustments to the code are being made and the descriptors are currently being adapted to the latest versions of OSM 8 and 10, as the original ones are for the version 5 of OSM.

D. Requested APIs

Table 7 lists the NetApp's requested APIs from the 5GASP platform.

Table 7 NetApp 4 API requirements

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Required
Kubernetes API	Not required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

The NetApp requires a client application running in a physical OBU and the automotive communication infrastructure.

E. Timeframe

The following table describes the tasks and their timeline associated with NetApp 4.

Table 8 NetApp 4 timeframe

Task description	Task completion
NetApp initial implementation available (basic functionalities, some improvements to be implemented).	Q3 2021
NetApp VNFD/NSD implementation for OSM8 completed. OSM10 version under development.	Q1-2 2022
GST/NEST initial version.	Q3 2021
NetApp-specific tests to be designed.	Q1-Q2 2022
Still not locally deployed.	Q1-Q2 2022

3.5 NetApp 5 'Vehicle-to-Cloud (V2C) Real-Time Communication'

A. High Level Design

The V2C R2C NetApp enables the vehicle to send and receive data at real-time. The NetApp is application-aware to ensure optimized data transmission based on the content being sent e.g., real-time video or voice have higher priority than telemetry data. Since the transmission of the HD video generated by the cameras must be realized at real-time (i.e. buffering and retransmissions cannot be used) the NetApp uses techniques such as Forward-Error-Correction (FEC), channel bonding and dynamic encoding bit rate to assure fast video delivery while maintaining maximum video quality. In addition, when multiple channels exist, the NetApp will prioritize the data delivered to the vehicle based on the performance of the channels.

B. Low Level Design

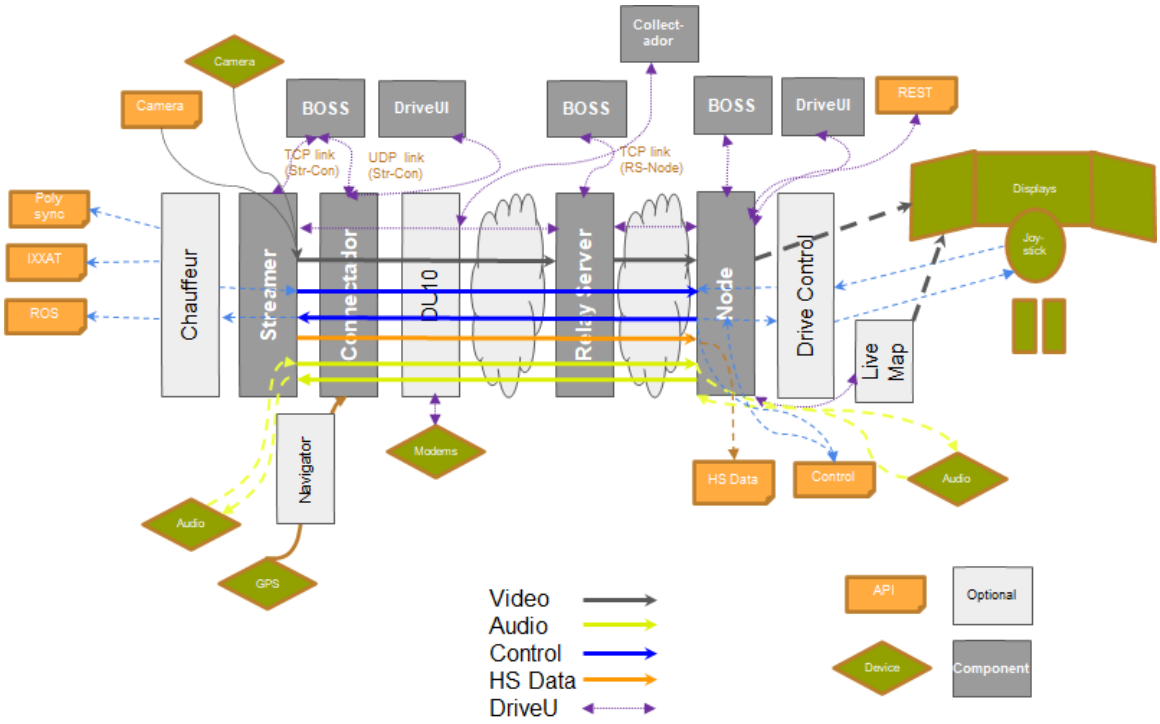

Figure 24 Component-level diagram of NetApp 5

Figure 24 displays the components in the DriveU system, end-to-end, from vehicle to relay server and then to the remote-control station. Note that due to the commercial status of the product, we are limited in our capability to reveal the low-level design of the whole product. However, we depict here the typical configuration for a system that enables remote driving the vehicle.

The following functionality is offered as we depict in Figure 24:

1. Video, captured directly from a camera or provided via the Camera API by the customer, is directly sent on the display screen in the remote-control station.
2. Two-way audio is streamed from the vehicle to the Driver Node, as well as back. The audio is directly connected to the devices (microphones and speakers).
3. Control messages are sent from a joystick/steering wheel/pedals in the remote control station to the vehicle, and the system is connected to drive-by-wire systems.
4. The system may be controlled via REST API, for example to change layouts.
5. A live map application is displaying the current location of the vehicle, based on GPS coordinates.

The application requires an NVIDIA video encoder installed on the transmitting side. The receiving side requires an NVIDIA GPU card for video decoding and video filtering. Some components' mechanisms are optimized for NVIDIA's CUDA engines. Specifically, the transmitting side uses the CUDA engine to create a properly encoded video frames in minimum time, while the receiving side is manipulating the video frame and enabling video filters to enhance the remote viewer/operator the situation awareness. At the current stages of the development, the latency added by the encoder and decoder technologies is critical. To minimize the impact of the encoder, NVIDIA video cards are used in the Streamer. The transmitting machine can be a low-end computer (currently tested with Jetson) or a standard computer equipped with NVIDIA card. The minimum required NVIDIA card is GTX1070.

The receiving side is normally deployed in the cloud or at clients' premises, therefore the receiving end can be a computer with a GTX1070 card or an EC2 instance with the A100 Nvidia card.

Finally, we would like to note that in some cases, the operation requires sending the video feed to the external device which is not equipped with video card or CUDA capabilities. To enable such nationalities, the receiving side will transcode the video. Such mode of operation will inherently create a delay in sending the video. To minimize the impact of the trans-coding, it is recommended to use a 'higher-end' video card.

C. Pre-installation information

This NetApp is an adaptation of DriveU's commercial product. During the reporting period, we designed and partially refined and further developed the deployment system so that it can operate in a containerized manner, leveraging the Kubernetes orchestrator.

D. Requested APIs

Table 9 lists the NetApp's requested APIs from the 5GASP platform.

Table 9 NetApp 5 API requirements

API	Required/Not Required
ETSI MANO SOL005	Required

OpenStack API	Not required
Kubernetes API	Required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

The NetApp is an automotive application and it is preferred to be deployed in the testbeds offering automotive capabilities i.e. the ITAv and OdinS testbeds. However, the NetApp is designed so that it can operate in a 'no mobility' mode. In this case, the NetApp can be onboarded, tested and certified at any testbed providing the API requirements that we described in Table 9 above.

E. Timeframe

The following table describes the tasks and their timeline associated with NetApp 5.

Table 10 NetApp 5 timeframe

Task description	Task completion
VNFDs/NSDs final design	Q2 2022
GST/NEST definition	Q1 2022
Definition of tests	Q2 2022
Onboarding on a 5GASP testbed	Q2 2022

3.6 NetApp 6 'Remote Human Driving NetApp–Teleoperation for assisting vehicles in complex situations'

A. High Level Design

There is an industry consensus today that autonomous vehicles will need help in making decisions, especially in unusual, dangerous situations that can happen on the road and may require violating the traffic laws (e.g. crossing double yellow lines). For these cases, the industry has started to adopt teleoperation/remote driving solutions that enable a remote human operator to monitor and take control over the car if needed.

DriveU and BLB present a solution that enables a remote operator to take full/partial control over an autonomous vehicle in unusual/dangerous situations that can happen on the road (e.g. let an autonomous vehicle crossing double yellow lines). Ensuring safe teleoperation and human remote assistance entails reliable transmission of high-quality real-time video with minimum latency.

B. Low Level Design

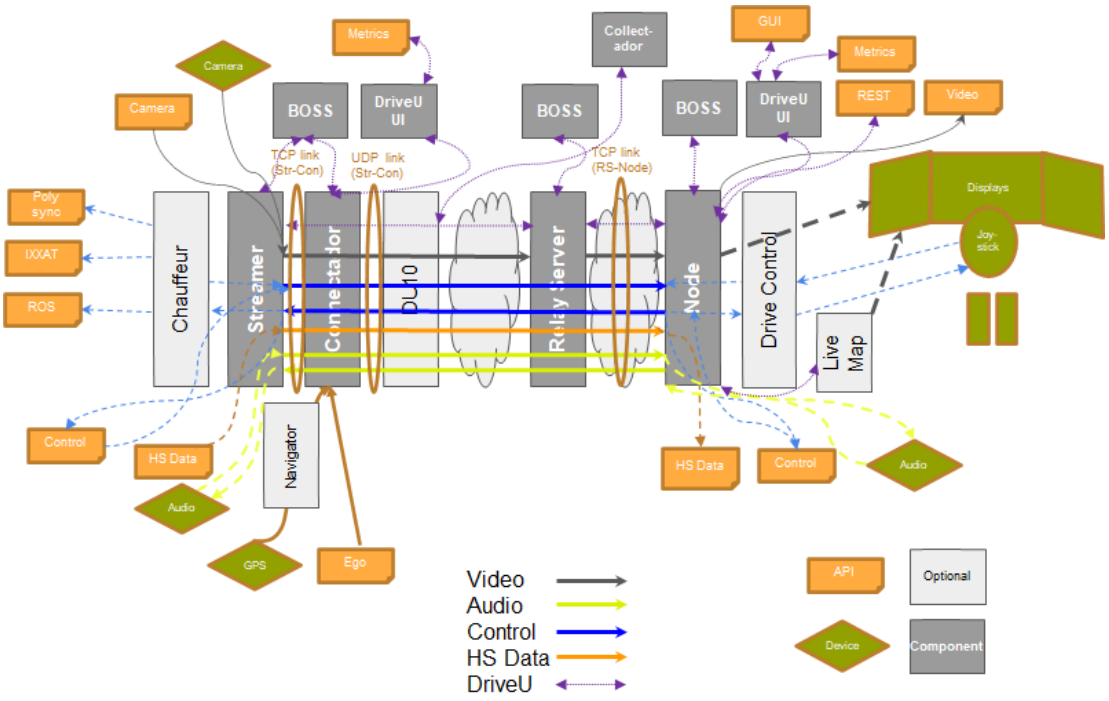

Figure 25 Component level diagram of NetApp 6

Figure 25 displays the main components in the DriveU system, end-to-end, from vehicle to relay server to the remote-control station. This is the typical configuration for a POC in "monitor only mode", or in cases DriveU provides the hardware for the system in a demo.

The diagram displays the following functionality:

1. Video, captured directly from a camera, sent to be shown directly on the display screen in the remote-control station.
2. Control messages are sent from vehicle to the Driver Node, though the system is not connected to the drive-by-wire system in the vehicle.
3. A live map application is displaying the current location of the vehicle, based on GPS coordinates.

The application requires Nvidia video encoder installed on the transmitting side. The receiving side requires Nvidia GPU card for video decoding and video filtering. Some of the NetApp mechanisms are optimized to use CUDA engines. The transmitting side uses the CUDA to create a properly encoded video frames in minimum time, while the receiving side is manipulating the video frame and enabling video filters to enhance the remote viewer/operator the situation awareness. At the current stages of the development, the latency added by the encoder and decoder technologies is critical. To minimize the impact of the encoder, Nvidia video cards are used in the Streamer. The

transmitting machine can be a low-end computer (currently tested with Jetson) or a standard computer equipped with Nvidia card. The minimum required Nvidia card is GTX1070.
The receiving side is normally deployed in the cloud or at client premises, therefore the receiving end can be a full-scale computer with GTX1070 card or a EC2 instance with A100 Nvidia card.

C. Pre-installation information

Similarly to our NetApp 5, this NetApp is again an adaptation of our commercial product. During the first year of the project, we designed and partially refined and further developed the system so that it shall run as in a CNF based manner over a Kubernetes-enabled testbed.

D. Requested APIs

Table 11Table 9 lists the NetApp's requested APIs from the 5GASP platform.

Table 11 NetApp 6 API requirements

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Not required
Kubernetes API	Required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

Alike NetApp 5 by DriveU, this NetApp belongs to the automotive vertical tackled by the 5GASP project. Thus, it is preferred to be deployed in the testbeds offering automotive capabilities i.e. the ITAv and OdinS testbeds. However, the NetApp is designed so that it can operate in a 'no mobility' mode. In this case, the NetApp can be onboarded, tested and certified at any testbed providing the API requirements listed in Table 11.

E. Timeframe

The following table describes the tasks and their timeline associated with NetApp 6.

Table 12 NetApp 6 timeframe

Task description	Task completion
VNFDs/NSDs final design	Q2 2022
GST/NEST definition	Q1 2022
Definition of tests	Q2 2022

3.7 NetApp 7 'Efficient MEC handover'

A. High Level Design

The efficient MEC handover NetApp is a supporting/enhancement application that aims to provide predictions to accommodate efficient orchestration decisions of other NetApps (namely, "*enhanced NetApps*") in a 5G radio environment. Examples of such predictions may refer to user handovers or traffic-related predictions, according to the enhanced applications requirements. The NetApp is a machine learning-based software that is currently developed and trained to predict the user connection handover in the context of a reference (known) 5G environment. The predictions would help delay-critical enhanced NetApps to achieve a near-zero downtime goal while the user is moving in the area from one access point to the other. The enhanced NetApps can utilise the information for placement, migration and scaling of the virtual network functions associated with the supported NetApps.

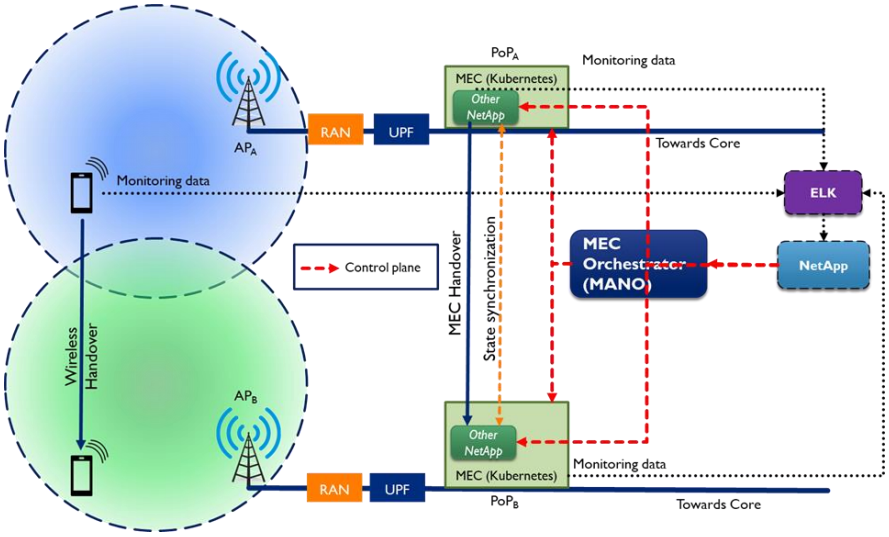

Figure 26 Generic architecture of Efficient MEC handover operation

B. Low Level Design

The NetApp is developed in Python3, and the main packages include Pandas and NumPy (for data processing), PyTorch (for machine learning). The NetApp has a predefined TCP socket to receive monitoring data from the ELK (Elastic Search, Logstash, Kibana) stack and provides handover predictions over the REST APIs to the other NetApps.

The Netapp will be deployed as a Network service using OSM. It will be composed of a Machine Learning module to make user's handover predictions supported by the ELK stack to provide the preprocessed data to the NetApp using the REST APIs as shown in

Figure 27. The handover decisions will be provided to the other NetApps using the NBI of the ML component.

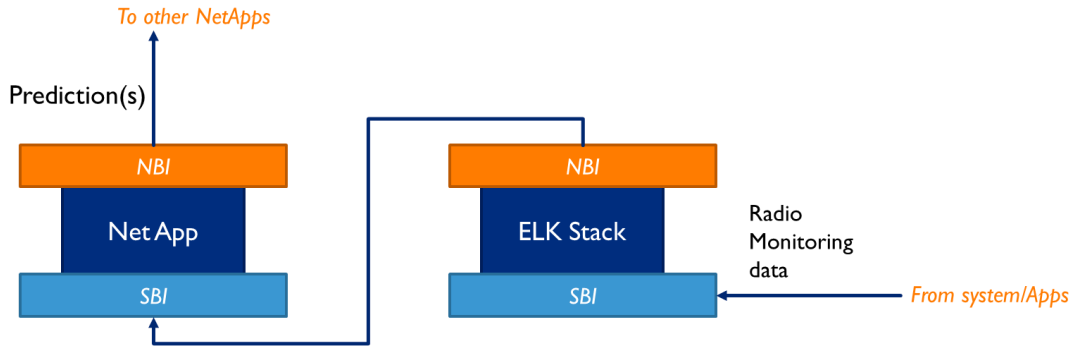

Figure 27 Component-level diagram

C. Pre-installation information

The Network Service Descriptor shown in Figure 29 is composed of two VNFs, Machine Learning VNF and ELK stack VNF. The mgmtnet is used for management access to the VNFs, the netAppnet serves the North-Bound-API for other netapps to consume the handover decisions from the Efficient MEC handover NetApp. The monnet serves the South-bound REST APIs to get the monitoring data from systems and/or the other monitoring applications.

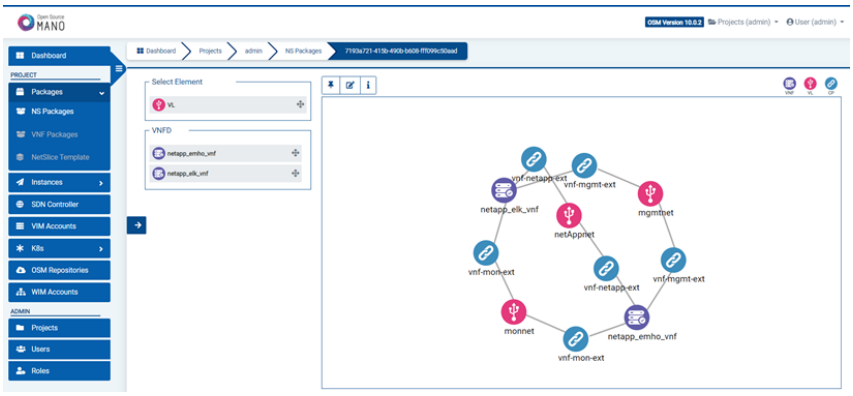

Figure 28 Network Service Descriptor with 2 VNFs

The ELK stack VNF descriptor, shown in Figure 30 has three interfaces for management access, to get the monitoring data from the underlying systems and to provide the processed monitoring data to the Efficient MEC Handover NetApp. The VNF requires 2 vCPUs, 5GB RAM and 60GB of storage. The other VNF in the Network Service is the ML component to predict the handover of the user in the mobile network. The VNF has three interfaces as well for management access, to get the pre-processed monitoring data from the ELK VNF and lastly, the NBI to provide the handover decisions to the other NetApps as shown in the Figure 31. The VNF requires, 4 vCPUs, 10GB RAM and 30GB storage.

Figure 29 Virtual Network Function Descriptor (ELK Stack)

Figure 30 Virtual Network Function Descriptor (Machine Learning App)

D. Requested APIs

Table 13Table 9 lists the NetApp's requested APIs from the 5GASP platform.

Table 13 NetApp 7 API requirements

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Required
Kubernetes API	Not required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

The NetApp will be deployed on Bristol's 5G testbed as VMs. It will support the other enhanced NetApps and provide an interface to communicate the handover predictions to the enhanced NetApps, which can in turn proceed with necessary actions (such as requesting orchestration actions from orchestrators or conducting application-specific actions) to enhance their service delivery. The NetApp can be hosted on any off-the-shelf compute nodes, however optionally it can make use of GPUs for faster processing of the machine learning models.

E. Timeframe

The following table describes the tasks and their timeline associated with NetApp 7.

Table 14 NetApp 7 timeframe

Task description	Task completion
VNFDs/NSDs definitions	Q2 2022
GST/NEST initial version	Q2-Q3 2022
Definition of tests	Q3-Q4 2022
Onboarding: Designing APIs as per the supported NetApps	Q3-Q4 2022

3.8 Integration status and roadmap of NetApp 8 'PrivacyAnalyzer'

A. High Level Design

- **General description of the system**

PrivacyAnalyzer offers the functionality for aiding the 5G/IoT integrators to assess the privacy strength of their applications or services against confidential information disclosure before they can release their solutions to their clients.

PrivacyAnalyzer receives testing sessions' traffic between the devices (UEs, IoT devices, etc.) participating in the integrator's test sessions, scans the sessions' message contents for privacy sensitive content and reports the outcomes of the analysis to the end-user (integrator) with the goal to enable the end-user to resolve the privacy issues detected in the test sessions before releasing her application or service to her client, typically a network/service operator or an IoT service provider.

- **High level architecture**

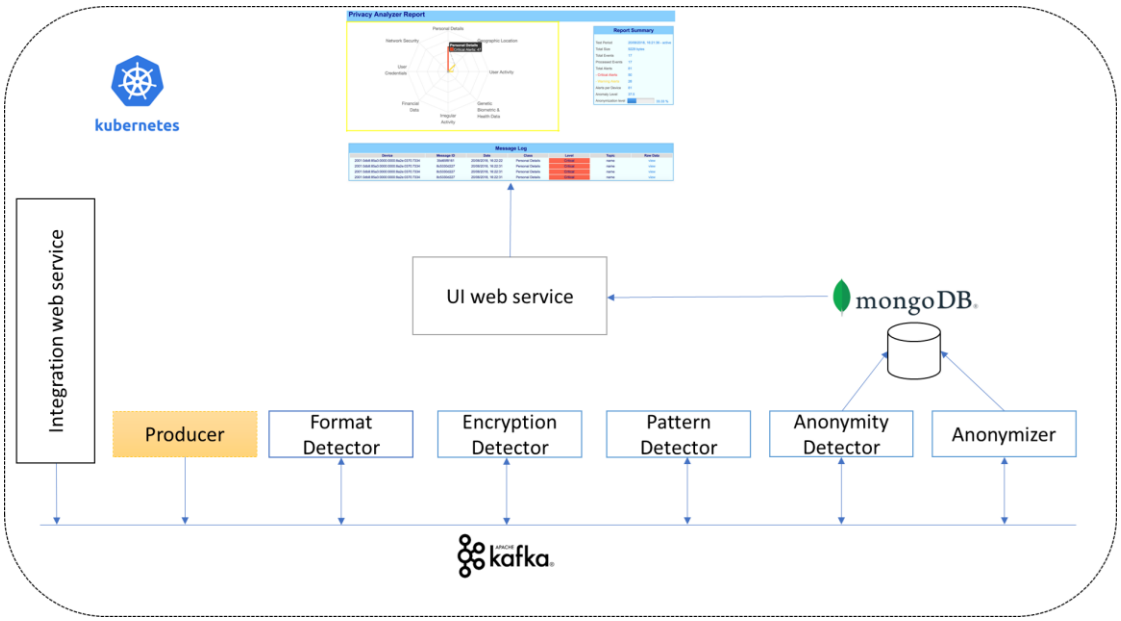

Figure 31 High level architecture of the PrivacyAnalyzer system

B. Low Level Design

- **Tools and programming languages used**

All components of the PrivacyAnalyzer core sub-system {format-detector, encryption-detector, pattern-detector, anonymity-detector, anonymizer} are written in Python 3 language and are packaged as Docker containers.

The MongoDB NoSQL database is used to store information relating to the analysis of the input datastreams. It is packaged as a Docker container. It is fed with JSONs by the 'anonymizer' component via a pymongo MongoClient call.

Interfacing with MongoDB to display information to end-users is performed via the 'web services' component which is written using Python Flask and which communicates with the MongoDB instance.

The UI of the end-user is an Angular User Interface running on Nginx.

- **Administration of the system**

PrivacyAnalyzer targets as end-users 5G/IoT integrator companies typically with no knowledge of privacy algorithms and technologies. The administrator User Interface must be user friendly manner so that it can be understood without domain specific knowledge.

The product's administrators are only required to configure a small set of configuration parameters. Specifically, the administrator is asked to choose the anonymization policy of the user's test sessions. For that purpose, a dropdown menu appears in the administrator UI from which the administrator selects one of the following policies {'Full Anonymization', 'Suppression of Identifiers', 'Supression of User Activity}.

The administrator also configures the user access via the administrator UI. Overall, it not mandatory that the administrator has any knowledge of the details of the privacy analysis components of the PrivacyAnalyzer 'core' sub-system as it is depicted in the high-level architecture of the system.

The product users can only view their dedicated User Interface which includes the privacy analysis of their devices and the history of the analysis of their devices. It is mandatory that no other user can access a specific user's devices and relevant datasets.

- **End-user functionality**

The PrivacyAnalyzer UI provides the following functionality to the end-user:

1. A report summary that holds aggregate statistics regarding the test session such as the number of processed messages and their total size in bytes.

2. A radar chart that renders the privacy alerts in distinct categories.

3. A message logs section. The message log contains important information such as the IPv6 address of the device, the message ID, the class, the level of anonymization and the message topic.

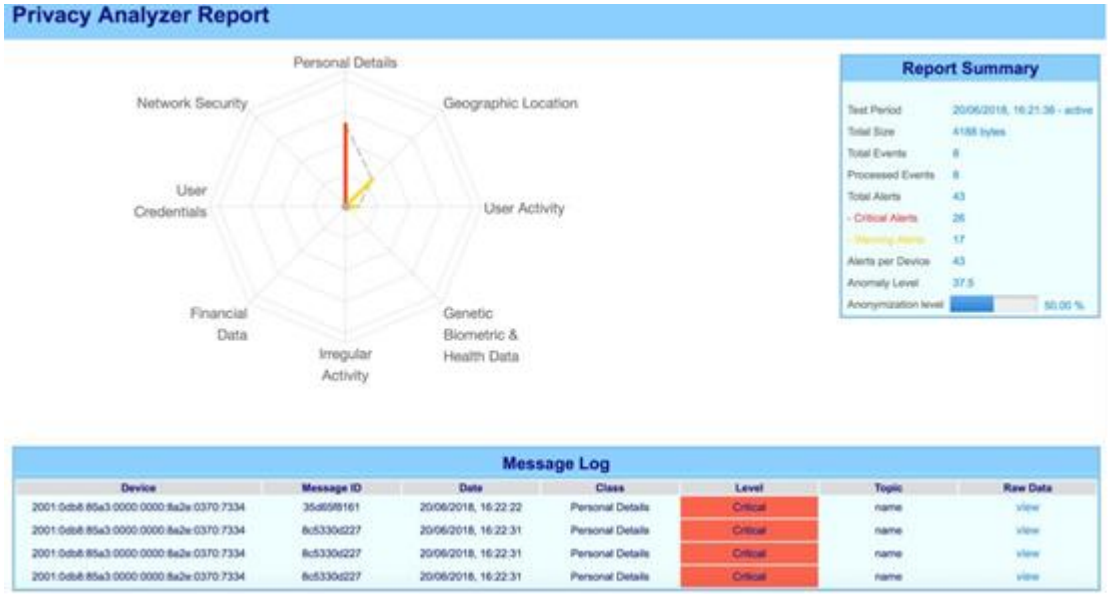

Figure 32 PrivacyAnalyzer User Interface

The raw data of each captured message that raised a privacy alert is available to the user through a table at the bottom of the web page as depicted in the following figure.

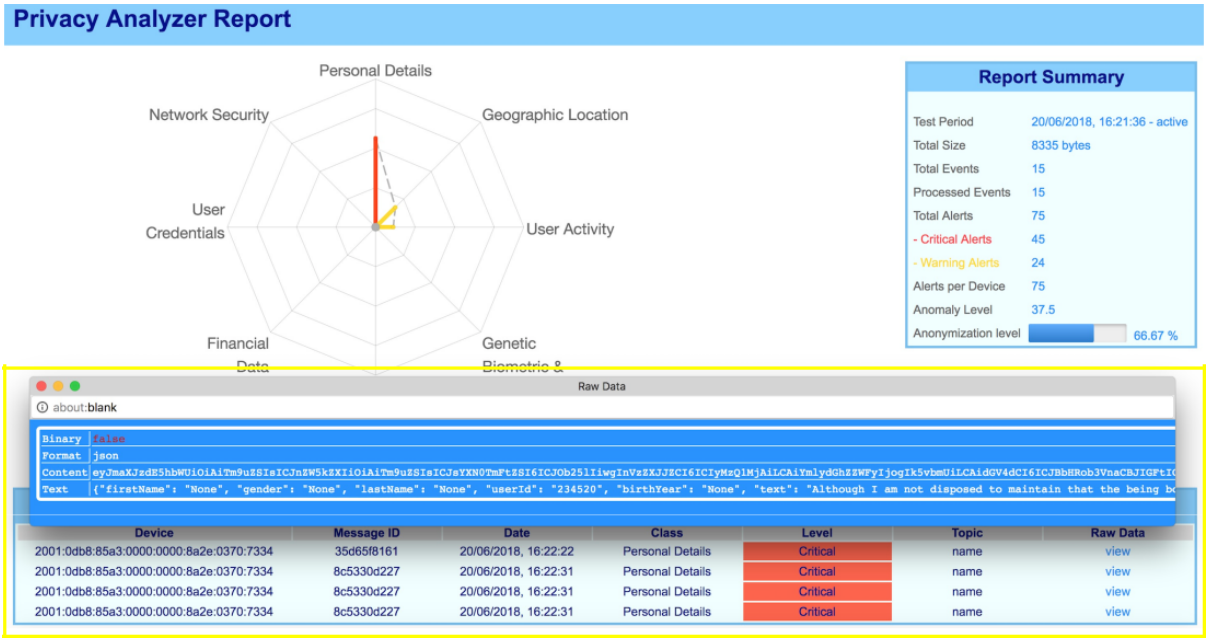

Figure 33 PrivacyAnalyzer message log

The user can hover over the radar chart and view the critical alerts for each privacy category. In the next figure, we can view alerts pertaining to Personal Details, Geographic Location and User Activity.

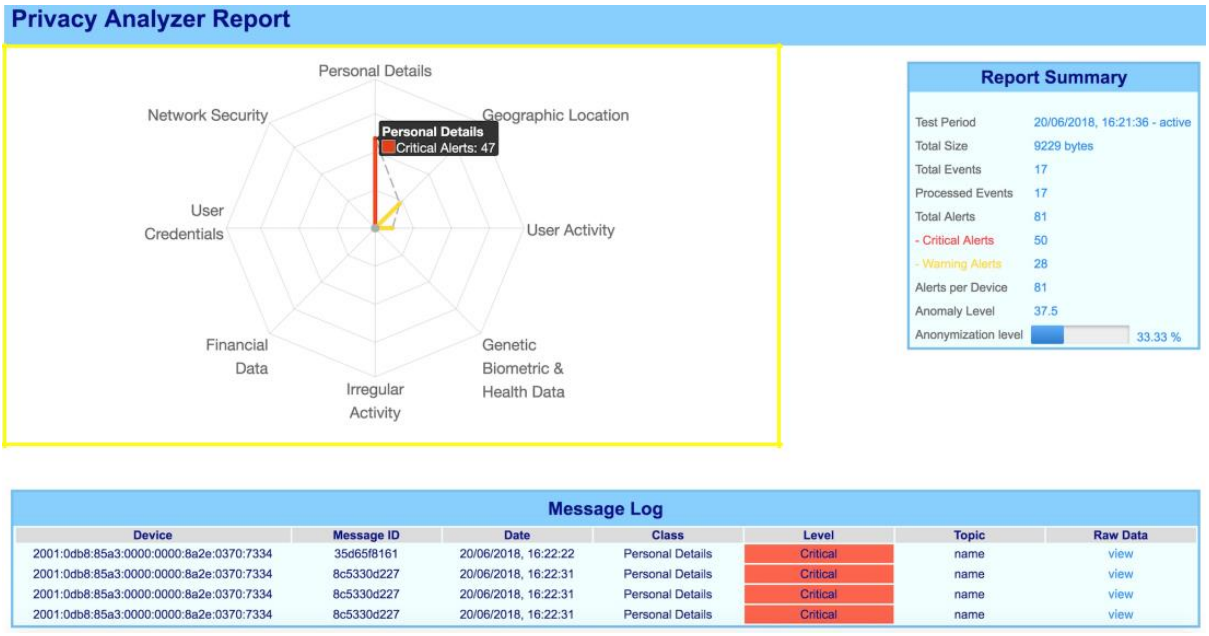

Figure 34 Alerts per privacy infringement category

As depicted in the HLD in section 3.8.A, the system consists of a sequence of processors (a process is implemented as a docker container) which communicate with messages through a Kafka cluster with the K8s deployment of the system. Each processor performs specific functionality as we shall latter discuss. Each individual process receives messages (often at high volume rates) from a specific Kafka topic and emits processed/enriched messages to an output Kafka topic. All topics are registered within the Kafka cluster.

In the following we discuss the main functionality of each processor implemented in the PrivacyAnalyzer system.

Format Detector: It detects whether the data is text or binary and the actual serialization format: XML, JSON or YAML for text messages and Avro or BSON for binary messages.

Encrypt Detector: It detects whether the content is compressed. This information is important since compressed content has similar entropy properties. It returns an entropy value (as discussed in [ZLIB]; we use python3 zlib entropy implementation) in the range [0..1] where values closer to 1 denote higher entropy and thus higher chance that the data is encrypted.

Pattern Detector: It tokenizes the message and classifies tokens to categories such as phones, addresses, emails, names, dates, etc. based on a combination of pattern matching, Named Entity Recognition (NER) and statistical analysis. The Pattern Detector also is able to detect birth years and geographic coordinates within the received messages.

Anonymity Detector: This processor classifies the detected tokens (note that tokens are produced by the Pattern Detector processor) either to de-identifiers or quasi-identifiers. E.g. email addresses are tagged as de-identifiers while detected dates are labeled as quasi-identifiers.

Anonymizer: The anonymizer processes all tokens marked as de-identifiers or quasi-identifiers by the anonymity classifier. De-identifiers are suppressed by replacing their value in the original stream with the placeholder character '*'. According to the input policy by the system administrator (see later in this sub-section) quasi-identifiers are either suppressed (gender) or generalized (dates, ages, geographic coordinates, etc.). The main objective of the Anonymizer is to detect whether the data contains de-identifiers, in which case it is de-identified, and/or it is anonymized according to the administrator policy. In case that quasi-identifiers are detected, they are either suppressed or generalized, also again according to the policy set by the administrator.

The results of the analyses performed by the 'Anonymity Detector' and 'Anonymizer' processors are stored in MongoDB cluster. The obtained results are used by the 'Web Service' component in the above high level architecture diagram in order to feed the Angular web app that is used by the end-user, so that the latter sees the results of the overall privacy analysis. In the next sub-section, we will present implementation details on user interface which is served by an instance of Nginx.

Note that our architecture also includes a component, named '**Stream Producer**' which is highlighted with the dashed yellow rectangle in the previous figure. 'Stream Producer' is used for testing purposes, i.e. it is not being used when the integration with the external system is performed. Its input is a large JSON file that contains a sequence of JSON-formatted records and emits a message for each record to the Kafka topic where the 'Format Detector' component is subscribed. We use various data sources to construct a

large realistic JSON file that emulates behaviors containing sensitive information. Specifically, one of the data sources used is the foursquare data sets from[4SQR-DATASET]. The 'Stream Producer' and its relevant input datasets containing large volumes of JSON data will also be used in the 5GASP project in order to ensure that the system functions correctly after its onboarding. Thus, it is envisaged as a NetApp-specific testing component which will aid us to validate the correct functioning of our software when it shall operate within the 5GASP environment.

- **Private Data Detection & Anonymization Policies**

Test sessions are associated with a set of configuration attributes that control the privacy detection and anonymization processes, as we discussed above. These attributes collectively constitute the privacy analysis policy and currently consist of the attributes listed in the following table.

Attribute	Description
anonymity_detection_deidentifier	Enables/disables the detection of de-identifiers.
anonymity_detection_quasiidentifier	Enables/disables the detection of quasi-identifiers.
anonymisation_deidentifier_suppress	Enables/disables the suppression of de-identifiers.
anonymisation_quasiidentifier_generalisation	Enables/disables the generalizationof quasi-identifiers
anonymisation_quasiidentifier_generalization_geo	Enables/disables quasi-identifiers related to geographic locations
anonymisation_quasiidentifier_generalization_time	Enables/disables the generalizationof quasi-identifiers holding times and dates.
anonymisation_quasiidentifier_generalization_bio	Enables/disables quasi-identifiers related to biometric data.

- **Privacy Conformance Policies & Anomaly Detection attributes**

Privacy conformance policies in PrivacyAnalyzer are externally provided policies that set user defined anonymization thresholds. After the anonymisation process, PrivacyAnalyzer compares the anonymization results against this set of anonymization thresholds. The distance between these is the '**Anomaly Level**' and denotes the conformance level of the anonymization process. This metric is displayed in the PrivacyAnalyzer UI and is being updated regularly during the test session. The attributes of the conformance policy that are currently supported by the PrivacyAnalyzer are listed in the following:

Attribute	Description
conformance_generalization_time	Minimal range size for time generalization in days.
conformance_generalization_geo_coord	Minimal range size for geographic coordinates generalization in degrees.
conformance_generalization_geo_address	Minimal range size for address generalization in number of discrete entries (size set) in the generalized

conformance_generalization_age	term[1]. Minimal range size for age generalization in years.

C. Pre-installation information

The core components {'format-detector', 'encrypt-detector', 'pattern-detector', 'anonym-detector', 'anonymizer'} are packaged as docker containers and pushed to our private repo as we described in the following Shell script.

```
#!/bin/sh

VERSION_STREAM=1.0
REPOSITORY=lamdaleon/privacyanalyzer

docker build -t $REPOSITORY:format-detector-$VERSION_STREAM -f
tools/stream/Dockerfiles/Dockerfile_format_detector tools/stream/
docker build -t $REPOSITORY:encrypt-detector-$VERSION_STREAM -f
tools/stream/Dockerfiles/Dockerfile_pattern_detector tools/stream/
docker build -t $REPOSITORY:pattern-detector-$VERSION_STREAM -f
tools/stream/Dockerfiles/Dockerfile_pattern_detector tools/stream/
docker build -t $REPOSITORY:anonym-detector-$VERSION_STREAM -f
tools/stream/Dockerfiles/Dockerfile_anonym_detector tools/stream/
docker build -t $REPOSITORY:anonymizer-$VERSION_STREAM -f
tools/stream/Dockerfiles/Dockerfile_anonymizer tools/stream/

if [ ! -z "$1" ]; then
  if [ $1 = "push" ]; then
    docker push $REPOSITORY:format-detector-$VERSION_STREAM
    docker push $REPOSITORY:encrypt-detector-$VERSION_STREAM
    docker push $REPOSITORY:pattern-detector-$VERSION_STREAM
    docker push $REPOSITORY:anonym-detector-$VERSION_STREAM
    docker push $REPOSITORY:anonymizer-$VERSION_STREAM
    docker push $REPOSITORY:anonym-estimator-$VERSION_STREAM
  fi
fi
```

The User Interface is an Angular5 based web application. The application can be built with the following command:

ng build --prod —aot=false

and then it is packaged as a Docker image and uploaded to Lamda Network's private Docker repository through the commands:

docker build –t lamdaleon/privacyanalyzer:ui-1.0

docker push lamdaleon/privacyanalyzer:ui-1.0

The fronted uses Nginx to serve the files as well as for the SSL termination (TLS 1.2) and authentication. The Nginx configuration files as well as the SSL certificates are packaged into the Docker image of the web app.

[1] As an example, when a quasi-identifier holding a city is generalized to the country where the city is located, the range size is the number of cities in that country.

Finally, in its SaaS version, the backend uses MongoDB, Kafka and Spark clusters for logging, fast message processing and aggregation of metrics during the detection of privacy vulnerabilities of incoming networked messages.

Packaging with Helm chart

A demo version of the PrivacyAnalyzer software has been packaged with a Helm Chart with the goal to be used for onboarding and instantiation purposes. Specifically, initially for checking locally the onboarding and instantiation, onto our local OSM 10 instance (as we will described in section E) as well as to prepare the respective deployments to the target 5GASP testbeds that we will describe in section F of this section.

The NS, KNF and Helm Chart of the demo version of the NetApp are described in the following three tables.

Table 15 NS of the demo version of PrivacyAnalyzer (producer_without_kafka nsd yaml)

```
nsd:
  nsd:
  - description: NS consisting of a single KNF producer_without_kafka_knf
connected to mgmt network
    designer: OSM and then modified by Lamda Networks
    df:
    - id: default-df
      vnf-profile:
      - id: producer_without_kafka
        virtual-link-connectivity:
        - constituent-cpd-id:
          - constituent-base-element-id: producer_without_kafka
            constituent-cpd-id: mgmt-ext
          virtual-link-profile-id: mgmtnet
        vnfd-id: producer_without_kafka_knf
    id: producer_without_kafka_ns
    name: producer_without_kafka_ns
    version: '1.0'
    virtual-link-desc:
    - id: mgmtnet
      mgmt-network: 'true'
    vnfd-id:
    - producer_without_kafka_knf
```

Table 16 producer_without_kafka VNFD yaml

```
vnfd:
  description: KNF with single KDU using the helm chart,named ll from repo
named lamdanetworks
  df:
  - id: default-df
  ext-cpd:
  - id: mgmt-ext
    k8s-cluster-net: mgmtnet
  id: producer_without_kafka_knf
  k8s-cluster:
    nets:
    - id: mgmtnet
  kdu:
  - name: producer_without_kafka
    helm-chart: lamdanetworks_repo/ll
  mgmt-cp: mgmt-ext
  product-name: producer_without_kafka_knf
  provider: Lamda Networks Inc
  version: '1.0'
```

Note that in the above KNF description, the name of the helm chart repository is **"lamdanetworks_repo"**. For our local experiment, we have performed this association with the following osm CLI command:

osm repo-add --type helm-chart --description "Repository for Lamda Networks private helm Chart" lamdanetworks_repo http://*GCP_public_IP*

where the IP address GCP_public_IP is the public IP of the google cloud VM which we are internally using to host Helm Charts using the Chartmuseum[2] software. Furthermore, ***ll*** is the name of the Helm Chart that shall be fetched upon OSM instantiation by the repository named **lamdanetworks_repo** configured in our local OSM.

Finally, the following table presents the configuration for our Helm Chart, ll-0.1.1. tgz, specifically the main configuration within the files *values.yaml* and *template/deployment.yaml*.

Table 17 Image location in values.yaml is the docker image tagged "producer-5gasp-demo-v1.0" in our public docker hub repository

```
image:
  repository: lamdaleon/docker101tutorial
  pullPolicy: Always
  tag: "producer-5gasp-demo-v1.0"
```

Table 18 Configuration of the K8s probes in the deployment.yaml file

```
 ports:
            - name: http
              containerPort: 5000
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /
              port: http
          readinessProbe:
            httpGet:
              path: /
              port: http
```

Note that port 5000 is used for the livenessProbe and readineProbe of our K8s deployment. For these to properly operate we have packaged our container using Python's flask web framework.

Local OSM onboarding

We first onboarded the NS and the KNF of our "privacyanalyzer-without-kafka:v1.0" demo version to OSM with the following commands:

osm nfpkg-create privacyanalyzer_without_kafka_knf
osm nspkg-create privacyanalyzer_without_kafka_ns

[2] https://chartmuseum.com/

Then we instatiated the NS on our local OSM 10 instance (running as a Virtual Box Ubuntu 18.04 virtual machine in a Mac book pro core i7, 16GB RAM laptop) using the osm command:

osm ns-create --ns_name producer_without_kafka_ns --nsd_name producer_kafka_ns --vim_account[3] lamdanetworks

The following figure displays the successful instantiation of our network service "producer_without_kafka_ns" within our local OSM 10 instance.

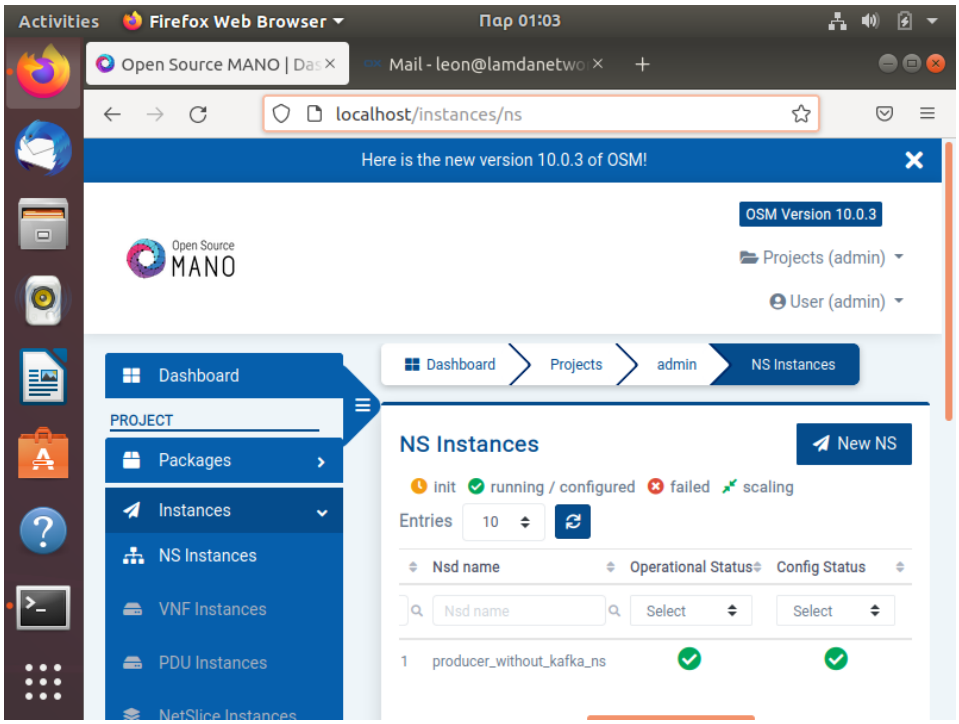

Figure 35 Local OSM 10 onboarding of the PrivacyAnalyzer demo

The NS and KNF have been locally onboarded as we described earlier, while the K8s deployment used the Helm Chart drawn from our Google VM instance running the Chartmuseum repo, as depicted in the following log file (Figure 36) of this VM.

[3] For the purposes of our lightweight demo, we do not use our private K8s cluster hosted at Google but an isolated K8s cluster used for local onboarding to an OSM instance, using `osm k8scluster-add cluster --creds .kube/config --vim lamdanetworks --k8s-nets '{k8s_net1: null}' --version "v1.15.9" --description="Isolated K8s cluster for local onboarding"`

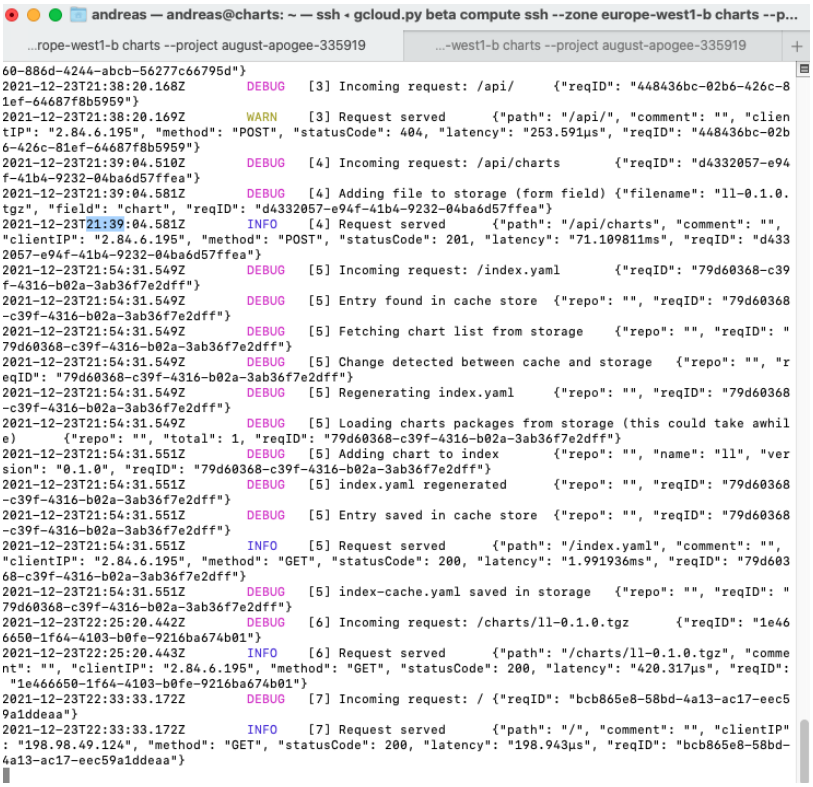

Figure 36 Logs from our private chartmuseum repo

Concluding this section, we have locally deployed a demo version of our software using a Helm Chart hosted in a private Chartmuseum repository and a demo docker image, uploaded as a public image on Docker Hub. The next steps, that will be described later in subsections F and G shall involve the instantiation of our the full fledge of our Tier-3 application.

D. Requested APIs

Table 19 APIs requested by PrivacyAnalyzer NetApp

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Not required
Kubernetes API	Required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

The NetApp shall be a CNF based NetApp thus there is no need for OpenStack API. In terms of hardware, interfacing with ININ hardware for the purposes of the PPDR use case shall be needed. We have discussed this issue with partner ININ and there will be a custom API sending messages from ININ portable h/w to the input endpoint of PrivacyAnalyzer.

In respect to the deployment in Patras or to any other 5GASP testbed supporting the NetApp's requirements expect from the dedicated H/W, we shall use pre-defined test messages with no hardware needed to test the h/w agnostic version of our NetApp.
In respect to the deployment in Bristol, our goal is to become an 'enhanced NetApp' leveraging the MEC offloading capabilities. We are discussing with partner UNIVBRIS the needs of the hardware in order to conduct realistic experiments on 5GUK.

E. Timeframe

The following table describes the tasks and their timeline so that we reach the goal that the PrivacyAnalyzer NetApp is deployed on three target testbeds which we have internally discussed could be used for the different usage scenarios of the NetApp.

Table 20 Timeframe of the PrivacyAnalyzer NetApp

Task description	Task completion
NetApp onboarding and instantiation on the Patras, ININ and Bristol testbeds using OSM 10 and their associated K8S VIMs.	Q1 2022
NEST versions a) for the centralized setup of the NetApp; and b) the distributed (MEC) setup of the NetApp.	Q2 2022
NetApp deployed on Patras's testbed	Q2 2022
NetApp deployed on ININ testbed	Q2 2022
NetApp deployed on Bristol testbed	Q3 2022

3.9 Integration status and roadmap of NetApp 9 '5G Isolated Operation for Public Safety - 5G IOPS'

A. High Level Design

- **General description of the system**

The Isolated Operations for Public Safety (5G IOPS) NetApp covers two operational scenarios related to PPDR operations, namely regular day-to-day operational scenario, and disaster phase operational scenario. As the scenarios' names indicates, an idea behind is to provide PPDR units with communication services in all circumstances. The infrastructure of the use case consists of edge node/cloud and core node/cloud, where disaster scenario reflects a case of a broken connection between the edge and the core, therefore edge node only needs to be capable of providing required PPDR communication services. The core network would usually be a public 5G core network. The NetApp consists of two VNFs to be developed and deployed, i.e., gNB VNF (provides functionalities of gNB/BBU) and CN VNF (provides core network functionalities).

Figure 37 depicts the high level architecture for both scenarios:

- regular day-to-day operational scenario, dataflow depicted by green line: both edge and core nodes/clouds are up and running, the edge providing gNB VNF (provided by this use case) and the core providing core functionalities which can be provided either by CN VNF (provided by this use case) or any other compatible core network solution, among others, core network functionalities provide internet access and access to specific 5G PPDR services,
- disaster phase operational scenario is triggered as a consequence of the uplink failure from gNB/edge to 5G mobile core, resulting in users being unable to use the 5G services and internet access. In case gNB detects uplink failure, it triggers the reconfiguration of the gNB to use the IOPS CN VNF provisioned on the edge and provide users the basic set of 5G services, i.e., Private 5G services.

- **High level architecture**

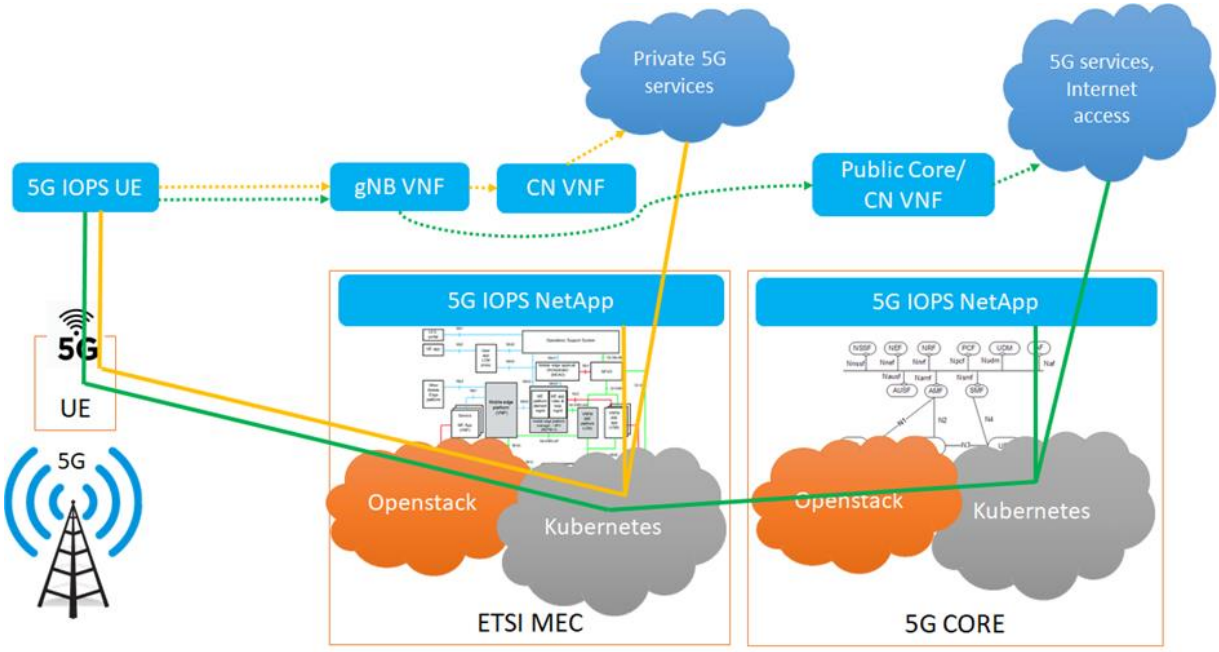

Figure 37 5G Isolated Operations for Public Safety NetApp high level architecture

B. Low Level Design

- **Tools and programming languages used**

The VNFs are based on commercial software binaries for gNB/5G CN and are used at a microservice level to build and package docker containers, i.e. separate containers for gNB and 5G CN. The gNB VNF component is enhanced with the additional software logic to support switching between 5G IOPS core and Public core network using standard Linux shell tools and Python.

- **Administration of the system**

Administration of the 5G IOPS NetApp will be fully done via 5GASP portal and orchestration mechanisms provided by OSM on each 5GASP testbed.

- **End-user functionality**

End users, i.e., PPDR users, expects basic services at any time which means there is a certain procedure expected in case of gNB/edge detects connection to core network has failed. Operational flow is as follows:

1. The UE is attached to the (public) 5G core network cloud accessing normal PPDR services (e.g. MCPTT).
2. The gNB detects loss of the backhaul to the (public) 5G core network cloud and in accordance with local operator policies decides to activate IOPS mode of operation. The gNb prevents any UEs from selecting the cell, using a suitable mechanism such as cell barring, until step 3 and step 4 are completed.
3. Local 5G CN VNF is activated.
4. The gNb establishes an N2/N3 connections to the Local 5G CN.
5. The gNb broadcasts the PLMN identity for IOPS operation with the Local 5G CN and indicates the IOPS PLMN cell(s) as "Not Barred" & "reserved" for operator use.
6. The UE detects the IOPS PLMN-Id and a decision is made to switch USIM application and the UE activates the IOPS USIM application.
7. The UE selects the IOPS PLMN-Id.
8. The UE attaches to the Local 5G CN and, for an IP PDN type, obtains a local IP address, if authorised.
9. Public safety services supported by the IOPS network can be accessed at this time.
10. At some point in time the gNb detects that the backhaul to the (public) 5G core network cloud has been restored.
11. N2/N3 connections to the Local 5G CN are released according to the IOPS network policies to move the UEs to idle mode.
12. The gNb stops its IOPS mode of operation and the Local 5G CN is de-activated.
13. The gNb establishes an N2/N3 connections to the (public) 5G CN.
14. The PLMN-Id of the (public) 5G CN is announced and the normal TAIs of the (public) 5G CN are advertised by the gNb so that UEs reselect the normal PLMN.
15. The UE detects the PLMN-Id of the (public) 5G CN and a decision is made to switch USIM application and the UE activates the normal USIM application.
16. The UE selects the normal PLMN-Id.
17. The UE attaches as normal to the (public) 5G CN.

C. Pre-installation information

The VNF components are packaged as VMs, each running a containerized component of a NetApp. The initial container images for gNB and 5G CN are prepared and use a local repository at ININ testbed.

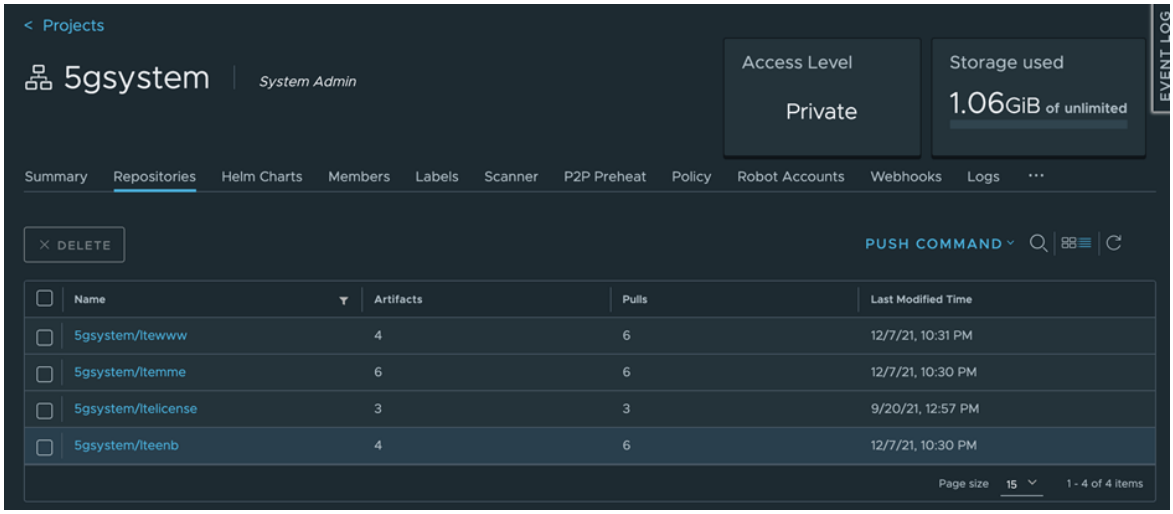

Figure 38 Local container repository containing NetApp docker components

In the following figures, “docker run” commands to run gNB and 5G CN are presented and are also showcasing some configuration options already available at container run time (i.e. PLMN). Further configuration options will be added in later stages of the project. These commands serve as a basis to run 5G services inside the VNFs.

```
sudo docker run -tid \
--name lteenb \
--mount type=bind,source="$(pwd)"/config/enb.template.cfg,target=/app/enb/config/enb.template.cfg \
--mount type=bind,source="$(pwd)"/logs,target=/app/screen_logs \
--mount type=bind,source="$(pwd)"/lteenb_service_logs,target=/tmp \
--net=amarisoft --ip 172.18.0.3 \
--cap-add=SYS_RAWIO -v /dev:/dev --privileged \
--env LOG_FILE_MAX_SIZE=1K \
--env LOG_MAX_NUBMER_OF_FILES=2 \
--env LICENSE_SERVER_ADDRESS=192.168.202.82 \
--env PLMN=00101 \
-p 9002:9000 \
core-harbor.qmon.eu/5gsystem/lteenb:sdr-2020-12-14
```

Figure 39 Example command to run gNB container component

```
sudo docker run -tid \
--name ltemme \
--mount type=bind,source="$(pwd)"/config/mme.template.cfg,target=/app/mme/config/mme.template.cfg \
--mount type=bind,source="$(pwd)"/config/ue_db-ims.cfg,target=/app/mme/config/ue_db-ims.cfg \
--mount type=bind,source="$(pwd)"/logs,target=/app/screen_logs \
--mount type=bind,source="$(pwd)"/ltemme_service_logs,target=/tmp \
--net=amarisoft --ip 172.18.0.2  \
--device /dev/net/tun:/dev/net/tun \
--cap-add=NET_ADMIN \
--env LOG_FILE_MAX_SIZE=1K \
--env LOG_MAX_NUBMER_OF_FILES=2 \
--env LICENSE_SERVER_ADDRESS=192.168.202.82 \
--env PLMN=00101 \
-p 9001:9000 \
core-harbor.qmon.eu/5gsystem/ltemme:sdr-2020-12-14
```

Figure 40 Example command to run 5G CN container component

Local OSM with VIM onboarding

The 5G IOPS NetApp is currently running as manually deployed NetApp, i.e. Docker containers inside VMs, with emphasis on end-to-end service testing. Also, the local instance of OSM10 is deployed and ready with onboarded OpenStack as VIM and Kubernetes, supporting testing VNF and CND orchestration.

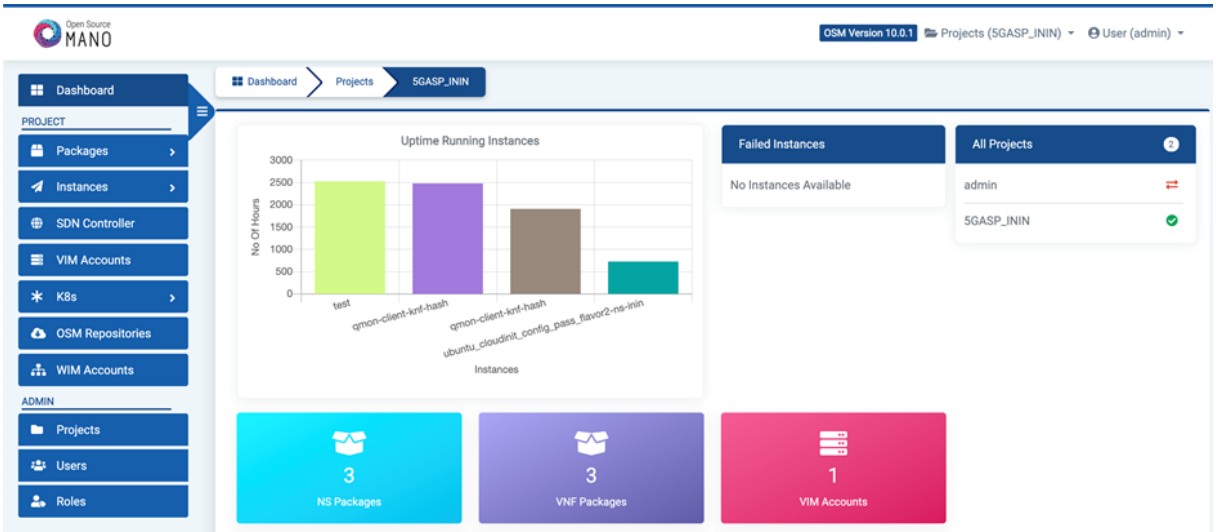

Figure 41 OSM Dashboard screenshot overviewing current (local) deployment

D. Requested APIs

Table 21 APIs requested by ININ's NetApp

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Required
Kubernetes API	Not required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Required

5G IOPS NetApp will be deployed on Ljubljana testbed. Since the NetApp will enable continuous communication services, it will also support other NetApps deployed on the same testbed, i.e., PrivacyAnalyzer. Next to this, Ljubljana testbed will also explore deployment of cross-border scenarios, such as PPDR related use cases involving Ljubljana and Patras testbeds.

E. Timeframe

The following table describes the tasks and their timeline associated with NetApp 9.

Table 22 5G IOPS NetApp timeframe

Task description	Task completion
NetApp initial implementation available (basic functionalities, some improvements to be implemented).	Q3 2021
NetApp VNFD/NSD implementation for OSM10.	Q4 2021
GST/NEST requirements for NetApp.	Q4 2021
NetApp generic and end-to-end test scenarios related to 5G system performance KPIs.	Q1-Q2 2022
NetApp onboarding on local OSM10 at ININ testbed.	Q1-Q2 2022

3.10 Integration status and roadmap of NetApp 10 'Vehicle Route Optimizer'

A. High Level Design

- **General description of the system**

Neobility's NetApp, NeoBus is a dynamic pooled transportation service (Demand Responsive Transport (DRT)) solution based on minibuses (±10 seats) that is as flexible as traditional ride-sharing, while being a fraction of the price.

- **High level architecture**

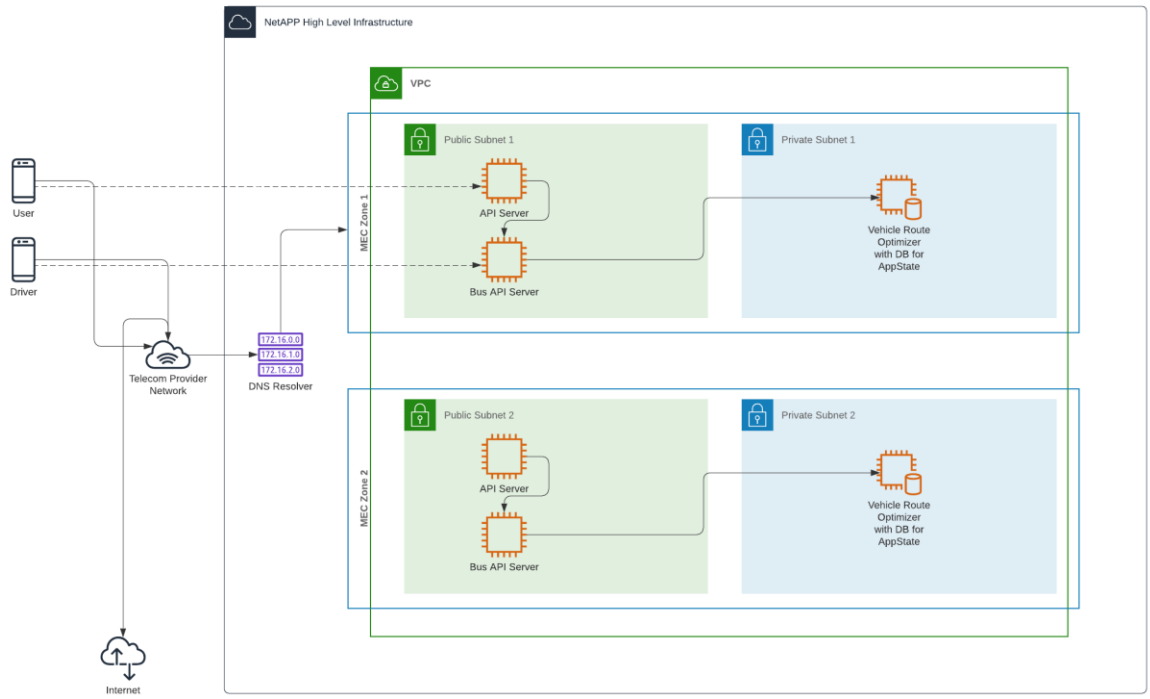

Figure 42 High level Architecture of the Vehicle Route Optimizer NetApp

B. Low Level Design

- **Tools and programming languages used**

The API Server and Bus API Server are being written in Typescript with Node.js 14 (at the time of writing, it is possible that will be upgraded to Node.js 16) and are packaged as Docker containers.

The Vehicle Route Optimizer is being written in Java 11 and packaged as a Docker container.
All services also have requirements for databases, caches and queues, which are also openly available as Docker containers. For example, the API Servers require transactional Postgres/MySQL databases and the Vehicle Route Optimizer requires a MongoDB NoSQL database to store information related to the routes generated by the algorithm. For caches, Redis will be used, as for queues this is not yet decided but one of two options will be used: either RSMQ over Redis or a standalone queue like RabbitMQ.

One of the extra main requirements of the Vehicle Route Optimizer is a Distance Matrix API Service. Multiple options are available, out of which OSRM (Open Source Routing Machine) can be deployed as a Docker container (it already have ready to use docker images) with the OpenStreetMap Data Extracts files downloaded manually or automatically from [GEOFABRIC].

As for the Admin UI(s) of the system, they are/will be written using a “Jamstack” architecture, most probably in React. The UI will not be tightly coupled to the backend. As for packaging they will also be packaged as Docker containers that simply serve the static content (most probably with Nginx).

For logging, logs will be asynchronously sent to a central ElasticSearch cluster, which will be available in the cloud and will be visualized using Kibana.

In conclusion, the full stack of the NetApp will be packaged as Docker containers.

- **Administration of the system**

NeoBus targets end-users that need to be transported on demand in the city or region where the service is available. The end-users only have access to a user app, with no administration features whatsoever.

The administration of the NetApp will be done solely by Neobility using the Admin UI(s) discussed in the previous point.

As for the orchestration part, deploying the services dynamically based on demand, this will be done in connection, that needs to be decided, between the NetApp APIs and portal/orchestration mechanisms provided by OSM.

- **End-user functionality**

The NeoBus NetApp will be embedded in the UrbanAir application as a distinct layer of mobility, easy to be accessed, users will have the possibility to hail a bus through the mobile app:

  - the user will get an estimation for the pick-up time and trip duration unlike regular buses, NeoBus will offer the possibility to hail a vehicle in real time, as well as preplan a pickup
  - the user will get to the virtual stations (around high-traffic areas), walking no more than 300 meters, as a tradeoff to help decrease the overall city traffic and minimize pickup times
  - the stations will be generated by the NeoBus app virtually anywhere considering the above condition and a maximum 15% detour for the onboard passengers (vs. initial estimation);
  - another big advantage of the NeoBus is that will allow the user to get from point A to B without changing lines/vehicles;
  - the algorithm behind the app will assure that the bus is never too crowded and will allow people to easily get in and out from the bus

C. Pre-installation information

The VNF components are to be packaged as virtual machines, each running several containerized components of the NetApp.

In the following table, “docker run” commands to run the components are presented. Complete configuration options will be added later in the project.

Table 23 NeoBus NetApp containers' configuration

```
#!/bin/bash

# OSRM requirement for the "routing-engine"
docker run --detach --env APP_ENV=dev --publish 5000:5000 neobility/osrm-backend-docker:latest

# NeoBus (Transport) API Server
docker run --detach --env NODE_ENV=dev --publish 3001:3000 neobility/transport-api:latest

# Routing engine
docker run --detach --env APP_ENV=dev --publish 3002:3000 neobility/routing-engine:latest

# API Server for the UrbanAir App
docker run --detach --env NODE_ENV=dev --env DEPLOYMENT=dev --publish 3003:3000 neobility/main-api:
```

The initial deployment is expected to be completed in Q1 2022. Ongoing developments towards the PoC of the NetApp that started in 2021 is being continued in 2022 along with deployment scripts and tests.

D. Required APIs

Table 24 APIs requested by NeoBus NetApp

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Required
Kubernetes API	Required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Not required

Ideally, Neobility's NetApp can be deployed on any testbed that is able to provide at least two (or more) smartphones with 5G and has coverage outside of the facility. Two (or more) persons will need to go outside and follow a testing procedure.

Although, the NetApp will also have a testing procedure that would not require a physical test, nor any UEs. It will emulate a series of "buses" and "ride requests". We use this to test the algorithm and we plan to expand it to the whole NetApp.

The Neobility NetApp can be deployed to all testbeds supporting the automotive usecase, but actual usage and pilot tests most likely can happen only where we are located (Bucharest).

Thus, the NeoBus NetApp will be deployed on the ORO testbed and on a second testbed supporting the automotive use case (ITAv or OdinS).

E. Timeframe

The following table describes the tasks and their timeline associated with NetApp 10.

Table 25 Timeframe of the NeoBus NetApp

Task description	Task completion
NetApp initial implementation available (basic functionalities, some improvements to be implemented)	Q1 2022
NetApp VNFD/NSD implementation for OSM10	Q1 2022
GST/NEST requirements for NetApp	Q1 2022
NetApp generic and end-to-end test scenarios related to 5G system performance KPIs	Q2-Q3 2022
NetApp onboarding on local OSM10 at ORO and on the second testbed	Q3-Q4 2022

3.11 Integration status and roadmap of NetApp 11 'Fire detection and ground assistance using drones '

A. High Level Design

The 'Fire detection and ground assistance using drones'(FIDEGAD) NetApp currently consists of the Image Acquisition Client that is going to be installed in the drone, and the Object Detection Service.

The Image Acquisition Client is installed on a drone, communicates with the backend image recognition/object detection NS over a websocket connection and transmits image/video data to the NS backend service.

The Object Detection Service consists of a deep neural network that provides real time object detection. The result of the object detection procedure is forwarded to the Control Center Web Interface where the end user can monitor the field.

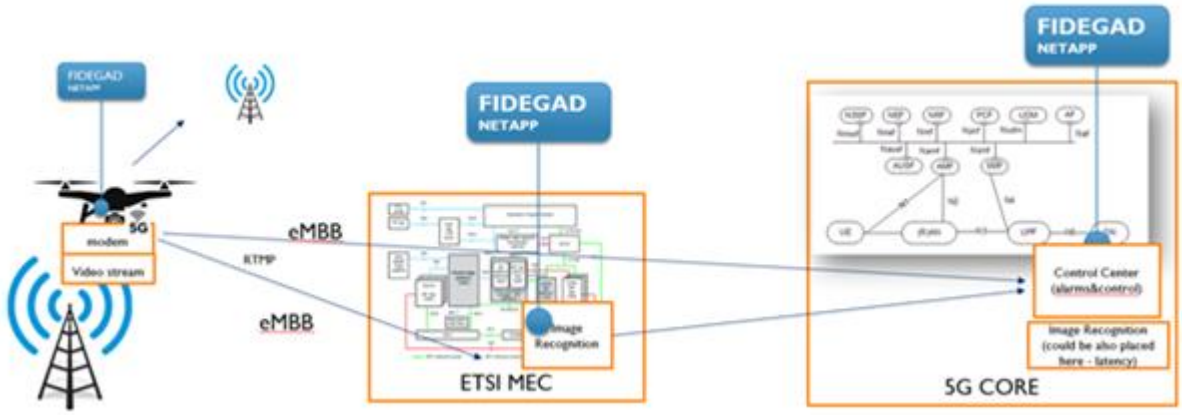

Figure 43 FIDEGAD NetApp high level architecture

B. Low Level Design

The **Object Detection NS** along with the **Control Panel Service** are implemented using python 3.7 and make use of the following dependencies.

```
-websockets==10.1
-tensorflow==2.4.0
-keras==2.4.3
-numpy==1.19.3
-pillow==7.0.0
-scipy==1.4.1
-h5py==2.10.0
-opencv-python
-keras-resnet==0.2.0
-imageai==2.1.6
```

The **Object Detection NS** utilizes the RESNET-50 architecture [RESNET-ARCH], currently trained on the Microsoft COCO dataset [COCO-DATASET]. The service extracts a set of detected objects, along with the related probabilities. This information is forwarded to the **Control Panel Service** which provides the results to the final user through a web interface.

C. Pre-installation information

The **Object Detection Service** along with the **Control Center** are currently packaged in a single container (cameraaistr). A docker image has been created and pushed to a private container registry. Additionally, a helm chart has been also created and pushed to a private helm repository, registered to the OSM, along with a VNFD and a NSD package.

The packages comprising the NetApp, i.e. the Object Detection Service, the Control Center (cameraaistr) and a python webserver (simplepythonwebserver) are presented below. Table 26 and Table 27 contain the NSD packages, whilst Table 28 and Table 29 contain the corresponding VNFD packages.

Table 26 FIDEGAD's cameraaistr NSD package

```
nsd:
  nsd:
    - description: NS for cameraaistr KNF connected to mgmt network
      designer: UoP
      df:
        - id: default-df
          vnf-profile:
            - id: '1'
              virtual-link-connectivity:
                - constituent-cpd-id:
                    - constituent-base-element-id: cameraaistr
                      constituent-cpd-id: mgmt-ext
                  virtual-link-profile-id: adminnet1
              vnfd-id: cameraaistr_knf
      id: cameraaistr_ns
      name: cameraaistr_ns
      version: '1.0'
      virtual-link-desc:
        - id: adminnet1
          mgmt-network: 'true'
      vnfd-id:
        - cameraaistr_knf
```

Table 27 FIDEGAD's simplepythonwebserver NSD package

```
nsd:
  nsd:
    - description: NS for simplepythonwebserver KNF connected to mgmt
network
      designer: UoP
      df:
        - id: default-df
          vnf-profile:
            - id: '1'
              virtual-link-connectivity:
                - constituent-cpd-id:
                    - constituent-base-element-id: simplepythonwebserver
                      constituent-cpd-id: mgmt-ext
                  virtual-link-profile-id: adminnet1
              vnfd-id: simplepythonwebserver_knf
      id: simplepythonwebserver_ns
      name: simplepythonwebserver_ns
      version: '1.0'
      virtual-link-desc:
        - id: adminnet1
          mgmt-network: 'true'
      vnfd-id:
        - simplepythonwebserver_knf
```

Table 28 FIDEGAD's cameraaistr VNFD package

```
vnfd:
  description: KNF with single KDU with cameraaistr using a helm-chart
  df:
    - id: default-df
  ext-cpd:
    - id: mgmt-ext
      k8s-cluster-net: mgmtnet
  id: cameraaistr_knf
  k8s-cluster:
    nets:
      - id: mgmtnet
  kdu:
    - name: cameraaistr
      helm-chart: NAM/cameraaistr
  mgmt-cp: mgmt-ext
  product-name: cameraaistr_knf
  provider: UoP
  version: '1.0'
```

Table 29 FIDEGAD's simplepythonwebserver VNFD package

```
vnfd:
  description: KNF with single KDU with simplepythonwebserver using a helm-
chart
  df:
    - id: default-df
  ext-cpd:
    - id: mgmt-ext
      k8s-cluster-net: mgmtnet
  id: simplepythonwebserver_knf
  k8s-cluster:
    nets:
      - id: mgmtnet
  kdu:
    - name: simplepythonwebserver
      helm-chart: NAM/simplepythonwebserver
  mgmt-cp: mgmt-ext
  product-name: simplepythonwebserver_knf
```

```
  provider: UoP
  version: '1.0'
```

Currently, the NetApp is in the process of finalizing the configurations required to be onboarded and deployed on Patras testbed. The expected timeframe for the aforementioned procedure is Q1 2022 (see following section E).

D. Required APIs

Table 30 APIs requested by FIDEGAD NetApp

API	Required/Not Required
ETSI MANO SOL005	Required
OpenStack API	Not required
Kubernetes API	Required
CI/CD API	Required
Monitoring API	Required
Slicing API	Required
Dedicated H/W API	Not required

The FIDEGAD NetApp will be deployed on Patras testbed. As the NetApp is container-based there is no need for OpenStack API support. Also, NetApp is designed in a way that it does not require a dedicated API for the hardware involved. The part of the NetApp that would be installed on a standalone hardware device, i.e. a Raspberry Pi 3, is the object detection service and is container-based as well. Thereby, a Kubernetes API exposed from the Raspberry Pi would suffice to meet NetApp's needs.

E. Timeframe

The following table describes the tasks and their timeline associated with the FIDEGAD NetApp.

Table 31 Timeframe of FIDEGAD NetApp

Task description	Task completion
NetApp's initial codebase (Prototype streaming video service and recognition algorithms).	Q1 2022
NetApp's CNFs implementation for OSM10	Q1 2022
NEST initial version	Q3 2021
NetApp-deployed in UoP's testbed	Q1 2022

4. Conclusions and Future Work

This documented presented the current status and the ongoing developments of the APIs that shall be exposed by the 5GASP testbeds to the NetApp developers of the project and also to 3rd party NetApp developers who shall leverage 5GASP's tools via their engagement with the 5GASP project and specifically through their participation in 5GASP's Community Portal.

Based on the testbed APIs and the requested APIs by the NetApps of the 5GASP consortium, we have concluded that currently most NetApps can be onboarded by one or more 5GASP testbeds, since most testbeds support SOL005, OpenStack, K8s and monitoring APIs which are required by all the NetApps of the project.

For the NetApps that do not have specific requirements for User Equipment, the current status is that these NetApps can be onboarded and instantiated on the 5GASP testbeds, however, there needs to be a unified onboarding and instantiation mechanism in place. This shall be realized in the next months via completing the Openslice deployment over the interconnected 5GASP testbeds.

For NetApps that have requirements bound to the PPDR vertical, the ININ has a rich UE offering which can be leveraged by the relevant NetApp developers. For NetApps bound to the Automotive domain, the ITAv and OdinS sites shall be the main ones that shall be used for automotive use cases and to that end, both testbed owners are working on implementing the necessary functionality for exposing their specific hardware to the relevant NetApp developers.

Future work shall also address the issue of providing a common monitoring API to NetApp developers so that it their NetApps can be onboarded and instantiated in a seamless way across any 5GASP facility and KPIs can be measured using a unified way through the expected common monitoring API that shall be exposed through the 5GASP portal.
Last but not least, as it is planned within the scope of Task 3.4 of the project, some 5GASP facilities must be tailored to cater for such vertical specific requirements associated to a subset of 5GASP NetApps and use cases. To that end, regular calls are held on a weekly basis in order that the relevant facilities progress the development of functionalities and APIs for offering vertical-specific enhancements to the relevant NetApps, combinations of interworking NetApps and vertical-specific use cases.

References

[4SQR-DATASET] Available from https://sites.google.com/site/yangdingqi/home/foursquare-dataset

[COCO-DATASET] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

[ETSI GS NFV-SOL005] https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.04.01_60/gs_NFV-SOL005v020401p.pdf

[GEOFABRIC] Available from https://download.geofabrik.de

[HLD] Available from https://en.wikipedia.org/wiki/High-level_design

[LLD] Available from https://en.wikipedia.org/wiki/Low-level_design

[NOVA-API] Available from https://docs.openstack.org/nova/latest/

[OPENSLICE] Available from https://openslice.readthedocs.io/en/stable/

[OPENSTACK-DOC] Available from https://docs.openstack.org/api-ref/compute/

[RESNET-ARCH] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[ZLIB] Available from https://www.euccas.me/zlib/