PROJECT "LOCUS": LOCalization and analytics on-demand embedded in the 5G ecosystem, for Ubiquitous vertical applicationS

Grant Agreement Number: 871249
(https://www.locus-project.eu/)

DELIVERABLE D4.3

"Implementation of the virtualization platform for network control and management: preliminary version"

Deliverable Type:	R
Dissemination Level:	Public
Contractual Date of Delivery to the EU:	31/03/2021
Actual Date of Delivery to the EU:	30/03/2021
WP contributing to the Deliverable:	WP4 – Localisation & Analytics for Smart Network Management
Editor(s):	Elian Kraja, Giacomo Bernini (Nextworks)
Author(s):	Elian Kraja, Giacomo Bernini, Francesco Bocchi, Micheal De Angelis (Nextworks) Athina Ropodi, Aristotelis Margaris, Ioannis Filippas (Incelligent) Marco Faltelli, Marco Bonola (CNIT) Maria Belesioti (OTE)

	Antonio Tarrias, Sergio Fortes, Eduardo Baena and Raquel Barco (UMA)
Internal Reviewer(s):	Takai Eddine Kennouche (Viavi) Tomasz Mach (Samsung)
Short Abstract:	The goal of this deliverable is to describe the design of the LOCUS virtualization platform, that allows to deploy and run the LOCUS functions as VNFs in distributed computing locations at the edge and core of the 5G network infrastructure. The document also reports on the design principles and architecture of the LOCUS MANO, responsible for the lifecycle management of the LOCUS analytics services on top of the virtualization platform. A first software prototype of the LOCUS MANO deployed and validated on top of a small-scale lab-based virtualization platform is also described.
Keyword List:	5G, localization analytics services, MANO, virtualization, NFV, edge, Openstack, Kubernetes

Executive Summary

LOCUS aims at providing a unified and generalized localization analytics platform for the deployment of localization analytics functions and services on top of 5G virtualized infrastructures, taking care to expose such services and functions towards Smart Network Management and 3rd party vertical applications to fulfil their localization requirements. With this kind of virtualization-based approach, LOCUS can enable the reuse of common localization and analytics functions for several purposes and applications scopes, thus enabling a flexible collection, manipulation, analysis and exposure of localization related data.

This deliverable covers two key aspects of the LOCUS platform. First, it provides the solution for the LOCUS virtualization platform, which allows to run the localization analytics services as combination and interconnection of virtual functions. This is to align with the 5G system architectural approach, that consider a high-degree of virtualization of network functions and services by design. In particular, LOCUS defines a hybrid virtualization platform that integrates different technologies and solutions to distribute the LOCUS virtual functions across edge and core computing locations within the 5G end-to-end infrastructures. Traditional Infrastructure as a Service solutions (based on Openstack) are combined in LOCUS with cloud-native virtualization platforms (based on Kubernetes) to run localization analytics function workloads at appropriate locations, exploiting computing resources at the far-edge (i.e. very close to users and 5G devices in general) and satisfy service requirements in terms of latency, availability, etc.

On the other hand, this deliverable also presents the LOCUS Management and Orchestration (MANO), which provides to the LOCUS platform full automation capabilities in the provisioning of localization analytics services, including deployment, configuration and operation of localization virtual functions on the virtualization platform. A detailed functional decomposition of the LOCUS MANO, together with principles and approach for lifecycle management of localization analytics and packaging of the related virtual functions is also reported. In practice, the LOCUS MANO enables to align with current trends in managing and orchestrating virtualized infrastructures in 5G, with a high level of automation and integration with NFV principles to dynamically and flexibly deploy and operate localization analytics services, as complementary functionalities of 5G network functions and services distributed across edge and core locations.

A preliminary software prototype is also reported in the deliverable, which combines an initial version of the LOCUS MANO (based on the ETSI OSM opensource platform) with a small-scale virtualization platform deployed in the Nextworks lab infrastructure and integrating an Openstack core compute location with a two nodes Kubernetes cluster at the edge.

VERSION CONTROL TABLE			
VERSION N.	**PURPOSE/CHANGES**	**AUTHOR (S)**	**DATE**
0.1	Agreed ToC	NXW	27/11/2021
0.2	First set of contributions (sections 2, 3, 4)	NXW, CNIT, INCE	18/12/2021
0.3	Second round of contributions (sections 3, 4)	NXW, INCE, OTE, UMA	22/01/2021
0.4	Consolidation of architectural principles (sections 3, 4)	NXW, INCE, CNIT	29/01/2021
0.4	Initial contribution for software prototype (section 5)	NXW	12/02/2021
0.5	Updates from partners on contributions (all sections)	NXW, INCE, OTE, UMA	19/02/2021
0.6	Finalization	NXW	26/2/2021
0.7	Version for internal review	NXW	16/03/2021
1.0	Final Version	NXW	29/03/2021
1.1	Final revision	Nicola Blefari Melazzi	30/3/2021

INDEX

List of Abbreviations

ABBREVIATION	FULL NAME
3GPP	3rd Generation Partnership Project
4G	Fourth generation technology standard for cellular networks
5G	Fifth generation technology standard for cellular networks
AI	Artificial Intelligence
API	Application Program Interface
BeFEMTO	Broadband Evolved FEMTO Networks
BS	Base Station
BSS	Business Support System
CNF	Containerized Network Function
CNI	Container Network Interface
CP	Connection Point
CRUD	Create, Read, Update, Delete
CSAR	Cloud Service ARchive
DL	Deep Learning
DPDK	Data Plane Development Kit
ETSI	European Telecommunications Standards Institute
FM	Fault management
GNSS	Global Navigation Satellite System
HTTP	Hypertext Transfer Protocol
HSS	Home Subscriber Server
IaaS	Infrastructure as a Service
ISG	Industry Specification Group
JSON	JavaScript Object Notation
KNF	Kubernetes Network Function
KPI	Key Performance Indicator
LCM	Lifecycle Manager
LCS	LoCation Service
LEN	Localization Enabler
LTE	Long-Term Evolution
MANO	Management and Orchestration

MEC	Multi-Access Edge Computing
ML	Machine Learning
MME	Mobility Management Entity
NAT	Network Address Translation
NMS	Network Management System
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NR	New Radio
NS	Network Service
NSD	Network Service Descriptor
NSE	New Services
OSM	Open Source MANO
OSS	Operations Support System
PCRF	Policy and Charging Rules Function
PGW	Packet Gateway
PNFs	Physical Network Functions
QoS	Quality of Service
REST	REpresentational State Transfer
SBA	Service Based Architecture
SBI	Service Based Interface
SGW	Serving Gateway
SNM	Smart Network Management
SON	Self Organizing Network
SRIOV	Single-root input/output virtualization
TOSCA	Topology and Orchestration Specification for Cloud Applications
UC	Use Case
UE	User Equipment
UWB	Ultra-Wideband
VDU	Virtual Deployment Unit
VIM	Virtualized Infrastructure Manager
VL	Virtual Link
VM	Virtual Machine

VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VNFFG	VNF Forwarding Graph
VNFM	VNF Manager
VPN	Virtual Private Network
WP	Work package
YAML	YAML Ain't Markup Language

Table 1: Abbreviation List

Figure Index

Table Index

1 Introduction

1.1 Document scope and objectives

This deliverable represents the first intermediate outcome of the task T4.3 “Virtualization platform for network control and management” activities. As part of the T4.3 objectives, the LOCUS Virtualization Platform and the Management and Orchestration (MANO) functions are first designed and then implemented. While the scope of the work in WP4 is mostly dedicated to Smart Network Management related functionalities, the virtualization platform and the MANO have the main goal to provide a common approach for automating the lifecycle management of any localization analytics service over 5G virtualized infrastructures. Therefore, the architecture design (as well as the preliminary prototype) reported in this document is to be considered applicable also for the New Services functionalities defined in the context of WP5 use cases.

This document builds on a reference baseline described in the deliverable D2.4 [1], where the overall principles for both the virtualization platforms and the LOCUS MANO are described, including their high-level functional description in the context of the preliminary LOCUS system architecture.

Starting from there, this deliverable aims at going deeper into the definition of the virtualization platform design and technical solution, assessing the feasibility of a hybrid approach where cloud-native and traditional Infrastructure as a Service virtualization solutions can coexist, being tightly integrated and exposed as a common virtualization platform. This allows to build a distributed platform that can leverage on de-facto standard solutions, such as Kubernetes and Openstack. Kubernetes can be used to deploy localization analytics workloads at the edge of the 5G infrastructure to run lightweight applications that requires proximity with users and devices. In any case, Kubernetes is also suitable for core compute locations to support also there cloud-native and container-based functions. On the other hand, Openstack can be used for running localization-related functions in core data centres with more traditional Virtual Machine based deployments.

Moreover, this document has the objective to provide a detailed definition of the LOCUS MANO functionalities, evolving the initial approach proposed in D2.4, that brought to the system architecture depicted in Figure 1. We target a smarter NFV management architecture, aligned with the current trends and standards in the area of 5G network management, which lays its foundation on automation and data analytics-based decision making. While the ETSI NFV principles and architecture are still at the foundations of this LOCUS work, our goal is to integrate the localization analytics service lifecycle management with Smart NFV Functions, helping in automation and data driven service operation. In addition, this deliverable has the objective to clarify how to manage the LOCUS functions and services as respectively VNFs and

virtualized NFV services, from packaging aspects up to runtime operation, including the definition of the LOCUS MANO external APIs.

Following the D2.4 architectural principles, the Service Based Architecture (SBA) approach enables each LOCUS function to provide atomized localization or analytics functionality and to be independently managed from other functions, while posing requirements for their interconnection, according to their produced (data) output and required (data) inputs. In this context, the LOCUS MANO becomes key to enforce the required configurations in the virtualized platform, as well as in the LOCUS functions themselves, in order to practically realize the SBA and logically chain the functions according to their data requirements.

As this is the initial outcome from task T4.3 (in terms of both design and prototype), it is expected that the LOCUS virtualization platform and the MANO functionalities will evolve in the following stages of the project, to align first of all with the work on LOCUS PoCs definition and design, in terms of scenarios to be supported as well as with the LOCUS testbed where to deploy the "production" virtualization platform. Another enabler for this evolution will be the final version of the LOCUS architecture, which will be described in deliverable D2.5. In the latter document, a few crucial aspects that have an impact on the virtualization platform and MANO aspects will be designed and modelled in their final version, including the integration and mapping with 3GPP 5G architecture, as well as the specific LOCUS data architecture and related solution for data exchange among the LOCUS functions building a localization analytics service (with related technologies to be used).

1.2 Document structure

This deliverable is structured into five main sections:

- Section 2 provides an overview of architectures and solutions for management and orchestration of services in 5G virtualized infrastructures, as well as how existing location-based service solutions have been deployed and managed in the state-of-the-art;
- Section 3 describes the LOCUS virtualization platform, providing the design approach for the hybrid solution based on the integration of Kubernetes clusters at the edge and Openstack compute locations at the core of the infrastructure. The available LOCUS infrastructures for the realization of the "production" virtualization platform are also described;
- Section 4 describes the LOCUS management and orchestration functionalities, including the new architecture approach (with respect to deliverable D2.4) with definition of features and APIs for localization and analytics functions and services automated deployment over the virtualization platform. The approach for the

integration with Smart Network Management functionalities from WP4 Use Cases is also detailed, including a monitoring platform for data collection and exposure, as well as a description on how to package the LOCUS functions to be managed by the MANO.

- Section 5 provides a description of the preliminary prototype that integrates a small-scale virtualization platform and an initial version of the LOCUS MANO in the Nextworks lab infrastructure. An early validation of such prototype with an exemplary LOCUS service is also described.
- Section 6 provides concluding remarks for this document.

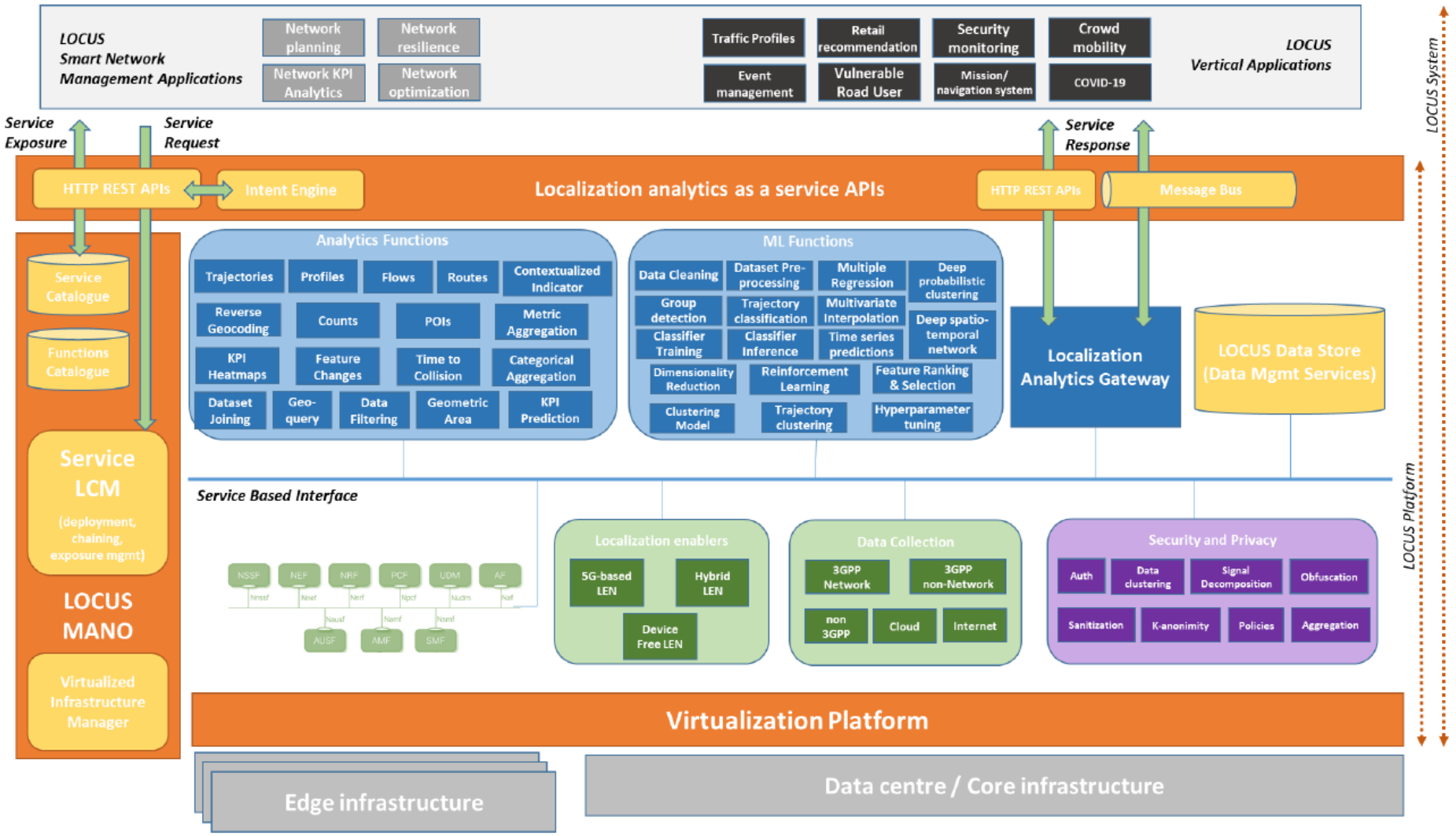


Figure 1: Locus System Architecture from D2.4

2 State of the art

2.1 Management and orchestration of services in virtualized environments

With 5G, the telecommunication industry is more and more looking at comprehensive orchestration solutions to ease the deployment of heterogeneous services across several technology domains and virtualized infrastructures distributed across edge and core locations. However, so far, most of the efforts in the industry and research communities have been put towards the delivery of end-to-end network services in support of vertical industry applications, without much focus on integrating location-based functionalities and services.

Indeed, already the concept of network slicing allows to jointly orchestrate resources (network, computing, storage) and virtualized or physical network functions, which are managed and delivered together to instantiate and compose network services over a shared network and computing infrastructure. Network slices can be dynamically created and customized according to the requirements of the services that will run on top of them, for example in terms of resource or function isolation and Quality of Services (QoS) guarantees, still mostly oriented to vertical industry services and applications.

In this respect, 3GPP has proposed a framework in [2] identifying functional components for the management and orchestration of end-to-end services and network slices, also identifying the three network segments (radio, access, core) that should be covered in end-to-end scenarios. Indeed, the network slicing principles are embracing heterogeneous technologies, spanning from 5G New Radio (5G NR) access technologies, to converged optical fronthaul and backhaul with Software Defined Networking, and hybrid 4G LTE and 5G Core services. Each of these segments and technology has normally its own stack of management and control tools and protocols that need to be tightly integrated for being able to automatize the provisioning of end-to-end services and slices. However, up to now, 3GPP yet does not provide any practical definition about the mechanisms needed to actually manage the lifecycle of these complex end-to-end services and slices across several technological domains.

The ETSI NFV ISG partially fills this gap in [3], providing an analysis of the relationship between the concepts of network slices and NFV network services. Here, the NFV network services (i.e. the combination of traditional VNFs and physical network functions) are presented as a resource-centric view of a network slice, thus mapping a network slice to one or more recursive network services. Following this approach, several phase-2 and phase-3 H2020 5G-PPP projects, like 5G-Transformer [4] , SliceNet [5], 5G-EVE [6], and 5Growth [7], among the others, propose service orchestration solutions based on “network slice management” functional entities operating on top of NFV Orchestrators, building end-to-end network slices through multiple NFV network services requested on-demand and across different domains. Vertical services are mapped to network slices according to service constraints and QoS

requirements, which are in turn instantiated as combination of NFV network services managed by the underlying NFV Orchestrators. However, the adoption of such kind of solutions still do not provide a truly end-to-end service orchestration approach as it is mostly limited to cover access and core network segments, partially considering radio domains.

Along the same line, Ericsson and Nokia, two of the main European solution providers and key players in the service orchestration arena, offer to their customers end-to-end service orchestration solutions to facilitate the delivery of 5G services, customized for different vertical requirements. Ericsson offers its Ericsson Orchestrator product as an integration of NFV Orchestration, Generic VNF management and Service Orchestration and Configuration Management features in support of end-to-end network slice coordination and provisioning [8]. Similarly, Nokia's network slice and NFV orchestration portfolio is built around the CloudBand product [9], which is an ETSI NFV management and orchestration system with proven reliability, automation, repeatability and security, that can be integrated with a network slicing solution for creating a flexible, scalable and dynamically reconfigurable virtual optical network spanning across access and core segments.

Besides this lack of truly end-to-end support, all of the mentioned approaches still are based on monolithic control and orchestration solutions, mostly dedicated to the management of NFV-like pipelined network services, where the lack of agility in the service lifecycle and operation is still a clear limitation. Moreover, the capability of such orchestration approaches to fulfil heterogeneous service constraints and requirements is still to be proven, as often requires per-service customizations and operation adjustments to support end-to-end deployments. In addition, the adoption of AI and ML technologies for service optimizations, including their interaction across the different technological domains and their tight integration with the service and slice lifecycle management is still at its early stages.

Beyond these current limitations, that are bringing to the design of cross-layer orchestration solutions (to be integrated as part of the next generation Network Management Systems (NMSs) and OSS platforms) where 5G NR, edge computing, SDN transport networks and cloud computing will be tightly integrated and coordinated, location-based information and data is still not yet integrated and processed in support of smarter network and service management. Indeed, location-based services and functions, together with geolocation information, are still mostly considered as independent and standalone aspects and features to be delivered with ad-hoc services and solutions, decoupled from the provisioning of end-to-end connectivity and vertical services. Such aspect is where LOCUS can contribute, with the aim of filling these location data and services gaps, by leveraging on the combination of LOCUS MANO and virtualized platform to dynamically and automatically manage the localization analytics services as combination of virtualized functions, exploiting edge and core distributed 5G virtualized infrastructures to ease the integration with regular 5G services for verticals and mobile connectivity.

2.2 Location based service solutions

The 5G architecture targets operability in a broad range of different contexts and use cases, and therefore will lead to a massive increase in the number of connected devices, thus creating a heterogeneous and very dense network. In this context, localization services play a fundamental part in order to accurately run and smartly manage these networks, as well as satisfying the data rate demands for the many connected devices.

3GPP has begun to discuss the integration of localization primitives into the 5G system architecture, starting from Release 16. To this purpose, the 3GPP Technical report 23.731 [10] describes the 5G Core LoCation Service (LCS) architecture, and addresses relevant security issues, including methods to address data eavesdropping and mobile location tracking, as well as other important aspects, e.g., combining LCS with low latency requirements, network slicing and scalability for the location support services.

In this context, the LOCUS architecture provides a natural solution for location services which operate in such a dense, highly connected network and to collect localization information coming from a wide variety of connected devices. It is essential to stress that state of the art localization systems and architectures, when compared to the LOCUS architecture, can only partially exploit the information coming from the heterogeneous sources. The reason for this limitation relies on the fact that such architectures usually handle data coming from subsets of sources, as for instance satellite location systems, intelligent transportation systems, or the network infrastructure [11][12]. Just to provide some examples, the authors of [12] propose a network-centric architecture for outdoor location and tracking of mobile devices in cellular networks. Such architecture relies on the joint exploitation of the measurements collected by the base stations, the angular sector covered by the antenna and open map data.

Trying to merge localization information with data coming from sources external to the cellular network ones (e.g., social data) has also been considered in order to increase the system's overall degree of knowledge.

To this aim, the general architecture of the system should be responsible for the integration between those sources (that could be provided by sources of information external to the cellular network). Previous works have tried to address such a challenge, as detailed below.

The European project BeFEMTO (Broadband Evolved FEMTO Networks) focused on efficient solutions for LTE-A femtocells management [13]. In this context, it identified localization as a key enabler for the cellular management. It also proposed an extension to the 3GPP architecture by adding core and local entities to minimize the related traffic into the operator's core and backhaul.

In the field of Self Organizing Networks (SON), an architecture for the integration of both internal and third-party location sources into cellular management, considering centralized, distributed and hybrid approaches was presented in [14]. This work also focused on

minimizing the impact of the use of the location fluxes of information (per each User Equipment (UE) and updated regularly) on the cellular network backhaul and core networks, so as to avoid congestion. It also included some discussion on the flows to exchange positioning and associated network management information between the different elements of the LTE network for both integrated solutions in the management/control plane of the network as well as by "over-the-top" solutions.

The integration of different sources of context information with SON mechanisms and related challenges (e.g., the defining of self-healing algorithms) have been discussed in [15], where a general scheme was defined.

A very high-level framework for the use of social-based event data into cellular management activities, such as main events prediction and their associated crowds, was discussed in [16].

Furthermore, some ad hoc solutions have also been developed, with the main shortcoming being their limited range of application and the fact that they are not part of the network infrastructure.

The work in [17] deals with satellite-based location systems with the goal of increasing their accuracy, especially in challenging environments (like tunnels or urban canyons), through the use of heterogeneous sensors. Moreover, the vehicles considered must be equipped with suitable sensors whose measurements are merged and integrated with those provided by other vehicles or a specific communication infrastructure.

The SELECT project [18] uses cooperating nodes for indoor person detection and tracking through the use of different radio technologies.

The ELAASTIC (European Location As A Service Targeting International Commerce) project [19] merges Galileo and the European Geostationary Navigation Overlay services with other available location enablers, providing a "Location as a Service" platform.

In the GEO VISION project [20] the position estimation done by the Global Navigation Satellite System (GNSS) is integrated with a local data and displayed with maps and geo-spatial information.

The work in [21] targets natural or man-made disasters and aims the location of isolated victims through the fusion of the position data obtained by the European GNSS system and those coming from Digital Cellular Technologies.

3 LOCUS Virtualization Platform

The virtualization platform is one of key enablers of the LOCUS system, in terms of advancements and innovation in how the location-based services can be deployed and operated in a very flexible and dynamic way. The main goal is indeed to provide a common environment where the localization and data analytics functions can run as virtual functions, following the current trends of cloud-native applications distributed across edge and core domains.

Indeed, the virtualization platform enables a smooth integration of the location-based services with the 5G infrastructures, as well as with other 5G services, whether providing services and functionalities for the 5G mobile connectivity (e.g. the 5G core services and functions) or services for vertical customers. 5G is bringing a new disruptive architectural approach, considering a high-degree of virtualization of the network functions and services by design. Moreover, to support emerging vertical services requirements, 5G infrastructures will natively host computing resources at the far-edge of the network infrastructures, to enable deployment of applications and network functions very close to users and 5G devices in general. In this scenario, the LOCUS virtualization platform aims at implementing a solution for location-based analytics services that integrates with, and supports, the SBA of 5G. On top of such virtualization platform, advanced management and orchestration techniques (as described in section 4) can help in delivering localization analytics services on-demand and with functions distributed over the 5G end-to-end infrastructure to satisfy the service requirements in terms of latency, availability, throughput, etc.

In practice, the LOCUS virtualization platform provides an abstraction of edge and core clouds in the 5G infrastructure exposing a unified virtualized infrastructure. It is capable to deploy on-demand the LOCUS functions as virtualized functions following NFV principles, facilitating the implementation of the functions in virtualized applications and make them run on commodity infrastructures. As described in deliverable D2.4 [1], the LOCUS functions can run in the virtualization platform as either containerized (i.e., implemented as Docker containers in a microservice approach) or Virtual Machine applications (to also support the traditional Infrastructure as a Service approach).

The LOCUS virtualization platform allows telecom operators and service providers to exploit localization functions for the implementation of new Smart Network Management and New Services applications, and for this allows to consider LOCUS functionality as an augmentation of the 5G capabilities.

3.1 Platform Design

The LOCUS virtualization platform is implemented as a 5G-enabled edge and core virtualized infrastructure. It is built leveraging on multiple virtualization technologies, allowing the execution of different UCs and Proof of Concept (PoC) scenarios. At the same time, it provides support to the LOCUS functions running on top of it to interact with other components of the architecture, including external tools, platform and applications. Considering these requirements, the design approach of the LOCUS virtualization platform is described in the following sub-sections.

3.1.1 *Multiple locations for computational resources*

The LOCUS virtualized infrastructure is based on the 5G distributed cloud architecture principles, as shown in Figure 2. Although the proposed virtualization platform aims at offering a unified and abstracted virtualization infrastructure that can integrate different technologies (that may be selected according to the location where they are deployed), it composed by multiple clusters of compute resources where LOCUS functions and services can be deployed.

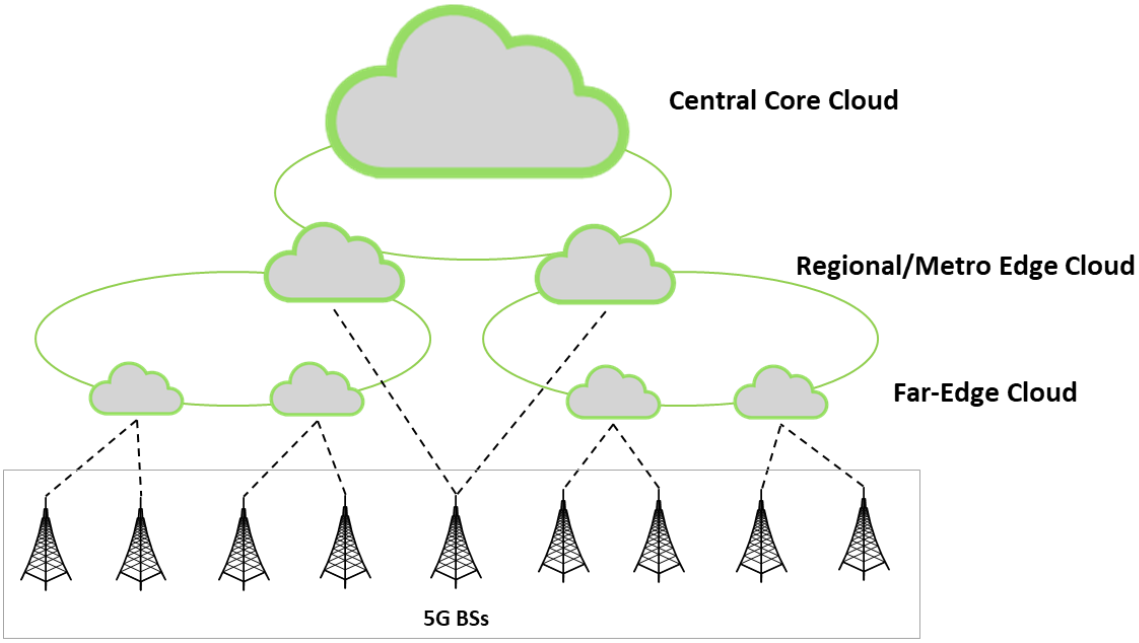


Figure 2: 5G distributed cloud architecture

As shown in Figure 2, 5G requires to distribute compute resources across the whole network infrastructure to support the requirements imposed by vertical industry services (as well as from end-users) in terms of latency, availability and throughput. The traditional cloud paradigm, mostly focused on running applications and services in remote centralized core clouds is increasingly being replaced by a highly distributed architecture, where computing resources are made available at regional/metro level as well as at the far-edge of the 5G

infrastructure to deploy and run the application workloads according to the service needs in terms of proximity with 5G users and devices.

Indeed, the evolution of network and cloud services has brought to more integrated architecture solutions, moving computing resources close to the user, where the physical space for computing hardware is limited when compared with core datacentres where the space for equipment is not a major constraint. The LOCUS virtualization platform is applying the edge/core architecture model to allow the LOCUS functions that require low latency to run in edge nodes and provide their localization or data analytics logic close to the 5G devices (and users). Other functions that require high computing capabilities (e.g.: for machine learning training purposes, or non-real time localization data analysis) can be run in core clouds where the resources are more adequate to their requirements.

3.1.2 *Supported technologies*

The LOCUS virtualization platform is one of the main enablers for the development and realization of the project Proof of Concepts (PoCs), together with the WP4 and WP5 UC functionalities that will run in them. These functionalities come with different technical requirements and should enable the final users or other LOCUS functions, including third-party vertical applications, to interact with them relying on specific characteristics.

Some of these functionalities will have strict requirements in terms of latency, others in terms of amount of computing resources. Other functions are expected to have requirements on their reaction time after a given event occurs (e.g.: spawning time for a LOCUS function).

Based on these heterogeneous requirements, the LOCUS virtualization platform should support different technologies, to allow the localization and analytics functions to perform as expected.

An architecture based on distributed edge and core clouds allows the usage of various technologies, like cloud-native solutions and traditional Infrastructure-as-a-Service (IaaS). If physical space is not an issue in the central core datacentres, the same cannot be said for edge locations, where the space that can be dedicated to the hardware is natively limited. This has an impact on the technologies that can be used in these locations, in terms of lightweight virtualization and computing requirements of the virtualization layer itself to make the LOCUS functions run as virtualized functions.

For this, the LOCUS virtualization platform, makes use of two different technologies for the virtualization of the LOCUS functions. The cloud-native solution, based on containerized applications and Kubernetes [22], as the virtualization management platform for containers, is used mainly at the edge to leverage on agility and lightweight capabilities. The traditional IaaS is used in the central core clouds, with OpenStack [23] as the cloud infrastructure for Virtual Machines based applications.

3.1.2.1 Kubernetes

Kubernetes (K8s) is an open-source platform for managing containerized workloads and services. Containers are similar to Virtual Machines, but they share the operating system among the applications. They have their filesystem while they share the CPU, memory and process space of the hosting machine or server. The major benefit is related to their portability across different clouds and operating system distributions. Figure 3 shows the Kubernetes architecture.

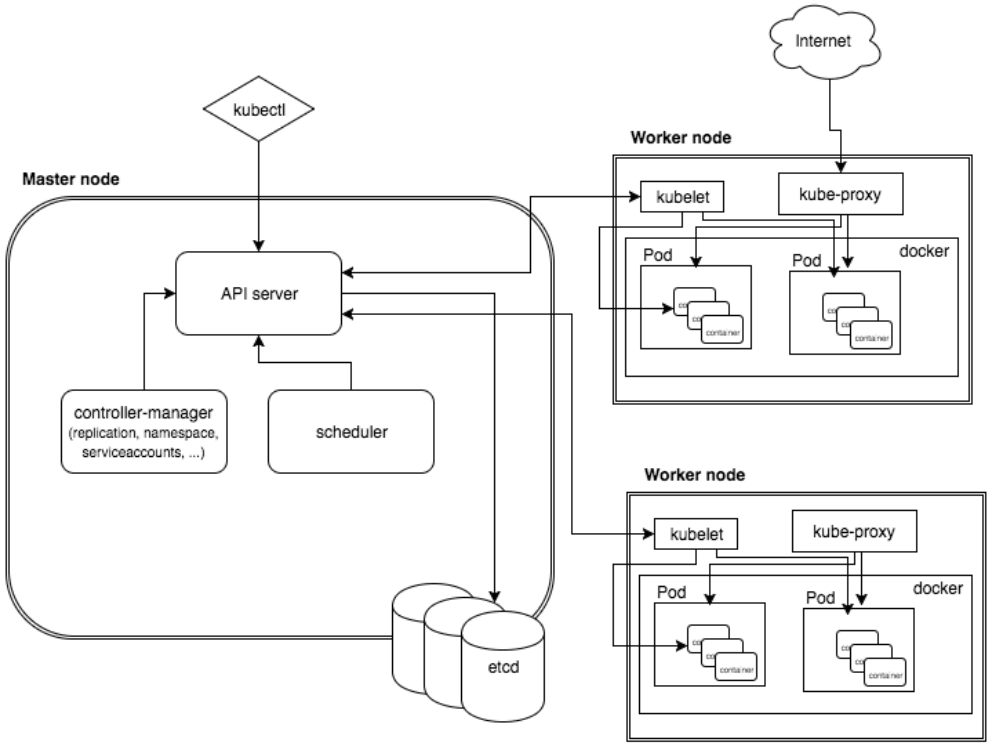


Figure 3: Kubernetes architecture

Kubernetes is organized in clusters; each cluster contains different nodes with different functionalities. The master node is the responsible for the management of the containers running in the cluster. This is the entry point for all those control and management functionalities that interact with the cluster, e.g. to create new Kubernetes services or workloads. In Kubernetes, the containers are run in "*PODs*", which usually run in the worker nodes and share network namespaces and storage volumes. The PODs are scalable objects, and the scaling is managed by the workload itself. A workload set the deployment rules for the POD and different type of workloads exist, based on the specific application requirements. On top PODs, Kubernetes allows to define Services. A Kubernetes Service is an abstraction that groups a logical set of PODs, and which define a specific rule to access them from external entities. This allows to expose one or more PODs (e.g. in terms of IP addresses and ports) depending on the specific deployment constraints (e.g. edge, core, etc.).

The Kubernetes components are classified in two major categories: control plane and node. The more relevant components of the control plane are i) *apiserver*, that expose the Kubernetes APIs, ii) *etcd*, a key-value database where all cluster information is stored and iii) *scheduler*, that is in charge to assign a node to the new created PODs. The node components are: i) *kubelet*, an agent that makes sure each container is running into a POD, and ii) *kube-proxy* that maintains network rules on the node and allows PODs to be exposed as Services.

Kubernetes is the de-facto standard for cloud-native application deployment and management, as it is driven, used and maintained by a wide industry and research community. In LOCUS it is the suitable choice for the virtualized platform and the implementation of the edge computing capabilities, leveraging on its agility, scalability and possibility to spawn applications and functions in short time when compared to Virtual Machine based deployments.

3.1.2.2 OpenStack

OpenStack is a cloud operating system able to control large pools of compute, storage, and networking resources throughout a datacentre. As already mentioned, Openstack is an IaaS cloud operating system solution and it is built as a set of interrelated services, each of them implementing specific tasks related to managed virtualized computing, networking and storage. The high-level architecture of OpenStack is shown in Figure 4.

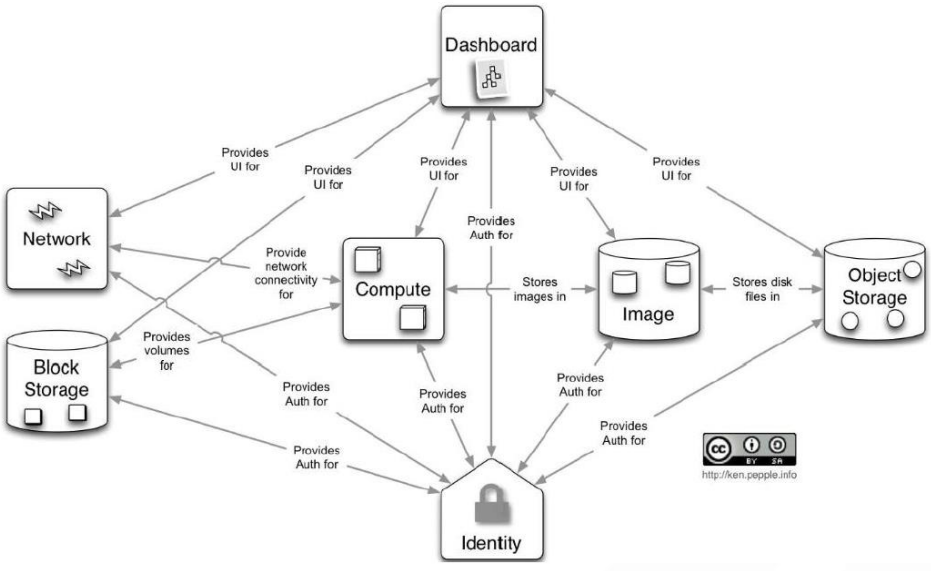


Figure 4: OpenStack High-Level Architecture

Each service offers an application programming interface (API) to facilitate integration with external services and applications. Some of these services are mandatory for each Openstack installation while others are optional. Identity, Network, Image and Compute services are mandatory, and are in charge respectively to handle the authentication and authorization of

users and services in OpenStack, networking among Virtual Machines and between Virtual Machines and external entities, software images, and lifecycle of Openstack instances. The more relevant optional services are the Dashboard and the different Storage services. OpenStack allows having different storage services running on an Openstack deployment. Block storages are persistent, and they can be detached from a Virtual Machine instance and attached to another one, while the data remain intact. On the other hand, Object storage is mostly used when the storage cluster is distributed across multiple datacentres. Openstack supports multi-tenancy, and up to now it is de-facto opensource reference and standard platform for most of the telco and service provider NFV deployments, and it is also used by many telco vendors as part of their commercial NFV products. Being it a widely supported and maintained industry driven opensource platform, Openstack is the suitable choice for the LOCUS virtualized platform and the implementation of the centralized core computing capabilities, leveraging on its scalability and possibility to manage the deployment and operation of Virtual Machines and their interconnection with external entities in a very customizable way.

3.2 Hybrid virtualization approach

The virtualization process for the LOCUS functions goes through the analysis of their requirements in terms of virtualized resources needs, instantiation time or latency. These characteristics are used to differentiate the type of cloud where the function should run. As already mentioned, the LOCUS virtualized platform is based on the distributed edge/core architecture. At the same time, to satisfy some of the requirements, like instantiation time, a different virtualization method should be used. Containers have shorter instantiation time and are easily portable. Multiple functions can run simultaneously sharing the same operating system. On the other hand, Virtual Machines run in dedicated operating system, guaranteeing more security at the cost of increased usage of resources.

Usually, the edge nodes provide limited computing resources, due to the limited space where the hardware is physically installed. On the contrary, the centralized core nodes are allocated in medium/large datacentres with high availability of computing resources.

To address all the possible requirements, with the available resources of the virtualized platform, a hybrid solution, integrating different virtualization techniques can be used to fulfil the varying requirements. The solution, described in detail below consists of the integration of the two virtualization solutions described above, Kubernetes for container management and Openstack for traditional Virtual Machine management, in a distributed and hierarchical approach as shown in Figure 6.

The entities that host the LOCUS functions are depicted in Figure 5. Those functions that are virtualized and packaged as traditional Virtual Machines, run in OpenStack instances. The

others, virtualized and packaged as containers, run in Kubernetes clusters. For each of these, a different Kubernetes POD is created at instantiation time. A Kubernetes POD can contain one or more LOCUS functions, allowing to share storage and network resources among these functions, but at the same time guaranteeing the isolation of the functions.

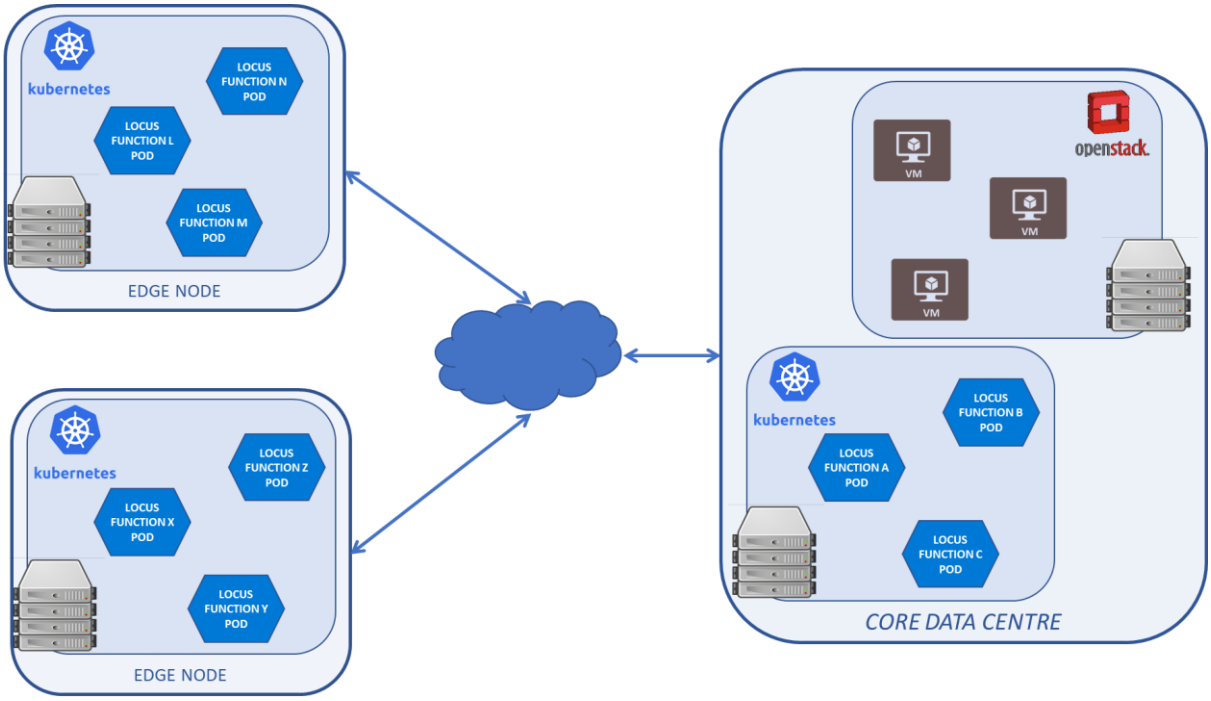


Figure 5: High-level hierarchical hybrid solution

This solution makes use of and combines de-facto standard opensource technologies for virtualized resources management. It has the flexibility to run the LOCUS functions in both edge and core nodes based on the function's requirements.

3.2.1 *Technical solution*

The LOCUS virtualization platform is based on the hybrid approach mentioned above. The LOCUS approach to the hybrid solutions is depicted in Figure 6. It makes usage of two different virtualization technologies, combining cloud-native solutions with IaaS solutions.

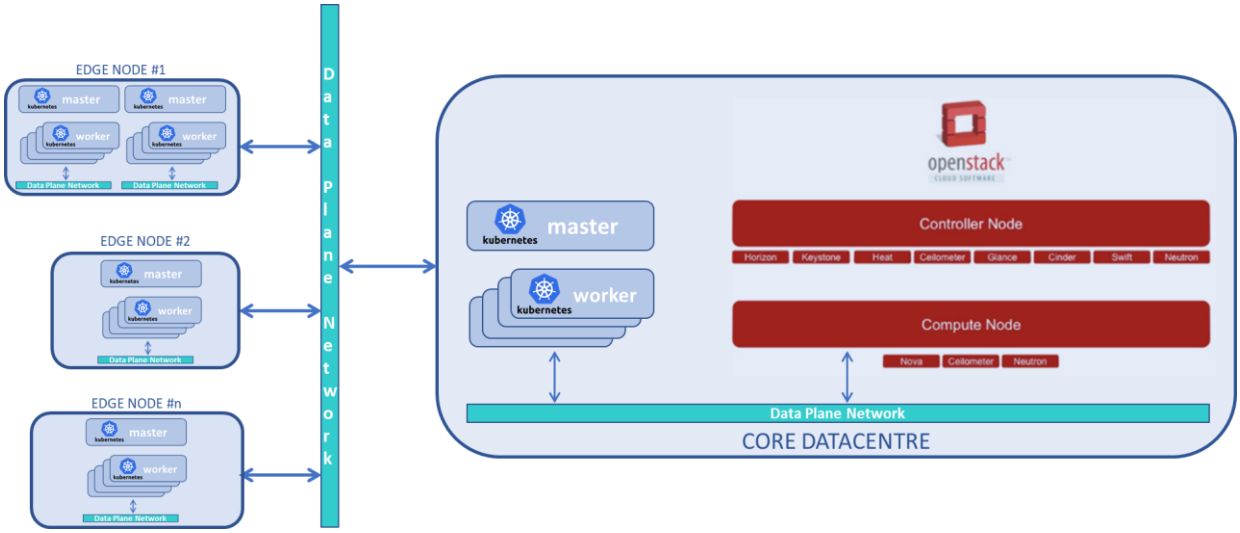


Figure 6: Architecture of a hybrid solution using K8s and OpenStack

The technologies involved are Kubernetes, as a cloud-native solution, that allows managing docker containers into Kubernetes PODs, and OpenStack, an IaaS solution that allows managing traditional virtual machines into OpenStack compute nodes. The LOCUS virtualized platform is making use of a core datacentre, where both Kubernetes and OpenStack solutions are available and different edge nodes. The edge nodes, as depicted in the Figure 6, do not implement the hybrid solution locally, due to hardware restrictions, instead they completely rely on Kubernetes.

The interconnectivity among the Virtual Machines and the Kubernetes PODs is implemented through a shared data plane network that leverages on networking modules available for both Openstack and Kubernetes for the local installations. When the nodes are not directly connected over a controlled data plane network, the nodes can rely on a preconfigured or dynamically established VPN connection, over which LOCUS functions' traffic is encapsulated, ensuring the reachability among workloads that run in different edge or core locations.

3.2.2 *OpenStack networking*

The LOCUS functions can have different networking interfaces, which enable the services related to the functions to communicate with other services or third-party applications. To enable this, Openstack has a core module, Neutron [24], that provides networking between interface devices. It comes with two different networking options:

- Provider networks
- Self-service networks

The provider networks option relies on the physical network infrastructure for layer-3 networking and can be seen as an extension of the infrastructure (external) networks inside the Openstack virtualized environment. When a new provider network is needed (i.e. inside

the Openstack environment), it requires system admin (manual) intervention to extend this network over the Openstack infrastructure. On the other hand, the self-service networks augment the capabilities of the first option by adding support for layer-3 networking, providing the opportunity to create different local virtual networks, and routing these networks to and from external infrastructure networks using Network Address Translation (NAT) techniques. This avoid the need of continuous system admin (manual) intervention when a new self-service network is created in Openstack. In practice, this enables external tools and applications (i.e. acting as Openstack clients) to dynamically create, configure and operate self-service networks to be used for communications among Virtual Machines. The two solutions are depicted respectively in Figure 7 and Figure 8

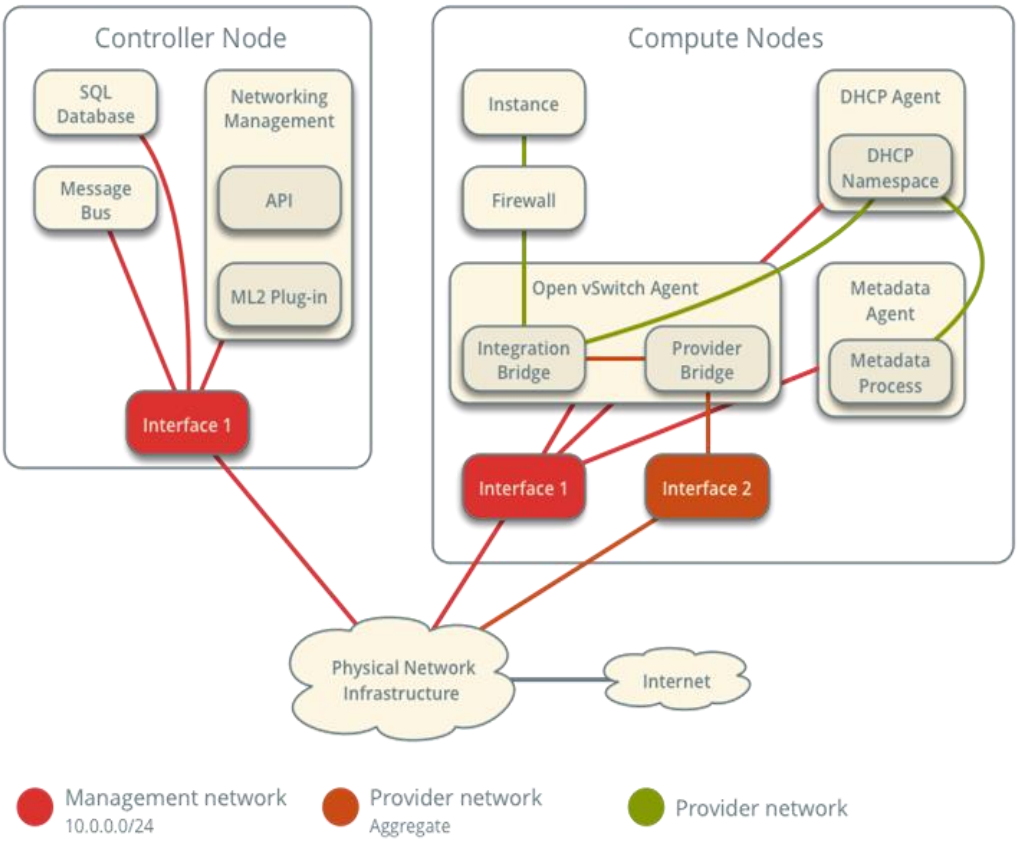


Figure 7: Provider networks architecture overview

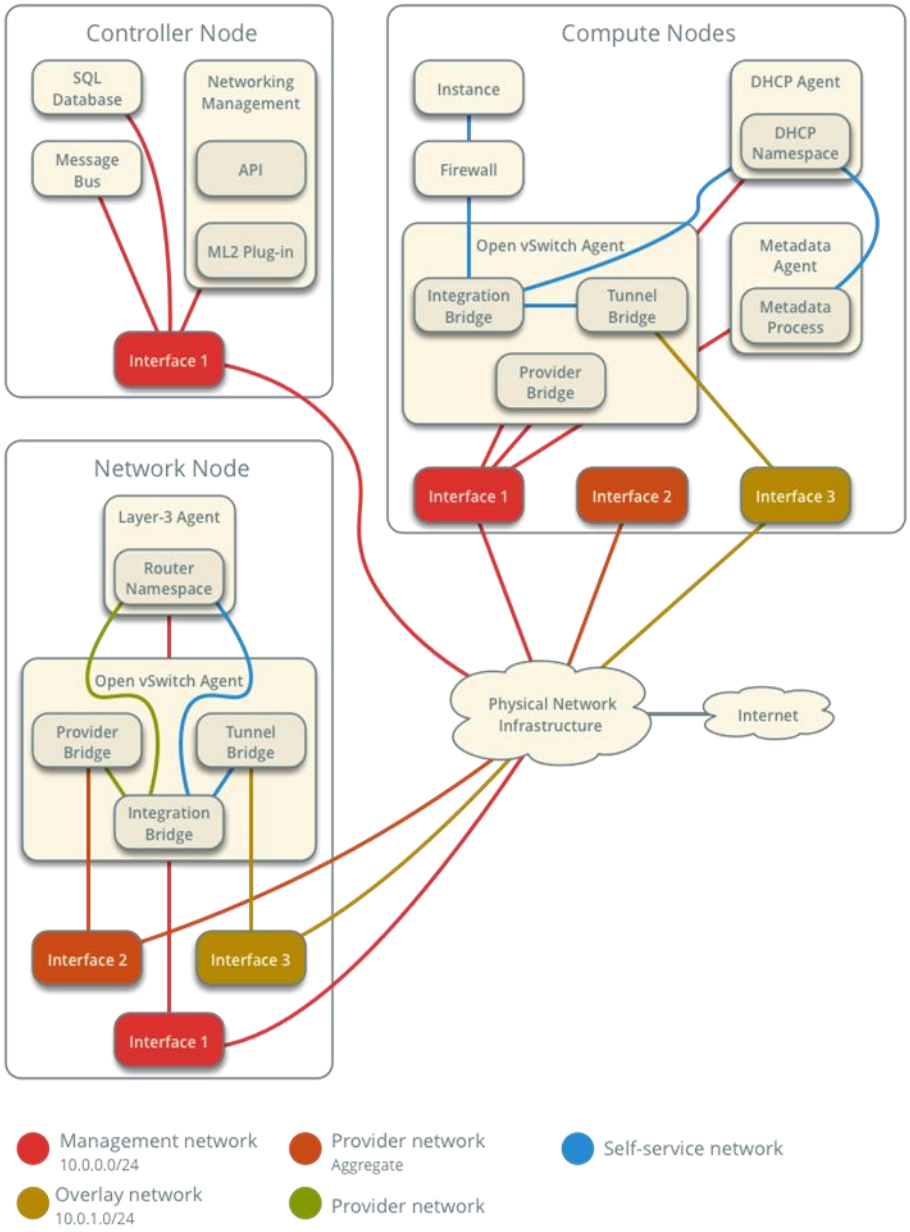


Figure 8: Self-service networking architecture overview

The approach used in the LOCUS virtualized platform, and in particular in the preliminary software prototype described in section 5 and implemented for now in the Nextworks lab infrastructure (section 3.3.3) is a hybrid networking solution. The LOCUS virtualized platform networking is based on the self-service option, that allows external tool (e.g. NFV management and orchestration platforms) to create local networks where to attach Virtual Machines that implement specific LOCUS functions without sharing the connectivity and network traffic with other LOCUS functions, e.g. belonging to different UCs and end-to-end services. The connectivity to and from external services, applications or other LOCUS functions is done by associating "floating IPs" to one of the local networks. The floating IP is associated via Neutron plugin to a Virtual Machine (and this can be done automatically, e.g. through

external tools such as NFV management and orchestration), in a transparent way for the Virtual Machine itself (and the application running in it). These floating IPs allow, on the one hand, a LOCUS function to reach other LOCUS functions running for example in other compute locations (e.g. edge or core). On the other hand, for those LOCUS functions that expose a given service, the floating IPs enable reachability to consume the service or data exposed.

At the same time, the LOCUS functions need to be managed (e.g. for configuration purposes) by the LOCUS management and orchestration functionalities, and other LOCUS platform components. This type of communication is accommodated using well-known networks (i.e.: Virtual Machines management network) that follows the provider network approach. These networks are defined and managed at the infrastructure level and used by OpenStack to enable reachability to the various Virtual Machines for management purposes.

3.2.3 *Kubernetes networking*

The Kubernetes networking covers four main communications aspects:

- Container to container communication
- POD-to-POD communication
- POD-to-Service communication
- External to service communication

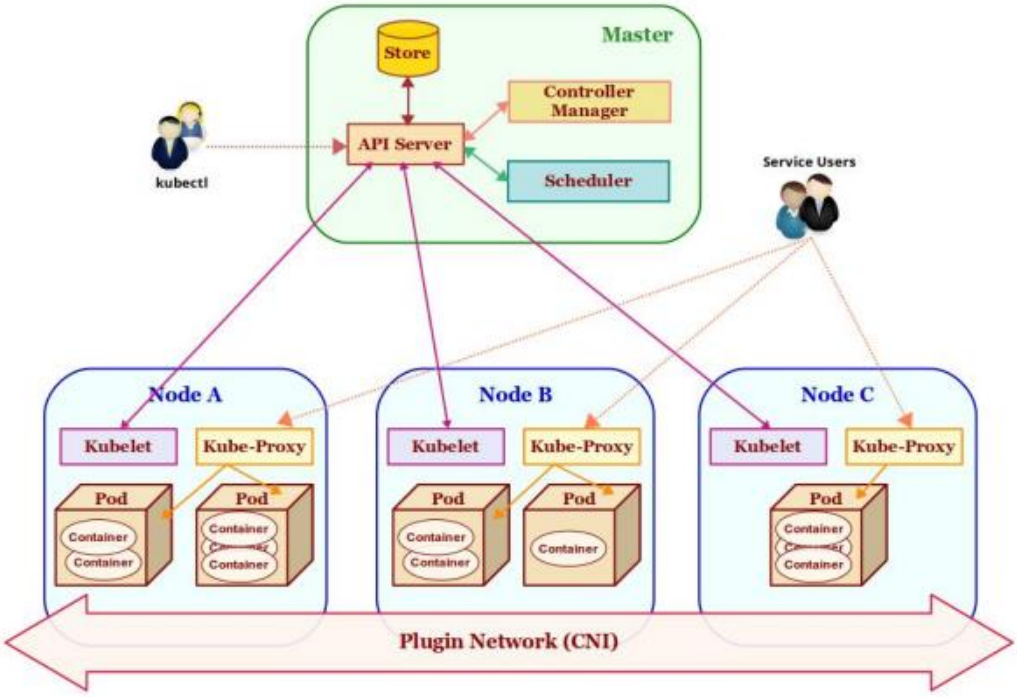


Figure 9: Kubernetes networking

The container-to-container communication is handled by the POD itself. Each Pod is assigned an IP address, derived from the **POD network**. Containers within the same POD share a

network namespace. They can communicate with each other, through localhost, but at the same time, containers running on the same POD should coordinate on port usage.

The POD-to-POD communication is handled by the CNI (Container Network Interface) specification as depicted in Figure 9. The communication among PODs and between agents and pods on the node does not rely on NAT. The purpose of a CNI is to offer a framework for the dynamic allocation of the networking capabilities for container instances running in Kubernetes. In the Kubernetes ecosystem, different CNI plugins exist that enables the networking features and satisfy the requirements of the cluster.

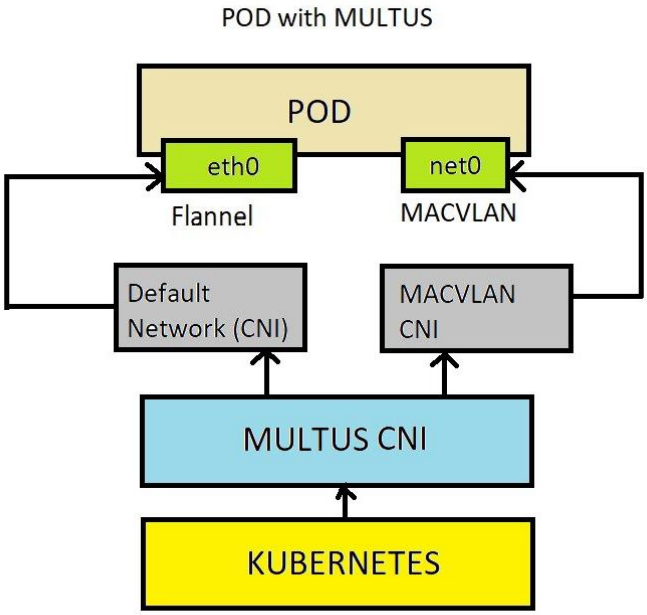


Figure 10: High-level architecture k8s with Multus

The LOCUS virtualization platform uses a subset of these CNI plugins, and in particular:

- ***Flannel***: it creates a flat overlay network which runs on top of the host network (i.e. the network among the nodes in the Kubernetes cluster), this is the so-called overlay network. All containers and PODs are assigned with IP addresses in this overlay network and they communicate with each other by calling each other's IP address directly;
- ***Multus***: it is a multi-CNI plugin and it allows to support multi-networking in Kubernetes. In practice it allows containers and PODs to have more than one network interface. It supports all the reference plugins (Flannel, macvlan, etc) as well as different networking acceleration technologies, like Single-root input/output virtualization (SRIOV) and Data Plane Development Kit (DPDK). The high-level architecture of Kubernetes using Multus plugin with flannel and macvlan is depicted in Figure 10

- ***Macvlan***: it is a specific CNI plugin that allows to directly connect a container or POD to the underlay network.

POD-to-Service communication is handled by the Kubernetes Service itself. It creates a proxy, through the *kube-proxy* service, that is configured to serve a set of PODs. The IP addresses for the Services are different from the ones of the PODs. This networks space is called **service network**.

External-to-Service communication is handled by the Service itself, providing a *ServiceType* parameter that allows to specify the scope. The *ClusterIP* service type makes the Service reachable within the cluster. The *NodePort* service type makes the Kubernetes Service reachable from outside of the cluster.

As for the LOCUS functions running in Openstack, the LOCUS functions running in Kubernetes may need different network interfaces connected to different networks. To allow this, the LOCUS virtualization platform make use of the *multus* and *macvlan* CNI plugins. These plugins allow to create additional container interfaces attached to the underlay network. This configuration has the aim to make LOCUS functions running on Kubernetes PODs reachable from external entities, or other LOCUS functions running in Openstack Virtual Machines.

3.3 Infrastructures Description

The LOCUS project relies on different testbed infrastructures to realize the hybrid virtualization platform approach described above. These are provided by the partners of the consortium, along with the necessary hardware and software resources to make possible the execution of the different WP6 PoCs and UCs functionalities.

Up to now, three different infrastructures have been considered to support the design and implementation of the LOCUS virtualization platform. First, the OTE testbed is being setup and prepared at the time of writing to be the main infrastructure resource for the LOCUS PoCs testbed. Moreover, UMA hosts also a testbed that is equipped with radio access network, based on LTE, and at the same time, there is an ongoing work to augment it with a 5G network deployment. Finally, for the sake of the preliminary prototype activities described in this document, Nextworks has made available its small-scale lab infrastructure. The following sub-sections provide a brief overview of these infrastructures.

3.3.1 *OTE testbed*

This section describes a high-level view and the main characteristics of the OTE's testbed, as well as the main available features. The detailed definition of the tests to be done will be provided in the future WP6 deliverables.

The Figure 11 shows the OTE's cloud testbed architecture.

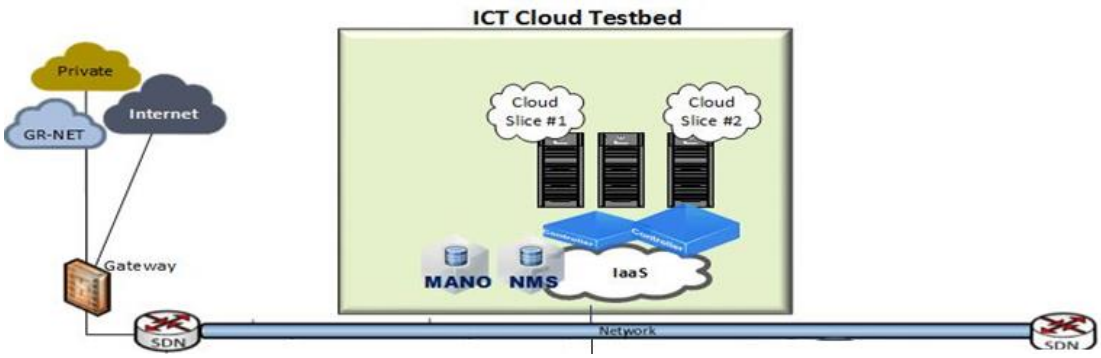


Figure 11: OTE's high level cloud testbed architecture

OTE has deployed and will make available for the needs of LOCUS PoCs an Openstack-based multi-cloud infrastructure. In the current setup OpenStack Pike and Queens releases are available on Ubuntu Server 16.04/18.04 LTS. The testbed collectively consists of >720 CPU cores, >1700GB RAM and >120TB storage space, and is interconnected (mostly) via 10Gbps fiber/copper links. Compute and storage resources can be made available either directly or for hosting relevant services on a per project basis e.g., a CI/CD platform.

More specifically, the setup can be split into one or more cloud slices (controllers/compute nodes/ hypervisors) of various sizes, either in bare metal or virtualized form, and allows high degrees of freedom for customized configurations to meet projects' needs. The OTE testbed will be used in LOCUS to deploy the hybrid virtualized platform described above and enable the execution of functions and services related to the three LOCUS PoCs defined in WP6.

3.3.2 *University of Malaga testbed*

Placed in the Superior Technical College of Telecommunication Engineering, the UMA's testbed encompasses a set of different radio technologies and elements implementing a heterogeneous network:

- LTE Network: A full indoor LTE network with 12 eNBs (Omnidirectional picocells) and a complete LTE mobile core network including various functions: Mobility Management Entity (MME), Serving Gateway (SGW), Packet Gateway (PGW), Home Subscriber Server (HSS) and Policy and Charging Rules Function (PCRF)
- Up to 12 mobile terminals and additional LTE dongles/sticks.
- 3 Wi-Fi APs
- Up to 12 Ultra-Wideband (UWB) devices (including anchors and tags)
- IoT devices: Presence sensors at the doors and neighbouring corridors

Figure 12 shows the scheme of the LTE Network, highlighting the options the algorithms have to interact with the network.

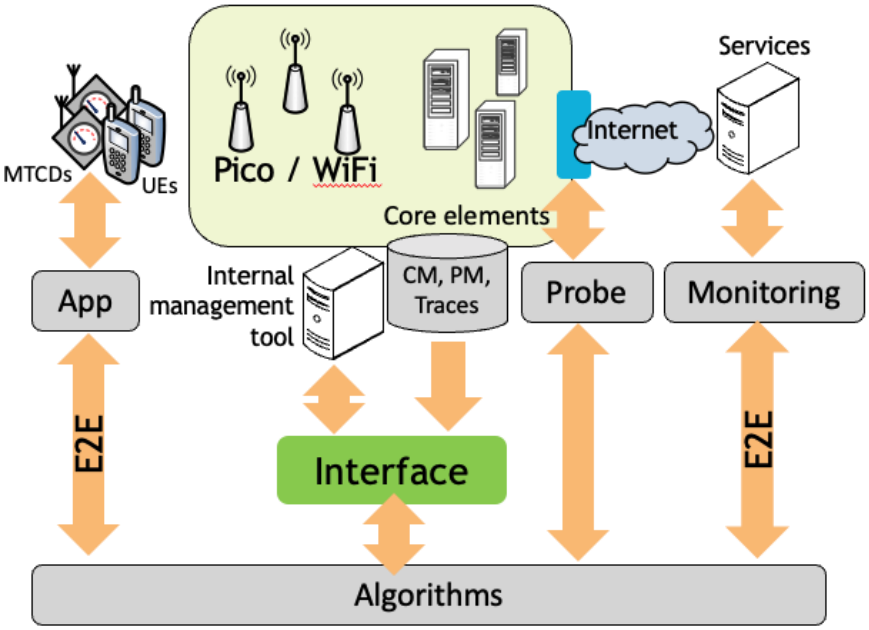


Figure 12: UMA's LTE Network Scheme

Regarding the LTE Network, a RESTful API allows to access measurements and change configuration parameters, such as handover margins (HOMs), transmission power or bandwidth. Multi-layer E2E monitoring can be done by counters, KPIs and traces. The network architecture is detailed in Figure 13, identifying the protocols and message types that are used in each network segment.

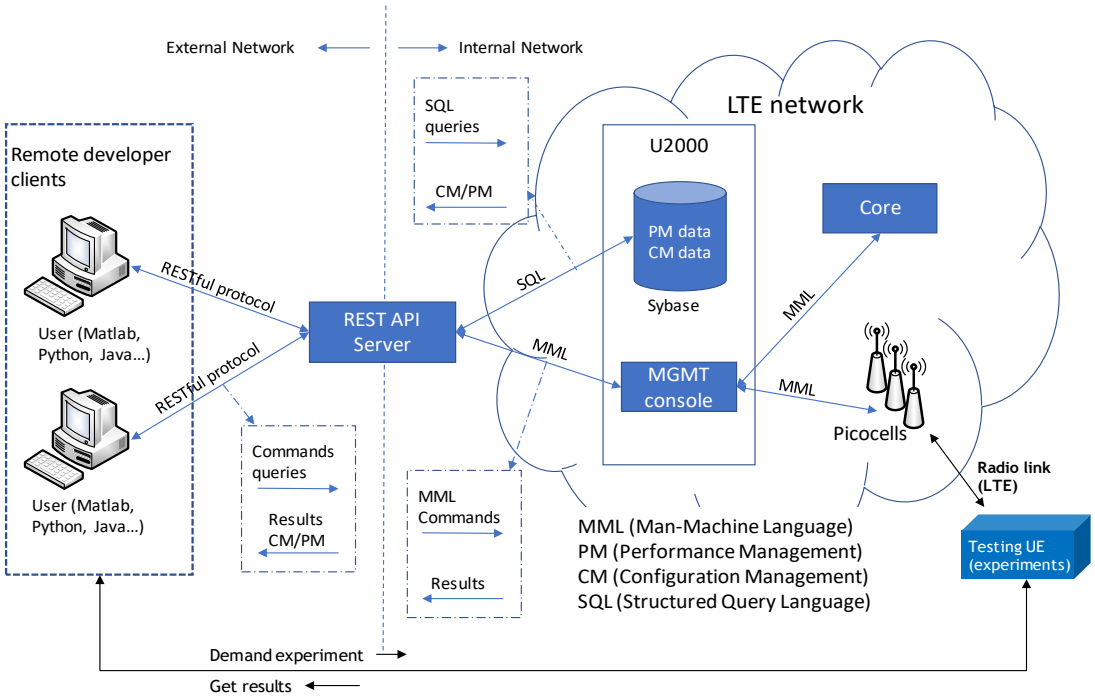


Figure 13: UMA's LTE Network Architecture

From the UE side, measurements can be obtained from different LTE cells and UWB measurements are retrieved in real time through a Bluetooth connection, with a precision down to 10 cm.

In this way the UMA network allows to gather radio measurements and localization information from both the UEs and the cellular network, supporting changes in its configuration at the same time. This will make this testbed a key platform for the testing of SNM algorithms developed in WP4.

3.3.3 *Nextworks lab infrastructure*

During the initial phase of scouting and testing, the Nextworks' lab infrastructure has been used to deploy, investigate, test and validate the virtualization technologies described in previous sections, including the various options for Kubernetes and Openstack interconnection and networking integration.

The Nextworks lab infrastructure, as shown in Figure 14, is physically separated in two areas. The NFV & SDN area, where the preliminary tests and validations with the LOCUS virtualization platform have been performed, and the IoT part, which is out of the scope of this deliverable. The NFV & SDN Area provides a large number of services for NFV & SDN orchestration, like ETSI OSM (in different versions and deployments), DevOps platforms like Gitlab and Jenkins etc. These services rely on the virtualization infrastructure (NFVI) composed by two main nodes.

The core node is based on an OpenStack multi-compute deployment, and the edge node is based on a Kubernetes cluster.

The Kubernetes cluster runs into three virtualized Virtual Machines. In particular, the whole cluster runs on a dedicated physical server, and it is based on Kubernetes v1.19. The cluster is manager by Rancher OS, a Kubernetes Management platform [25].

The OpenStack cluster consists of five physical servers, that represent four computing locations and one controller node, and it is based on OpenStack Queens [26]. OpenStack Virtual Machines and Kubernetes PODs are interconnected through a data plane network. The configuration of this data plane network has been done leveraging on the networking modules already available for OpenStack and Kubernetes, like Neutron [24] for OpenStack and macvlan [27] and multus-cni [28] for Kubernetes.

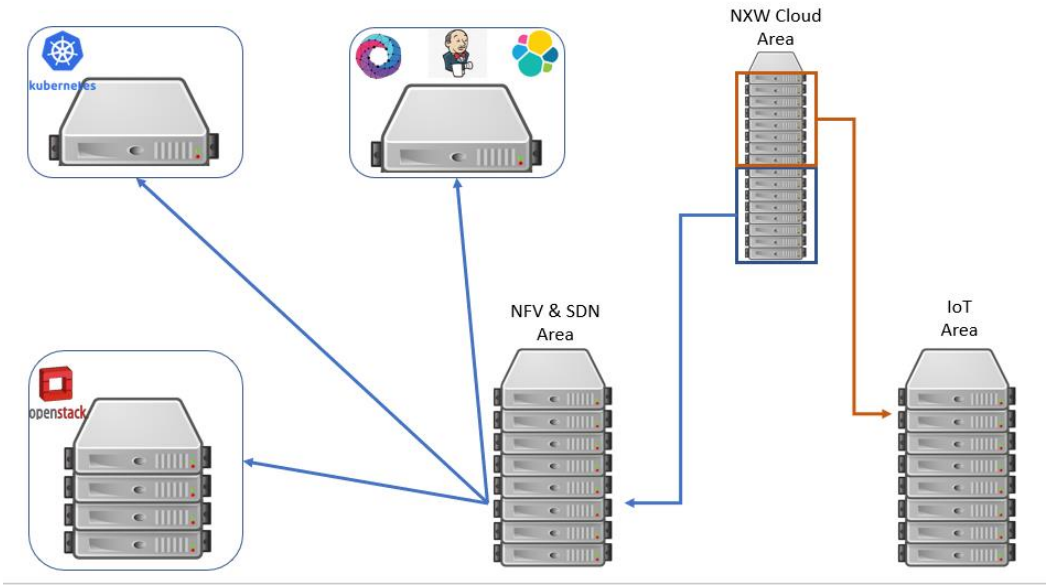


Figure 14: Nextworks' Cloud infrastructure

Based on the initial requirements for the LOCUS virtualized platform, part of the Nextworks' lab infrastructure has been dedicated to the preliminary work reported in this document. In Openstack a new tenant has been created dedicated to LOCUS, to the isolation to and from the other internal projects, and at the same time to dedicate a slice of compute resources to the LOCUS virtualized platform. The Kubernetes cluster is on the other hand dedicated to LOCUS related deployments, tests and validation. In any case, the isolation (e.g. among different services or UC functionalities) is guaranteed by the multiple namespaces defined within Kubernetes cluster.

The setup of the LOCUS-dedicated environment in the Nextworks lab infrastructure has been completed with a dedicated ETSI OSM instance, an OSM Release 9, acting as the LOCUS MANO that has been installed for the initial scouting phase. The physical resources used for this purpose are highlighted in the overall Nextworks' lab infrastructure as depicted in the following Figure 15.

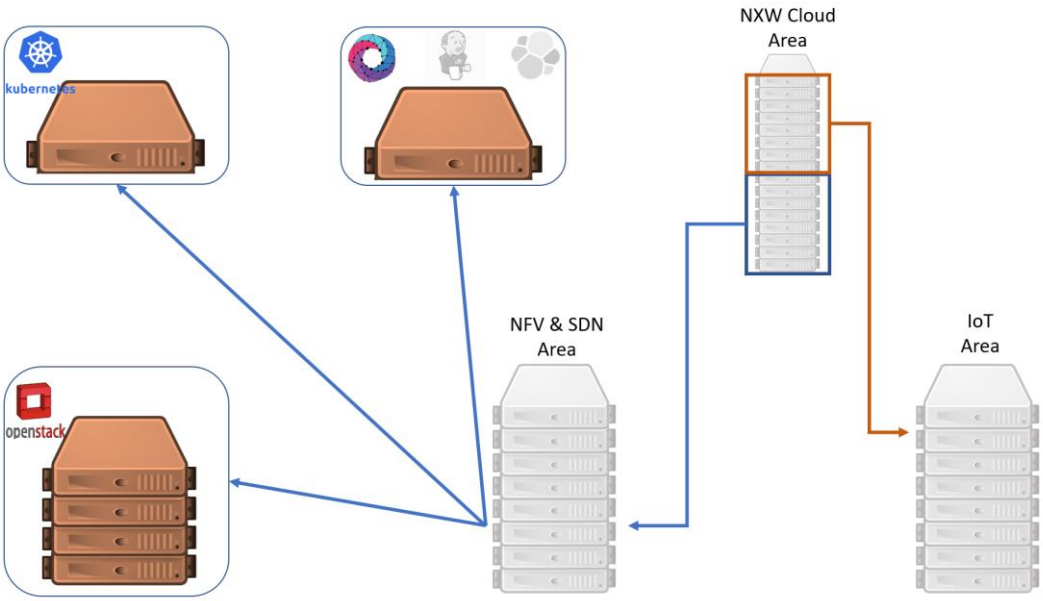


Figure 15: Networks LOCUS scouting testbed

3.4 Future work

The hybrid virtualization platform approach defined for LOCUS and described in the previous sections, with the integration of Kubernetes clusters at the edge and Openstack compute locations at the core of the 5G network infrastructure, has been developed in the form of a preliminary prototype in the Nextworks testbed, as described in section 5.

Beyond this initial work performed in the Nextworks lab infrastructure, that aimed at validating the proposed hybrid solution in a small-scale environment, the final target is to deploy and implement the virtualized platform in the LOCUS WP6 testbed for PoCs validation and demonstration. Therefore, as part of the future work, a close collaboration with WP6 is required to first dimension and then configure the hybrid virtualization platform in a common project infrastructure to foster the PoC activities. In this respect, a joint study and analysis concerning the amount of virtualized resources required to run the various LOCUS functions (developed in WP4 and WP5) in support of the PoCs is already happening. This is where T4.3 is already (and will continue in the next phases of the project) supporting WP6 related activities to operate and maintain the LOCUS virtualization platform.

While the main target infrastructures identified are the OTE and UMA testbeds (as described in section 3.3), some preliminary options for interconnecting them in order to setup an edge/core scenario are currently being evaluated. This would allow to have a realistic scenario, as UMA is enhancing its testbed with 5G capabilities (i.e. 5G new radio and devices) and could possibly host a dedicated edge node to run LOCUS functions on site. A site-to-site VPN, as depicted in Figure 16 will be considered to enable LOCUS functions running in OTE testbed to reach the services exposed at the UMA side, and vice versa. The potential impact of such kind

of solutions (e.g. in terms of possible latency, bandwidth bottlenecks, etc.) will be investigated as part of the WP6 testbed preparation work, with full support from T4.3 in terms of required adjustments and specific configurations at the level of the virtualized platform.

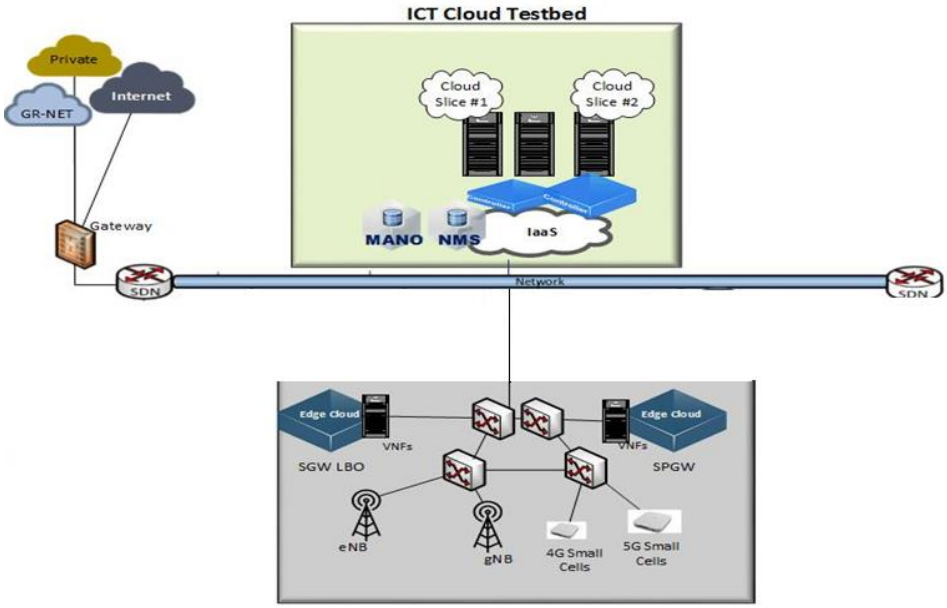


Figure 16: Interconnection between UMA and OTE testbeds

4 LOCUS Management and Orchestration

The LOCUS Management and Orchestration (MANO) is a crucial component within the LOCUS platform and enables the use of the virtualization platform to deploy and operate the LOCUS functions as virtualized functions across edge and core locations.

While the provisioning of localization related services is currently mostly performed as a chain of manual operations, including deployment, configuration and operation of localization functions and consumption of their output, the main target of the LOCUS MANO is to automate the full lifecycle of these localization functions. This way, different combinations of localization and analytics functions can be easily composed as services, and they can be exposed through the Localization Analytics as a Service APIs that are under definition and implementation in WP5.

Beyond that, the LOCUS MANO is a key enabler for the Smart Network Management (SNM) functionalities and applications described in D4.1 [29]. It allows deploying the localization and analytics services on-demand as combination of virtualized functions and expose the produced analytics and geolocation related data as a service, to be consumed by the SNM applications. On the other hand, the LOCUS MANO itself is a fundamental smart management tool and it guarantees automated provisioning and operation of localization analytics functions in virtualized environments.

The LOCUS MANO allows the LOCUS platform to align with current trends in managing and orchestrating virtualized infrastructures in 5G and beyond including various scenarios and infrastructures. Indeed, 5G network management and orchestration requires a high level of automation and integration with NFV principles to dynamically and flexibly deploy and operate 5G services for verticals. In this context, LOCUS provides innovative complementary functionalities for the provisioning and exposure of localization functions as services, fully compliant with the virtualization of network functions and services happening in 5G with very distributed deployments across edge and core locations. Therefore, it can be seen as an add-on with respect to the classical NFV MANO tools that take care of the provisioning of pure virtualized network related services and functions only.

For this reason, with respect to the original system architecture presented in D2.4 [1], the LOCUS MANO is now evolved towards a more comprehensive platform integrating Smart NFV Management functionalities, as described in the next section.

Moreover, while the LOCUS MANO is part of the Smart Network Management technological area in WP4, it covers management and orchestration features for the analytics and ML functions developed in WP5 as well, and that are conceived to be deployed and run in the common LOCUS virtualization platform.

4.1 Principles and architecture

With respect to the LOCUS system architecture presented in D2.4, and recalled in the introduction chapter, some advancements and evolutions are taking place in WP2 at the time of writing this deliverable, to properly map and integrate the LOCUS platform with the 3GPP 5G architecture. Beyond that, NFV MANO functionalities are key enablers for automated management of 5G network functions and services as part of the network slicing paradigm and are indeed essential in the 5G network management architecture: 3GPP [2] and ETSI NFV [3] share a common approach in presenting NFV network services (i.e., the chaining of VNFs to realize a consistent service) as a resource-centric view of a network slice, thus mapping the network slice to one or more recursive NFV network service.

Following this approach, the LOCUS MANO provides additional capabilities to manage and coordinate the provisioning and operation of the localization analytics functions and services in support of Smart Network Management and 3rd party vertical applications, e.g., as additional per-network-slice functionalities to be offered on-demand.

In this respect, LOCUS lays its foundations on the separation of concerns principle, where the SNM functionalities are to be considered as LOCUS applications (i.e. sitting on top of the LOCUS platform) which focus on exploiting analytics and geolocation related data in order to enable an advanced, smarter and AI/ML empowered management of radio resources in the 5G infrastructure. Similarly, the 3rd party vertical applications exploit the analytics and geolocation related data to improve the end-user New Services (NSE) they offer. On the other hand, the LOCUS platform, through its MANO component, allows to manage the localization and analytics service on top of virtualized infrastructures while exposing the analytics data as a service towards the LOCUS applications, leveraging on the Localization Analytics as a Service APIs.

This is aligned with current evolution trends of Network Management Systems (NMSs) and OSS platforms towards cross-layer management and orchestration frameworks that coordinate end-to-end network slices, network and physical infrastructure resources, NFV services and virtualized infrastructure leveraging on different management functions. As shown in Figure 17, such cross-layer approach provides a complete integration of 5G radio resources and infrastructures, Software Defined Networking (SDN) enabled network resources and segments, NFV enabled core and edge virtualized infrastructures. Therefore, end-to-end service and slice management functions are consumers, among the others, of the NFV MANO to request, for the deployment of virtualized services and functions at edge and core computing locations in support of the NFV Network Service resource-centric view of a network slice described above. At the same time, end-to-end service and slice management functions interface with 5G radio equipment and controllers, as well as with SDN enabled network controllers in backhaul and transport domains to deliver end-to-end connectivity

services in support of end-to-end network slices. In this context, the LOCUS platform, and as a consequence the LOCUS MANO, becomes an enhanced NFV MANO framework able to manage localization analytics services beyond the traditional NFV services (e.g. those providing 5G network functions and services) required to implement end-to-end network slices. Such innovative LOCUS services can be then consumed for two main purposes: a) Smart Network Management at the level of end-to-end service and slice management, that will be able to offer advanced 5G radio resource management capabilities, as well as b) 3rd party vertical applications that will be able to improve vertical services with localization-based analytics data processing.

For this reason, LOCUS considers the SNM functionalities developed in the context of UC2 (Location-Aware Planning of 5G Networks), UC3 (Network Optimization based on Geolocation Knowledge) and UC4 (Location-Aware Network Diagnosis/Troubleshooting in 5G), as LOCUS applications, thus running outside the LOCUS platform (and from a logical perspective integrated within a cross-layer management and orchestration framework) but still within the scope of LOCUS. They leverage on the Localization Analytics as a Service APIs to request for localization services and consume the related output analytics data. On the other hand, in line with D4.1, the SNM functionalities developed in the context of UC1 (Knowledge Building for Network Management functionalities) represent exactly those localization analytics services (which offer analytics and geolocation related data) required by the other SNM UC functionalities to perform their 5G radio resource management operations.

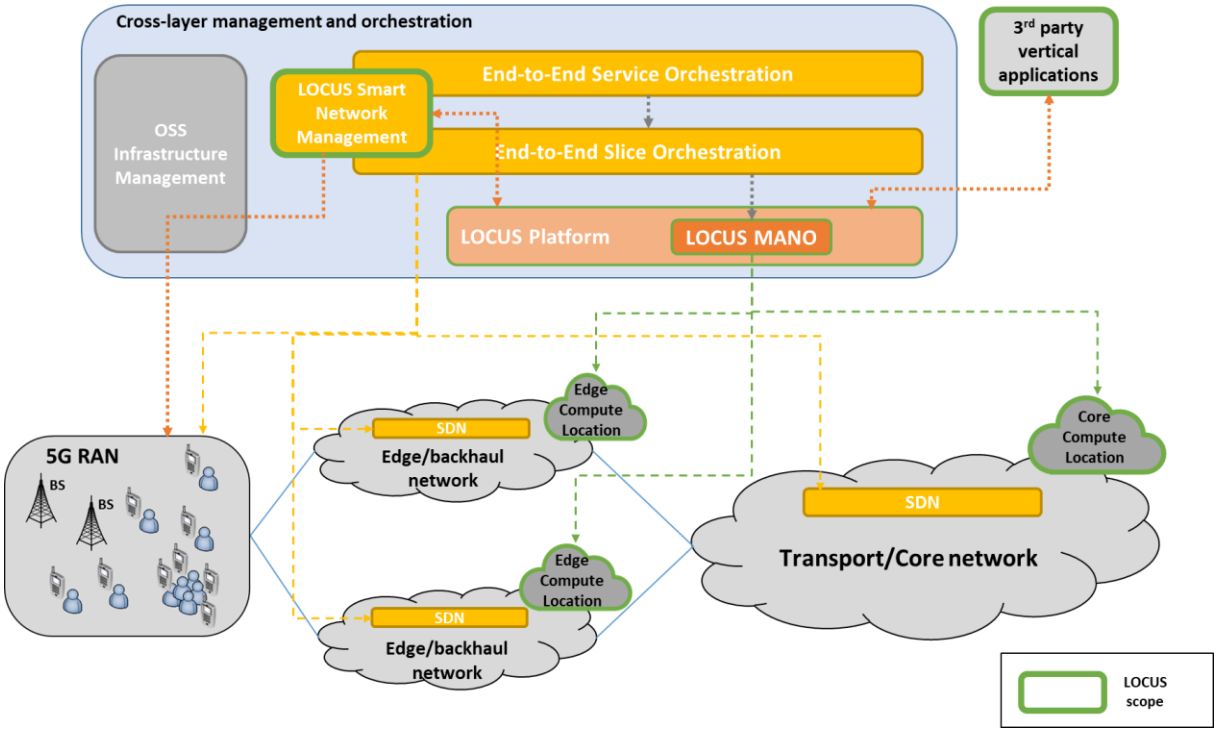


Figure 17 LOCUS positioning in 5G end-to-end network management and orchestration scenario

However, current NFV MANO solutions are increasingly moving towards fully automated smart management frameworks themselves, that aims at closing the management and control loops for optimizing the network functions and services runtime operation. This is fully aligned with the LOCUS Smart Network Management principles and functionalities reported in D4.1, and therefore it is clear how advanced management functionalities can be integrated as part of the LOCUS MANO as well. This drives an alignment with the current trends of AI/ML enabled NFV MANO systems that use data analytics techniques on top of heterogeneous data (e.g. obtained by monitoring the various virtualized resources, functions and services as well as by collecting statistics from the network functions) to make service design, runtime optimization and failure management more effective and efficient in the NFV domain.

As a result of these considerations, Figure 18 focuses on one part of the LOCUS platform architecture (with respect to the one presented in D2.4) to show where and how the LOCUS MANO is evolved towards a more comprehensive framework that is built of three main building blocks: NFV MANO, Monitoring, Smart NFV Functions. The NFV MANO (which corresponds to the previous LOCUS MANO defined in D2.4) provides the basic functionalities for lifecycle management of localization analytics functions and services, thus taking care of their on-demand deployment. The Monitoring platform allows collecting data and statistics from heterogenous sources, including network and computing virtualized resources as well as network and localization and analytics virtualized functions, and make them available to the other LOCUS MANO functions for advanced performance management, fault management, runtime optimization, etc. The Smart NFV Functions are now integrated with the LOCUS MANO and make it smarter and more automated in how the various services, functions and virtualized resources are operated at runtime, by leveraging on data offered by the monitoring platform and covering different aspects of their management (design, runtime optimization, fault management).

While, as said above, the overall LOCUS system architecture is currently being revisited in WP2, mostly in terms of mapping and integration of the various components with the 3GPP 5G architecture and network functions, including APIs and data exchanged among them, Figure 18 intentionally shows a simplified view of the 5G system functionalities and how they relate with the various LOCUS localization and analytics functions (called simply LOCUS functions in the figure). Similarly, the Security & Privacy functions, and the Localization Analytics Gateway are those defined in D2.4 and implemented in WP2 and WP5 and kept in the figure at high level as they are not directly part of the work described in this document. Still, the adoption of the Service Based Architecture (SBA) approach is valid for the LOCUS architecture shown in Figure 18, thus each LOCUS function provides a cohesive atomized localization or analytics functionality and can be independently managed from other functions, producing specific outputs based on specific (data) inputs.

The following sections provide a detailed description of the three main functional blocks of the LOCUS MANO, namely the NFV MANO, the Monitoring platform and the Smart NFV Functions.

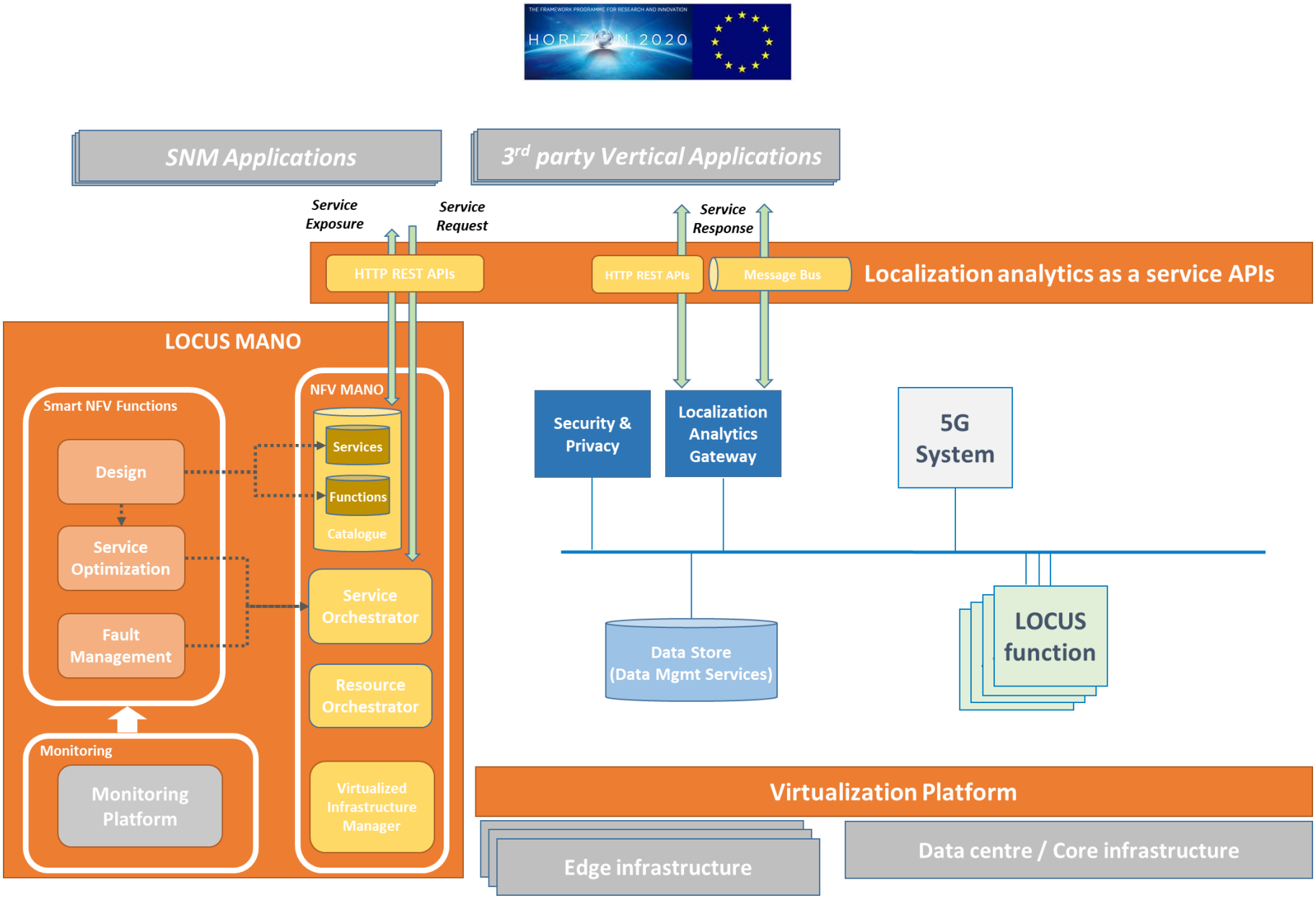


Figure 18 Evolved LOCUS MANO in the LOCUS platform

4.1.1 *LOCUS NFV MANO*

The LOCUS MANO builds its foundation on the ETSI NFV principles, which allows managing the localization and analytics functions developed in WP3, WP4 and WP5 as VNFs to be deployed in the virtualization platform. Moreover, combination of localization and analytics VNFs can be modelled as NFV Network Services to realize more complex analytics functions, such as machine learning pipelines.

Therefore, the NFV MANO can be considered as the core engine of the LOCUS MANO as it practically realizes the automation in delivering on-demand localization analytics services, to be exposed through the Localization Analytics as a Service APIs, and feed with their monitoring data the Smart NFV Functions for implementing advanced management routines, including design, runtime optimization and fault management for the various virtualized services.

As shown in Figure 19, the LOCUS NFV MANO is compliant with the standard ETSI NFV architecture [30], and is composed by four main functions: the Catalogue, the Service Orchestrator, the Resource Orchestrator and the Virtualized Infrastructure Manager (VIM).

This functional split is inspired by the ETSI OSM opensource platform [31], which is the ETSI-driven NFV MANO reference implementation that is supported by a wide industry community. ETSI OSM is the reference baseline for the software implementation of the LOCUS NFV MANO, as detailed later in this document. Adopting such kind of approach has the main objective of aligning LOCUS with a de facto standard NFV MANO platform like ETSI OSM, that can assure an easier and facilitated integration with the 5G core network functions and 5G services in general. Indeed, ETSI OSM is already capable to deploy and manage 4G/EPC virtualized services [32] based on the Magma opensource software platform [33] and automatically manage a mobile core network. Thus, it will be able to do the same with (opensource) 5G core VNFs and services as soon as they will be available; in addition, the integration of the LOCUS localization and analytics service will be facilitated, following the strategy that will be defined in WP2 as part of the final LOCUS architecture. Moreover, ETSI OSM has already generated a wide ecosystem where heterogeneous industry and research-related activities are taking place [34], including development of compatible VNFs and VIMs, consulting and integration services, research initiatives, as well as industry driven Proof of Concepts in relevant 5G environments [35]. The following sections provide a brief description of the LOCUS NFV MANO main functional components.

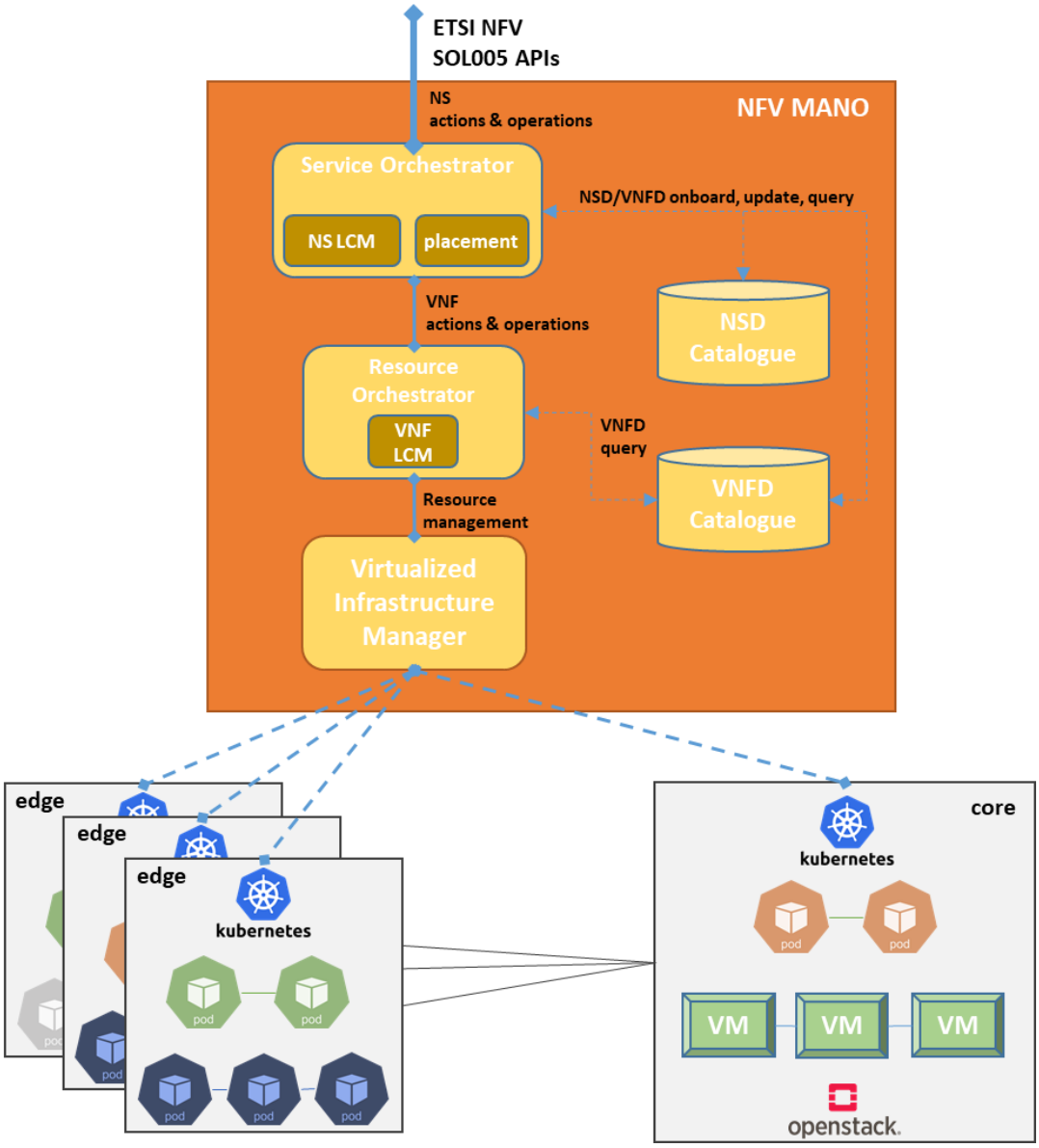


Figure 19 LOCUS NFV MANO architecture

4.1.1.1 Catalogue

The Catalogue is the centralized repository service that hosts the description of the VNFs and Network Services that can be managed by the NFV MANO and therefore instantiated on demand on the virtualized platform. In LOCUS, it is therefore the database of the available localization and analytics functions and services that are modelled following the ETSI NFV standard data model for VNFs and Network Services, which are called VNF Descriptor (VNFD) and Network Service Descriptor (NSD) respectively.

In line with the latest release of the ETSI OSM software platform, i.e. Release 9, LOCUS supports the ETSI NFV SOL006 specification [36] for VNFDs and NSDs, which is based on a YANG data model [37]. In particular, both VNFD and NSD data models provide the necessary

information to the Resource Orchestrator and Service Orchestrator respectively, to perform their lifecycle management routines on functions and services, including allocation of required virtualized resources,

Figure 20 shows the VNFD data model tree (with only the top-level attributes information highlighted); the main relevant attributes for VNF lifecycle management in LOCUS (beyond those for identifying a VNF name, provider, version, etc.) can be briefly described as follows:

- *vdu*: it is the constituent element of the VNF, as it represents the Virtual Deployment Unit, that in practice is a self-contained component of a VNF. At minimum, a VNF is composed by one vdu, but it can have more vdus when the VNF is designed as a more complex application (i.e. more components that are interconnected). A vdu is described in terms of its requirements in terms virtualized resources required, by referring to virtual-compute-desc and virtual-storage-desc attributes;
- *virtual-compute-desc:* it provides the requirements for one or more vdus in terms of computing resources (i.e. virtual CPUs) to be allocated in the virtualized platform for properly running the VNF;
- *virtual-storage-desc*: it provides the requirements in terms of storage resources (i.e. virtual disk) to be allocated in the virtualized platform for properly running the VNF;

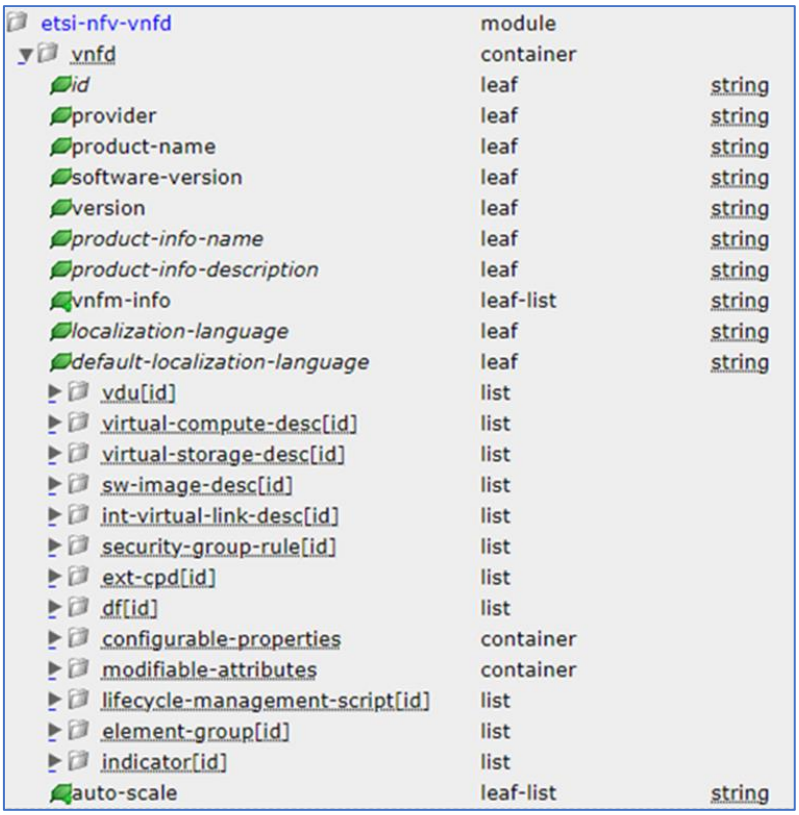


Figure 20 VNFD data model

- *sw-image-desc*: it provides the reference to the software images required to deploy and run the VNF in the virtualized infrastructure. The software image can be either a Docker image (to run the VNF as one or more Docker containers in a microservice-like deployment) or a Virtual Machine image (to run the VNF as a more Infrastructure as a Service-like deployment)
- *ext-cpd*: it represents and describes the external connection point, that is the virtual network port resource to be allocated in the virtualize platform to enable the VNF to be interconnected with other VNFs, or in any case with any external service (with respect to its own virtualized running environment, i.e. a Docker container or a Virtual Machine). They can be of different nature, e.g. for management or data traffic purposes;
- *df:* it describes the list of deployment flavours available for the VNF, that in practice is a list of options in terms of which and how many vdus to be instantiated in the virtualized platform, with specific requirements in terms of virtualized resources required. This means that the same VNFs can be deployed, and scaled, in different ways (e.g. with different compute and storage requirements) if multiple deployment flavours are available
- *lifecyle-management-scripts*: these scripts provide a set of instructions and primitives to be run in the VNF when a specific lifecycle management event happens (e.g. soon after VNF instantiation, upon VNF scaling, before VNF termination, etc.)
- *indicator*: it provides the list of KPIs the VNF can offer for performance monitoring purposes to the NFV MANO. They can be related to specific VNF application statistics.

On the other hand, Figure 21 shows the NSD data model tree (again with only the top-level attributes information highlighted); the main relevant attributes for the Network Service lifecycle management in LOCUS can be briefly described as follows:

- *vnfd-id:* it represents the list of VNFs that are part of the Network Service. Each VNF is identified with its own VNFD identifier;
- *nested-nsd-id:* it is an optional list of other NSDs that are required to deploy the current Network Service. It follows the concept of nesting and composition of Network Services, that allows to create complex Network Services as the combination of simpler Network Services;
- *sapd:* it is the list of Service Access Points for the Network Service. In practice they are mapped to virtual network resources (e.g. ports associated to a VNF ext-cpd) that give access to the Network Service from any external service. They can be of different nature, e.g. for management or data traffic purposes;

- *virtual-link-desc:* it provides a description of the virtual links interconnecting the various VNFs that form the Network Service (including QoS requirements), as well as those required to expose the Network Service towards any other external service.
- *vnffgd:* it stays for VNF forwarding graph descriptor, and it describes how the various VNF in the Network Services are interconnected in terms of data traffic forwarded among them, if required. It allows the Resource Orchestrator to enforce the proper configurations in the virtualization platform for enabling the data traffic to flow across the VNFs in the desired order.
- *lifecycle-management-script:* these scripts provide a set of instructions and primitives to be run in one or more VNFs (possibly with cross-VNF dependency in terms of parameters or values) when a specific lifecycle management event happens (e.g., soon after Network Service instantiation, upon Network Service scaling, before Network Services termination, etc.)
- *df*: it describes the list of deployment flavours available for the Network Services. In this case, it represents a list of options in terms of combination of profiles for the different VNFs to be deployed in the virtualization platform, with specific requirements in terms of number of instances for each VNF, their own deployment flavours to be used, and possibly the vnffgd to be applied. This means that the same Network Service can be deployed, and scaled, in different ways (e.g. with combination of VNFs) if multiple deployment flavours are available.

Figure 21 NSD data model

Beyond the standard ETSI NFV SOL006 descriptor attributes described above, the LOCUS MANO, in line with the ETSI OSM approach, supports additional features (called YANG model

augmentations) that allow to extend the VNFD and NSD descriptors with enhanced capabilities to enable a VNF to be modelled as a Kubernetes Network Function (KNF), and thus be natively and transparently deployed in Kubernetes clusters by leveraging on the integration with Kubernetes Helm charts descriptors [38], as described in section 5.

The Catalogue is fed with VNFDs and NSDs during the localization and analytics functions and services onboarding processes, that in practice allow to upload in the NFV MANO new or updated descriptors for functions and services. This procedure is managed by the Service Orchestrator through its northbound APIs (as depicted in Figure 19) in accordance with the ETSI NFV SOL005 NSD and VNF Package Management procedures [39]. Once onboarded, NSDs and VNFDs maintained in the Catalogue can be possibly updated at runtime upon trigger from the Smart NFV Management design related functions, whenever existing localization analytics service and function descriptors could be improved (e.g. in terms of virtualized resource capabilities) to better support actual service runtime requirements and behaviour in the virtualization platform.

The Catalogue accepts the VNFDs and NSDs in the form of packages, which are software archives (e.g. zip files) that contain the descriptor files themselves (in the ETSI NFV SOL006 format) together with additional metadata information. The process of building a VNFD or NSD related packages is typically an offline procedure, that requires the execution of several steps to properly design and implement both the software images and the VNFD/NSD descriptors, as described in detail in section 4.3.

4.1.1.2 Service Orchestrator

The Service Orchestrator is the core localization analytics service engine of the LOCUS NFV MANO; it takes care of the localization analytics service coordination, in practice by orchestrating a set of lifecycle management steps for deploying in the virtualization platform the individual localization and analytics functions modelled as VNFs in the given Network Service. Indeed, the Service Orchestrator builds its coordination logic on the content of the NSD that models the given localization analytics service, in terms of which location function VNFs have to be created and how they should be interconnected according to the virtual link and topology requirements expressed in the NSD.

As mentioned in the previous section, an NSD can reference other NSDs to build complex localization analytics services by combining atomic services. For this reason, the Service Orchestrator can manage Network Services as a combination or a composition of multiple Network Services. This allows to possibly re-use and share existing localization analytics service instances as part of more complex (or even distributed across edge and core locations) ones following either a nesting (i.e. hierarchical integration of Network Services) or composition (i.e. in-line Network Service combination) approach. In particular, the Service

Orchestrator manages the localization analytics services according to the principles described in section 4.2.

As part of its coordination logic, the Service Orchestrator implements a VNF placement logic with the aim of having an optimized deployment of functions and services at specific locations (in terms of available edge and core computing locations). Such localization and analytics VNF placement functionality can consider several constraints as part of the optimization process:

- virtualized compute and storage resource requirements of the VNFs
- interconnection and virtual link requirements for the Network Service (e.g. latency)
- cost for using virtual resources in the given edge or core location.

Of course, the placement optimization can be performed based on multiple criteria, that could be aiming at minimizing or maximizing specific objective functions, such as:

- minimization of virtual resource usage at higher cost edge location
- maximization of virtual resource usage at cloud location
- minimization or maximization of distribution of VNFs for the same Network Services across multiple locations.

Beyond the above high-level principles, the localization and analytics functions and services placement optimization procedures and logics are planned to be defined and implemented in the following stages of the project and described and released as part of deliverable D4.4. By the way, to implement such placement optimization functions, the Service Orchestrator need to access the information of the localization and analytics functions VNFDs stored in the Catalogue to retrieve the specific requirements for their instantiation.

In practice, the Service Orchestrator is the overarching LOCUS NFV MANO service coordination entity, and it leverages the Resource Orchestrator to actually deploy the various VNFs (with the related virtualized resources) in the virtualization platform. There is therefore a clear split between the service and (virtualized) resource layers management.

As part of the localization analytics services runtime operation, the Service Orchestrator allows service modifications to be requested by the Smart NFV Functions for automated service optimization. In particular, following the ETSI NFV management procedures (and aligned with the NSD and VNFD capabilities described in the previous section), the Service Orchestrator supports the following Network Service runtime modifications:

- service scaling: for scaling in or out specific localization function VNFs (i.e. by adding or removing entire VNF instances), e.g. to react to high load or consumption of computing resources,
- service update: to move from one Network Service deployment flavour to another, possibly implying a change in the service topology (e.g. a new VNF type to be instantiated, or a new nested Network Service to be chained to the original one).

While the closure of the automated control and management loop is not a key requirement and target feature in LOCUS, the LOCUS MANO architecture is anyway supporting it from a design perspective, to make it aligned with current trends of full automation in 5G service management and operation.

With respect to the ETSI NFV architecture [30] the Service Orchestrator is mapped to the NFV Orchestration functional component, and for this it is the main and unique entry point of the entire LOCUS MANO functional box for what concerns all of the actions that relate to the localization analytics services, including onboarding of descriptors and lifecycle management operations (instantiation, scale, update, termination). For this purpose, the Service Orchestrator is compliant with the ETSI NFV approach, and it supports the ETSI NFV SOL005 APIs [39] at its northbound interface, as described in section 4.5. These ETSI NFV SOL005 based APIs are exposed towards the LOCUS localization analytics as a service API layer, which governs the interactions with the LOCUS applications for exposure of both the LOCUS NFV MANO Catalogue content and Service Orchestrator lifecycle management operations. In addition, the Service Orchestrator northbound APIs can be used by the Smart NFV Functions as well to request for runtime localization analytics service update or scaling, as described above.

4.1.1.3 Resource Orchestrator

The Resource Orchestrator is a key component within the LOCUS NFV MANO as it is responsible to interface with the virtualization platform by interacting with the VIM, and in practice taking care to allocate, configure and maintain the virtualized resources required to run and operate the individual LOCUS functions VNFs that build a given localization analytics service.

With respect to the ETSI NFV architecture [30] the Resource Orchestrator is mapped to the VNF Manager functional component in the direct mode, which means that it has the rights to actually manage the virtualized resources in the virtualization platform. In practice, it manages the lifecycle of the LOCUS functions VNF, from instantiation, to update, scale and termination according to the evolution of the localization analytics services where the VNF is used.

As depicted in Figure 22, the Resource Orchestrator can also optionally take care of the coordination and provisioning of network resources in those network domains interconnecting various edge and core compute locations (e.g. when this is not covered by end-to-end slice management functions as per Figure 17), that normally are backhaul and transport networks. This is supported whenever the underlying network infrastructure offers to the NFV MANO access to SDN controllers that can dynamically and on-demand network provision connectivity services in the fronthaul, backhaul or transport networks to enable the

distributed LOCUS function VNFs to be interconnected according to the requirements expressed in the localization analytics service NSD.

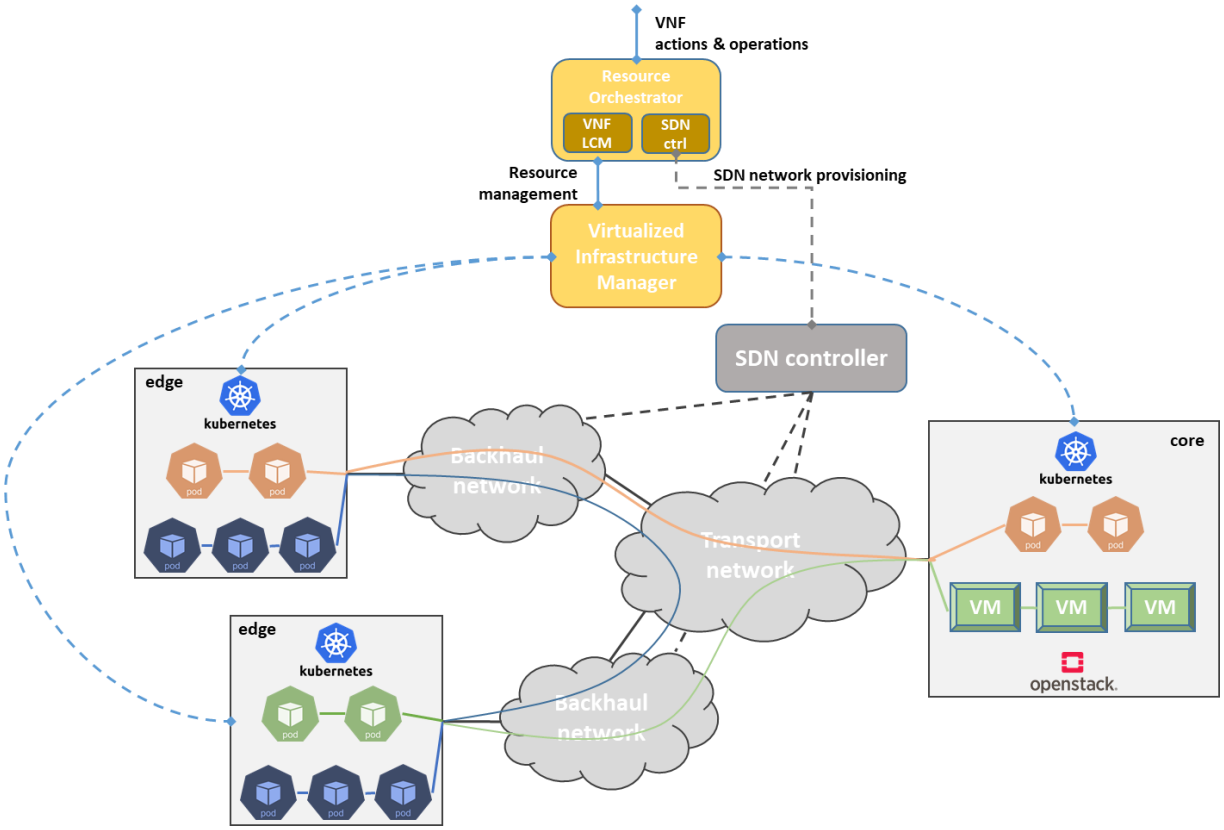


Figure 22 Resource Orchestrator interaction with the virtualization platform

While these SDN based network control features are included as part of the design of the LOCUS NFV MANO, they are not explored in the implementation and demonstration perspectives. Indeed, the main focus of the LOCUS MANO is to validate the automated deployment and operation of the localization analytics functions and services in the virtualized platform, satisfying their requirements in terms of resource and data constraints irrespectively of the solution adopted to interconnect at the network level the various edge and core locations, as well as the various VNFs running in them.

As highlighted in Figure 19, runtime modifications related to VNFs and the correspondent virtualized resources (network, storage, compute) are not exposed as a service by the Resource Orchestrator outside the LOCUS NFV MANO. Indeed, all of these modifications are under the coordination of the Service Orchestrator as part of the localization analytics service scale or update procedures. In particular, it is up to the Service Orchestrator to translate a Network Service scale or update request into one or more VNF scale or update request

towards the Resource Orchestrator. From a practical VNF perspective the Resource Orchestrator offers to the Service Orchestrator the following modification operations:

- LOCUS function VNF scaling: for scaling in or out a VNF instance, in terms of (i.e. by adding or removing virtualized resources available for the VNF)
- LOCUS function VNF update: to move from one VNF deployment flavour to another, possibly implying a change in the number and types of vdus composing the VNF.

4.1.1.4 Virtualized Infrastructure Manager

The VIM is the direct interface of the LOCUS NFV MANO towards the virtualization platform where the localization and analytics VNFs have to be deployed. It therefore offers a common interface towards the various available virtualized resources.

While logically the VIM is part of the LOCUS NFV MANO, from an implementation perspective it represents the control features of the virtual infrastructure managers used in a given deployment scenario, and therefore is typically co-located with the virtual infrastructure itself (i.e. it runs as a control service in one of the servers of the infrastructure). Since, as detailed in section 3, LOCUS adopts a hybrid virtualization approach with an integration of Openstack and Kubernetes, the LOCUS NFV MANO VIM is split into two main functionalities:

- the Openstack VIM, which is implemented by the Openstack controller component [40]. It provides a common interface for managing all of the virtual resources available in the different Openstack compute nodes. It allows to create virtual ports, virtual networks, Virtual Machines and configure how these are linked together. Indeed, the compute nodes are those compute locations where the LOCUS function VNFs can be deployed as Virtual Machines.
- the Kubernetes VIM, that is implemented by the Kubernetes Control Plane components [41]. In particular, the Kubernetes master node hosts these components, and offers to the LOCUS NFV MANO its kube-api-server APIs for managing the allocation, configuration and operation of the Kubernetes nodes, thus allowing to deploy the LOCUS function VNFs as Kubernetes PODs (or KNFs).

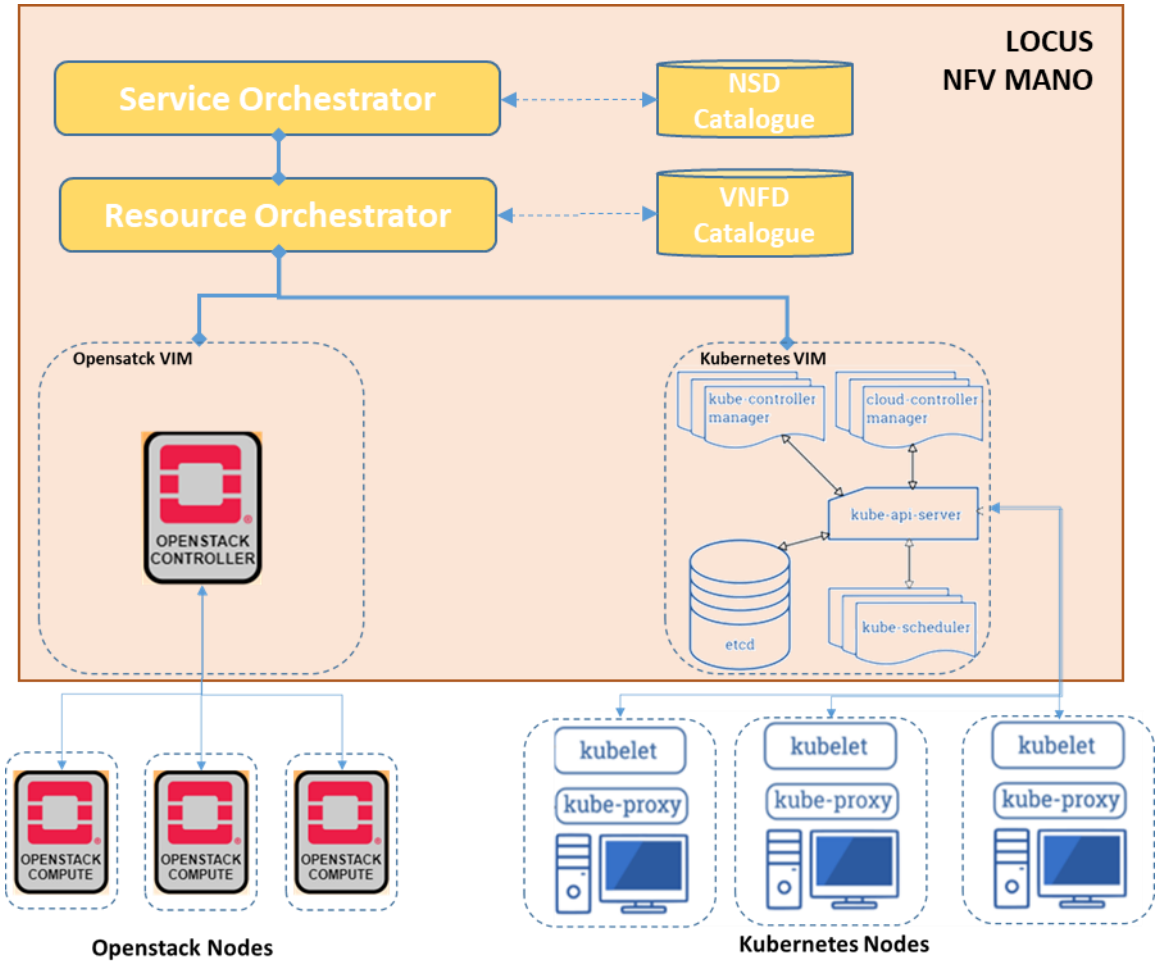


Figure 23 Openstack and Kubernetes VIMs

4.1.2 *Monitoring platform*

As anticipated above in the description of the evolved LOCUS MANO architecture, the implementation of a smart NFV management framework requires a set of functionalities to enable automation in the virtualized service management, from instantiation to runtime operation. Among these, a key enabler functionality is surely the capability to collect data from heterogeneous sources to retrieve the status of the network, the physical and virtualized infrastructure, as well as the current behaviour of the deployed virtual functions and services over such infrastructures.

In practice, this means that a Monitoring platform is required to be part of the LOCUS MANO to collect such heterogeneous set of data that include virtualized resource usage metrics (in the form of performance monitoring) as well as application-level statistics that the LOCUS localization and analytics VNFs may be capable to offer. This platform is therefore responsible to maintain historical data in the form of time series data sets and to make such data consumable by the Smart NFV Functions to apply their logic.

The data collected and maintained by the Monitoring platform can be used to perform simple runtime service management operations, such as threshold-based scaling of LOCUS functions VNFs or Networks Services (e.g. when CPU consumption for one more VNFs at the edge or core cross a pre-defined threshold) to dynamically adapt the localization analytics service to the current conditions of its functions as well as of the virtualization platform. On the other hand, more complex functionalities, e.g. that makes use of data analytics and machine learning techniques, can be enabled as well to implement proactive Network Service optimization, predictive fault management and automated NSDs and VNFDs deployment flavours update to accommodate actual requirements (in terms of virtualized resources like CPU and memory) of running services and functions.

Figure 24 shows the LOCUS Monitoring platform internal architecture. It combines two data management approaches, a query-based data collection and exposure (based on Prometheus [42], and a publish/subscribe model for data exchange based on Kafka topics [43] This allows to support heterogeneous data consumers (i.e. query based and notification based) and thus enable flexibility in the implementation of the Smart NFV Functions and the related data processing and analytics logics.

On the one hand, Prometheus is a widely used platform that natively offers exporters for multiple data sources. It is equipped with an Alarm Manager component that allows to generate alarms based on custom rules that can be configured based on the data and metrics collected and aggregated by Prometheus. It is in charge of exposing data through HTTP REST API queries to the data consumers. Moreover, Prometheus provides an easy integration with data lakes to store the raw collected data in timeseries format. In the proposed approach, the InfluxDB [44] is used. On the other hand, Kafka follows a subscribe-notify pattern for data streaming, and also allows for hierarchical deployments to integrate with distributed data movement architectures to enable data exchange among the LOCUS functions in a given localization analytics service (as currently under investigation in the context of WP2 and WP5). As shown in the figure, data consumers can have direct access to the Kafka data streams following the publish-subscribe approach.

With such kind of approach, a wide range of data sources can be supported, as shown in Figure 24. First of all, the LOCUS functions VNFs can provide monitoring data, e.g. related to VNF status, application-level metrics and resource consumption (e.g. CPU, memory, etc). Additional data sources can be the LOCUS NFV MANO VIM control services (i.e. those based on Openstack and Kubernetes as described in the previous section) which could provide overall status and consumption of the virtualized resources in the virtualization platform. Moreover, external data sources can be integrated as well, e.g. to collect data from SDN controllers concerning network resource and connectivity service status and usage in backhaul and transport networks.

Each of these data sources is expected to integrate with the monitoring platform Kafka integration bus. To do this, multiple choices and options are available to integrate data shippers within the data sources. One of the most commonly used approach is to use Telegraf [45], a plugin-driven server agent for collecting and reporting metrics, that in the specific case of LOCUS fits very well with the localization and analytics VNF approach.

The configuration of the Monitoring Platform is performed by the Config Manager, driven by the Service Orchestrator, that can customize data collection for each localization analytics service. As shown in Figure 24 it configures Prometheus (to set how aggregate which data), the Alarm Manager (to set thresholds and type of alarms to generate), as well as the Kafka topics (e.g. one for each localization analytics service).

It is worth mentioning that this platform primary goal is to collect, maintain and expose monitoring data from the NFV domain, thus targeting LOCUS functions, virtualization platform and network resources status and related statistics (possibly including application-level ones).

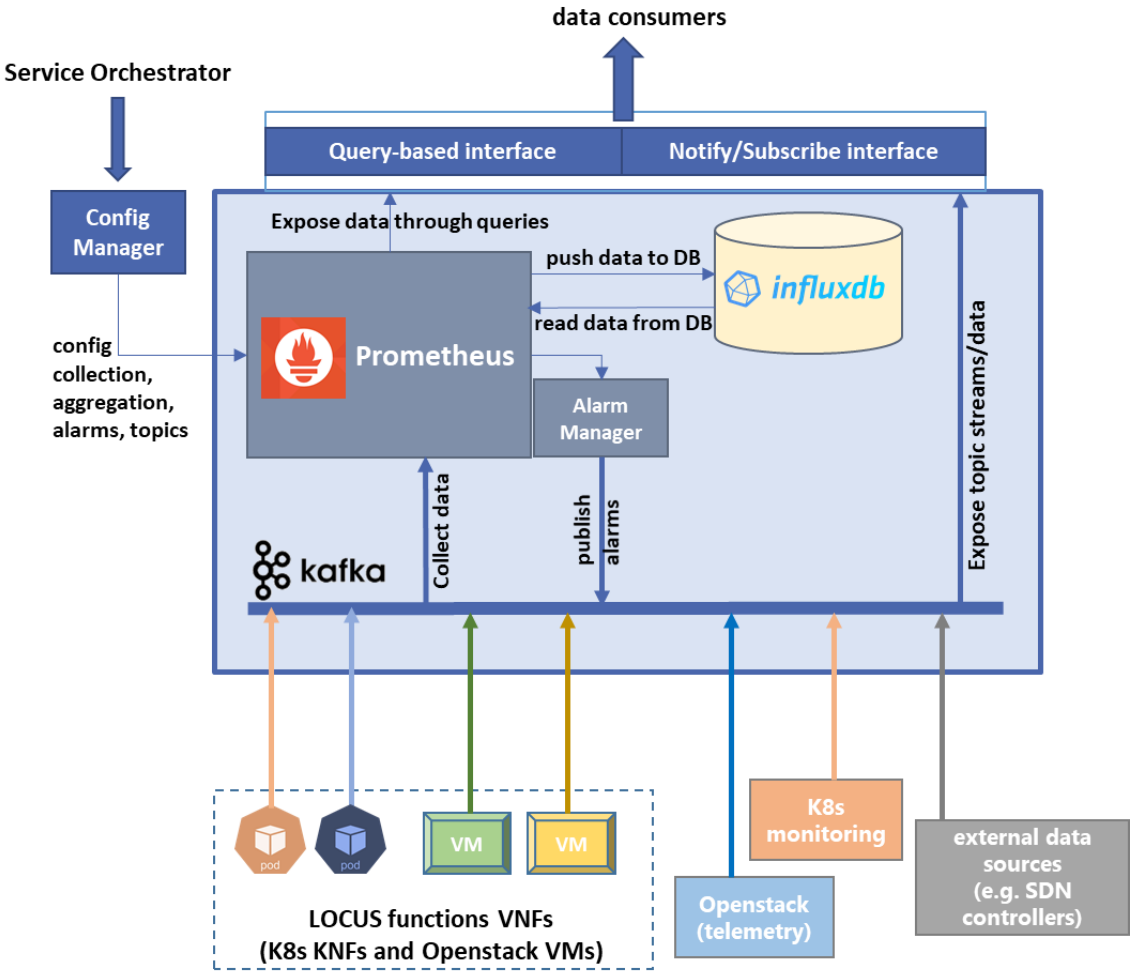


Figure 24 LOCUS MANO monitoring platform

4.1.3 *Smart NFV Functions*

The Smart NFV Functions within the LOCUS MANO include all those functionalities that provide additional NFV Network Service and VNF management capabilities leveraging on the data collected from the localization and analytics virtual functions deployed as part of a service, as well as from the virtualization platform. As anticipated in section 4.1, this approach goes in the direction of making the whole LOCUS MANO smarter, by enabling more automation in the localization analytics service management on top of virtualized infrastructures on the one hand, and a more comprehensive NFV management platform aligned with 5G network management architecture principles that exploits the available management information and collected data.

Three main types of Smart NFV Functions are envisaged within the LOCUS MANO, as described in Table 2: Design, Service Optimization and Fault Management.

Table 2 Mapping between Smart Network Functions and SNM UCs

Smart NFV Function Type	Description
Design	These functions aim at adapting NSD and VNFD content in terms of virtualized resource requirements according to the actual behaviour of running Network Services and VNFs. It may happen the initial modelling (as per workflow in section 4.4) of Network Service and VNFs is not adequate to their actual (or predicted) load when they are deployed in the virtualized platform. The idea is to process and analyse virtualized resource consumption information for the various VNFs to derive the optimal NSD and VNFD requirements. Moreover, these types of functions can support other Smart NFV Functions (e.g. Service Optimization ones) to fine-tune specific attributes or algorithms configurations. In particular, LOCUS considers the following aspects and attributes for the Design Smart NFV Functions: • Amount of CPU, RAM and storage virtualized resources required for specific *vdus* and deployment flavours inside a VNFD, • Default deployment flavour to be used for Network Service and VNF instantiation, • Values of thresholds that trigger automated NFV Network Service and VNF scaling (as specific case of Service Optimization)

	For this, the Design Smart NFV Functions interact with the NFV MANO Catalogue (through the ETSI NFV SOL005 APIs for NSD and VNF Package management), as well as with specific Service Optimization functions.
Service Optimization	These functions allow to automatically adjust NFV Network Services at runtime, with the aim of optimizing the service topology, size and deployment (in terms of which VNF runs where) according to the actual or predicted status of service itself, in terms of consumption of virtualized resources and load of the various VNFs against the available resources in the various virtualized platforms compute locations. In particular, LOCUS considers the following scenarios for the Service Optimization Smart NFV Functions, covering both reactive and predictive cases: • NFV Network Service automated scaling: to automatically trigger (towards the Service Orchestrator) the scaling of specific VNFs in a Network Service instance (i.e. by adding or removing entire VNF instances), as a reaction or prediction of high load or consumption of virtualized resources for the given VNF, • NFV Network Service automated service update: to automatically trigger (towards the Service Orchestrator) the modification of the deployment flavour in use. This could imply a reactive or predictive change in the service topology, e.g. with a new VNF type to be instantiated in the service, or a new VNF instance deployed in a different edge or core location to better satisfy the specific service and function requirements. The Service Optimization Smart NFV Functions interact with the Service Orchestrator through the ETSI NFV SOL005 APIs for Network Service lifecycle management operations at runtime.
Fault Management	These functions allow to automatically manage failure conditions that occurs either in the virtualized platform and that affects one or more resources allocated to running VNFs,

	or in the VNF instances themselves (e.g. at application level). Indeed, VNFs can be classified as faulty not only when they do not properly work due to faults in the virtual resources they use, but also when for some internal or external (to the VNF itself) reason their behaviour, reachability or manageability is compromised. In particular, LOCUS considers these faulty scenarios by processing and analysing monitoring data collected from the virtualized platform and the various VNF instances to detect or predict faults affecting running VNFs. As outcome, these Smart NFV Functions aims at triggering NFV Network Service automated service updates (similar to the Service Optimization above) to replace or redeploy faulty VNFs (e.g. in new edge or core compute locations) and dynamically modify the service topology while aiming at minimizing the service downtime. The Fault Management Smart NFV Functions interact with the Service Orchestrator through the ETSI NFV SOL005 APIs to automatically trigger Network Service lifecycle management operations at runtime to react to detected (or anticipate predicted) faulty conditions.

As anticipated in the previous section, the Monitoring Platform is the main enabler and data provider for the implementation of these Smart NFV Functions. In practice, following the approach of Figure 24, the data stored by the Monitoring Platform can be consumed by the Smart NFV Functions in two different ways:

- As HTTP REST services for stateless access to information stored in the Monitoring Platform and produced by the LOCUS functions and virtualized platform
- As Message Bus services, under the Publish-Subscribe (Pub-Sub) paradigm, for consuming streams of data generated by the LOCUS functions and virtualized platform through Kafka.

Based on these, the following types of data services exposure can be described:

- **CRUD operations on collection of data;** meaning an HTTP Rest CRUD (Create, Read, Update, Delete) service for stateless access of data entities and resources, allowing the Smart NFV Functions to consume existing data or even possibly push new data;
- **Data consumption as a Kafka Topic**; a pub-sub approach of consuming the output of one or more LOCUS functions, services or virtualized platform data source. In this

context, the Smart NFV Function connect to the Kafka messaging system based on the specific localization analytics service topic, where a broker handles all requests, the producer send records to the broker and the consumer consumes these records.

While this document provides a design approach for such Smart NFV Functions, the actual implementation and experimental validation of one Service Optimization and one Fault Management function is planned for the following stages of the project and thus will be reported in deliverable D4.4.

4.2 Management of localization analytics services

The LOCUS MANO has the main objective to manage the lifecycle of the localization analytics services following automated procedures, thus taking care of deploying the LOCUS function VNFs in the virtualization platform according to their specific requirements.

However, the localization analytics services are special cases of NFV Network Services, as they do not produce any strictly network-specific or network-related service, but rather they are focused on providing a data analytics service, thus producing a specific data output based on a specific data input requirement. This is also valid for each of the LOCUS function VNF that is part of a given localization analytics service. They are self-contained components of the data analytics service with specific input and output data requirements combined and interconnected to realize NFV Network Services.

In practice the legacy VNFs chaining happening in NFV services (i.e. requirements for their interconnection in terms of network traffic forwarding graph and service topology), is translated in LOCUS into requirements for data exchange among the various VNFs in a localization analytics service.

The SBA approach perfectly matches this scenario, as depicted in Figure 25, as each LOCUS function VNF (i.e. those running as Virtual Machines in the virtualization platform) and KNF (i.e. those running as Kubernetes PODs, following the ETSI OSM terminology described in section 4.1.1.1) provide an atomized localization or analytics functionality and can be independently managed from other functions, producing specific outputs based on specific inputs. This means that the LOCUS MANO is responsible to enforce all the required configurations in the LOCUS function VNFs and KNFs to make such SBA approach happen, and in practice satisfy the input and output data requirements for each function, thus logically chaining them and enable them to exchange analytics data (as shown with the red line in the figure).

As anticipated in previous sections, the LOCUS data architecture and the related approach for analytics data exchange across the various LOCUS functions is currently under definition as part of the WP2 and WP5 work. By the way, irrespectively of the final data exchange approach that will be chosen (also in terms of technology to be used, e.g. based on distributed Kafka

architecture to integrate data produced and consumed at edge and core locations), the LOCUS MANO already supports dedicated mechanisms for VNFs and KNFs configuration and customization at different phases of their lifecycle. These mechanisms can be effectively used to instruct VNFs and KNFs on how to reach each other, providing the required information in the form of configuration parameters, attributes, scripts to be run, etc. Such mechanisms can be summarized as:

- *Day0 configuration*: it refers to a VNF/KNF configuration that is applied during the spawning of the related Kubernetes POD or Virtual Machine instance (i.e. as part of the boot process) in the virtualization platform. It is also called "cloud-init", as the format of the configuration file (which is static and included in the VNF Package). It can be used to launch specific services or applications within the VNF/KNF instance, as well as to set static parameters or variables for service aspects that are immutable (e.g. the IP/port of a data exchange or monitoring Kafka bus);
- *Day1 configuration*: it refers to a VNF/KNF configuration that is applied during its instantiation, therefore soon after the boot of the Virtual Machine or the Kubernetes POD. It is performed through the VNF or Network Service *lifecyle-management-scripts* which are referenced in the descriptors as described in section 4.1.1.1. These scripts are invoked by the LOCUS NFV MANO and can be used to set specific parameters or launch VNF/KNF internal services that depends on dynamic attributes of the given VNF/KNF instance (e.g. identifiers of virtualized resources, IP addresses of virtual ports, etc.). At Network Service level, it can be used to apply configurations which have cross-VNF/KNF dependencies;
- *Day2 configuration*: it is a VNF/KNF configuration applied upon the occurrence of any other lifecycle management operation that is not the instantiation. Indeed, it is again an automated configuration performed by the LOCUS NFV MANO through the VNF or Network Service *lifecyle-management-scripts* referenced in the descriptors, and it can be used to update VNF/KNF internal services or configurations when the given lifecycle event occurs (e.g. VNF/KNF or Network Service scale, deployment flavour modification, termination).

ETSI OSM supports all of the three mechanisms above, through the use of Juju Proxy Charms [46], which requires the implementation of custom configuration scripts (i.e., per VNF/KNF) to be automatically invoked by ETSI OSM. Such scripts need to be included as part of the VNF or Network Service packaging, as detailed in section 4.3.

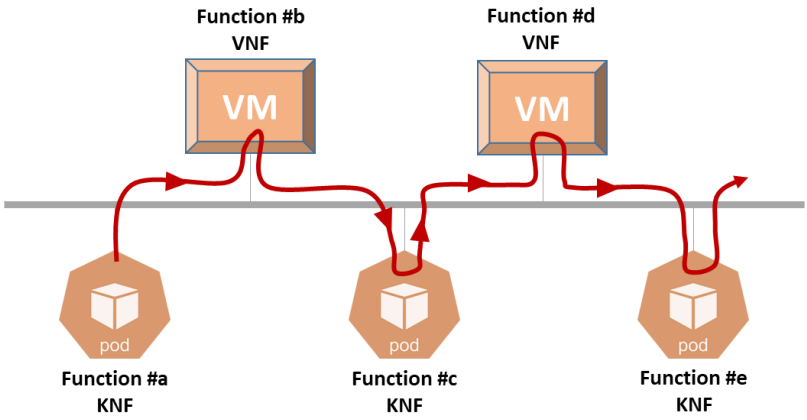


Figure 25 Logical representation of a localization analytics service and its LOCUS VNFs and KNFs

This kind of approach for the management of the localization analytics services, i.e. an SBA based integration and interconnection of the individual LOCUS function VNFs and KNFs based on their data requirements, facilitates the integration of the LOCUS services with the 5G Core network functions and services required to deliver 5G mobile connectivity to users and devices.

Indeed, leveraging on the NFV Network Services nesting approach, the LOCUS localization analytics services can be combined with 5G core services (still deployed as NFV network Services). This allows to realize composite and complex Network Services that can be managed by the LOCUS MANO and that deliver a 5G core augmented with LOCUS localization services, as shown in Figure 26.

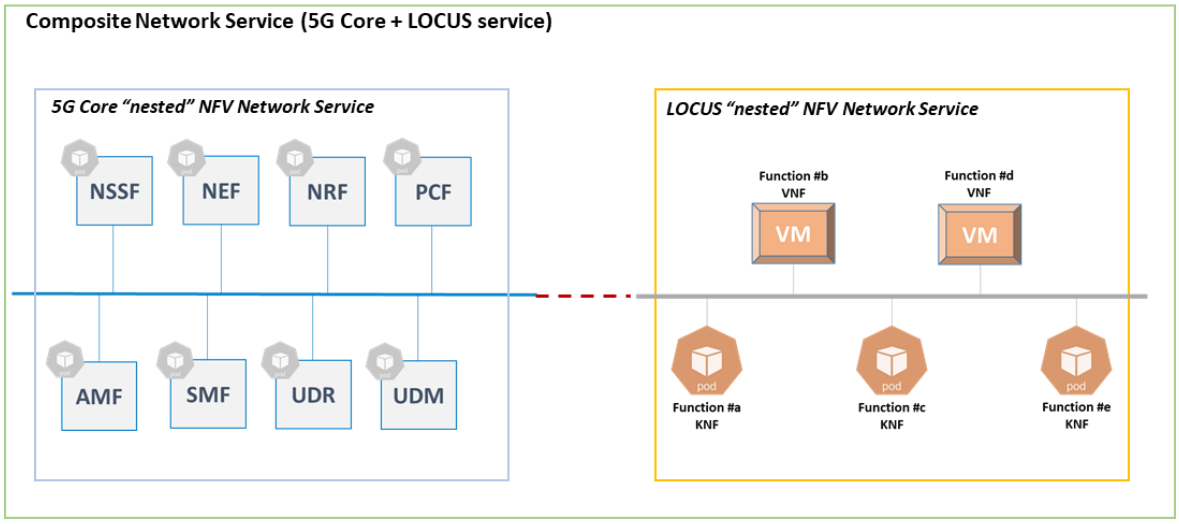


Figure 26 NFV Network Service nesting to combine 5G core and LOCUS services

4.3 Packaging of LOCUS localization and analytics functions

The use of localization and analytics functions within the LOCUS platform requires to prepare them as VNF Packages, following the LOCUS NFV MANO approach. The packaging process is

therefore a crucial operation to make a LOCUS function ready to be deployed in the virtualization platform, and it goes through different steps. In particular, the overall required operations can be summarized as follows:

- Create the function's software image that contains all the software and the common libraries to make the functions executable in the virtualization platform.
- Create a VNF Package to be onboarded into the LOCUS NFV MANO and have it available in the Catalogue for on-demand instantiation of the LOCUS function VNF (or KNF) as part of a localization analytics service.

With the aim of easing the two main operations above while trying to follow a common pattern for all of the LOCUS functions developed in WP4 (i.e. the SNM functionalities described in D4.1), a template has been prepared to collect relevant information for each UC functionality. Therefore, each LOCUS function developer is asked to complete this template and provide all the necessary information to the packaging the function.

The following subsections describe how the LOCUS function template is built, and then used to perform the two packaging steps listed above.

4.3.1 *LOCUS function template*

The LOCUS SNM function template is reported in Table 3. As said, the scope of this template is to drive the packaging process to properly package the LOCUS functions, by identifying the list of requirements in terms of software (i.e.: libraries, compilers, etc.), virtualization resources, input/output data, etc.

Table 3: LOCUS functionality template

Functional decomposition	Diagram representing the functional/software modules that implement the SNM functionality. E.g., it is important to define if the SNM functionality is formed by one or more software modules and how they relate each other
Programming language	Python, java, go (etc.)
Dependency list	• Internal dependencies (i.e.: libraries to run the software, internal databases, internal message bus) • External dependencies (other LOCUS components, external tools)

Input data	Where the input data should be gathered: • Message bus • Rest API • Dataset / Data store • Other (specify protocol/method)
Input data format	Expected datatype (i.e., data model, or JSON schema, or OpenAPI) and periodicity
Input parameters	Other configuration parameters required by the function
Output data	Where the input data should be gathered • Message bus • Rest API • Dataset / Data store • Other (specify protocol/method
Output data format	Produced datatype (i.e., data model, or JSON schema, or OpenAPI) and periodicity
Requirements (1)	In terms of CPU, RAM and DISK space
Requirements (2)	In terms of placement/location where to run the function(s) (close to user/device, in the cloud/core datacenter, any)
Requirements (3)	In terms of configuration/reconfiguration: if and how the (re)configuration is required (e.g., config file, management API, etc.)
Status of the implementation	If partial, when do you expect to have it finished?

The *Functional Decomposition* allows to understand the different modules of the function itself, how they are interconnected.

The *Programming Language* and software dependencies are crucial in the next phase, the creation of the software image, to allow a correct execution of the software.

The *input and output data* and *data format* have an impact on how to support the interconnection and communication with other LOCUS entities, including datastores or other LOCUS functions.

The *Requirements (1-3)* are related to the virtualization resources and technologies. Based on the provided data, the packaging process is able to create a software image ready to be run in the virtualization platform (i.e. in the Kubernetes environment or in the Openstack one).

The Annex A provides an example of LOCUS function template compiled for the SNM UC1 "Identification of PoIs" functionality. As shown in the Figure 27, this function is implemented as the concatenation of four components, each of them requiring input data, and after performing the internal manipulation of the data, the output is provided to the next component. Each of them can be a candidate for being packaged as an individual software image and thus realize the SNM UC1 functionality as a combination of four atomic functions.

As already mentioned, the LOCUS functions can run as Kubernetes service (and implemented as Docker containers) or Virtual Machines. According to the filled template, the set of functions to implement this SNM UC1 functionality can be packaged as Docker images.

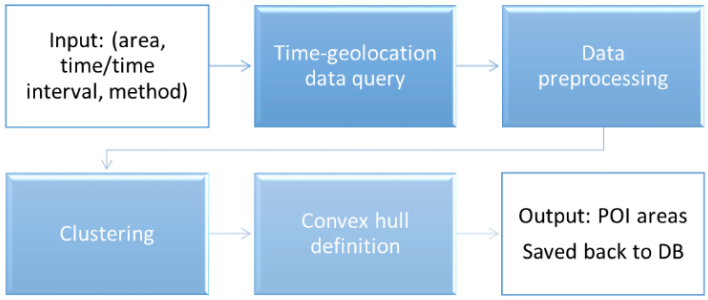


Figure 27: Functional decomposition of SNM UC1 Identification of PoIs

4.3.2 *LOCUS functions software packaging*

Once the LOCUS function template is compiled, together with the software code or binary of each component, the second step concerns the preparation of the software images. The software packaging procedures supported in LOCUS have been already defined in deliverable D5.1 [47]. In summary, two alternatives are available: i) package functions as Docker images, ii) package functions as Openstack Virtual Machines. For each of these options, software packaging process covers two main aspects:

1. the identification of common software baselines, including set of libraries (e.g. for API handling, data exchange, etc.) and common applications to interact with the LOCUS platform (e.g. the NFV MANO for Day1/Day2 operations, monitoring data collection, etc.) as well as with other LOCUS functions and services.
2. the preparation of the actual software image, built by integrating the common software baselines with the function application logic. As said, the software image can be a Docker container or a Virtual Machine image.

The selection of which software image to use is performed according to the requirements expressed by the LOCUS function provider in the template of Table 3. Irrespectively of the choice, the software image content for a given LOCUS function follows the structure depicted in Figure 28. Indeed, each software image integrates in the same running environment (either a Docker container or a Virtual Machine): i) the core application logic of the function with its requirements in terms of libraries (generically "x", "y" and "z" in the figure), ii) a monitoring plugin (e.g. based on Telegraf [45]) to connect with the Monitoring Platform, iii) the Day0/Day1/Day2 lifecycle scripts to allow the LOCUS NFV MANO to customize the function configuration at runtime, iv) a data exchange plugin (e.g. based on a Kafka connector).

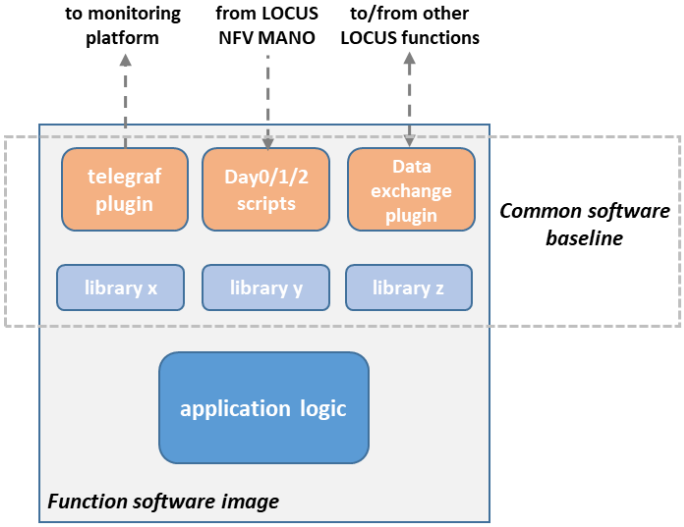


Figure 28 Software image content

4.3.3 *LOCUS functions and services packaging for NFV MANO*

The final step of the packaging process is represented by the preparation of the software artifacts to be onboarded in the LOCUS NFV MANO to enable the automated lifecycle management of VNFs/KNFs and Network Services.

The LOCUS functions and the localization analytics services are packaged as VNF Packages and NS Packages, following the approach defined by ETSI NFV in the SOL004 [48] and SOL007 [49] specifications. ETSI OSM, that in LOCUS is used as the implementation of the NFV MANO, is also aligned with this standard ETSI NFV approach, and a package is a software archive (in the *tar.gz* format) that contains all the information that the Service Orchestrator and the Resource Orchestrator need to manage LOCUS functions and services.

Within OSM there are two types of packages:

The VNF Package archive, as shown in Figure 29, is structured as follows:

- a *vnfd* YAML file, which is compliant with the ETSI NFV SOL006 [36] data model described in section 4.1.1.1. It includes KNF details if the function has to be deployed in the form of Kubernetes POD;

- a *cloud_init* folder: which includes a single file compliant with the cloud-init [50] approach, and that lists the instructions to be performed as part of the VNF Day0 configuration procedure;
- a *charms* folder: which includes the set of lifecycle management scripts to execute the Day1 and Day2 configuration procedures in the VNF. From an implementation perspective, they are Juju proxy charms [46], i.e. a set of python scripts which use SSH to get into the VNF/KNF instances and configure them. They can have a *vdu* level scope, with individual actions for each vdu in the VNF/KNF, or VNF level scope, when they run globally for the whole VNF/KNF. The charms have to be referenced in the VNFD as described in section 4.1.1.1;
- an optional *images* folder: it can include the software images required to run the VNF/KNF in the virtualization platform. Normally it is empty as the images can be references in the VNFD file.

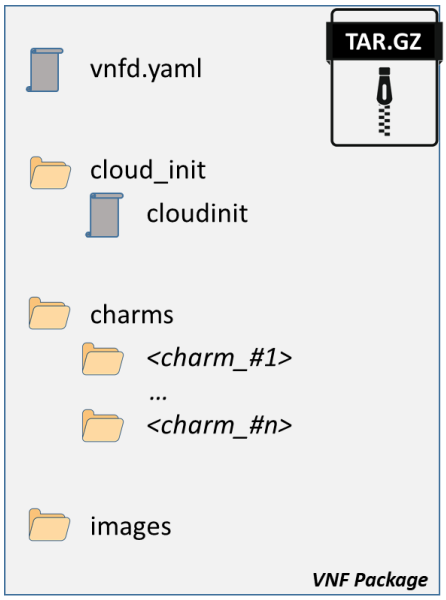


Figure 29: VNF Package content (in ETSI OSM)

On the other hand, as shown in Figure 30, the NSD package archive is structured as follows:

- a *nsd* YAML file: which is again compliant with the ETSI NFV SOL006 [36] data model described in section 4.1.1.1 and describes how the localization analytics service is composed in terms of functions and interconnection requirements.
- a *charms* folder: which includes the set of lifecycle management scripts to execute the Day1 and Day2 configuration procedures at the level of the Network Service, again implemented as Juju Proxy Charms that can be used to set cross-VNF/KNF configurations. These charms have to be referenced in the VNFD as described in section 4.1.1.1.

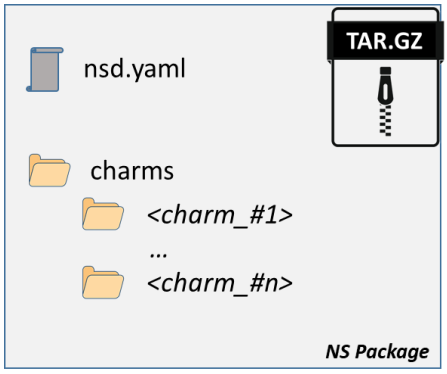


Figure 30: NS package content (in ETSI OSM)

4.4 Localization analytics services lifecycle workflows

As said, the LOCUS MANO makes use of ETSI OSM to implement the LOCUS NFV MANO functionalities. OSM leverages a unified northbound interface (NBI), based on ETSI NFV SOL005 [39], which enables the full lifecycle control of both its Network Services and VNFs/KNFs. In fact, OSM's NBI provides as a service all the necessary abstractions to allow the complete control, operation and supervision of the NS/NSI lifecycle by client systems, avoiding the exposure of unnecessary details of its constituent elements. OSM's northbound interface is based on CRUD APIs and it can therefore be seen as the entry point to the system for external clients.

The workflows implemented to support the lifecycle of the LOCUS localization analytics services can be split in different subsequent phases: modelling, onboarding, Network Service creation, Network Service runtime operation, Network Service termination.

The first design phase consists of the service **modelling** to be instantiated. On the basis of the scheme drawn up, the VNF and NSD packages are defined, which respectively contain the VNFD defining the topology of the function and its management procedures, and the NSD containing the service topology. It is necessary to design a scheme on what will be the connection points of the service, both internal and external, the images and flavours of each VDUs (in terms of vCPU, RAM, DISK) composing the VNFs.

Once the LOCUS localization functions and services are packed, they are injected into the system so they can be later on used for NS creation: this process is called **onboarding**. CRUD APIs are used in order to handle such packages. OSM also performs checks in order to validate in-model and cross-model consistency for the LOCUS services. OSM also requires a description for each LOCUS VNF component as well as the requirements in terms of vCPUs, RAM and storage.

Once the LOCUS functions are onboarded, they are used as a template for the actual Network Service **creation** (also referred to as instantiation, or "Day0" and "Day1" phase). In this case,

OSM takes as input the created package and, optionally, a set of additional deployment constraints (e.g., target deployment locations for specific VNFs of the Network Services) and parameters to particularize in the Network Service, as explicitly allowed by the NSD package.

At this point, our localization functions and services are running, and **runtime operations** can be performed: these are API-driven and are divided in two main categories. The first one is related to common lifecycle operations (e.g., scaling actions, pausing/resuming, delete, on-demand monitoring requests): these operations can either be triggered by users or also automated by defining policies at Network Service and VNF level to be automatically executed when certain events occur (e.g., monitoring a certain metric). The second one is related to the specific actions that each function/service supports. This phase is also known as "Day2".

The **terminate** operation takes care to deleting an existing Network Service instance (and all the correspondent VNFs), thus interacting with the virtualization platform as well as to de-commission the related virtualized resources.

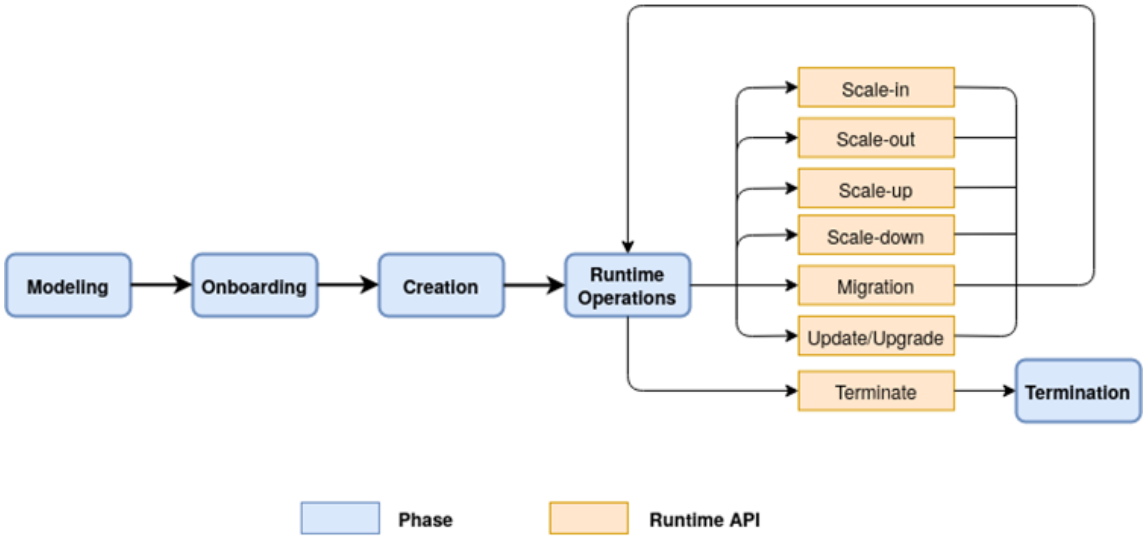


Figure 31: LOCUS localization analytics service lifecycle workflows

4.5 LOCUS MANO northbound APIs

The LOCUS MANO exposes a single set of northbound APIs, which are those offered by the Service Orchestrator, as anticipated in section 4.1.1.2. These APIs are compliant with the ETSI NFV SOL005 [39] specification and are conceived to expose Network Service and VNF/KNF lifecycle management interfaces towards the Localization as a Service APIs layer, which is the main consumer of these APIs within the LOCUS platform.

ETSI OSM, i.e. the LOCUS NFV MANO implementation, provides support for most of these standard APIs, as REST interfaces based on the HTTP protocol and JSON messages, which include additional custom operations for virtualized infrastructures management, admin, authentication, identity management ones. While the full specification of these northbound

APIs supported by OSM is available in the form of OpenAPI representations [51], the main relevant REST APIs which are actually in scope for the LOCUS platform are:

- *NSD Package Management APIs*: used to onboard VNF packages and query the content of the NFV MANO NSD catalogue
- *VNF Package Management APIs*: used to onboard VNF packages and query the content of the NFV MANO VNFD catalogue
- *NetSlice Templates APIs*: the *netslice* represents the OSM implementation of the NFV Network Service nesting concept. These APIs allow to manage composite Network Services onboarding operations.
- *NS Instance Management APIs*: used to manage the lifecycle of Network Service and VNF/KNF instances
- *NetSlice Instance Management APIs*: used to manage the lifecycle of composite Network Services.

At the time of writing, considering the identified and processed requirements for the lifecycle management of the LOCUS localization analytics services, these OSM APIs can be used as they are. Any required extension to the OSM northbound APIs that will emerge as part of the future work will be reported and implemented as part of the final version of this document, i.e., D4.4.

For the sake of completeness of this document, the Annex B provides a brief overview in table format of the OSM northbound APIs used in LOCUS.

4.6 Future work

The work reported in the previous sections covers most of the architecture and functionality design for the LOCUS MANO. However, some of the aspects described will be further elaborated and defined in the coming months. Among these, the placement optimization for LOCUS functions and services will be further elaborated. Similarly, the way the LOCUS NFV MANO will support the data exchange and data movement between the various LOCUS functions in the localization analytics services will be aligned with the approach (and related technologies) that will be soon agreed in WP2 and WP5. This will also include the integration and support in the LOCUS MANO of the New Services UCs machine learning pipelines lifecycle management from WP5, as ultimate target towards common management of localization analytics services developed in WP4 and WP5. And together with these, the integration (in terms of data required) of the Monitoring Platform with the Smart NFV Functions will be further investigated and defined in detail.

From an implementation perspective, the preliminary prototype of the LOCUS MANO integrated with the virtualization platform is described in section 5, and reports its early

validation with an exemplary LOCUS service. Of course, as detailed in section 5.2, this just includes a first set of basic functionalities which will be evolved towards the final release of the LOCUS virtualization platform and MANO with D4.4.

5 Preliminary prototype description

A preliminary version of the LOCUS MANO, integrated with a small-scale edge/core virtualization platform, has been developed in the Nextworks lab infrastructure with the aim of validating the basic concepts, principles and functionalities described in section 3 and section 4.

First of all, this first implementation work aims at setting the ground for the continuation of the activities in WP4 and WP5 for what concerns the development of the various LOCUS functions, including their integration in the virtualization platform and their automated deployment as localization analytics services through the LOCUS MANO. Moreover, this initial prototype development is already feeding the setup of the PoCs environments and scenarios in WP6, mostly in terms of requirements, dimensioning and configuration of the infrastructures where the final version of the virtualization platform will be deployed. Indeed, the virtualization platform and the LOCUS MANO will be crucial in the demonstration of the SNM and NSE UCs in WP6, as main enablers of a common deployment, integration and validation platform for the various localization analytics services.

The next section provides details about this initial prototype, describing the deployment scenario and the early validations performed.

5.1 Deployment scenario and early validation

The preliminary prototype has been developed with the aim of providing an initial integration of the core LOCUS MANO functionalities with a virtualization platform that follows the hybrid virtualization approach described in section 3. Therefore, the main goal has been to validate the basic architecture and functional concepts described in this document into a lab environment, setting up a simplified edge/core virtualization platform managed by ETSI OSM where to deploy exemplary LOCUS functions, exploiting the possibility to combine Kubernetes and Openstack technologies for mixed services.

As shown in Figure 32, such prototype has been developed and deployed in the Nextworks lab infrastructure (the one described in section 3.3.3), and is composed by three main parts:

- a set of **Control** functionalities, which implements a basic version of the LOCUS MANO, and includes ETSI OSM Release 9 and an Openstack controller (Queens version);
- a single **Core** computing location, implemented as an Openstack compute node (Queens version), where LOCUS VNFs packaged as Virtual Machines can be deployed
- a single **Edge** computing location, implemented as a Kubernetes cluster (v1.17), composed by one master node and two worker nodes. Rancher OS is also used on top

the cluster for management purposes and to have a graphical interface for debugging and troubleshooting purposes.

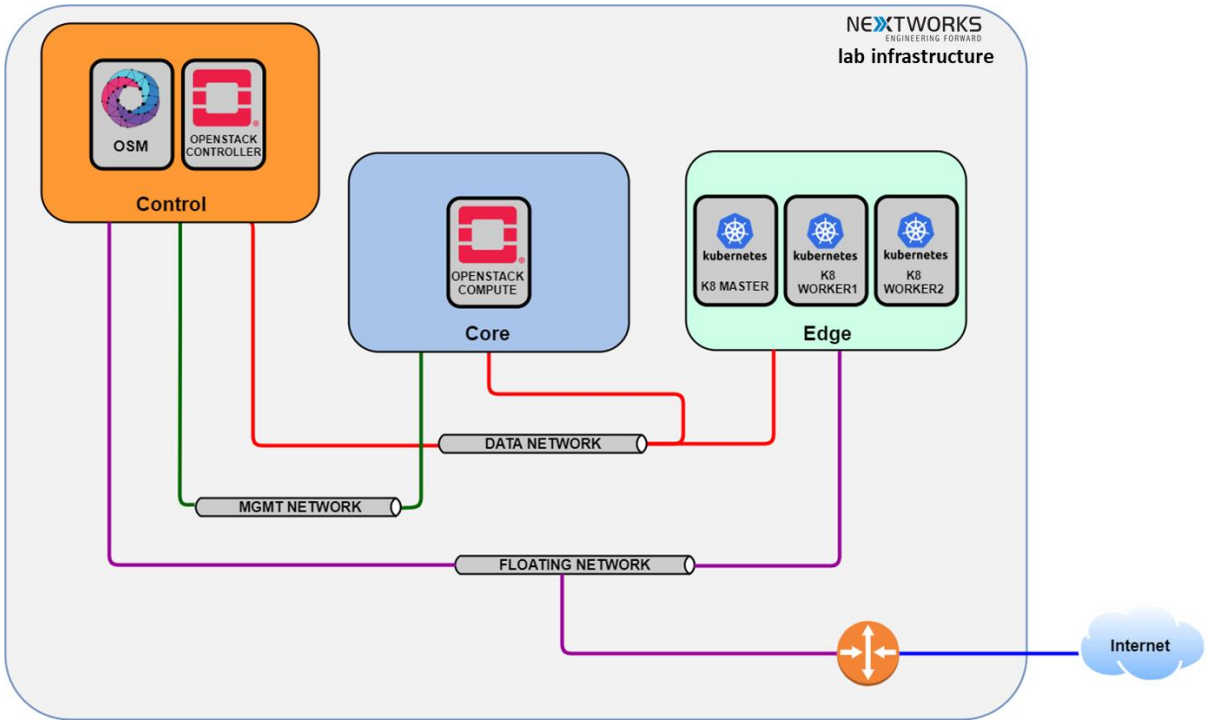


Figure 32: Preliminary prototype setup

Following the hybrid virtualization platform approach described in section 3, the three components of this preliminary prototype (LOCUS MANO, core computing location, edge computing location) have been interconnected through networks jointly configured in the Openstack and Kubernetes environments:

- *mgmt network:* it is the network to enable ETSI OSM to have control over the Openstack core compute location, and in particular to have reachability of the VNFs deployed there for Day1/Day2 configurations;
- *data network:* it is the network used for data traffic to be exchanged between the VNFs and KNFs that are deployed by ETSI OSM in the two compute locations;
- *floating network*: it is the network that allows the data and management networks and traffic to be reachable from Internet (or anyway from outside the lab environment). In the Kubernetes edge computing location, this network is also used as management network to enable KNFs deployment and Day1/Day2 configurations through ETSI OSM.

In the Kubernetes cluster, the *multus* and *macvlan* CNI plugins have been used and properly configured to enable the KNFs running at the edge location to be reachable by external entities

(i.e. the VNFs running in the core Openstack location, as well as the ETSI OSM for management purposes) through additional container interfaces.

While the LOCUS functions are mostly still under development (or packaging) in WP4 and WP5, the validation of the hybrid virtualization platform integrating Kubernetes edge and Openstack core locations and managed by ETSI OSM as LOCUS NFV MANO has been performed through an exemplary test service, composed by one VNF and one KNF. With the aim of validating the concept of NFV Network Service nesting as well, this reference validation service has been modelled as a composite service, as shown in Figure 33.

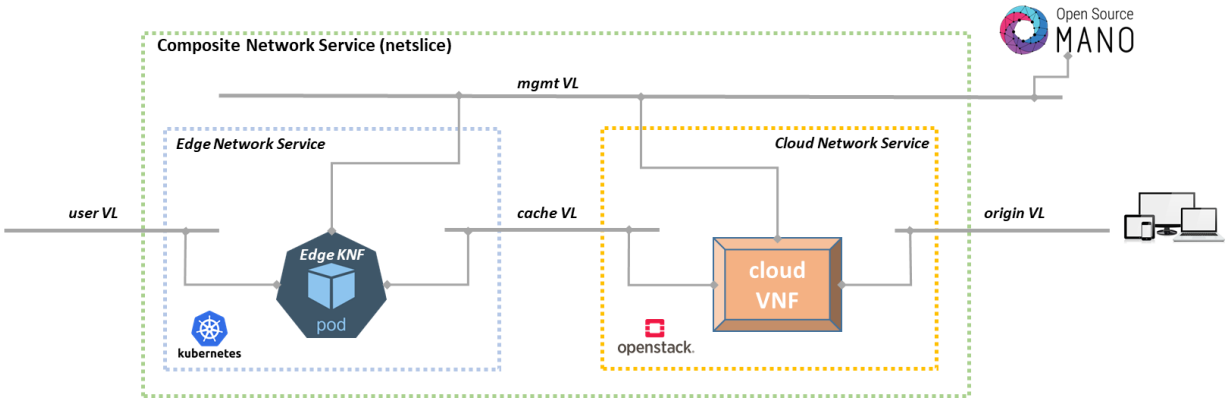


Figure 33 Exemplary LOCUS composite NFV Network Service

In particular, this exemplary LOCUS composite NFV Network Service is built by:

- a single VNF "*cloud network service*" to be deployed in the Openstack core location (thus packaged as a Virtual Machine)
- a single KNF "*edge network service*" to be deployed in the Kubernetes edge location (thus packaged as a Docker container)
- a virtual link (*origin VL*) to access the core VNF, that would represent the exposure of the LOCUS service data output towards other (external) entities.
- a virtual link (*cache VL*) for interconnecting the two nested Network Services, enabling the communication among the VNF and the KNF;
- a virtual link (*user VL*) to enable the edge KNF to possibly retrieve data from user devices or other edge applications (not actually used in the validation);
- a virtual link (*mgmt VL*) for management traffic and allowing ETSI OSM to manage the lifecycle of the VNF and KNF instances.

Indeed, from an application point of view, this exemplary composite Network Service supports a simple data query workflow. As shown in Figure 34, the core VNF exposes an HTTP REST endpoint to perform a simple synchronous data query, and therefore:

1. the external consumer (e.g. a Unix terminal) query the endpoint exposed by the cloud VNF.
2. the cloud VNF processes the request and forward it to the edge KNF, that exposes the same HTTP REST endpoint.
3. the edge KNF receives the request, generates random data in JSON format (in the specific case a list of 5 random integers 0-100) and send it back to the cloud VNF.
4. the cloud VNF forwards back the data to the external consumer.

This simple approach had the objective to validate the integration of VNFs and KNFs deployed at edge/core locations, without focusing too much on the data analytics aspects (including how the data is exchanged among them) as they will be validated with "real" LOCUS functions developed in WP4 and WP5.

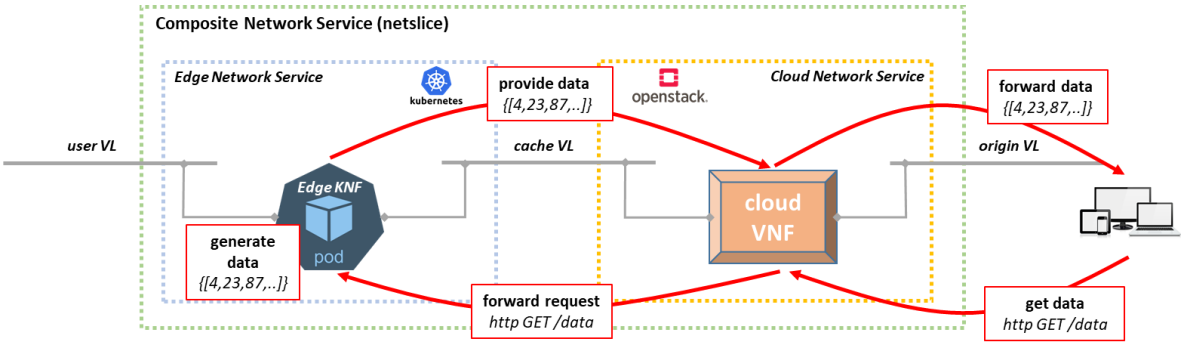


Figure 34 Application workflow for the exemplary LOCUS service

To implement this exemplary composite NFV Network Service, the first step is to prepare the VNF and KNF descriptors according to the specifications in section 4.1.1.1 and 4.3.3, therefore compliant with the ETSI NFV SOL006 data models. As shown in Figure 35, the VNF and KNF descriptors include all the required information and requirements for the LOCUS NFV MANO (i.e. ETSI OSM) to manage their automated deployment in the virtualization platform, including virtual resource and connectivity constraints. For the KNF, as anticipated in section 4.3.3, the descriptor is practically a VNFD with additional information required to handle its deployment as a Helm Chart [38], that is a Kubernetes standard deployment template that leverage on the Helm orchestration tool. The Helm chart used for the edge KNF is shown in Figure 36, and it describes how the Kubernetes workload has to be deployed in the cluster, including instructions on how Multus shall enable the container interfaces to allow the KNF to communicate with external entities.

```
vnfd:
    description: descriptor for cloud VNF
    df:
    -   id: default-df
        instantiation-level:
        -   id: default-instantiation-level
            vdu-level:
            -   number-of-instances: 1
    ext-cpd:
    -   id: cp_mgmt-ext
        int-cpd:
            cpd: ens3-int
            vdu-id: cloud_vdu
    -   id: cp_cache-ext
        int-cpd:
            cpd: ens6-int
            vdu-id: cloud_vdu
    -   id: cp_origin-ext
        int-cpd:
            cpd: ens7-int
            vdu-id: cloud_vdu
    id: b0c92f8c-7c69-40bd-a570-54a7a3df5b14
    product-name: cloud_vnf
    provider: NXW
    sw-image-desc:
    -   id: vCache-ubuntu-52-shrink
        image: vCache-ubuntu-52-shrink
    vdu:
    -   description: Cloud vdu
        id: cloud_vdu
        int-cpd:
        -   id: ens3-int
            virtual-network-interface-requirement:
            -   name: ens3
                position: 1
        -   id: ens6-int
            virtual-network-interface-requirement:
            -   name: ens6
                position: 2
        -   id: ens7-int
            virtual-network-interface-requirement:
            -   name: ens7
                position: 3
        name: cloud_vdu
        sw-image-desc: vCache-ubuntu-52-shrink
        virtual-compute-desc: cloud_vdu-compute
        virtual-storage-desc:
        - cloud_vdu-storage
    version: '1.0'
    virtual-compute-desc:
    -   id: cloud_vdu-compute
        virtual-cpu:
            num-virtual-cpu: 4
        virtual-memory:
            size: 8.0
    virtual-storage-desc:
    -   id: cloud_vdu-storage
        size-of-storage: 20
```

cloud VNF

```
vnfd:
    description: descriptor for Edge KNF
    df:
    -   id: default-df
    ext-cpd:
    -   id: cp_ext-ext
        k8s-cluster-net: external_net
    -   id: cp_user-ext
        k8s-cluster-net: user_net
    -   id: cp_cache-ext
        k8s-cluster-net: cache_net
    id: bb7d54e9-b60b-45ae-bded-06b14387816b
    k8s-cluster:
        nets:
        -   id: external_net
        -   id: user_net
        -   id: cache_net
    kdu:
    -   helm-chart: github-repo/edge_cache:0.1.3
        name: edge_cache
    mgmt-cp: cp_ext-ext
    product-name: edge_knf_vnfd
    provider: Nextworks
    version: '1.0'
```

edge KNF

Figure 35: VNF and KNF Descriptors

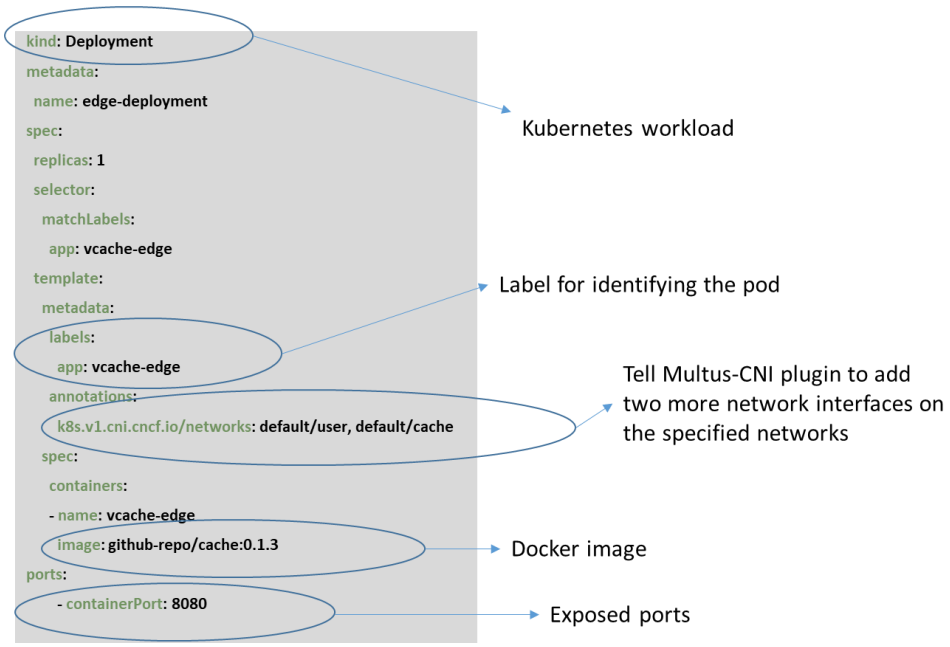


Figure 36: KNF helm chart

After having the VNF and KNF descriptors ready, the VNF packages can be prepared according to the procedure described in section 4.3.3 in the form of tar.gz archives to be onboarded on the LOCUS NFV MANO. For this, ETSI OSM implements a graphical user interface to upload the VNF packages archives, as shown in Figure 37.

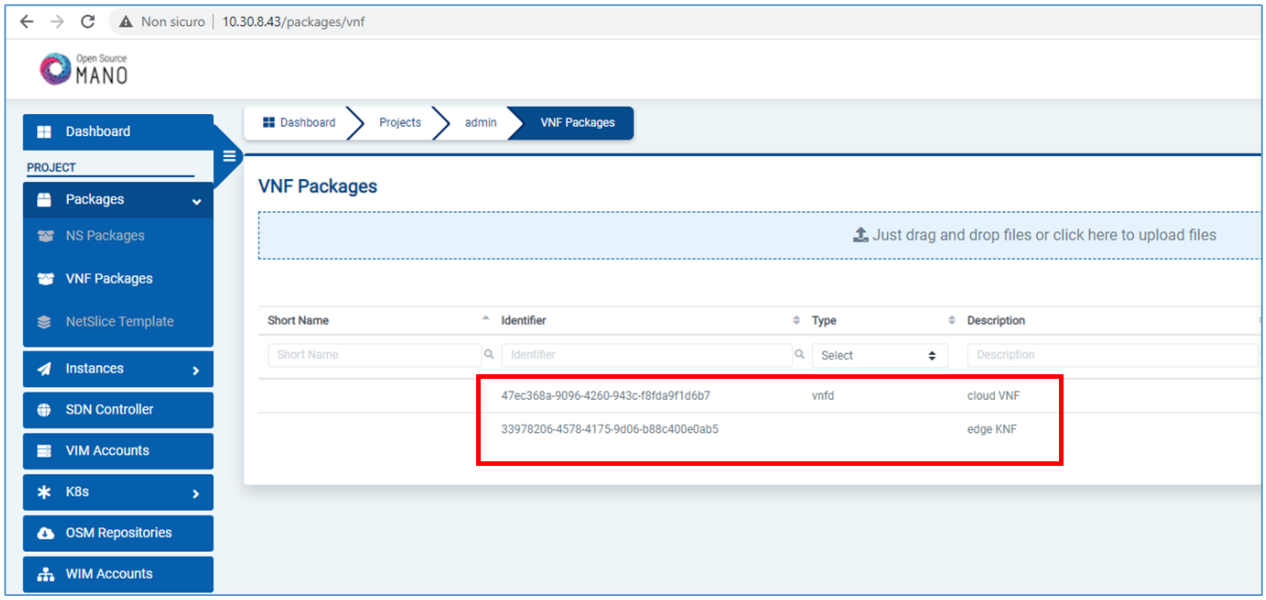


Figure 37: VNF packages onboarded in ETSI OSM.

With the VNF Packages now available in ETSI OSM, the design of the two nested Network Services (the cloud and the edge ones), as well as of the composite Network Service, can be performed. This requires the preparation of three NSDs to model the two nested Network Services and the composite one of Figure 33. The two nested NSDs (i.e. the cloud NSD for the cloud VNF and the edge NSD for the edge KNF) are shown in Figure 38, where the various requirements in terms of interconnection through the virtual link described above are set. The composite NSD is shown in Figure 39, in the form of a ETSI OSM netslice. The netslice is the ETSI OSM implementation of the NFV Network Service nesting concept and allows combining more NSDs together to be managed together and combined into a common higher-level NSD, that indeed is the composite NSD (i.e. netslice template).

cloud NSD

```
nsd:
    nsd:
    -   description: Cloud NSD - single VNF
        designer: Nextworks
        df:
        -   id: default-df
            vnf-profile:
            -   id: '1'
                virtual-link-connectivity:
                -   constituent-cpd-id:
                    -   constituent-base-element-id: '1'
                        constituent-cpd-id: cp_mgmt-ext
                    virtual-link-profile-id: mgmt_net
                -   constituent-cpd-id:
                    -   constituent-base-element-id: '1'
                        constituent-cpd-id: cp_origin-ext
                    virtual-link-profile-id: origin_net
                -   constituent-cpd-id:
                    -   constituent-base-element-id: '1'
                        constituent-cpd-id: cp_cache-ext
                    virtual-link-profile-id: cache_net
                vnfd-id: b0c92f8c-7c69-40bd-a570-54a7a3df5b14
        id: flea73e3-0f94-45e5-be56-973a2b0085e8
        name: cloud_nsd
        version: '1.0'
        virtual-link-desc:
        -   id: mgmt_net
            mgmt-network: true
            vim-network-name: VM_MGMT
        -   id: origin_net
            mgmt-network: true
            vim-network-name: origin_net
        -   id: cache_net
            mgmt-network: true
            vim-network-name: cache_net
        vnfd-id:
        - b0c92f8c-7c69-40bd-a570-54a7a3df5b14
```

edge NSD

```
nsd:
    nsd:
    -   description: KNF edge NSD - single knf
        designer: Nextworks
        df:
        -   id: default-df
            vnf-profile:
            -   id: '1'
                virtual-link-connectivity:
                -   constituent-cpd-id:
                    -   constituent-base-element-id: '1'
                        constituent-cpd-id: cp_ext-ext
                    virtual-link-profile-id: external_net
                -   constituent-cpd-id:
                    -   constituent-base-element-id: '1'
                        constituent-cpd-id: cp_cache-ext
                    virtual-link-profile-id: cache_net
                -   constituent-cpd-id:
                    -   constituent-base-element-id: '1'
                        constituent-cpd-id: cp_user-ext
                    virtual-link-profile-id: user_net
                vnfd-id: bb7d54e9-b60b-45ae-bded-06b14387816b
        id: flea73e3-0f94-45e5-be56-973a2b0085e7
        name: knf_edge_nsd
        version: '1.0'
        virtual-link-desc:
        -   id: external_net
            mgmt-network: true
        -   id: cache_net
            mgmt-network: true
        -   id: user_net
            mgmt-network: true
        vnfd-id:
        - bb7d54e9-b60b-45ae-bded-06b14387816b
```

Figure 38: Cloud and edge nested NSDs

```
nst:
-   SNSSAI-identifier:
        slice-service-type: eMBB
    id: 895166bd-6ce8-498b-8ad6-be46a26df0d2
    name: nst_knf_edge_vnf_cloud
    netslice-subnet:
    -   description: cloud NS
        id: cloud_nsd
        is-shared-nss: true
        nsd-ref: flea73e3-0f94-45e5-be56-973a2b0085e8
    -   description: edge NS
        id: edge_nsd
        is-shared-nss: false
        nsd-ref: flea73e3-0f94-45e5-be56-973a2b0085e7
    netslice-vld:
    -   id: VM_MGMT
        mgmt-network: true
        name: VM_MGMT
        nss-connection-point-ref:
        -   nsd-connection-point-ref: cp_mgmt
            nss-ref: vCDN_cloud_nsd
        type: ELAN
    -   id: external_net
        mgmt-network: true
        name: external_net
        nss-connection-point-ref:
        -   nsd-connection-point-ref: cp_ext
            nss-ref: vCDN_edge_nsd
        type: ELAN
    -   id: user_net
        mgmt-network: true
        name: user_net
        nss-connection-point-ref:
        -   nsd-connection-point-ref: cp_user
            nss-ref: vCDN_edge_nsd
        type: ELAN
    -   id: cache_net
        mgmt-network: true
        name: cache_net
        nss-connection-point-ref:
        -   nsd-connection-point-ref: cp_cache
            nss-ref: vCDN_cloud_nsd
        -   nsd-connection-point-ref: cp_cache
            nss-ref: vCDN_edge_nsd
        type: ELAN
    -   id: origin_net
        mgmt-network: true
        name: origin_net
        nss-connection-point-ref:
        -   nsd-connection-point-ref: cp_origin
            nss-ref: vCDN_cloud_nsd
        type: ELAN
```

Figure 39: Composite NSDs in the form of OSM netslice

As for the VNF and KNF, with the NSDs ready, the NS Packages can be prepared according to the procedure detailed in section 4.3.3 and then onboarded to ETSI OSM through its graphical

interface, as shown in Figure 40 for the nested NS Packages and in Figure 41 for the netslice template (i.e. the composite NSD).

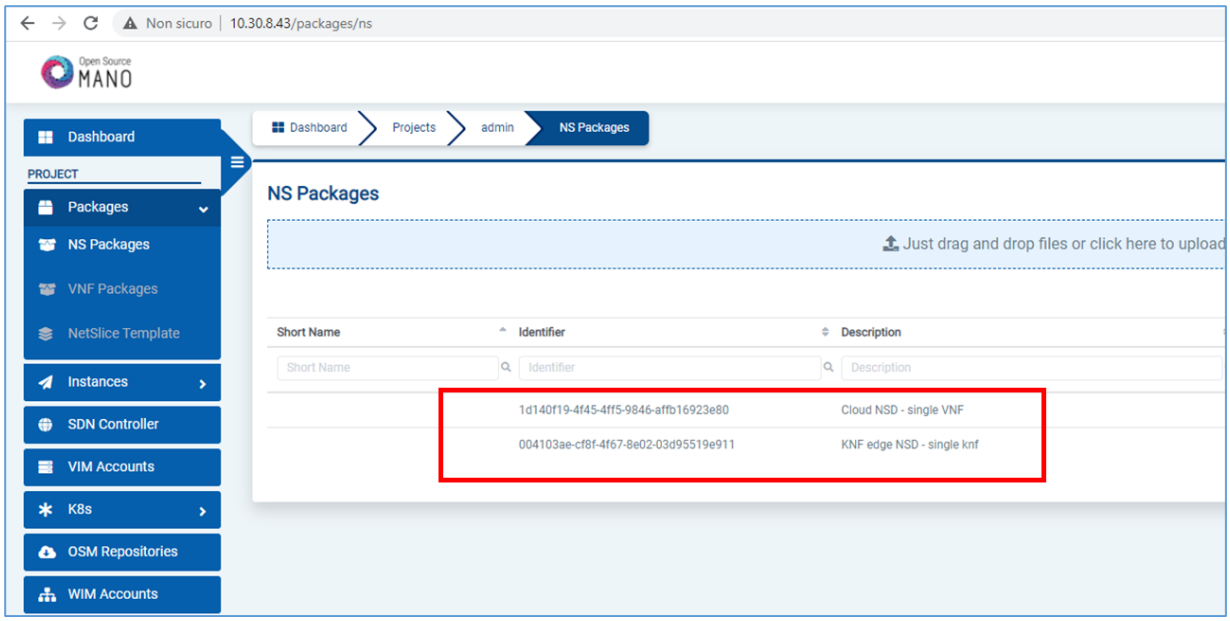


Figure 40: Network Service Packages onboarded in OSM.

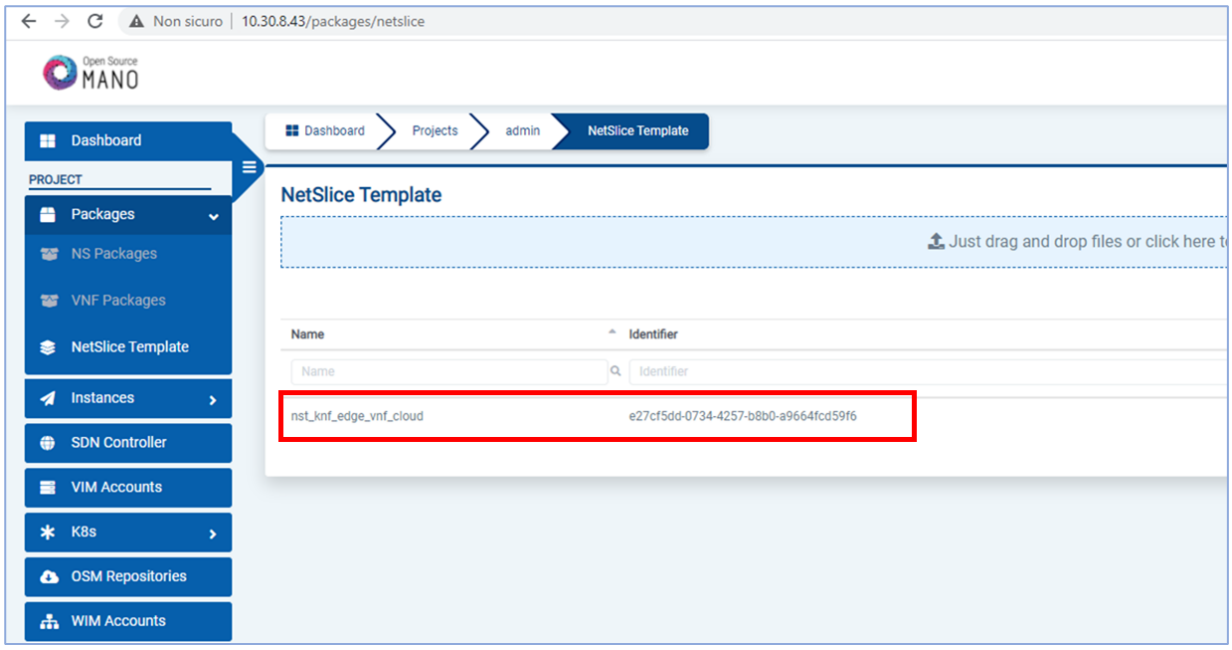


Figure 41: netslice template onboarded in OSM.

At this point, the exemplary LOCUS service can be instantiated in the virtualized platform, delegating to the NFV MANO to automatically create all of the required resources and artifacts in the edge and core compute locations to satisfy the service requirements. Indeed, upon issuing the service instantiation request through the graphical user interface, ETSI OSM takes care to parse and process the NSDs and netslice descriptors first to split the instantiation

phase in two separate actions for the two cloud and edge Network Services, and then for each of them to interact with the virtualized platform to create the required virtual resources (Kubernetes pods, Openstack Virtual Machines, virtual interfaces, virtual links, etc.) according to the requirements set in the descriptors.

As a result, when the instantiation process is completed, ETSI OSM makes the two Network Service instances, the VNF and KNF instances available as “running” and “configured” on the graphical user interface, so that the services can be consumed, as shown in Figure 42 and Figure 43. Moreover, connecting to the Rancher OS graphical user interface, it is possible also to verify that the Kubernetes pod for the edge KNF has been properly created and configured, as shown in Figure 44.

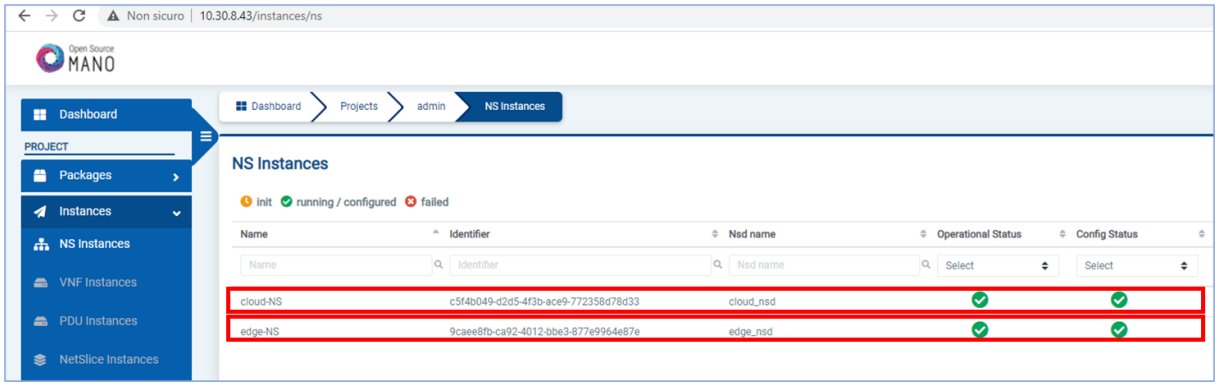


Figure 42: Network Service instances in OSM

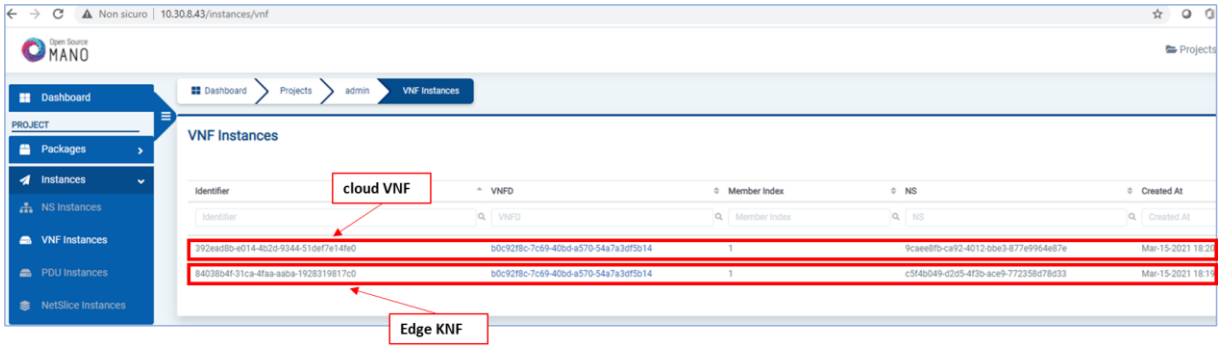


Figure 43: VNF and KNF instances in OSM

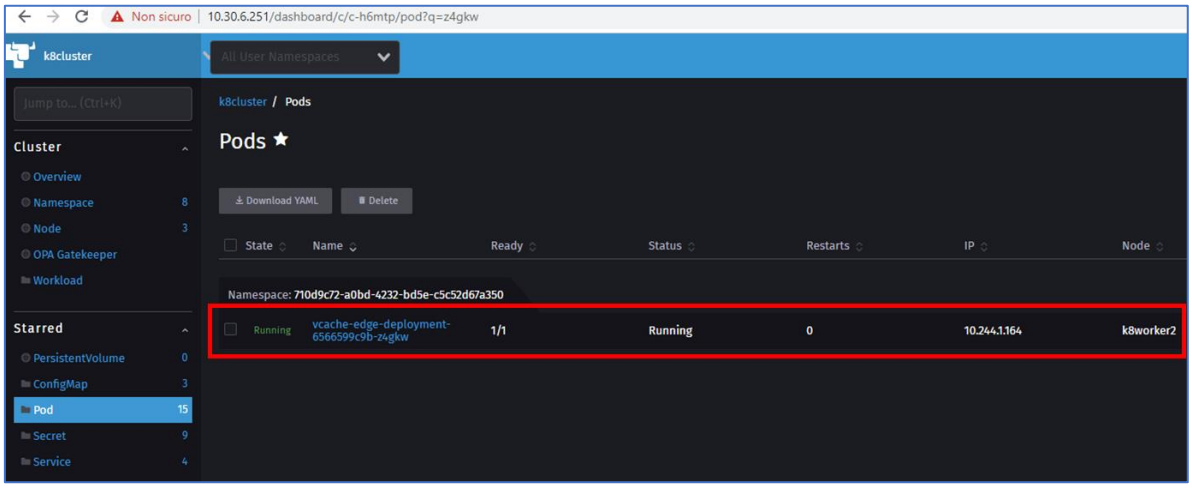


Figure 44: Kubernetes pod on Rancher OS

5.2 Future work

The preliminary prototype described above represents a first step towards the development of the complete LOCUS virtualization platform to be deployed in the WP6 LOCUS testbed, and fully managed by the LOCUS MANO described in section 4. Indeed, this is already started as initial liaise with WP6 for the deployment of the “production” virtualization platform in the for PoCs validation and demonstration activities.

While for this initial implementation work the focus has been on validating the hybrid virtualization approach and its integration with the ETSI OSM management and orchestration for an exemplary LOCUS service, the immediate next step is to integrate the functions developed in WP4 and WP5 with the LOCUS NFV MANO, in terms of packaging (as soon as function templates will be provided by the function developers) and automated deployment in the virtualization platform at edge and core locations. In addition, the LOCUS NFV MANO will be evolved to support the architecture defined in section 4, thus including integration with Smart Management Functions developed as SNM functionalities as well as with the Monitoring Platform.

6 Conclusions

This virtualization platform and the management and orchestration framework represent two key aspects and building blocks in LOCUS to enable the implementation of the localization analytics services platform. This deliverable has presented the first iteration of their design and implementation, together with a preliminary software prototype that integrates them in a small-scale lab environment.

The LOCUS virtualization platform is designed following a hybrid approach to leverage on existing virtualization technologies and support distributed deployments in line with the high degree of infrastructure virtualization required by 5G. In particular, LOCUS integrates Infrastructure as a Service and cloud-native virtualization technologies to realize a distributed edge/core platform where the LOCUS virtual functions can be deployed following different paradigms, i.e. to have them implemented as either Virtual Machines in Openstack core clouds or containerized applications suitable for running in Kubernetes edge clouds. This allows to ease the interconnection with the 5G system and related network functions that can provide the required localization information and data to be consumed and processed by the LOCUS virtual functions. This is achieved by considering the LOCUS virtualization platform as seamlessly integrated with 5G networks and infrastructures, with the aim of providing edge and core computing capabilities to host and run the LOCUS localization and analytics functions.

On the other hand, the LOCUS MANO has been described in this deliverable in terms of detailed functional decomposition, with the aim of identifying the main capabilities in support of the LOCUS localization analytics services platform. The LOCUS MANO enables an automated lifecycle management of the localization analytics functions and services, fully exploiting the NFV principles for the deployment and operation of VNFs and NFV Network Services that implement WP4 and WP5 use case functionalities. This includes various aspects, from the packaging of the LOCUS functions applications into virtualization-ready software images first, and MANO descriptors later to enable their automated management on top of the distributed edge/core LOCUS virtualization platform following the 5G SBA approach and leveraging on monitoring capabilities. This approach allows to integrate the LOCUS MANO (and the LOCUS platform as a consequence) with cross-layer service and slice management and orchestration frameworks (as next generation NSM and OSS platforms) and augment traditional NFV MANO tools with additional localization related capabilities and analytics services to be offered in support of either Smart Network Management or 3rd vertical applications.

As said, this document also reports a preliminary prototype that integrates a first version of the LOCUS MANO (based on the ETSI OSM opensource platform) with a small-scale virtualization platform (composed by the integration of one Openstack core computation location with a single Kubernetes edge cluster with two worker nodes), with the aim of

validating the proposed hybrid edge/core virtualization approach with an exemplary distributed service automatically deployed and managed following the LOCUS MANO principles.

As part of the next steps, the hybrid virtualization platform approach described in this document (and early validated as part of the small-scale prototype reported here) will be implemented in the LOCUS WP6 testbed, with the aim of supporting the PoCs activities and enable the integration of the LOCUS virtual functions developed in WP4 and WP5. Moreover, for what concerns the LOCUS MANO, it will be evolved (at both design and implementation levels) in the next stages of the project to support the LOCUS final architecture specifications in terms of mapping and integration with the 5G system, as well as in terms of data exchange among the LOCUS functions in support of the LOCUS PoCs in WP6.

References

[1] Deliverable D2.4 EU LOCUS project “System Architecture: preliminary version”

[2] 3GPP TR 28.801, “Telecommunication management; Study on management and orchestration of network slicing for next generation network”, Release 15

[3] ETSI GR NFV-EVE 012, “Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework”, December 2017

[4] H2020 5G-Transformer, http://5g-transformer.eu/

[5] H2020 SliceNet, https://slicenet.eu/

[6] H2020 5G-EVE, https://www.5g-eve.eu/

[7] H2020 5Growth, https://5growth.eu/

[8] Ericsson Orchestrator, https://www.ericsson.com/en/portfolio/digital-services/automated-network-operations/orchestration/ericsson-orchestrator

[9] NOKIA CloudBand Solution Elements, https://www.nokia.com/networks/solutions/cloudband/#solution-elements

[10] 3GPP Technical report, TR 23.731, “Study on enhancement to the 5GC LoCation Services (LCS)”, December 2018

[11] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements," in IEEE Signal Processing Magazine, vol. 22, no. 4, pp. 41-53, July 2005.

[12] J. Trogh, D. Plets, E. Surewaard et al., “Outdoor location tracking of mobile devices in cellular networks”, J Wireless Com Network, 115 (2019)

[13] BeFEMTO project consortium, “BeFEMTO: Broadband Evolved FEMTO Networks”, http://www.ict-befemto.eu

[14] S. Fortes, A. Aguilar-García, R. Barco, F. B. Barba, J. A. Fernández-luque and A. Fernández-Durán, "Management architecture for location-aware self-organizing LTE/LTE-a small cell networks," in IEEE Communications Magazine, vol. 53, no. 1, pp. 294-302, January 2015.

[15] S. Fortes, “Context-Aware Self-Healing for Small Cell Networks” PhD. thesis, IC, University of Málaga 2017. https://riuma.uma.es/xmlui/handle/10630/15782

[16] S. Fortes, D. Palacios, I. Serrano and R. Barco, "Applying Social Event Data for the Management of Cellular Networks," in IEEE Communications Magazine, vol. 56, no. 11, pp. 36-43, November 2018, doi: 10.1109/MCOM.2018.1700580.

[17] HORIZON 2020 Programme, “High precision positioning for cooperative ITS applications”, http://www.hights.eu.

[18] FP7 Programme, “SELECT: Smart Efficient Location, idEntification and Cooperation Techniques”, https://cordis.europa.eu/project/id/257544/it.

[19] HORIZON 2020 Programme, “European Location As A Service Targeting International Commerce”, https://cordis.europa.eu/project/id/641526/it

[20] HORIZON 2020 Programme, “GNSS driven EO and Verifiable Image and Sensor Integration for mission-critical Operational Networks”, https://www.gsa.europa.eu/gnss-driven-eo-and-verifiable-image-and-sensor-integration-mission-critical-operational-networks.

[21] HORIZON 2002 Programme, “MOBNET MOBile NETwork for people's location in natural and man-made disasters”, https://cordis.europa.eu/project/id/687338

[22] Kubernetes, https://kubernetes.io/

[23] Openstack, https://www.openstack.org/

[24] Openstack Neutron, https://docs.openstack.org/neutron/latest/

[25] Rancher OS, https://rancher.com/

[26] Openstack Queens version, https://www.openstack.org/software/queens/

[27] Macvlan Kubernetes CNI, https://www.cni.dev/plugins/main/macvlan/

[28] Multus Kubernetes CNI, https://github.com/k8snetworkplumbingwg/multus-cni/blob/master/README.md

[29] Deliverable D4.1 EU LOCUS project “Localization & Analytics for Smart Network Management”

[30] ETSI GS NFV-MAN 001, “Network Functions Virtualisation (NFV); Management and Orchestration”, December 2014

[31] ETSI Open Source MANO, https://osm.etsi.org/

[32] ETSI OSM management of 4G/LTE EPC, https://osm.etsi.org/gitlab/vnf-onboarding/osm-packages/tree/master/magma

[33] Magma EPC, https://www.magmacore.org/

[34] ETSI OSM Ecosystem, https://osm.etsi.org/wikipub/index.php/OSM_Ecosystem

[35] ETSI OSM PoCs, https://osm.etsi.org/wikipub/index.php/OSM_PoCs

[36] ETSI GS NFV-SOL 006, “Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification”, August 2020

[37] M. Bjorklund, “The YANG 1.1 Data Modeling Language”, IETF RFC 7950, October 2016

[38] Helm Charts, https://helm.sh/docs/topics/charts/

[39] ETSI GS NFV-SOL 005, “Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point”, September 2020

[40] Openstack controller node, https://docs.openstack.org/newton/install-guide-rdo/overview.html

[41] Kubernetes Control Plane, https://kubernetes.io/it/docs/concepts/overview/components/

[42] Prometheus, https://prometheus.io

[43] Apache Kafka, https://kafka.apache.org/

[44] Influx DB, https://www.influxdata.com/

[45] Telegraf, https://docs.influxdata.com/telegraf/v1.17/

[46] OSM Juju Proxy Charms, https://osm.etsi.org/wikipub/index.php/Creating_your_VNF_Charm

[47] Deliverable D5.1 EU LOCUS project “Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics”, preliminary version

[48] ETSI GS NFV-SOL 004, “Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; VNF Package and PNFD Archive specification”

[49] ETSI GS NFV-SOL 007, https://docbox.etsi.org/ISG/NFV/open/Publications_pdf/Specs-Reports/NFV-SOL%20007v3.3.1%20-%20GS%20-%20NSD%20file%20structure%20spec.pdf

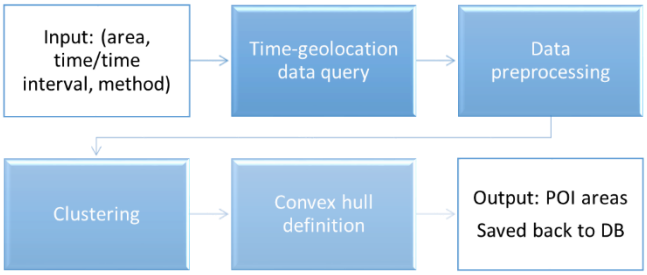
[50] cloud-init, https://cloud-init.io/

[51] ETSI OSM NBI OpenAPIs, https://osm.etsi.org/gitweb/?p=osm/SOL005.git;a=blob_plain;f=osm-openapi.yaml;hb=HEAD

Annex A – LOCUS Function template

SNM-UC1: Knowledge Building for Network Management

Table 4: SNM-UC1 Identification of PoIs functionality template

Functional decomposition	Focusing on the POI functionality and not the analytics on top of POIs: 
Programming language	Python

Dependency list	• libraries for ML:
	  • numpy==1.18.5
	  • matplotlib==3.3.1
	  • scipy==1.4.1
	  • pandas==1.1.1
	  • scikit_learn==0.23.2
	• Dependency to internal data store
	• Dependency –in terms of data availability- on timely arrival to locus internal database of location information from LEN functions

Input data	Dataset/ Data store. The functionality assumes that UE location at time X always stored. Then by a simple query for a given time or time frame we can get the data (to be filtered by the area in question). I.e. Function requests data and data store return reference to data URL (http, db/tcp, hdfs) through a REST API. POI identification is not time critical and does not have to be executed with high frequency. The data acquisition periodicity can indicatively be set to 1 batch UE location dataset/ 5mins.
Input data format	Currently the proposed format for input that can be easily supported is CSV, JSON, XML.
Input parameters	Area, time/time interval definition, POI identification method (HCA, DBSCAN or OPTICS)
Output data	• Dataset / Data store SNM Apps can then have access to the saved results in the LOCUS data store
Output data format	JSON or XML. A result every 5-10 minutes would be OK.
Requirements (1)	It would depend on the amount of UE entries in a given area. An example run in a laptop with Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz, 8 processors, 16GB RAM, 512GB SSD: • HCA: 0.041sec, CPU 55% (800 entries)/ 0.394sec, CPU 170%, 295.680MiB Mem (3200 entries) • OPTICS: 2.952sec, CPU 151% (800 entries)/ 7.09837036 seconds, CPU 171%, 296.059 MiB (3200 entries) • DBSCAN: 0.081sec, CPU 157% (800 rows)/ 0.277sec, CPU 172%, 294.082 MiB (3200 rows) (percentages in CPU, are peak/max percentages)
Requirements (2)	Cloud/ core datacenter

Requirements (3)	Runs clustering/POI detection every 5-10 mins Data acquisition periodicity may differ Possibly, to retrain model if the silhouette metric is too low with additional hyperparameters. This has not been implemented
Status of the implementation	Implementation ready. Minor alignments for functional decomposition as per LOCUS architecture needed. If a more “big data” ML implementation is necessary, then some extra effort would be required.

Annex B – LOCUS MANO REST APIs

B.1 NSD Package Management APIs

Table 5 Create a new NSD information resource

Endpoint	/nsd/v1/ns_descriptors
HTTP Verb	POST
Description	Create a new NSD information resource
Request Content	name, description
Response Content	nsdInfoId
Response Code (on success)	201

Table 6 Upload a new NSD Package

Endpoint	/nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content
HTTP Verb	PUT
Description	Upload a new NSD Package
Request Content	NSD Package archive in tar.gz format
Response Content	none
Response Code (on success)	201

Table 7 Retrieve the NSD Package content

Endpoint	/nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content
HTTP Verb	GET
Description	Retrieve the NSD Package content
Request Content	none
Response Content	NSD Package archive in tar.gz format
Response Code (on success)	200

Table 8 Read NSD content of an onboarded NS package

Endpoint	/nsd/v1/ns_descriptors/{nsdInfoId}/nsd
HTTP Verb	GET
Description	Read NSD content of an onboarded NS package
Request Content	none
Response Content	NSD content in YAML format, compliant with ETSI NFV SOL006 format
Response Code (on success)	200

Table 9 Delete an existing NSD Packageresource

Endpoint	/nsd/v1/ns_descriptors/{nsdInfoId}
HTTP Verb	DELETE
Description	Delete an existing NSD Package resource and its content
Request Content	none
Response Content	none
Response Code (on success)	204

B.2 VNF Package Management APIs

Table 10 Create a new VNF Package information resource

Endpoint	/vnfpkgm/v1/vnf_packages
HTTP Verb	POST
Description	Create a new VNF Package information resource
Request Content	name, description
Response Content	vnfPckgId
Response Code (on success)	201

Table 11 Upload a new VNF Package

Endpoint	/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content

HTTP Verb	PUT
Description	Upload a new VNF Package
Request Content	VNF Package archive in tar.gz format
Response Content	none
Response Code (on success)	201

Table 12 Retrieve the VNF Package content

Endpoint	/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content
HTTP Verb	GET
Description	Retrieve the VNF Package content
Request Content	none
Response Content	VNF Package archive in tar.gz format
Response Code (on success)	200

Table 13 Read VNFD content of an onboarded VNF package

Endpoint	/vnfpkgm/v1/vnf_packages/{vnfPkgId}/vnfd
HTTP Verb	GET
Description	Read VNFD content of an onboarded VNF package
Request Content	none
Response Content	VNFD content in YAML format, compliant with ETSI NFV SOL006 format
Response Code (on success)	200

Table 14 Delete an existing VNF Package resource and its content

Endpoint	/vnfpkgm/v1/vnf_packages/{vnfPkgId}
HTTP Verb	DELETE
Description	Delete an existing VNF Package and its content
Request Content	none

Response Content	none
Response Code (on success)	204

B.3 NetSlice Templates APIs

Table 15 Create a new netslice template resource

Endpoint	/nst/v1/netslice_templates
HTTP Verb	POST
Description	Create a new netslice template resource
Request Content	name, description
Response Content	netsliceTemplateId
Response Code (on success)	204

Table 16 Upload a new netslice template

Endpoint	/nst/v1/netslice_templates/{netsliceTemplateId}/nst_content
HTTP Verb	PUT
Description	Upload a new netslice template
Request Content	netslice template file in YAML format
Response Content	none
Response Code (on success)	201

Table 17 Retrieve the netslice tempalte content

Endpoint	/nst/v1/netslice_templates/{netsliceTemplateId}/nst_content
HTTP Verb	GET
Description	Retrieve the netslice template content
Request Content	none
Response Content	netslice template file in YAML format

Response Code (on success)	200

Table 18 Delete an existing netslice template resource

Endpoint	/nst/v1/netslice_templates/{netsliceTemplateId}
HTTP Verb	DELETE
Description	Delete an existing netslice template and its content
Request Content	none
Response Content	none
Response Code (on success)	204

B.4 NS Instance Management APIs

Table 19 Create a new Network Service instance

Endpoint	/nslcm/v1/ns_instances_content
HTTP Verb	POST
Description	Create a new Network Service instance
Request Content	nsdId, nsName
Response Content	nsInstanceId, nsLcmOpOccId
Response Code (on success)	202

Table 20 Trigger a scale operation for a Network Service Instance

Endpoint	/nslcm/v1/ns_instances/{nsInstanceId}/scale
HTTP Verb	POST
Description	Trigger a scale operation for a Network Service Instance in instantiated status (can be used to scale VNF instances with the use of scaleVnfData request parameters)
Request Content	scaleType, timeout, scaleVnfData
Response Content	nsLcmOpOccId

Response Code (on success)	202

Table 21 Trigger an action for a Network Service Instance

Endpoint	/nslcm/v1/ns_instances/{nsInstanceId}/action
HTTP Verb	POST
Description	Trigger the execution of a Day2 configuration or action (i.e. a primitive of Juju Proxy Charm) for a Network Service instance or a VNF/KNF in instantiated status
Request Content	primitive, primitiveParameters
Response Content	nsLcmOpOccId
Response Code (on success)	202

Table 22 Trigger termination a Network Service Instance

Endpoint	/nslcm/v1/ns_instances/{nsInstanceId}/terminate
HTTP Verb	POST
Description	Trigger the termination for a Network Service instance and all its related VNFs/KNFs
Request Content	timeout
Response Content	nsLcmOpOccId
Response Code (on success)	202

Table 23 Retrieve status of a Network Service Instance operation

Endpoint	/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}
HTTP Verb	GET
Description	Retrieve status of a Network Service Instance operation
Request Content	none
Response Content	status, operationType
Response Code (on success)	200

B.5 NetSlice Instance Management APIs

Table 24 Create a new composed Network Service instance

Endpoint	/nsilcm/v1/netslice_instances_content
HTTP Verb	POST
Description	Create a new composed Network Service instance (in the form of a OSM netslice)
Request Content	netSliceTemplateId, nsiName
Response Content	netsliceInstanceId, nsiLcmOpOccId
Response Code (on success)	202

Table 25 Trigger an action for a composite Network Service Instance

Endpoint	/nsilcm/v1/netslice_instances/{netsliceInstanceId}/action
HTTP Verb	POST
Description	Trigger the execution of a Day2 configuration or action (i.e. a primitive of Juju Proxy Charm) for a composite Network Service instance. The Day2 configuration to execute shall refer to one of the nested Network Services in the composite on, or one of their VNF/KNF instance
Request Content	primitive, primitiveParameters
Response Content	nsiLcmOpOccId
Response Code (on success)	202

Table 26 Trigger termination a composite Network Service Instance

Endpoint	/nsilcm/v1/netslice_instances/{netsliceInstanceId}/terminate
HTTP Verb	POST
Description	Trigger the termination for a composite Network Service instance and all its nested Network Services and related VNFs/KNFs
Request Content	timeout
Response Content	nsiLcmOpOccId
Response Code (on success)	202

Table 27 Retrieve status of a composite Network Service Instance operation

Endpoint	/nsilcm/v1/nsi_lcm_op_occs/{nsiLcmOpOccId}
HTTP Verb	GET
Description	Retrieve status of a composite Network Service Instance operation
Request Content	none
Response Content	status, operationType
Response Code (on success)	200