UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

NETWORK FUNCTION VIRTUALIZATION TECHNOLOGIES APPLIED TO CELLULAR SYSTEMS

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Joel Padilla Pinedo

In partial fulfilment

of the requirements for the degree of

MASTER IN TELECOMMUNICATIONS ENGINEERING

Advisor: Jordi Pérez Romero

Barcelona, January 2020

Title of the thesis:

Network Function Virtualization technologies applied to cellular systems.

Author:

Joel Padilla Pinedo

Advisor:

Jordi Pérez Romero

Abstract

Future 5G networks open a new spectrum of possibilities, both at the level of services it can offer, and at the level of its deployment. This thesis aims to make a study of some of the technologies that make possible the arrival of 5G, such as virtualization and virtualization applied to networks, NFV. In order to better understand the defined standard for NFV, the analysis of market NFV-MANO available tools is included. In addition, the study and evaluation of the deployment process of a virtualized 5G network scenario has been performed with HPE NFV Director.

Acknowledgements

For the development of this project it has been fundamental the support of different people, who have given me the strength to fulfil this thesis.

In first place, I would like to thank Dr. Jordi Pérez Romero, the tutor of this thesis, for giving me the chance to develop this project. I express my gratitude for his technical advices, his patience when correcting the documentation and his encouraging and motivating attitude during the course of the project. Moreover, I want to acknowledge HPE Chief Technologist Rod Anliker, who has been involved during all the thesis development giving me technical and mental support as well as correcting the present document. I would also like to thank HPE NFV Director Product Architect Ignacio Aldama for giving me the opportunity to use the HPE NFV Director tool and for all the time spent in solving my doubts.

Last but not least, I would like to thank Irene for her invaluable help, and my family for supporting me and cheering me up during these months in which I have been committed to this thesis.

Thank you very much all of you.

Revision history and approval record

Revision	**Date**	**Purpose**
0	10/01/2020	Document creation
1	17/01/2020	Document revision
2	29/01/2020	Document correction

Written by:		Reviewed and approved by:	
Date	10/01/2020	Date	29/01/2020
Name	Joel Padilla	Name	Jordi Pérez Romero
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1 Introduction

5G networks have been conceived as a cloud native technology that will exploit the inherent flexibility associated to the introduction of service base model infrastructures. In comparison to previous technologies like 2G, 3G, and 4G, which were all deployed as fixed hardware appliances, 5G has a more modular design, achieved by specifying a set of Network Functions (NFs). This allows a stronger decoupling between logical and physical architecture, facilitating the virtualization of the different NFs running on generic computer hardware.

This can be achieved by means of virtualization tools. Virtualization in conjunction with cloud computing have resulted in lowering the cost of computing by pooling resources in shared data centers. By using virtualization, 5G core networks can be shrunk in physical size by running various components of the core network as communicating virtual machines (VMs).

In this direction, Network Function Virtualization (NFV) appears as the technology that will help telco providers to deploy 5G networks, reducing CAPEX and OPEX costs, as well as making networks more flexible and faster to update and change. In this sense, NFV can be used to create and manage NFs for both the core network and Radio Access Network (RAN) parts of 5G through the software implementation of network functions running on general purpose computing/storage resources. The advent of the NFV paradigm provides an inherent capability to add new functionalities, extend, upgrade or evolve existing functionalities and to customize the network on a per-tenant basis.

During the 5G development phase, telco and IT companies have started working together to develop tools that allow the deployment of 5G networks by using the technologies above mentioned. Given the novelty of these tools and its incipient phase of development, this project intended to make an analysis of the current virtualization technologies, platforms and tools existing nowadays in the market and study their applicability in the realization of a virtualized 5G network, focusing on the core network part functions.

This Thesis consists of seven chapters, starting by this introductory chapter, which also covers the Thesis' objectives and a project plan.

The second chapter consists in the state-of-the-art of the 5G system, which encompasses its evolution from 4G, the 5G service-based architecture and a description of related technologies such as network slicing and Multi-access Edge Computing (MEC). Moreover, the 5G network functions are described, including the core and NG-RAN parts of the system.

The third chapter contains a description of the concepts related to virtualization, including the advantages of virtualization, the role of the hypervisor and a comparison of different virtualization approaches.

The fourth chapter revises the background and basic concepts of the Network Function Virtualization (NFV) technologies. Then, a description of the NFV architecture standardized by ETSI is provided and, based on this, the main advantages of NFV are discussed. Moreover, different NFV-Management-And-Orchestration (NFV-MANO) solutions in the market are introduced and compared, including the Open Source MANO

(OSM), the Open Network Automation Platform (ONAP) and the Hewlett Packard Enterprise (HPE) Network Function Virtualization Director (HPE NFVD).

The fifth chapter includes the definition and deployment of a scenario containing the main 5G core network functions by means of the HPE NFVD framework. After defining the scenario, a description of the HPE NFVD graphical interface is provided. Then, the procedure to deploy a generic VNF is also described. After this, the deployment of the 5G core defined scenario containing two slices is performed. Finally, an evaluation of the ease of use, development time and limitations of the employed tool is provided.

Finally, in the sixth chapter it can be found the cost assessment of the Thesis while in the seventh chapter the conclusions of the Thesis and a discussion about future work are provided.

1.1 Objectives

The main objectives of the project are the following:

- Perform a state-of-the-art analysis of 5G system architecture.
- Perform a state-of-the-art analysis of the existing network function virtualization technologies and standards as well as the current NFV tools in the market
- Study and evaluate the applicability of a market tool for virtualizing the 5G core network functions.

1.2 Project Plan and deviations

The project has been developed in different phases, as it can be observed in the following Gantt diagram.

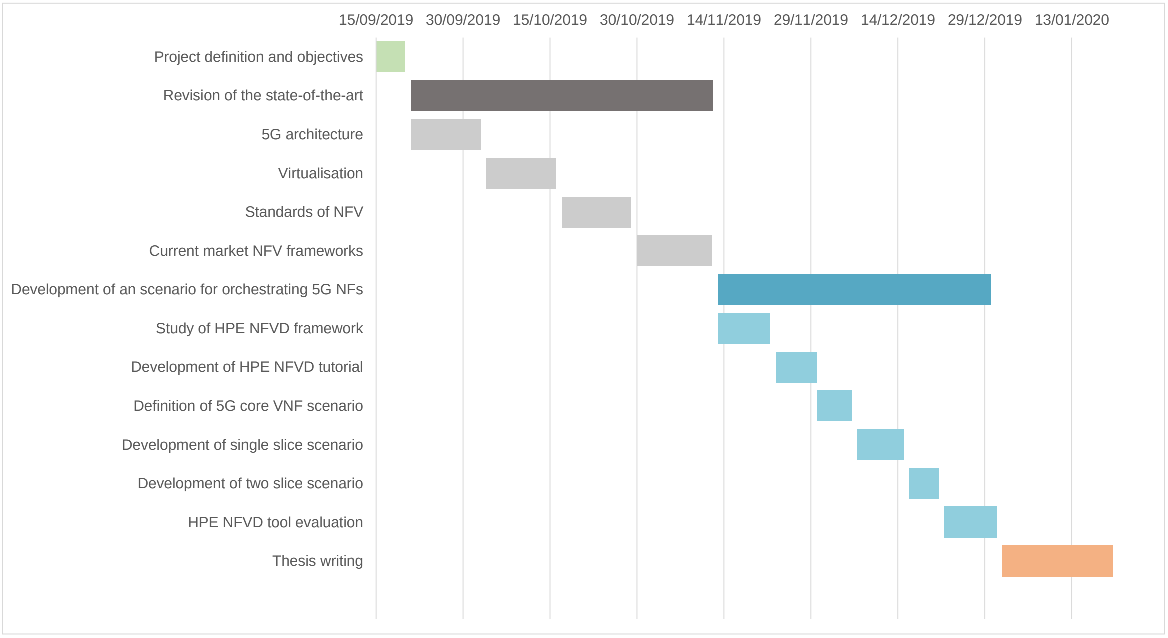


Fig. 1. Project plan Gantt diagram

The first phase included the literature review related to the topic of the Thesis. This involved the revision of the state-of-the-art of the 5G and its architecture as well as the virtualization technologies and standards required to support it. Then, a research of the current NFV frameworks available in the market has been performed.

The second phase of the thesis consists in developing a 5G core VNF deployment test by using the HPE NFVD framework, whose access has been provided temporally by HPE. For this purpose, a detailed study of the framework has been carried out, focusing on the identification of the necessary steps to deploy a generic VNF. After this, a simplified 5G core VNF scenario involving two slices has been defined for its latter deployment in HPE NFVD framework. Finally, an evaluation of the tool in terms of its ease of use, development time and limitations has been performed.

The last phase of the development of the Thesis was the writing of this document.

2 5G system

This chapter explains the 5G system architecture as a background for the project. It firstly introduces the 5G architecture reference model and points out the key evolution factors from 4G system. Then, the service-based architecture is described and a description of network slicing and MEC technologies are included. The chapter finishes with the explanation of the different network functions devised for 5G systems regarding the core network and NG-RAN parts.

2.1 5G Architecture Reference Model

Like in previous systems (e.g. LTE), the 5G system architecture encompasses three big constitutive blocks [1]:

- **User Equipment (UE)**
- **Next Generation – Radio Access Network (NG-RAN):** Responsible for all the radio-related functionality of the system.
- **5G Core Network (5GC):** Responsible of the non-radio access related functions such as authentication, charging, set-up of end-to-end connections, mobility management, etc.

Fig. 2 shows a representation of the three big constitutive blocks:

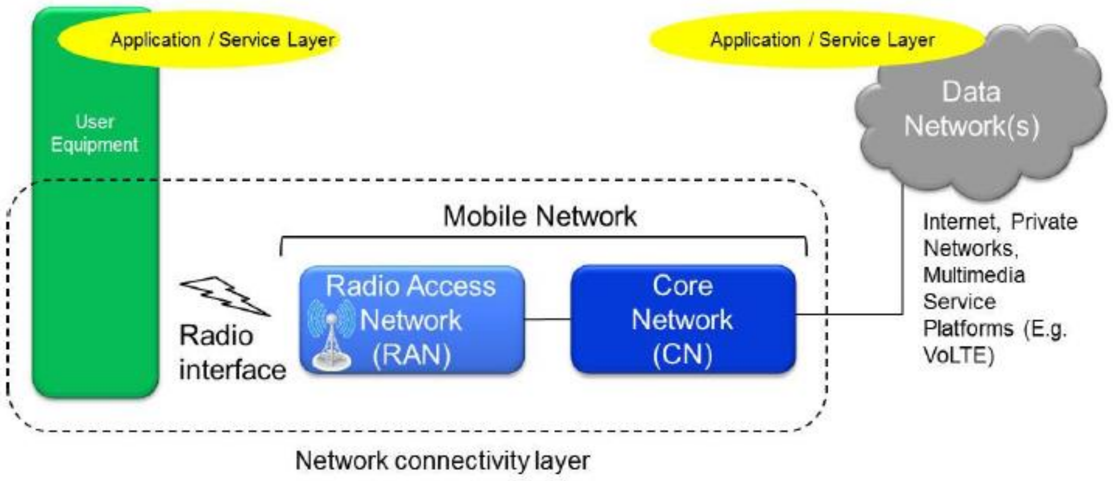

Application / Service Layer
User Equipment
Application / Service Layer
Data Network(s)
Mobile Network
Radio interface
Radio Access Network (RAN)
Core Network (CN)
Internet, Private Networks, Multimedia Service Platforms (E.g. VoLTE)
Network connectivity layer

Fig. 2 5G constitutive blocks, UE, NG-RAN and 5GC [1]

The latter aspects are discussed in the following subsections.

2.2 Evolution from 4G EPC

LTE Architecture is defined in terms of network entities, each grouping a number of mobile network functions. Interactions between network entities are specified through interfaces connecting them as shown in Fig. 3. A brief definition of the different LTE network entities is provided in the following:

E-UTRAN Node B (eNB): Provides connectivity between user equipment (UE) and the Evolved Packet Core (EPC) backbone network.

Serving Gateway (S-GW): The S-GW routes and forwards user data packets, while also acting as the mobility anchor for the user plane during inter eNB handovers and as the anchor for mobility between LTE and other 3GPP technologies.

Mobility Management Entity (MME): The MME is the key control-node for the LTE access-network. It is responsible for idle mode UE paging and tagging procedure including retransmissions. It is involved in the bearer activation/deactivation process and is also responsible for choosing the S-GW for a UE at the initial attach and at time of intra-LTE handover involving Core Network (CN) node relocation.

Packet Data Network (PDN) Gateway (P-GW): The P-GW provides connectivity from the UE to external packet data networks by being the point of exit and entry of traffic for the UE. A UE may have simultaneous connectivity with more than one P-GW for accessing multiple PDNs. The P-GW performs policy enforcement, packet filtering for each user, charging support, lawful interception and packet screening. Another key role of the P-GW is to act as the anchor for mobility between 3GPP and non-3GPP technologies such as WiMAX and 3GPP2.

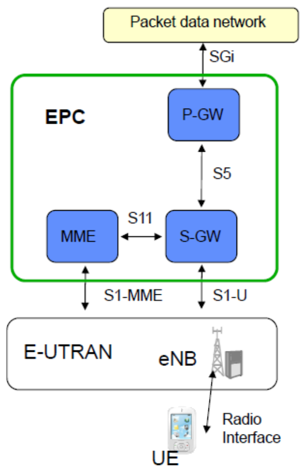


Fig. 3 LTE Architecture

This architecture presents some limitations:

• It is not possible to split parts of a functional entity and place them at different locations (e.g. place the user plane functions of the S-GW for delay critical services close to the eNB).

• A network entity includes both control and user plane functionality.

• Hard to optimize/customize the network to provide different behaviors for applications with very different types of requirements (e.g. delay critical vs. bit rate demanding applications).

In order to overcome these limitations, the 5G core has evolved from the 4G EPC in two steps:

- Control and User Plane Separation (CUPS) of the 4G EPC.
- Reorganizing the 4G EPC CUPS functions into services.

2.3 Service based architecture

Compared to previous generations the 3GPP 5G system architecture is service based. That means wherever suitable the architectural elements are defined as network functions that offer their services via interfaces of a common framework to any network functions that are permitted to make use of these provided services. Network Repository Functions (NRF) allow every network function to discover the services offered by other network functions. This architectural model, which further adopts principles like modularity, reusability and self-containment of network functions, is chosen to enable deployments to take advantage of the latest virtualization and software technologies [2]. Instead of specifying network entities, a more modular design is achieved by specifying a set of NFs which allows stronger decoupling between logical and physical architecture, facilitating the virtualization of the different NFs running on generic computer hardware. Furthermore, NFs can be physically implemented in different ways (e.g. all of them in a single physical node, distributed across multiple nodes or running on a cloud platform). Fig. 4 shows the Service-Based architecture of the 5G system. In Sections 2.6 and 2.7 a description of the core and RAN NFs is included, respectively.

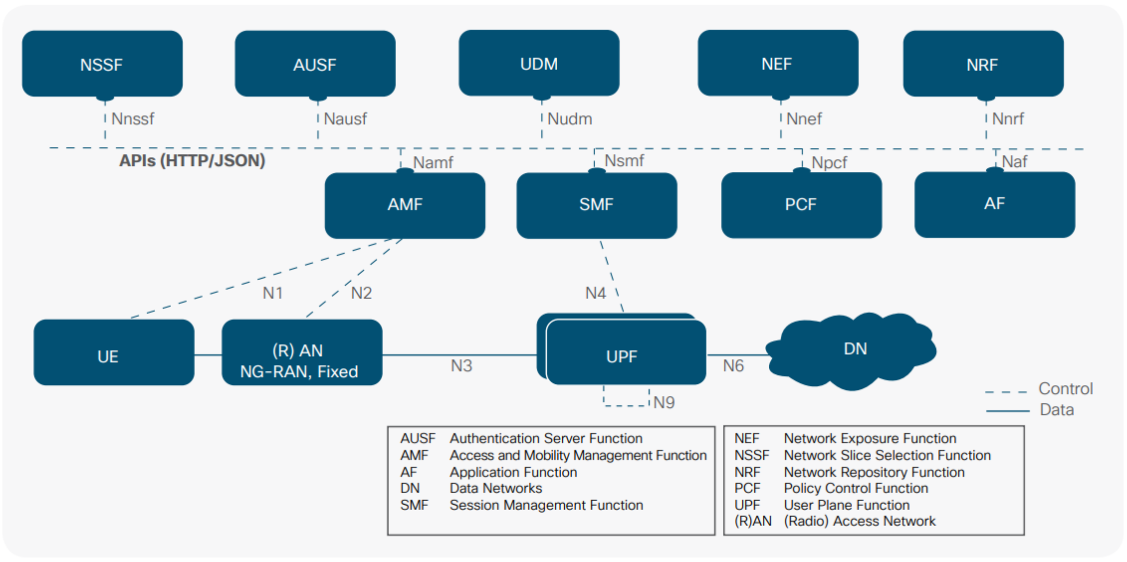

NSSF
AUSF
UDM
NEF
NRF
Nnssf
Nausf
Nudm
Nnef
Nnrf
APIs (HTTP/JSON)
Namf
Nsmf
Npcf
Naf
AMF
SMF
PCF
AF
N1
N2
N4
UE
(R) AN
NG-RAN, Fixed
N3
UPF
N6
DN
N9
Control
Data
AUSF Authentication Server Function
AMF Access and Mobility Management Function
AF Application Function
DN Data Networks
SMF Session Management Function
NEF Network Exposure Function
NSSF Network Slice Selection Function
NRF Network Repository Function
PCF Policy Control Function
UPF User Plane Function
(R)AN (Radio) Access Network

Fig. 4 Service-based representation of the 5G system architecture [3]

With the split of control plane and user plane, NFs responsible of the control plane are different from those responsible of the user plane allowing independent scalability and evolution (e.g. allocating more capacity to the control plane without affecting the user plane). Moreover, it allows flexible deployments, e.g. centralized location for control plane or distributed (remote) location for user plane.

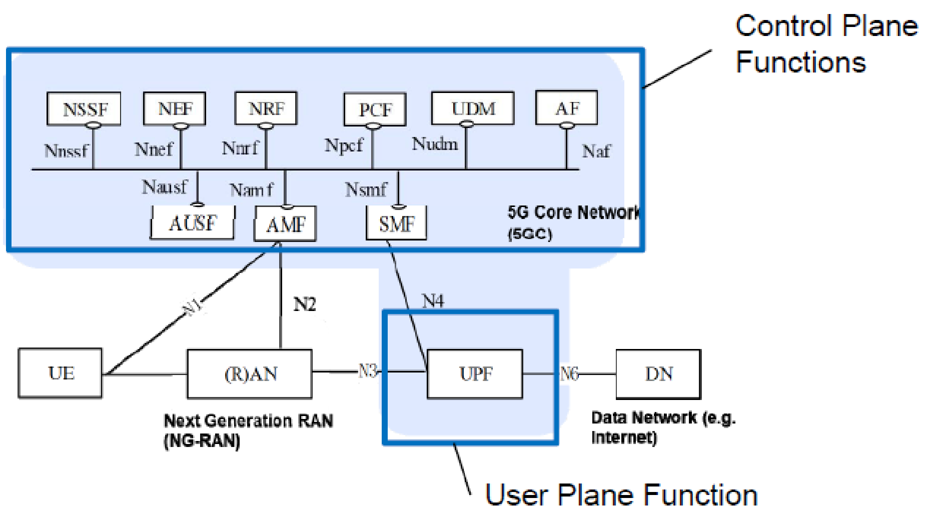


Fig. 5 Split of control plane and user plane [4]

2.4 Support of Network slicing

A distinct key feature of the 5G system architecture is network slicing. The previous generation supported certain aspects of this with the functionality for dedicated Core Networks. Compared to this 5G network slicing is a more powerful concept and includes the whole Public Land Mobile Network (PLMN). Within the scope of the 3GPP 5G system architecture a network slice refers to the set of 3GPP defined features and functionalities that together form a complete PLMN for providing services to UEs. Network slicing allows for controlled composition of a PLMN from the specified network functions with their specifics and provided services that are required for a specific usage scenario. Earlier system architectures enabled typically rather a single deployment of a PLMN to provide all features, capabilities and services required for all wanted usage scenarios. Much of the capabilities and features provided by the single, common deployment were in fact required for only a subset of the PLMN's users/UEs. Network slicing enables the network operator to deploy multiple, independent PLMNs where each is customized by instantiating only the features, capabilities and services required to satisfy the subset of the served users/UEs or a related business customer needs [2].

With 5G, NFs can be individually instantiated for each network slice, and placed where appropriate. In this way, multiple network slices can be created, each one composed by a collection of control plane and user plane NFs customized to the needs of the slice. E.g. one network slice can include the NFs to support mobile broadband services with full mobility support, and another one to support non-mobile, latency-critical industry applications. Fig. 6 is an abstract representation of a network deploying network slices.

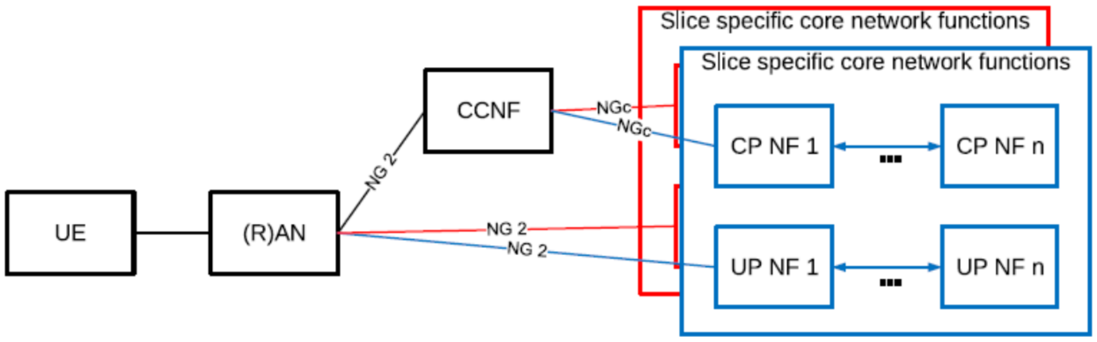

Slice specific core network functions
Slice specific core network functions
CCNF
NGc
NGc
NG 2
UE
(R)AN
NG 2
NG 2
CP NF 1
CP NF n
UP NF 1
UP NF n

Fig. 6 Representation of 2 network slices

Fig. 7 presents more specifics of 3GPP network slicing. In that figure, network slice #3 is a straightforward deployment where all network functions serve a single network slice only. The figure also shows how a UE receives service from multiple network slices, #1 and #2. In such deployments, there are network functions in common for a set of slices, including the AMF and the related policy control (PCF) and NRF. This is because there is a single access control and mobility management instance per UE that is responsible for all services of a UE. The user plane services, specifically the data services, can be obtained via multiple, separate network slices. In the figure, slice #1 provides the UE with data services for Data Network #1, and slice #2 for Data Network #2. Those slices and the data services are independent of each other apart from interaction with common access and mobility control that applies for all services of the user/UE. This makes it possible to tailor each slice for e.g. different QoS data services or different application functions, all determined by means of the policy control framework.

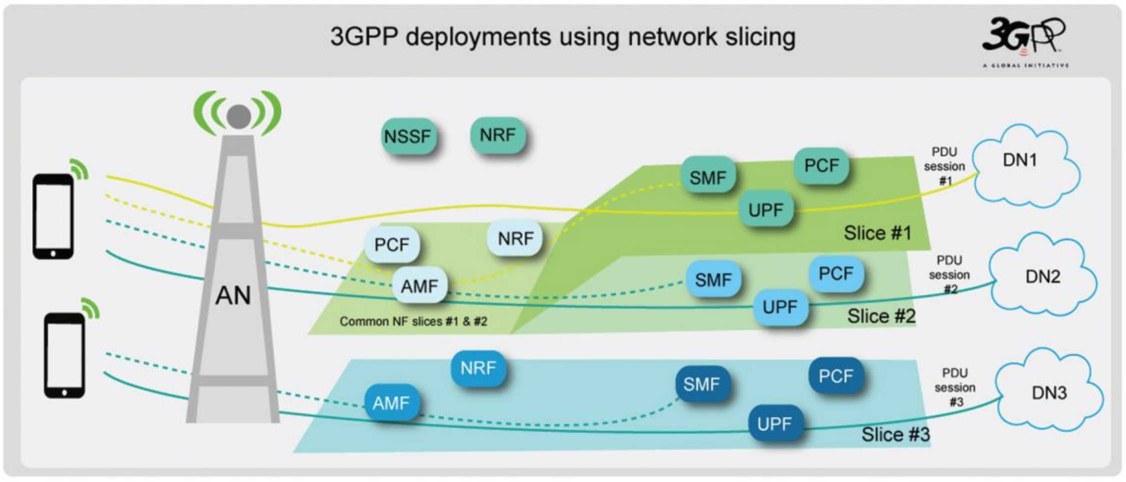


Fig. 7 Network functions composing network slices [2]

2.5 <u>Support of Multi-access Edge Computing (MEC)</u>

MEC uses the wireless access network to provide services and cloud computing functions required by telecom users, and to construct a carrier class service environment with high performance, low latency and high bandwidth to improve communication experience of mobile users. In 2014, the ETSI formally included MEC into the standards. Edge computing is an evolution of cloud computing that brings application hosting from centralized data centers down to the network edge, close to the consumers and the data generated by applications. The main advantages are the computation offloading, distributed content delivery and caching and low latency services.

In order to take profit of the advantages provided by the edge computing in 5G, the technology needs to support to connect the 5G core to a local area data network (LADN) where the applications are implemented and the UPF must be able to perform the local routing of certain traffic to the LADN. Fig. 8 shows how the MEC system is deployed in an integrated manner in 5G network.

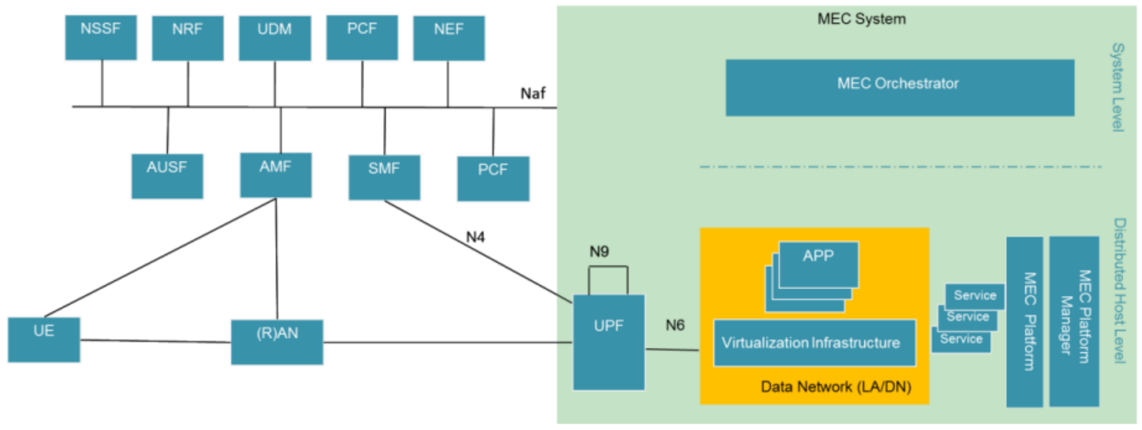


Fig. 8 Integrated MEC deployment in 5G network [5]

In the MEC system on the right-hand side of Fig. 8 the MEC orchestrator is a MEC system level functional entity that, acting as an Application Function (AF), can interact with the Network Exposure Function (NEF), or in some scenarios directly with the target 5G NFs. On the MEC host level it is the MEC platform that can interact with these 5G NFs, again in the role of an AF. The MEC host, i.e. the host level functional entities, are most often deployed in a data network in the 5G system. While the NEF as a Core Network function is a system level entity deployed centrally together with similar NFs, an instance of NEF can also be deployed in the edge to allow low latency, high throughput service access from a MEC host [5].

In terms of physical deployment of MEC hosts, there are multiple options available based on various operational, performance or security related requirements. The following figure gives an outline of some of the feasible options for the physical location of MEC.

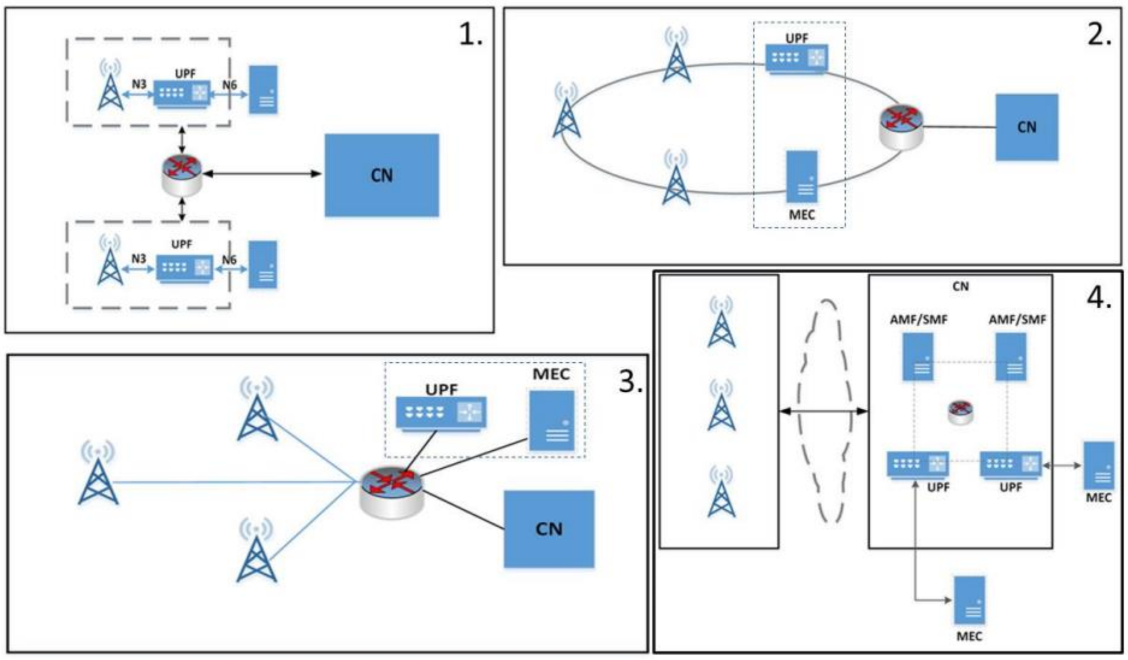


Fig. 9 Examples of the physical deployment of MEC

1. MEC and the local UPF collocated with the Base Station.
2. MEC collocated with a transmission node, possibly with a local UPF.
3. MEC and the local UPF collocated with a network aggregation point.
4. MEC collocated with the Core Network functions (i.e. in the same data center).

The options presented above show that MEC can be flexibly deployed in different locations from near the Base Station to the central Data Network. Common for all deployments is the UPF that is deployed and used to steer the traffic towards the targeted MEC applications and towards the network.

The MEC has the characteristics of localization, proximity, low latency, location awareness, and access to network context information. With MEC technology, network operators are capable of making use of existing network infrastructure to achieve low latency and real time processing at network edge, thus effectively reducing operating costs while improving service levels. The MEC can create a new value chain and ecosystem. Mobile network operators should work with upstream and downstream vendors to jointly promote MEC technology research and development and commercialization to accelerate the development and deployment of 5G.

2.6 <u>5G core NFs</u>

The 5G core is a mesh of interconnected services as shown in Fig. 4. The most important functions are the User Plane Function (UPF), Access and Mobility management Function (AMF) and the Sessions Management Functions. In the following a brief explanation of each function is included:

User Plane Function (UPF): The User Plane Function deals with the user plane communication in the 5GC, acting as a gateway between the RAN and the external Data Network (DN) (e.g. Internet). The main functionalities are: packet routing and forwarding, downlink packet buffering and downlink data notification triggering, QoS handling and traffic measurements.

Access and Mobility management Function (AMF): The Access and Mobility management Function is a control-plane function in charge of handling the control signaling between the UE and the 5GC. The main functionalities are: registration management, connection management to establish the control plane signaling with the UE, mobility management (e.g. idle mode UE reachability), control and execution of paging and support of infra-system and inter-system mobility.

Session Management Function (SMF): The Session Management Function is a control-plane function in charge of the following main functionalities: session establishment, modification and release, UE IP address allocation and management, control of policy enforcement and QoS and configuration of traffic steering at UPF to route traffic to proper destination.

Other functions are included in the following:

Policy Control Function (PCF): Provides policy rules (e.g. authorized QoS for each service data flow) to the network functions in charge of enforcing them (e.g. SMF).

Application Function (AF): Allows interacting with the applications making use of the network. This can be used for applications that require dynamic policy control, e.g. for dynamically modifying the bit rate to be provided. Based on interactions with these applications policy requirements are provided to the PCF.

Unified Data Management (UDM): User identification handling, subscription management, access authorization based on subscription data (e.g. roaming restrictions), generation of authentication credentials. It uses subscription data that may be stored in the UDR (Unified Data Repository).

Authentication Server Function (AUSF): Provides the UE authentication service.

Network Slice Selection Function (NSSF): Supports the selection of the Network Slice instance(s) serving a UE.

Network Exposure Function (NEF): It is used to expose services of the 5G core towards other networks (e.g. third party providers, verticals, etc.). This allows the fast creation of new services making use of the 5G core.

Network function Repository Function (NRF): Repository with the profile of the available NF instances (id, PLMN ID, network slice identifiers, capacity information, etc.) and their supported services. It allows a NF to discover the services offered by the other NFs of the core network. This provides a lot of flexibility for defining the interactions between NFs, and allows that any NF can directly interact with another one.

2.7 5G NG-RAN NF

The NG-RAN is the network function that connects the User Equipment with the 5G Core. It can be composed by two types of nodes:

- gNode B (gNB): it operates with the 5G New Radio (NR) technology
- next generation eNodeB (ng-eNB): it operates with the LTE technology

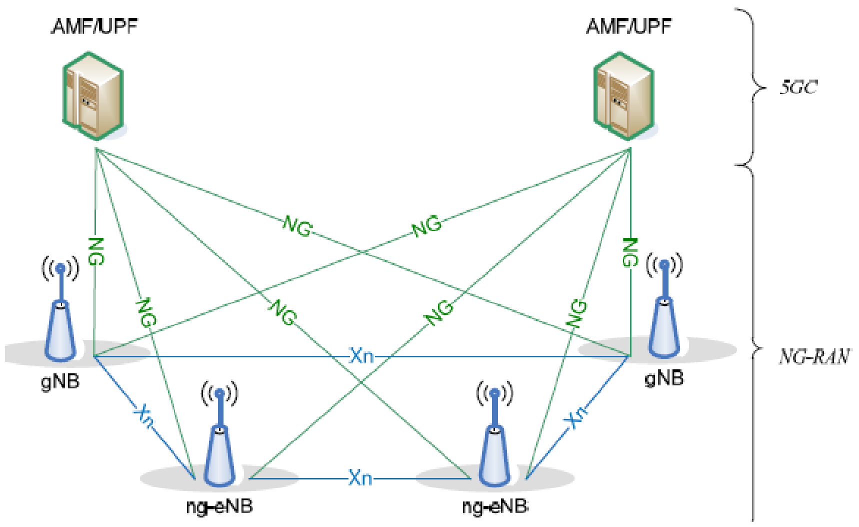


Fig. 10 NG-RAN in relation to the 5G system [6]

The gNB is responsible for all the radio-related functions associated to one or multiple cells supporting the 5G NR technology. The main functionalities are:

- Radio resource management (e.g. admission control, packet scheduling, connection mobility control, etc.).
- Routing of user-plane information to the UPF and control-plane information to the AMF.
- Measurement and measurement reporting configuration.
- QoS flow management.
- Connection setup and release.
- Scheduling and transmission of paging messages.
- Scheduling and transmission of system broadcast information.

3 Virtualization Technologies

The above described 5G system architecture is a service-based model where each function of the technology can be abstracted from the hardware by means of virtualization. Given the relevance of virtualization in 5G, this chapter aims at providing the main concepts of virtualization and the state-of-the-art of virtualization approaches.

3.1 Virtualization

When applications were more monolithic back in the 1960s through the 1990s, the User Interface (UI), application logic, storage management, data management and networking functions were all hosted together on a single system [7]. Back then, the industry focus was on making the systems themselves "fault tolerant". This was accomplished by designing mainframe systems that used multiple processors, stacks of memory, storage adapters and network adapters; they included system firmware that monitored the health of individual components and moved workloads to surviving components in case a component failed or became unresponsive. These systems were extremely expensive when compared to standard off-the-shelf configurations, and were only used to host the most critical workloads.

The high cost of redundant mainframe systems, lead the industry to the study of new ways of assuring business workloads continuity. Virtualization appeared as a tool that allowed companies to have their resources in high availability, thus allowing them to ensure the continuity of workloads with much lower costs.

In computing, virtualization consists in the generation of a virtual version of some technological resources through the use of software tools [8]. These technological resources can be hardware, operating systems, storage and networking, among others. In other words, virtualization can be understood as the abstraction of computer resources. Fig. 11 shows the evolution of infrastructure virtualization, starting by a traditional monolithic server to fully virtualized infrastructure with several applications running on top of it.

In order to execute the abstraction of the resources, it is necessary the use of a software tool, usually called Hypervisor or VMM (Virtual Machine Monitor). The Hypervisor manages, controls and arbitrates the four main resources of a computer: CPU, Memory, Input/Output (I/O), and Network connectivity. This allows the dynamic distribution of the resources among all the Virtual Machines (VMs) defined in the mainframe. In that way, it is possible to have different virtual computers working on top of the same physical computer, also referred as “host”.

Virtualization is responsible of creating an external interface that encapsulates the underlying implementation through the combination of resources housed in different physical locations or by simplifying the control system. In general, the VM simulates a stand-alone hardware, including a complete Operating System (OS) that runs as if it was installed. For example, a computer that is running Microsoft Windows may host a VM simulating a computer with the Ubuntu Linux operating system [9].

There are different forms of virtualization: the virtualization of server hardware, server software, user sessions, applications, and also the creation of VMs on a desktop computer.

In recent years, the advanced development of new platforms and virtualization technologies has made attention to this concept, making it possible to develop its use in different technological fields, such as mobile systems and communication networks in general.

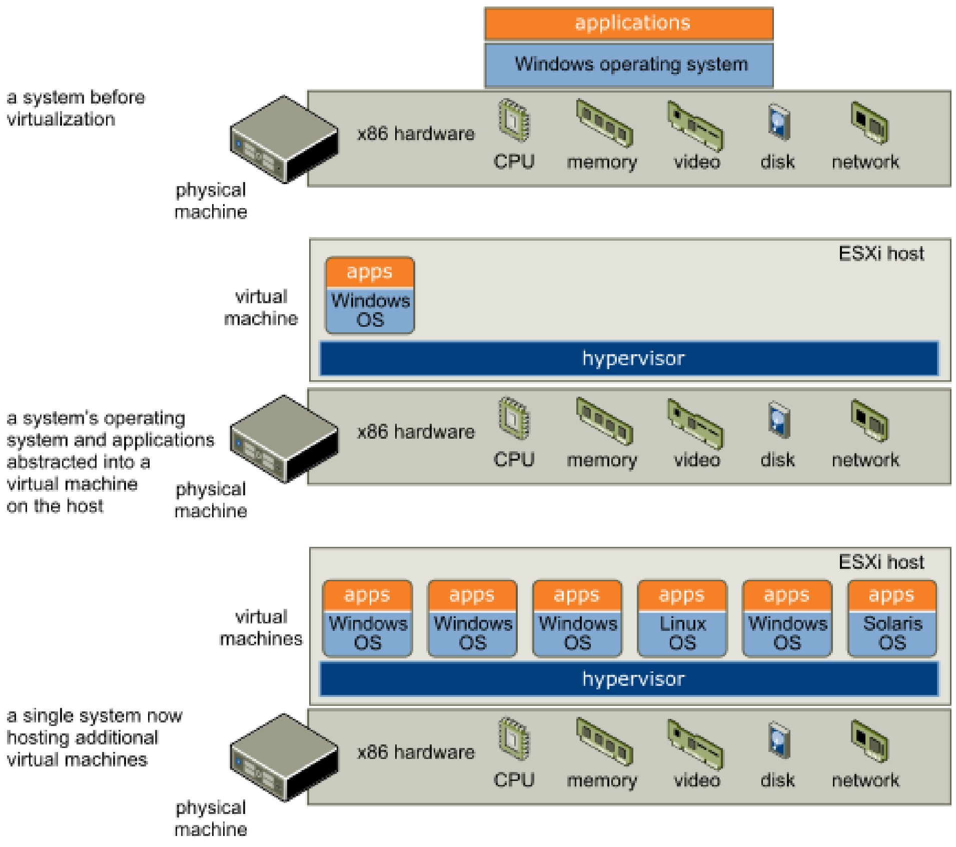

applications
Windows operating system
a system before virtualization
x86 hardware
CPU
memory
video
disk
network
physical machine
ESXi host
apps
Windows OS
virtual machine
hypervisor
a system's operating system and applications abstracted into a virtual machine on the host
virtual machines
Linux OS
Solaris OS
a single system now hosting additional virtual machines

Fig. 11. Virtualizing the Infrastructure. ESXI refers to the VMware hypervisor in [8].

3.1.1 Advantages of virtualization

In the following, the different advantages of virtualization in comparison to former legacy systems are included [10].

Higher utilization rates: Before virtualization, the typical rates of server and storage utilization were around 10%-15%. Virtualization allows rescheduling and moving workloads in order to make a better use of the infrastructure increasing the utilization rates up to 60%-80%.

Resource consolidation: Through virtualization it is possible to share the storage and computing resources among multiple users, resulting in cost savings and better efficiency.

Greater energy efficiency: Using virtualization for resource consolidations you can save significantly on total energy consumption.

Reducing footprint: Server extension appears as a serious problem in most enterprise data centers. It is not always an option, because construction costs are very high. Virtualization can relieve tension by consolidating many virtual systems into fewer physical systems.

Disaster Recovery: The virtualization facilitates the recovery of the systems, being able to raise a VM in a matter of minutes in another server.

Reduced operating costs: The average company spends 8$ on maintenance for every 1$ invested in new infrastructure [11]. Virtualization can change the service-to-administration ratio, reduce the total administrative workload and save operating costs.

Fast provisioning: Ability to provision new applications in minutes, rather than days or weeks.

Centralized management: Centralized and simplified global administration. It allows managing the data center as a pool of resources, memory, network and storage capacity available in our infrastructure.

Testbeds: Improvement in the processes of cloning and copying of systems, greater ease to create test environments that allow new applications to be launched without impacting production, speeding up the testing process.

Isolation: A general system failure of a VM does not affect the rest of VMs. For instance, if a problem occurs in any of the VMs, it is possible to reboot the application in a new VM in a matter of minutes to have the least possible impact.

Live migration: Hot migration of VMs (without loss of service) from one physical server to another, eliminating the need for planned shutdowns for maintenance of physical servers.

3.1.2 Hypervisor

A Hypervisor is a software program that is capable of hosting different VMs with different OSs installed and running over the same single hardware resources [12][13]. Hence, it has the flexibility to support several VMs with multiple OS and applications running on a single hardware resource. Moreover, a hypervisor is responsible for resource allocation to the VM as well as responsible for monitoring and managing VMs through coordination with the underlying hardware primary OS.

Hypervisors can be classified in two types, as shown in Fig. 12:

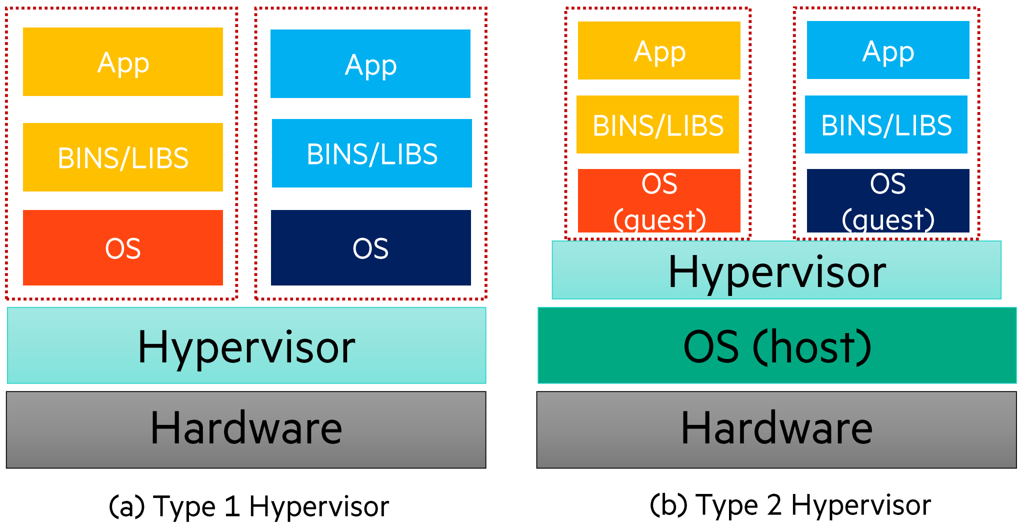


Fig. 12. Hypervisor types.

Type 1 hypervisors, also known as bare metal or native or embedded hypervisors (hardware-based hypervisors), do not need any host OS because the communication to hardware resources is direct with full visibility of hardware resources [13]. The most known are the following [14]:

VMware ESXi: This hypervisor offer advanced features and scalability, and require licensing. There are some lower-cost bundles that VMware offers and they can make hypervisor technology more affordable for small infrastructures. VMware is the leader in the Type-1 hypervisors (Fig. 13). Their vSphere/ESXi product are available in a free edition and 5 commercial editions.

Microsoft Hyper-V: The Microsoft hypervisor, Hyper-V doesn't offer many of the advanced features that VMware's products provide. However, with vSphere/ESXi, Hyper-V is one of the top 2 Type-1 hypervisors (Fig. 13). It was first released with Windows Server, but now Hyper-V has been greatly enhanced with Windows Server 2012 Hyper-V. Hyper-V is available in both a free edition and 4 commercial editions.

Citrix XenServer: It began as an open source project. The core hypervisor technology is free, but like VMware's free ESXi, it has almost no advanced features. Today, the Xen open source projects and community are at Xen.org.

Oracle VM: The Oracle hypervisor is based on the open source Xen but support and product updates have a cost. Oracle VM lacks many of the advanced features found in other bare-metal virtualization hypervisors.

KVM: This is a virtualization infrastructure for the Linux kernel. It supports native virtualization on processors with hardware virtualization extensions. The open-source KVM (or Kernel-Based Virtual Machine) is a Linux-based hypervisor that can be added to most Linux operating systems including Ubuntu, Debian, SUSE, and Red Hat Enterprise Linux, but also Solaris, and Windows.

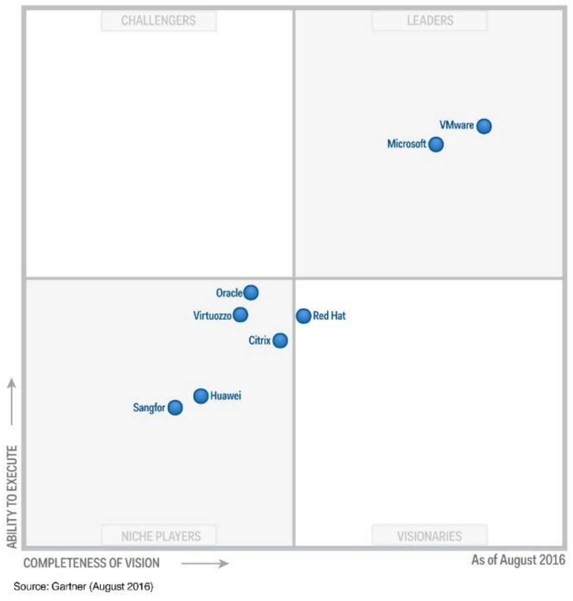


Fig. 13 Magic Quadrant for x86 Server Virtualization infrastructure [15].

The Type 2 hypervisors, also known as hosted or embedded hypervisor (software-based hypervisor), requires a host OS because the Type 2 hypervisors run on top of the supported OS (an additional layer that interacts with the underlying hardware resources in order to manage VM/Server). The most known are the following:

VMware Workstation/Fusion/Player: VMware Player is a free virtualization hypervisor. It is intended to run only one VM and does not allow creating VMs. VMware Workstation is a more robust hypervisor with some advanced features, such as record-and-replay and VM snapshot support. VMware Workstation has three major use cases: for running multiple different operating systems or versions of one OS on one desktop, for developers that need sandbox environments and snapshots, or for labs and demonstration purposes.

VMware Server: VMware Server is a free, hosted virtualization hypervisor that's very similar to the VMware Workstation. VMware has halted development on Server since 2009.

Microsoft Virtual PC: This is the latest Microsoft's version of this hypervisor technology, Windows Virtual PC and runs only on Windows 7 and supports only Windows operating systems running on it.

Oracle VM VirtualBox: VirtualBox hypervisor technology provides reasonable performance and features if you want to virtualize on a budget. Despite being a free, hosted product with a very small footprint, VirtualBox shares many features with VMware vSphere and Microsoft Hyper-V.

Red Hat Enterprise Virtualization: Red Hat's Kernel-based Virtual Machine (KVM) has qualities of both a hosted and a bare-metal virtualization hypervisor. It can turn the Linux kernel itself into a hypervisor so the VMs have direct access to the physical hardware.

In order to compare the two types of hypervisors, Table 1 has been included.

Table 1 Type 1 and Type 2 hypervisor comparison [13]

Type 1 Hypervisor	**Type 2 Hypervisor**
Directly runs on server hardware	Runs on top of the supported OS
Minimizing overhead due to direct hypervisor interaction with hardware resources	Incur overhead as hypervisor runs on top of the supported OS
Provide better hardware resource utilization	Provide less hardware resource utilization
More secure due to hardware based hypervisor	Less secure due to a software based hypervisor
Hard to Set-up	Easy to Set-up

3.2 <u>Virtualization approaches</u>

Given that the virtualization can be applied in a wide range of application types with different requirements (i.e. level of isolation, image size, instantiation, etc.), diverse virtualization approaches have been developed in the market. In the following, the main state-of-the-art framework approaches are described: Virtual Machines, Software Containers and UniKernels.

3.2.1 Virtual Machines

As we have already described, a VM is hosted on a physical server with certain hardware resources. Through a hypervisor, it can be generated a file that contains all the data necessary to create a VM to which is generally assigned a part of the host's physical resources. The sizing of the VM will depend on the resources available and the characteristics of the application that it is going to be executed on this machine. In Fig. 14, it is included a high-level scheme of a physical server or host hosting *n* VMs. In this case, the figure shows a type 2 hypervisor although for type 1 it would be very similar. The only difference is that for the type 1 hypervisor we would not have a Host Operating System and the hypervisor would run directly on the hardware.

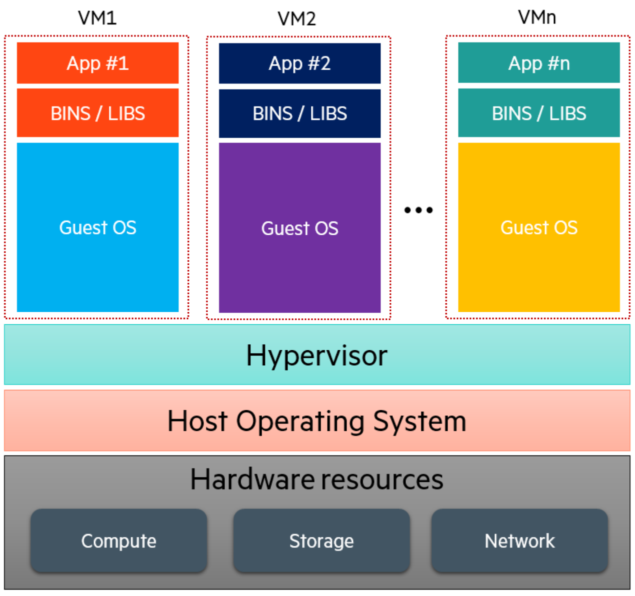


Fig. 14 High-level VM framework

A VM consists of several files that are stored on a storage device. The key files are the configuration file, virtual disk file, NVRAM setting file, and log file.

Table 2 Virtual Machine Files

File	**Description**
.vmx	Virtual machine configuration file
.vmxf	Additional virtual machine configuration files
.vmdk	Virtual disk characteristics
-flat.vmdk	Virtual machine data disk
.nvram	Virtual machine BIOS configuration
.vmsd	Virtual machine snapshots
.vmsn	Virtual machine snapshots data file
.vswp	Virtual machine swap file
.vmss	Virtual machine suspend file

.log	Current virtual machine log file
-#.log	Old virtual machine log files

Some of the most interesting functions that VMs enable are the following [8]:

Snapshots: A snapshot is a state of a VM, and generally its storage devices, at an exact point in time. A snapshot enables the VM's state at the time of the snapshot to be restored later, effectively undoing any changes that occurred afterwards. This capability is useful as a backup technique, for example, prior to performing a risky operation.

Migration: The snapshots described above can be moved to another host machine with its own hypervisor; when the VM is temporarily stopped, snapshotted, moved, and then resumed on the new host, this is known as migration. If the older snapshots are kept in synchronization regularly, this operation can be quite fast, and allow the VM to provide uninterrupted service while its prior physical host is, for example, taken down for physical maintenance.

Failover: Similar to the migration mechanism described above, failover allows the VM to continue operations if the host fails. Generally it occurs if the migration has stopped working. However, in this case, the VM continues operation from the last-known coherent state, rather than the current state, based on whatever materials the backup server was last provided with.

Within the VM, it is possible to run different OS depending on the necessities or requirements of the application running on top of it. VMs running proprietary operating systems require licensing, regardless of the host machine's operating system or hypervisor. For example, installing Microsoft Windows into a VM guest requires its licensing requirements to be satisfied. The hypervisor can be with or without a license. The most used hypervisors by market share are VMware ESXi and Microsoft Hyper-V, both requiring a license [16].

3.2.2 Software Containers

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another [17]. Each container may look like real computers from the point of view of programs running on them. The container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Virtualization in the case of software containers is performed at OS level, in which the kernel of an OS allows the existence of multiple isolated user-spaces instead of just one. Containerization can be performed with or without hypervisor. The most common approach is the second case, where all the containers share a common OS and the hypervisor is substituted by a container engine that is installed on top of the server OS. The container engine is responsible for the automatization of applications deployment inside software containers, providing an additional layer of abstraction and automation of virtualization of applications in multiple operating systems.

Fig. 15 shows a high-level example of the structure of *n* software containers running in one host without the use of hypervisor. Note that another possible approach is that all containers share the libraries (i.e. BINS/LIBS), which can be beneficial in those cases where different containers run the same application.

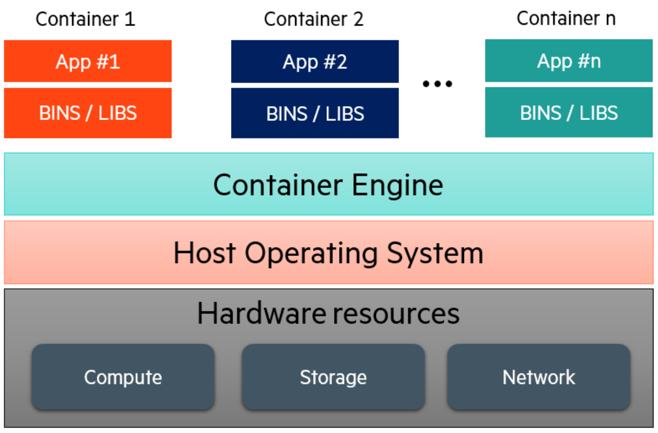


Fig. 15 High-level Container framework

The most used container engines are listed below:

Docker: it is a popular open-source project based on Linux containers. Docker is written in GO and developed by Dotcloud. Docker is basically a container engine which uses the Linux Kernel features like namespaces and control groups to create containers on top of an operating system and automates application deployment on the container.

rkt: container manager created by corers.

runC: container implementation based on the OpenContainers proposal.

lxc: original container implementation now handled by Linux Containers.

singularity: proposal of containers for scientific computing.

Currently, the most popular container engine is Docker due to its extensive integration with third-party services, such as Amazon Web Services, Microsoft Azure, Google Cloud Platform, and OpenStack among others. Fig. 16 shows the Docker architecture:

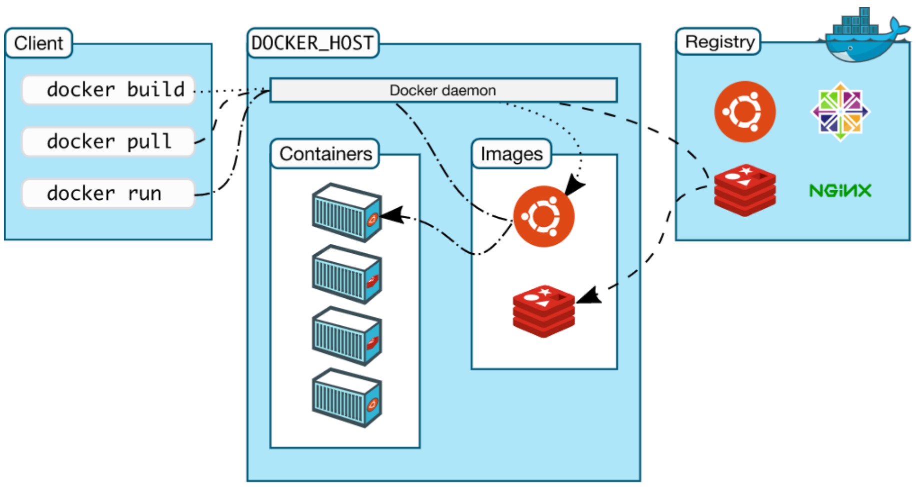


Fig. 16 Docker Architecture [18]

In order to create containers using Docker, the client uses the API to communicate with the server that performs the necessary operations to create the images to be run in each of the containers. The images can be stored in the public Registry of Docker. Then, the containers are executed from the constructed images in the Docker Host [18].

Comparing containers with VMs, both have similar resource isolation and allocation benefits, but they operate differently because containers virtualize the operating system instead of hardware. That makes containers more portable, lightweight and efficient in general. Note that it is possible to use both technologies together, even allocate containers inside VMs.

Table 3. VM and Containers comparison

Virtual Machines	Containers
Heavyweight	Lightweight
Limited performance	Native Performance
Each VM runs in its own OS	All containers share the host OS
Hardware-level virtualization	OS virtualization
Startup time in minutes	Startup time in milliseconds
Allocates required memory	Requires less memory space
Fully isolated and hence more secure	Process-level isolation, possibly less secure

3.2.3 Unikernels

A unikernel is a specialized, single address space machine image constructed by using library operating systems [19]. A developer selects, from a modular stack, the minimal set of libraries which correspond to the OS constructs required for their application to run. These libraries are then compiled with the application and configuration code to build sealed, fixed-purpose images (unikernels) which run directly on a hypervisor or hardware without an intervening OS such as Linux or Windows. Fig. 17 below shows a high level unikernel framework:

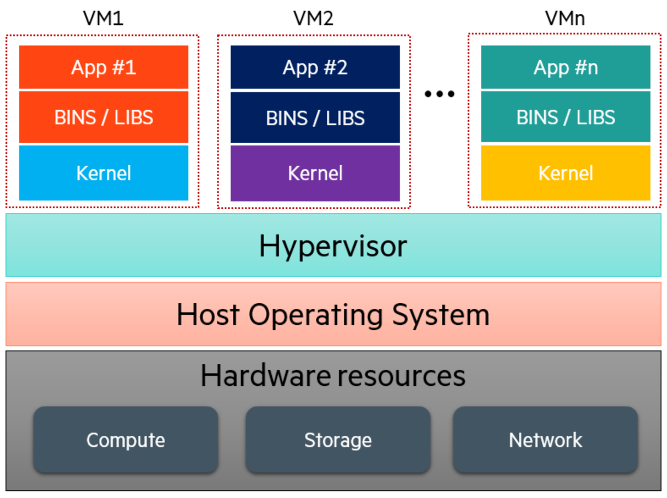


Fig. 17 High-level Unikernel framework

Unikernels as a third option primarily target the drawbacks of legacy VMs by compressing the kernel and shared libraries to the bare minimum while maintaining compatibility with the traditional cloud virtualization stacks (hypervisors, controllers, MANO). Differently from containers, which have a shared kernel, each unikernel has its own kernel. This allows performing better isolation than containers. Fig. 18 illustrates the main differences between VM, containers and unikernels.

Table 4 Virtualization technologies comparison [19]

Feature	Virtual Machines	Containers	Unikernels
Isolation	Strong	Weak	Strong
Size	Large	Small	Small
Instantiation	Slow	Fast	Fast
Overhead	High	Low	Medium
Toolset	Strong	Strong	Weak
Resource Overhead	High	Low	Medium

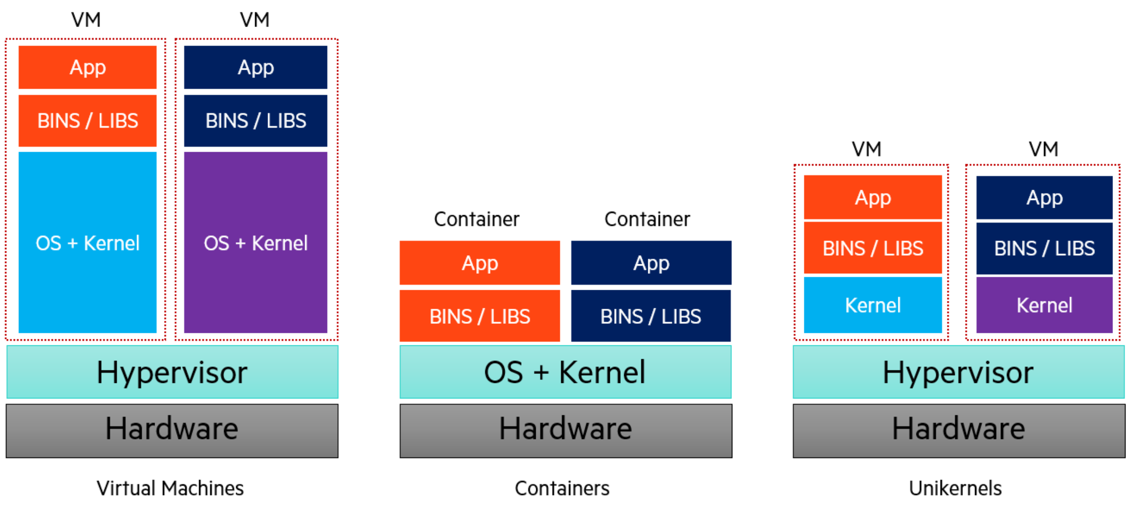


Fig. 18 High-level Virtualization strategies comparison

There are a number of new approaches to constructing unikernels [20]:

ClickOS: is a high-performance, virtualized software middle box platform based on open source virtualization. Early performance analysis shows that ClickOS VMs are small (5MB), boot quickly (as little as 20 milliseconds), add little delay (45 microseconds) and more than 100 can be concurrently run while saturating a 10Gb pipe on an inexpensive commodity server.

Clive: is an operating system designed to work in distributed and cloud computing environments, written in the Go programming language.

Drawbridge: is a research prototype of a new form of virtualization for application sandboxing. Drawbridge combines two core technologies: a picoprocess, which is a process-based isolation container with a minimal kernel API surface, and a library OS, which is a version of Windows enlightened to run efficiently within a picoprocess.

Graphene Library OS: is a Linux-compatible library operating system that focuses on securing multi-process, server-type or shell-type legacy applications. Graphene spans a multi-process application across multiple picoprocesses, with inter-process abstractions (e.g., signals, message queues, semaphores) coordinated over simple pipe-like streams. For applications with multiple security principles, Graphene can dynamically sandbox an insecure picoprocess.

The Haskell Lightweight Virtual Machine (HaLVM): is a port of the Glasgow Haskell Compiler tool suite that enables developers to write high-level, lightweight VMs that can run directly on the Xen hypervisor.

HermitCore: is a novel unikernel targeting a scalable and predictable runtime behavior for HPC and cloud environments. HermitCore supports C, C++, FORTRAN, and Go, Pthreads, OpenMP and iRCCE as message passing library. It is a research project that extends the multi-kernel approach and combines it with unikernel features. HermitCore can directly run on the KVM hypervisor but also natively on x86_64 architecture.

IncludeOS: is a minimal, service oriented, open source, includable library operating system for cloud services. It's designed for running C++ code on virtual hardware.

LING: is a unikernel based on the Erlang/OTP and understands .beam files. Developers can create code in Erlang and deploy it as LING unikernels. LING removes the majority of vector files, uses only three external libraries and no OpenSSL.

MirageOS: is a clean-slate library operating system that constructs unikernels for secure, high-performance network applications across a variety of cloud computing and mobile platforms. There are now more than 100 MirageOS libraries and a growing number of compatible libraries within the wider OCaml ecosystem.

OSv: is a new OS designed specifically for cloud VMs from Cloudius Systems. Able to boot in less than a second, OSv is designed from the ground up to execute a single application on top of any hypervisor, resulting in superior performance, speed and effortless management. OSv can run unmodified Linux executables (with some limitations), and support for C, C++, JVM, Ruby and Node.js application stacks is available.

Rumprun: is a software stack which enables running existing unmodified POSIX software as a unikernel. Rumprun supports multiple platforms, from bare ARM hardware to hypervisors such as Xen. It is based on the NetBSD rump kernel which provide free, portable, componentized, kernel quality drivers such as file systems, POSIX system call handlers, PCI device drivers, a SCSI protocol stack, virtio and a TCP/IP stack.

Runtime.js: is an open-source library operating system for the cloud that runs on JavaScript VM, could be bundled up with an application and deployed as a lightweight and immutable VM image. Runtime.js is built on the V8 Javascript engine and currently supports QEMU/KVM hypervisor.

Nanos: is a unikernel that runs on the QEMU/KVM hypervisor. It can load arbitrary ELF binaries and is implemented by the builder/orchestrator "OPS".

4 Network Function Virtualization (NFV) technologies

Whereas virtualization can be applied to several fields, Network Function Virtualization (NFV) provides the framework that allows the deployment and management of the different virtualized elements of telecom networks, and more specifically those defined for the 5G architecture. This chapter presents the background and basics of NFV, describes the ETSI architectural framework and identifies the different Network Functions Virtualization Management and Orchestration (NFV-MANO) solutions currently available in the market.

4.1 NFV Background

The hardware, physical, and network elements have always been difficult or very expensive to change or update [21]. Patching, installing new firmware or updating the software on these network elements can take months if it has to be done on all the elements of the network. As a result of this inconvenience, Over-The-Top (OTT) operators, that is, operators without their own infrastructure that exploit the infrastructure of other operators have been gaining market share because they will only use the network and do not inherit their hardware problems. This situation allows OTT operators to deploy a software-based internet platform that can be implemented almost immediately and run on existing hardware.

The concept of Network Function Virtualization (NFV) first appeared in November 2012 as part of the ETSI ISG with the resolution to reduce the CAPEX and OPEX related to the network hardware and with the promise of making networks more flexible and faster to update and change [22]. As more developments have been made with NFV, the agility of the service has become one of the main drivers for the development of virtualization of network functions.

4.2 NFV Basics

The virtualization of network elements, NFV, is a concept that virtualizes the main elements of a network. In this sense, instead of having a dedicated hardware element to provide a function of the network, software running on general hardware is used. In this way, entire classes of network node functions can be set up as building blocks that can be connected to create overall telecommunications networks.

NFV uses traditional server virtualization tools, but extends the concept significantly. In this sense, one or more VMs hosted on standard servers, running different software and providing different processes, are able to provide the network functions of a switch or router, for example, instead of having custom hardware devices to each function.

Examples of the virtualized functions that can be provided include: virtualized load balancers, firewalls, intrusion detection devices, WAN accelerators, routers, access control and billing.

NFV envisages the implementation of NFs as software-only entities that run over the NFV Infrastructure (NFVI) [23]. Fig. 19 illustrates the high-level framework. As such, three main working domains are identified in NFV:

Virtualized network functions, VNF: The virtualized network functions comprise the software used to create the various network functions in their virtualized format. These are then deployed onto the hardware, i.e. the Network Function Virtualization Infrastructure.

Network function virtualization infrastructure, NFVI: The NFVI consists of all the hardware and software components which are contained within the environment in which VNFs are deployed. One of the advantages of NFV is that the NFV-Infrastructure, NFVI can be located across several physical locations, allowing operators to typically place their centers at the most convenient locations. The network providing connectivity between these locations is part of the NFV-Infrastructure.

Network functions virtualization management and orchestration (NFV-MANO) Architectural Framework: NFV- MANO consists of the various functional blocks in whatever form that enables the exchange of information, manipulation and storage needed to manage and run the NFVI and VNFs.

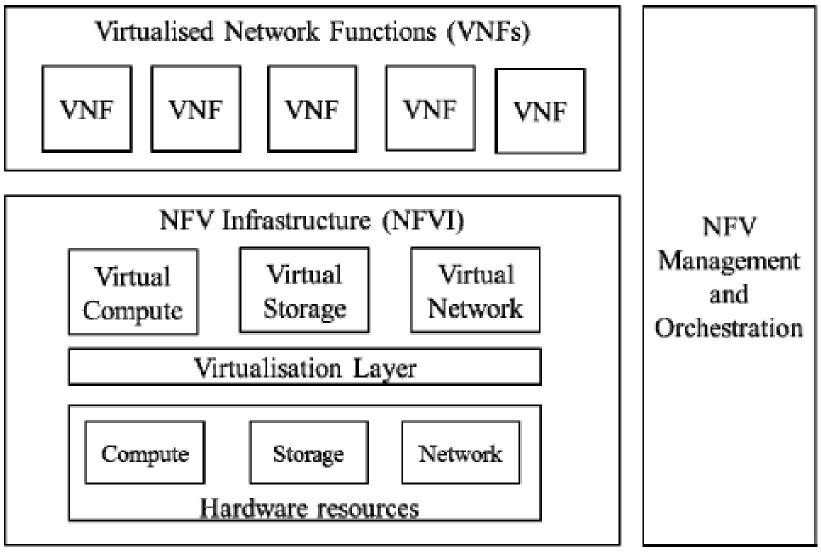


Fig. 19. High level NFV framework [23].

4.3 NFV Architectural Framework

The NFV architectural framework establishes the relations between the three working domains of Fig. 19 and identifies functional blocks and the main reference points between such blocks. Some of these are already present in current deployments, whilst others might be necessary additions in order to support the virtualization process and consequent operation. Fig. 20 shows the detailed NFV architectural framework functional blocks and reference points

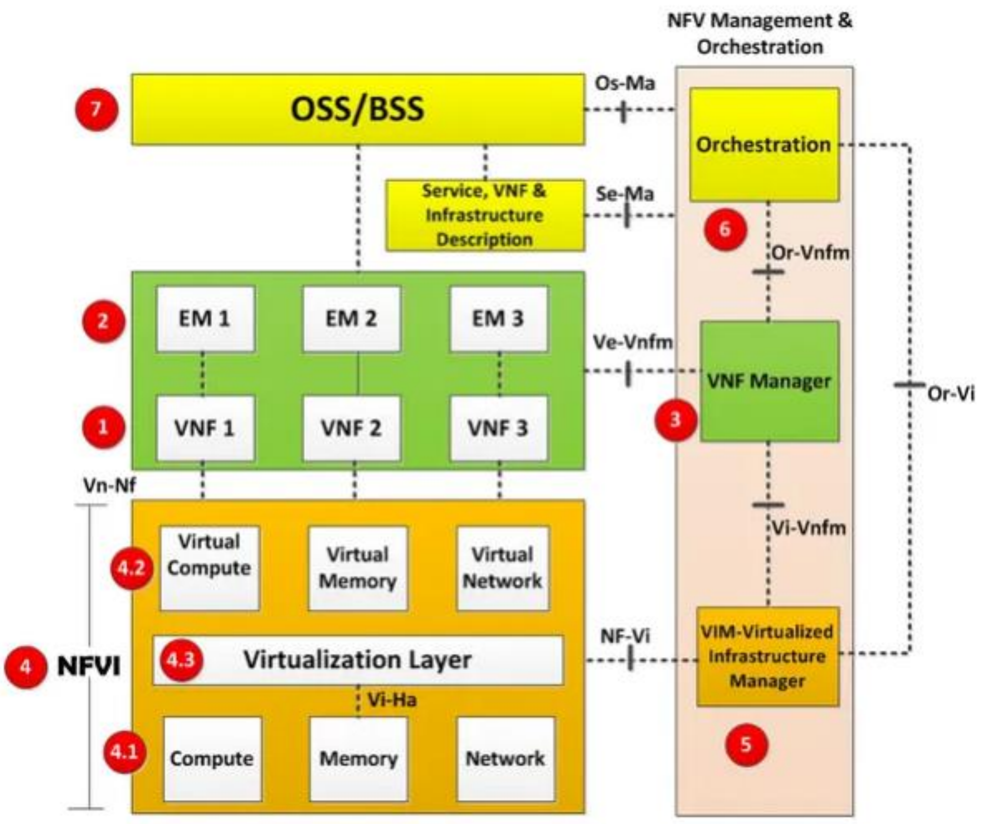


Fig. 20 NFV Architectural functional blocks [24]

In the following, a description of the different blocks numbered in the Fig. 20 is included [23]:

1. **VNF (Virtual Network Function):** A VNF is the virtualization of a network function in a legacy non-virtualized network. Examples of Network Functions (NFs) that can be virtualized are routers, base stations, gateways, firewalls etc. Note that the sub functions that compose a NF are also consider VNFs. For instance, various sub-functions of the router can be separated VNFs which together function as virtual router.
2. **Element Management (EM):** The Element Management is responsible for the functional management of VNFs. An example of management function is the Fault, Configuration, Accounting, Performance and Security Management (FCAPS). An EM can manage one or multiple VNFs depending on the NF. In fact, the EM itself can be considered a VNF.
3. **VNF Manager**: VNF Manager (VNFM) is the responsible of lifecycle management of VNFs, which consist in setting up/ maintaining and tearing down VNFs. The difference between EM and VNFM is that the EM manages the specific functionality of the VNF while the VNFM does the management for the virtual components. For instance, in the case of virtualizing a mobile core, the EM would manage issues related to mobile signaling, while VNFM would bring up a VNF itself.

4. **NFVI (Network Function Virtualization Infrastructure):** NFVI is the environment in which VNFs are deployed, managed and executed. This includes physical resources, virtual resources and virtualization layer, described below.
    1. **Compute, Memory and Networking Resources:** This is the physical part (i.e., hardware) in NFVI. Virtual resources are instantiated on these physical resources. This category includes any commodity switch or physical compute/storage hardware.
    2. **Virtual Resources:** Consists in the compute, memory and networking physical virtualized resources. This virtualization is performed by the virtualization layer.
    3. **Virtualization Layer:** This layer is responsible for abstracting physical resources into virtual resources. This abstraction is commonly performed by a hypervisor.
5. **VIM (Virtualized Infrastructure Manager):** This is the management system for NFVI. It is responsible of controlling and managing the interaction of a VNF with the underlying computing, storage and networking resources and their virtualization. It performs resource management tasks such as inventory of hypervisors, allocation of VMs onto hypervisors or increasing resources to VMs. Moreover, it performs operations for visibility of the NFVI, analysis of performance data of the infrastructure and collection of information for capacity planning, monitoring and optimization.
6. **NFV Orchestrator:** Generates, maintains and tears down network services (NSs). Hence, in order to deploy a given NS, the orchestrator communicates with the VNFM and VIM to enable the required VNFs and NFVI resources, respectively, required by this NS.
7. **OSS/BSS:** includes collection of applications that a service provider uses to operate its business. While OSS deals with network management, fault management, configuration management and service management, BSS deals with customer management, product management and order management.

Moreover, Table 5 includes the definitions of the main reference points in Fig. 20 [23], which are the connections between the different functional blocks.

Table 5 Reference Points

Reference Point	Boundaries	Use Defined in the framework
Vi-Ha	Virtualization Layer <> Hardware	Interfaces the virtualization layer to hardware resources to create an execution environment for VNFs, and collect relevant hardware resource state information for managing VNFs without being dependent on any hardware platform.
Vn-Nf	NFVI <> VNF	This is the execution environment provided by the NFVI to the VNF. It does not assume any specific control protocol. It is in the scope of NFV in order to guarantee hardware independent

		lifecycle, performance and portability requirements of the VNF.
Or-Vnfm	NFVO <> VNFM	Resource related request, e.g. authorization, validation, reservation, allocation by the VNFM. Sending configuration information to the VNFM, so that the VNF can be configured appropriately to function within the VNF Forwarding Graph in the network service. Collecting state information of the VNF necessary for network service lifecycle management.
Vi-Vnfm	VIM <> VNFM	Resource allocation requests by the VNFM. Virtualized hardware resource configuration and state information (e.g. events) exchange.
Or-Vi	NFVO <> VIM	Resource reservation and/or allocation request by the Orchestrator. Virtualized hardware resource configuration and state information (e.g. events) exchange.
Nf-Vi	NFVI <> VIM	Specific assignment of virtualized resource in response to resource allocation request. Forwarding of virtualized resource state information. Hardware resource configuration and state information (e.g. events) exchange.
Os-Ma	OSS/BSS <> NFV-MANO	Request for network service lifecycle management. Request for VNF lifecycle management. Forwarding of NFV related state information. Policy management exchanges. Data analytics exchanges. Forwarding of NFV related accounting and usage records. NFVI capacity and inventory information exchange.
Ve-Vnfm	EM <> VNFM	Request for VNF lifecycle management. Exchanging configuration information. Exchanging state information necessary for network service lifecycle management.
Se-Ma	Service, VNF and Infrastructure Description <>	Used for retrieving information regarding the VNF deployment template, VNF Forwarding Graph, service-related

	NFV-MANO	information, and NFVI information models. The information provided is used by NFV-MANO.

4.3.1 End-to-End Flow in the ETSI NFV Framework

Below is shown how the above-mentioned framework works end to end, taking the example of a simple network service and examining how the functional blocks defined in the ETSI framework collectively interact to implement the service. Fig. 21 and the accompanying list depict a simplified flow as an example to help understand the framework.

- **Step 1:** The full view of the current state of the end-to-end topology is visible to the NFVO, including the already instanced VNFs and the state of the NFVI.
- **Step 2:** The NFVO instantiates the required VNFs and communicate this to the VNFM.
- **Step 3:** VNFM determines the number of VMs needed as well as the resources that each of these will need and reverts back to NFVO with these requirements to be able to fulfill the VNF creation.
- **Step 4:** NFVO validates if there are enough resources available for the VMs to be created. The NFVO now needs to initiate a request to have these VMs created.
- **Step 5:** NFVO sends request to VIM to create the VMs and allocate the necessary resources to those VMs.
- **Step 6:** VIM ask the virtualization layer to create these VMs.
- **Step 7:** Once the VMs are successfully created, VIM acknowledge this back to NFVO.
- **Step 8:** NFVO notifies VNFM that the VMs it needs are available to bring up the VNFs.
- **Step 9:** VNFM now configures the VNFs with specific parameters.
- **Step 10:** Upon successful configuration of the VNFs, VNFM communicates to NFVO that the VNFs are ready, configured, and available to use.

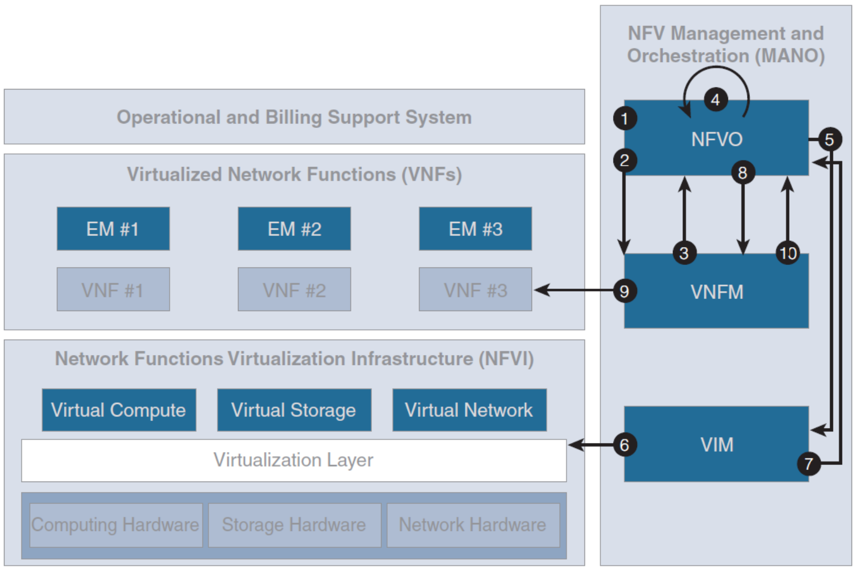


Fig. 21 End-to-End Flow in the ETSI NFV Framework [25]

The framework does not restrict the implementation to a single VNFM to manage all the VNFs. It is possible that the vendor that owns the VNF requires its own VNFM to manage that VNF. Therefore, there can be NFV deployments where multiple VNFMs are managing multiple VNFs or a single VNFM manages a single VNF. An example deployment with multiple VNFMs is shown in Fig. 22, where VNFM of "Vendor A" manages multiple VNFs and VNFM of "Vendor B" only manages VNF #1.

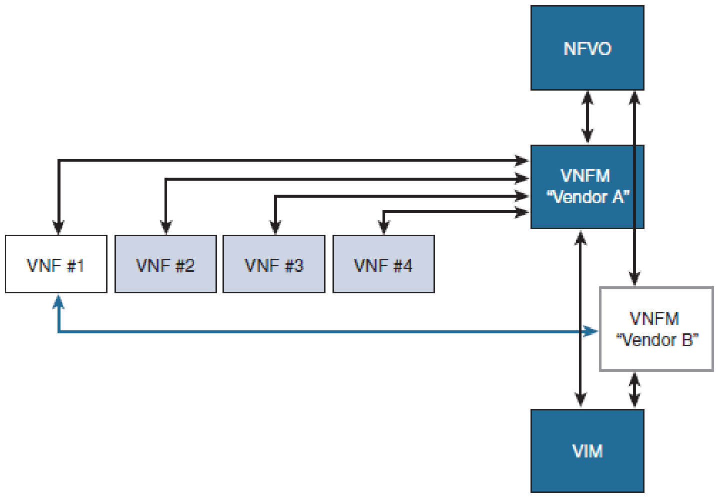


Fig. 22 Multiple VNFMs Managing Separate VNFs [25]

4.4 NFV Advantages

NFV offers several advantages to Communication Service Providers (CSPs) as compared to the existing hardware-based network functions. The five main advantages of NFV are [13]:

- NFV enables the efficient use of Information Communication Technologies (ICTs) infrastructure through softwarization of Network Functions (NFs). Hence, it brings more flexibility and agility and eases the management of the telecommunications systems.
- NFV offers flexibility to CSPs to create, deploy and manage network services since VNFs can be deployed in general purpose hardware. This avoids depending on vendor-specific hardware devices with special configuration needs.
- CSPs appreciate NFV because network functions are virtualized in a way that they can be chained together to create and deploy network services on the fly. Thus, it enables both dynamic scaling and service provisioning.
- NFV offers flexibility to adapt rapidly to technological innovation and provides a better return on investment for CSPs than the case of hardware-based appliances. As product life-cycles are becoming shorter, this can often become critical to support new network services.
- NFV enables infrastructure to be deployed as a Telco cloud, rather than as appliances. The transition from network appliances being statically configured to cloud infrastructure hosting containerized network functions is a major change for operators. That change is enabled by NFV. NFV has a major impact on how hardware for 5G is being deployed. While 5G is a cloud, 4G, 3G, and 2G were all deployed as fixed hardware appliances.

4.5 NFV-MANO Solutions

The Network Functions Virtualization Management and Orchestration (NFV-MANO) architectural framework has the role to manage the NFVI and orchestrate the allocation of resources needed by the Network Services (NSs) and VNFs [26]. Such coordination is necessary now because of the decoupling of the Network Functions software from the NFVI.

The development of platforms for NFV-MANO faces different challenges [26]:

Convergence: Management and standardization of end-to-end services by OSS/BSS requires convergence on a common approach to present management services and management information from both legacy network systems and NFV based systems, so that accurate end-to-end management views can be derived.

Automation: Operators need to speed up the ability to dynamically incorporate changes. The agile management of network functions, applications, and services provides new opportunities for revenue while reducing costs. The key challenge in this area is to provide a flexible and dynamic policy-based management approach.

Real time processing: The need for real-time processing (in conjunction with offline processing) is an additional key challenge that plays an increasingly important role. It is desirable to support real-time processing by automation.

Data analytics: The support of the processing of a huge amount of data, including data analytics, based on several data sources (e.g. structured, semi-structured, and unstructured data from the infrastructure as well as other sources), also in real time.

Currently, several NFV related projects such as OSM [27], ONAP [28], and HPE NFVD [29], are expected to speed up NFV deployments, mainly focusing on different aspects of NFV-MANO framework.

In the following, some proposals of the different entities leading the development of NFV-MANO tools are presented.

4.5.1 Open Source MANO (OSM)

The Open Source MANO project was officially launched at the Mobile World Congress (MWC) in 2016. Starting with several founding members, including Mirantis, Telefónica, BT, Canonical, Intel, RIFT.io, Telekom Austria Group and Telenor, the OSM community now includes 55 different organizations. The OSM project is hosted at ETSI facilities and targets delivering an open source MANO stack closely aligned with the ETSI NFV reference architecture. While ETSI has always focused on standards, the OSM initiative is based on open-source tools like GitHub. Hosted by ETSI, the initiative is also based on components that have been around for some time like Telefonica's OpenMANO project. In [30] a list of 5G projects and research activities that are using or contributing to OSM is included.

Fig. 23 includes the OSM protocol and its relation with the ETSI NFV framework. It consists of two major components: design-time scope and run-time scope.

The **design-time scope**, colored in green in Fig. 23, allows the definition of the NS to be deployed by means of a model-driven environment fully aligned with the ETSI NFV standard. It also includes functions for the creation, update and deletion of service definitions. Finally, it supplies a Graphical User Interface (GUI) to accelerate service design time and includes a VNF package generation.

The **run-time scope** component, colored in purple in Fig. 23, comprises of four main blocks:

a) **Network Service Orchestrator (NSO):** Responsible for end-to-end NS orchestration and provisioning. The NSO stores the VNF definitions and NS catalogs, manages workflow of the service deployment and can query the status of already deployed services.
b) **Resource Orchestrator (RO):** It is used to provision services over a particular Infrastructure as a Service (IaaS) provider in a given location. It includes VIM/SDN Connectors. The NSO and RO components can be jointly mapped to the NFVO entity in the ETSI MANO architecture.
c) **VNF Configuration and Abstraction (VCA):** This module performs the initial VNF configuration. Considering this purpose, the VCA module can be considered as a generic VNFM with a limited feature set.
d) **Open VIM:** Open VIM is a light implementation of an NFV VIM. OSM can also be operated with Openstack and VMware VIM.

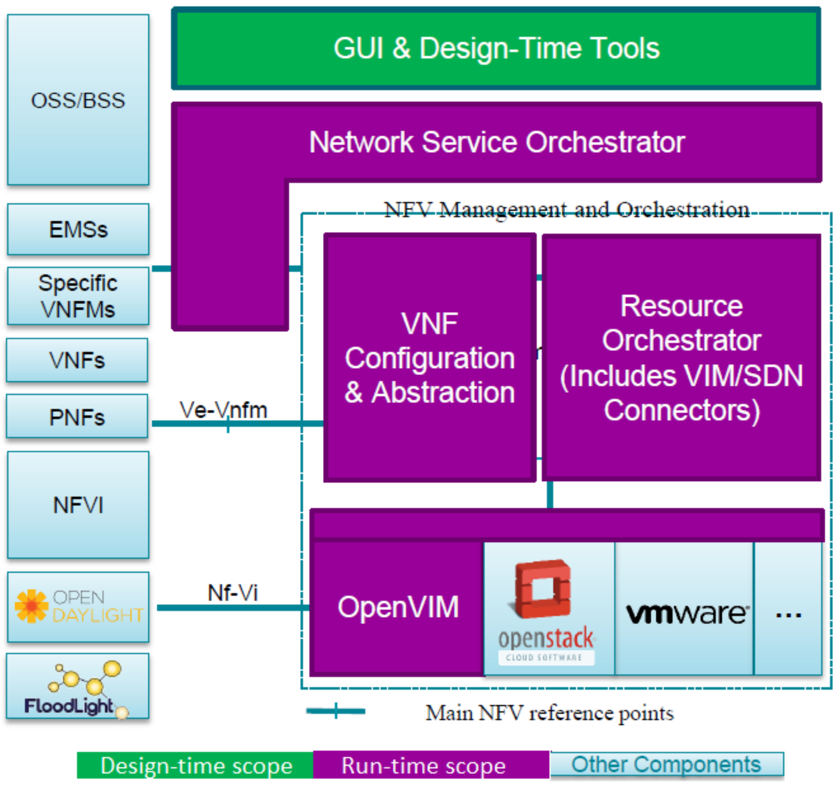

GUI & Design-Time Tools
OSS/BSS
Network Service Orchestrator
NFV Management and Orchestration
EMSs
Specific VNFMs
VNFs
PNFs
NFVI
VNF Configuration & Abstraction
Resource Orchestrator (Includes VIM/SDN Connectors)
Ve-Vnfm
Nf-Vi
OpenVIM
openstack
vmware
...
OPEN DAYLIGHT
FloodLight
Main NFV reference points
Design-time scope
Run-time scope
Other Components

Fig. 23 OSM Project architecture [27]

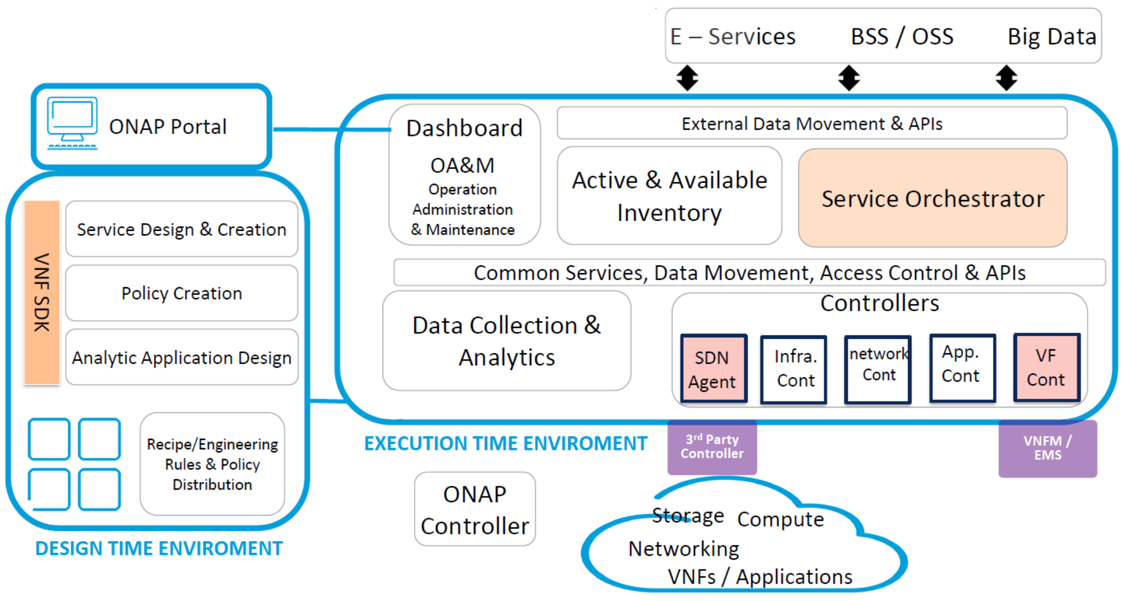
4.5.2 Open Network Automation Platform (ONAP)

ONAP is an open-source software platform that enables the design, creation, and orchestration of services on an infrastructure layer on top of individual VNFs. ONAP was formed as a combination of the Linux Foundation's OPEN-Orchestrator Project (OPEN-O), and the AT&T ECOMP (Enhanced Control, Orchestration, Management and Policy) platform to form one unique code [28]. ONAP architecture consists of two major environments, the design-time environment and the execution-time environment with numerous separate subsystems operating within them, as illustrated in Fig. 24.

The design-time environment is used as a development environment with all the functions and libraries needed for the development of new capabilities; the environment can be used to upgrade of existing capabilities through portal, role-based interfaces, and command-line. Within the design-time environment framework, the Service Design and Creation (SDC) can be used as environment to define service and resource, constraints, instantiation and modifications recipes, while the policy subsystem is used to associate anomalous and actionable conditions with automated remedy actions. Moreover, a Software Development Kit (SDK) to onboard and certify Vendors VNFs is provided.

The execution-time environment framework is used to execute all policies and rules prepared in the design-time environment. This environment is composed of different functionalities:

a) **Active & Available Inventory (A&AI):** This function allows visualizing a real-time topology map of the virtual networks, services and applications that are running. In addition, network resources are available in a database record.
b) **Service Orchestration (SO):** It handles the orchestration and management of the delivery, modification or removal of networks and services as well as it provides cross domain orchestration and coordination.
c) **Data Collection, Analytics & Events (DCAE):** Through this function, telemetry data from VNFs and other sources are collected and analyzed to detect anomalous conditions.
d) **Controllers:** Diverse controllers are included to manage and configure VNFs, services and applications of VFs and physical resources of the infrastructure.



E – Services
BSS / OSS
Big Data
ONAP Portal
Dashboard
OA&M
Operation Administration & Maintenance
External Data Movement & APIs
Active & Available Inventory
Service Orchestrator
VNF SDK
Service Design & Creation
Policy Creation
Analytic Application Design
Common Services, Data Movement, Access Control & APIs
Data Collection & Analytics
Controllers
SDN Agent
Infra. Cont
network Cont
App. Cont
VF Cont
Recipe/Engineering Rules & Policy Distribution
EXECUTION TIME ENVIROMENT
3rd Party Controller
VNFM / EMS
ONAP Controller
Storage
Compute
Networking
VNFs / Applications
DESIGN TIME ENVIROMENT

Fig. 24 ONAP Platform Architecture [31]

4.5.3 HPE Network Function Virtualization Director (HPE NFVD)

NFV Director (NFVD) is a proprietary software tool developed by HPE. NFVD provides consistent management and behavior of multivendor VNFs to efficiently run on heterogeneous COTS hardware platforms and virtualization environments. It takes responsibility for automatically managing the end-to-end service across VNF, VNF forwarding graphs (VNF-FGs) and network services (NSs).

HPE NFVD is designed to meet the evolving ETSI specifications for the NFV orchestrator functionality. This includes the orchestration and management of VNFs and NSs, providing the global resource management, and consistently applying global, cross-VNF, and VNF-specific policies. Within the ETSI model, VNF managers are responsible for the VNFs actions—deciding to scale in or out, for example. HPE NFV Director can work with external VNFMs, when supplied by the vendors. It also can provide the VNFM functionality through its embedded functions, which compensate for completely or partially missing VNFM functionality in the vendor solutions or in virtual network functions created from basics by carriers. Fig. 25 shows a representation of the NFV Architectural Functional Blocks run by NFV Director in relation with the ETSI NFV MANO Model:

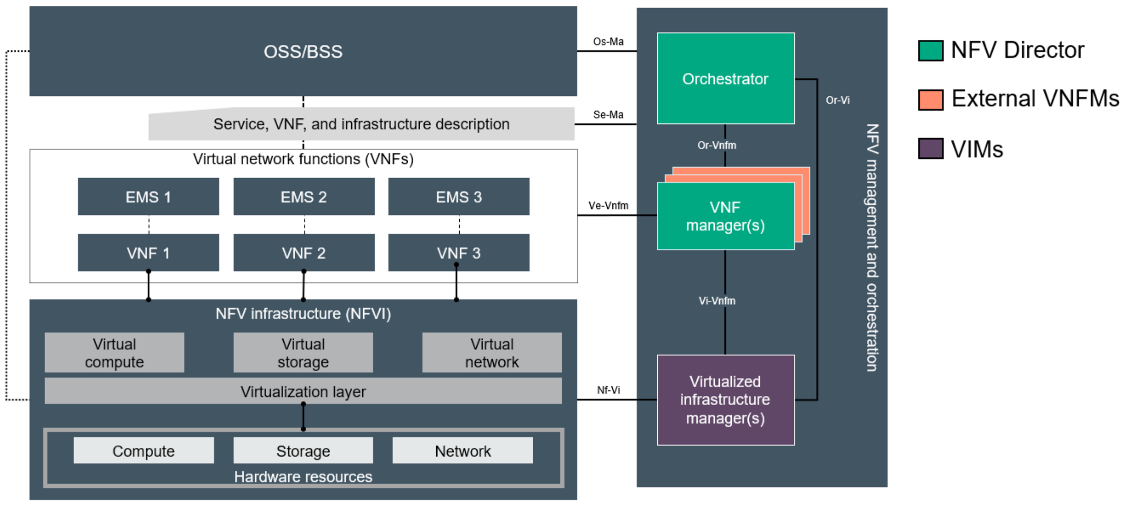


Fig. 25. NFVD as NFVO + VNFM

Different users can interact with the HPE NFVD platform at diverse levels depending on their role. HPE NFVD is designed to allow multitenancy and multi-VIM support so there are two main user categories defined: “Domain user” and “Tenant users”. The “Domain user” is the NFVD platform administrator (i.e. a CSP), and the “Tenant users” are the clients of the “Domain user” (i.e. any company that consume resources from a CSP). NFVD platform establishes a user level hierarchy in order to organize the access to resources and permissions:

Domain category: The Domain user is the platform administrator and has permission for creating any type of user, trigger discovery and onboard new datacenters, see resources and consumptions, see discovered elements and navigate through all Datacenters.

Tenant category user’s roles: Tenant category is divided in three user’s roles: Organization, Virtual Data Center (VDC), and VNF Group.

1 **Organization users:** represent a real organization and have rights to:
   a. Create a new VDC.
   b. Assign resources (catalogue items, VDC, VNFs).
   c. Model VDC with VDC assigned resources.

2 **VDC users**: correspond to users belonging to the organization responsible of the VDC management. They have right to:
   a. Manage VNFs users.
   b. Deploy elements of the catalogue in VDC in any VNF Group.

   c. Design VNFs.
   d. Create virtual links that connects elements in the VDC.
   e. Upload images and distribute the rights among VNF Group users.

3 **VNF Group users:** represent a group of people in charge of deploying/operating a well-defined set of VNF templates / descriptors. They have rights to:
   a. See all the virtual links of the VDC.
   b. Deploy/Undeployed VNFs from templates.

Fig. 26 shows a diagram of the user level hierarchy of NFV Director:

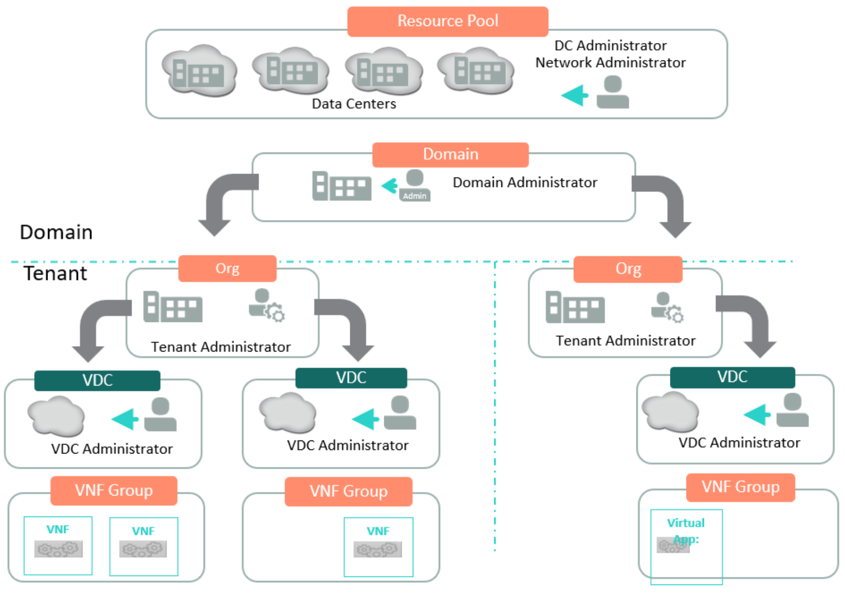


Fig. 26. User level hierarchy of NFV Director

Regarding the HPE NFVD framework, Fig. 27 illustrates its functional blocks. Among all components, the following four are of special relevance: model-based coordination and control, agentless monitoring, dynamic topology-based correlation, and rules-based autonomous actions.

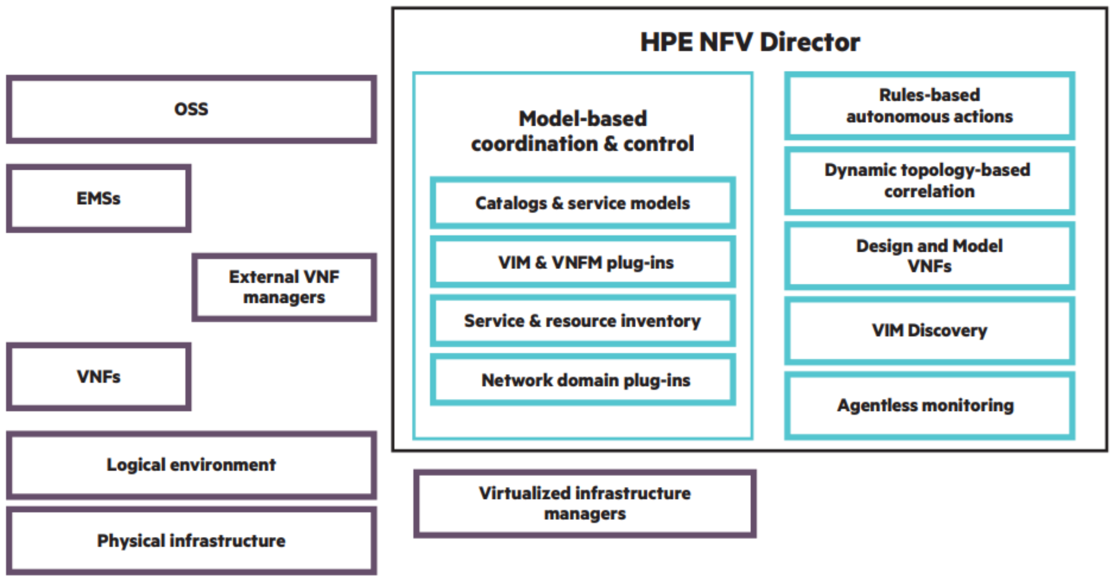
a) **Model-based coordination and control:** The core component of HPE NFV Director is the model-based coordination and control function. Its role is to maintain the data models and global resource inventory while orchestrating the required changes to manage VNFs. The model consists of definitions, templates, and instances:
   - *Definitions*: they provide the constructs that are known by NFV Director, defining what is allowed through abstract concepts (like generic “VNF” or “policy”).
   - *Templates*: they provide the catalog of the VNFs and its components, describing the catalog items to orchestrate. By just filling in the request-specific information, templates allow for a creation of new instances.
   - *Instances*: NFV Director uses a set of workflows to manage the process instantiating or modifying changes in the model. Any required changes are done in a transactional manner, ensuring that things are always left in a known state in case of errors.

This component, allows the operator to assign quotas for resource usage in terms of CPU, memory, disk and networking resources to be provided to the tenants, which are the clients of the operator. The quotas can be allocated for virtual compute and network resources, templates and catalogues. NFV Director ensures that the quotas allocated are not violated during resource allocation for VNF deployment. To ensure that, NFV Director performs periodic resource discovery and reconciles the discovered resources with the global resource inventory and recalculates the quota availability. This quota mechanism ensures that the resources are managed efficiently and made available justly to the different tenants that compete for the resources.

b) **Agent-less monitoring:** It can monitor a wide variety of points, issuing events, or executing commands when predefined thresholds are crossed. Since different VNFs have different monitoring needs, the monitoring points and thresholds are automatically configured by HPE NFV Director when the VNF is provisioned or modified. As an agentless solution, HPE NFV Director does not require the installation of monitoring agents on the target systems. The individual VNF managers are responsible for monitoring their own internal faults and performance. When VNFM functionality is provided out of HPE NFV Director (through the embedded VNFM), it may be necessary to collect application-specific monitoring information. This is done using application-specific monitors delivered as a part of a VNF-specific plug-in.

c) **Dynamic topology-based correlation:** This component provides the two levels of correlations for which a NFVO is responsible. The NFVO must be able to correlate data from all the layers involved to understand the root-cause analysis in case of a problem. The correlation is performed with data from top (service or function related) and from bottom (infrastructure related) layers. For the top layer, the correlation function will take information from the VNFM about the status of each VNF involved in a VNF forwarding graph or a NS and correlate this into a view about the NS status. For the bottom layer, NFVD correlates information from the physical infrastructure into the logical view of the virtual infrastructure used by each VNF or tenant.

d) **Rules-based autonomous actions:** An important role of a NFVO is to consistently provide common policies across multiple VNFs and different sites. Through a defined and extensible set of rules, the policies are implemented through a set of actions. These actions can be internal to HPE NFV Director or directed at the VNFs, the infrastructure, or an external system. Actions can be configured to execute in open loop (request operator confirmation before executing) mode or closed loop (automatically execute) mode.



HPE NFV Director
OSS
EMSs
External VNF managers
VNFs
Logical environment
Physical infrastructure
Model-based coordination & control
Catalogs & service models
VIM & VNFM plug-ins
Service & resource inventory
Network domain plug-ins
Rules-based autonomous actions
Dynamic topology-based correlation
Design and Model VNFs
VIM Discovery
Agentless monitoring
Virtualized infrastructure managers

Fig. 27 HPE NFVD components

Other components of lower relevance are described below:

a) **Design and Model VNFs:** To aid quicker onboarding of the VNFs, NFV Director provides a graphical tool to model and design VNF descriptors. NFV Director provides Component and VNF designer to model and design complex VNF templates from simple building block templates. This allows operators to have control over the VNF placement, monitoring, and resource management as per their specific needs.

b) **Discover VIM and maintain global resource inventory:** HPE NFV Director has a discovery component that discovers the available virtual and physical resources exposed by underlying VIMs. NFV Director uses VIM plug-ins that perform resource discovery by using VIM APIs. The resources discovered from multiple VIMs are maintained and reconciled in the resource inventory databases. Currently, NFV Director supports VMware vCenter and OpenStack VIMs.

c) **External VNF manager and VNFs provisioning:** When dealing with an external VNF manager, HPE NFV Director can support both of the interaction modes discussed by the ETSI:

   - HPE NFV Director provides global resource allocation while VNF managers interact directly with each VIM within the scope of the resources they have been allocated.
   - HPE NFV Director interacts with the VIM on behalf of the VNF manager.

   The latter model ensures a simpler interaction among the VNF manager, the VIM, and the HPE NFV Director, delivering better consistency in the resource usage.

   Based on defined policies, HPE NFV Director can deploy a VNF in a single site or across multiple sites. These sites can come from different vendors and use different types of VIM, enabling different VNF components to be in diverse environments. For example,

- A virtual Customer-Premises equipment (CPE) network service could have some of its components deployed at the customer premises on a VMware vCenter VIM while other components may be deployed centrally in CSP's carrier data centers hosted on OpenStack VIM.
- A virtual Evolved Packet Core (EPC) service where VNFs (such as Home Subscriber Service (HSS)) could be deployed in redundant configuration in 2 different CSP data centers hosted on OpenStack VIM.

Just creating the VNF components is often not sufficient to have a working service. The network connectivity between different VNFs and even between different components within the same VNF also needs to be provisioned. Whether the network is a traditional network or is based on SDN technology, HPE NFV Director can automatically provision and configure the end-to-end connectivity, ensuring that the complete network service is consistently and correctly set up.

5 Development of a scenario for orchestrating 5G network functions

Once studied and understood the different elements that come into play in the development and management of 5G networks and the technologies used for virtualization, we propose to develop a 5G core VNF deployment test by using the HPE NFVD framework, whose full licensed access has been provided temporally by HPE freely.

The goal of this section is to create a resource orchestration scenario based on the ETSI NFV architecture (MANO component). In order to be able to deploy functions of orchestration and administration of the life cycle of the virtual and physical resources of the 5G network, the scenario must contain three functional blocks: NFV Orchestrator (NFVO), VNF Manager (VNFM) and Virtualized Infrastructure Manager (VIM). The resources to orchestrate in the scenario correspond to the main functions of the 5G core network, that is, UPF, SMF and AMF.

This chapter firstly introduces the 5G core VNF deployment scenario. Then, the HPE NFVD graphical interface is presented and the onboarding of a generic VNF is performed. After this, the presented scenario is deployed and evaluated.

5.1 NFV scenario definition

The considered scenario is composed of two tenants, named “Tenant1” and “Tenant2” that aim at providing 5G connectivity to their users. For this purpose, a Telecommunication Provider, who owns infrastructure, provides each of the tenants with a different network slice that satisfies the tenant requirements. In this sense, “Slice 1” is provided to “Tenant 1” and “Slice 2” is provided to “Tenant 2”.

The Telecommunication Provider is responsible of creating, orchestrating and managing each of the slices, as well as providing the required NFs. In this case scenario, the Telecommunication Provider provisions each of the slices with the three main NFs of the 5G core: UPF, SMF and AMF. The SMF and UPF VNFs are instantiated in a per-slice basis. In turn, the AMF VNF is shared by the two slices, meaning that only one AMF instance exists in the scenario and it is accessed by the two slices. By sharing the AMF, access and mobility data can be accessed by both tenants thus a more efficient deployment can be achieved.

Regarding the interfaces and connections, the different VNFs are connected through the control plane (dotted lines) and each UPF is also provided with an interface to the data network (DN) through the data plane.

Fig. 28 shows a high-level scheme of the considered scenario:

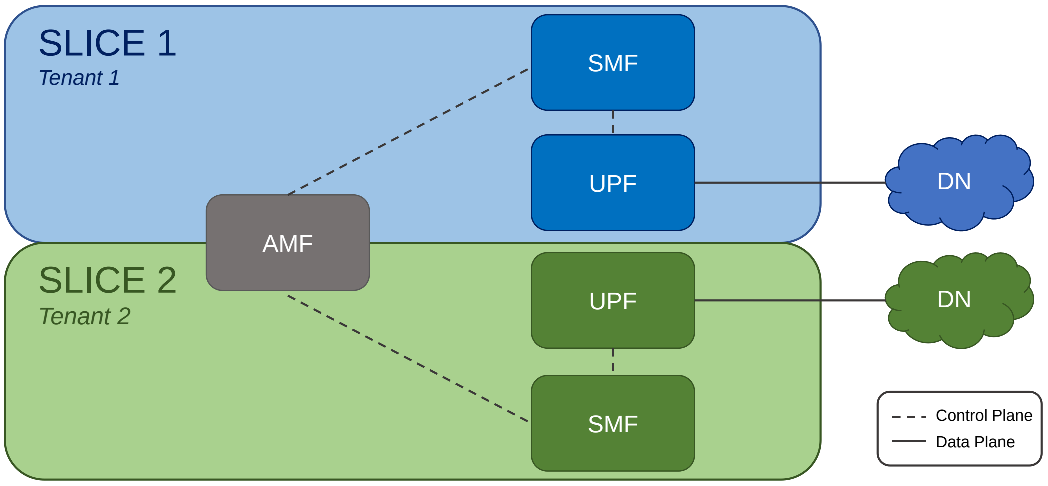


Fig. 28. Abstraction of the 5G core VNF deployment scenario concerning two slices

5.2 <u>Implementation with HPE NFVD</u>

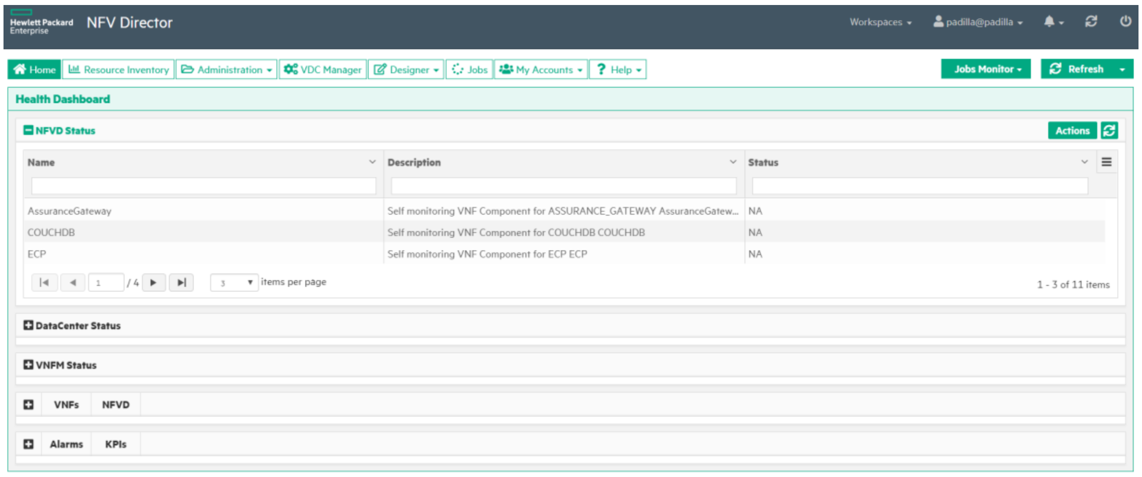
HPE has provided access to a full license version of the NFVD tool in order to develop the considered scenario. Although NFVD software can be installed over bare metal, over VMs on a vCenter (VMware management server) outside of the cloud and over Virtual Machines that reside in the cloud (even if that same cloud is managed by NFVD), the provided access is over bare metal.

As a previous step to the deployment of the considered scenario, this section presents the Graphical User Interface (GUI) of the NFVD and the onboarding of a generic VNF to provide an understanding of the tool.

5.2.1 HPE NFV Director Graphical User Interface (GUI)

HPE NFVD offers a complete graphical interface with different panels to interact with, which is composed of different dashboards. The most relevant ones are introduced in the following in order to review the tool and its functionalities.

Fig. 29 shows the first screen after login in. The first dashboard that appears is the *Home* view, which shows information about the status of different parameters, such as the status of NFVD, Datacenter, VNFM, VNFs.

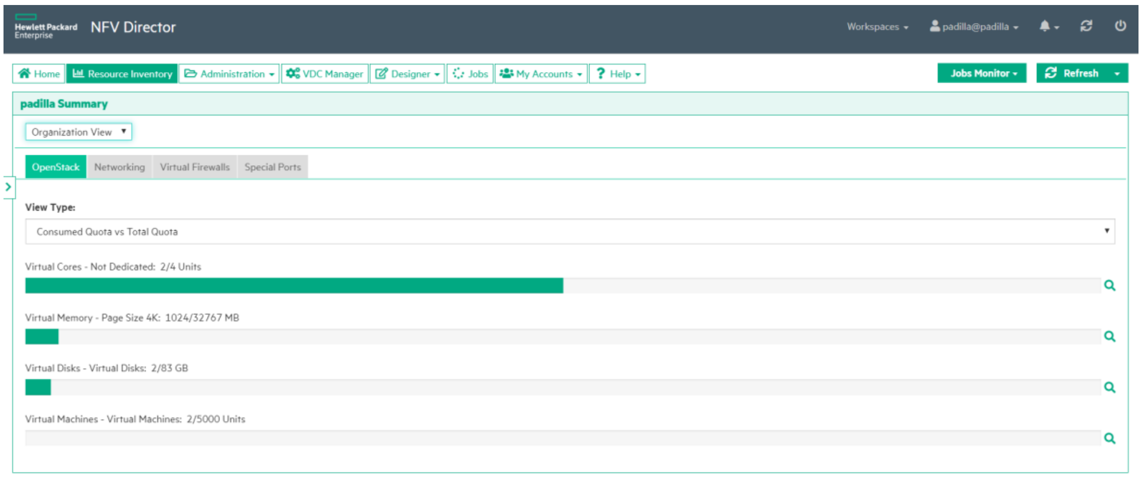


Name	Description	Status
AssuranceGateway	Self monitoring VNF Component for ASSURANCE_GATEWAY AssuranceGatew...	NA
COUCHDB	Self monitoring VNF Component for COUCHDB COUCHDB	NA
ECP	Self monitoring VNF Component for ECP ECP	NA

1 / 4 3 items per page 1 - 3 of 11 items

Fig. 29 NFV Director Home

The next dashboard is the *Resource Inventory*, where the allocated and consumed resources in the VIM (OpenStack in this case) from different points of views can be checked: Organization View, VDC View, and VNF Group View. Fig. 30 and Fig. 31 shows the *Resource Inventory* and the different elements that can be checked, respectively.



View Type: Consumed Quota vs Total Quota

Virtual Cores - Not Dedicated: 2/4 Units

Virtual Memory - Page Size 4K: 1024/32767 MB

Virtual Disks - Virtual Disks: 2/83 GB

Virtual Machines - Virtual Machines: 2/5000 Units

Fig. 30 NFV Director Resource Inventory

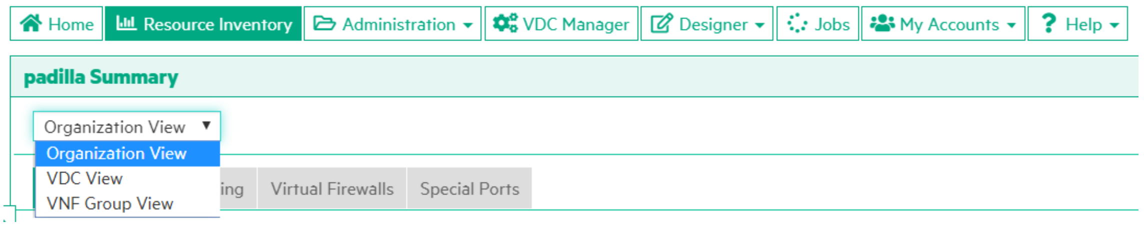
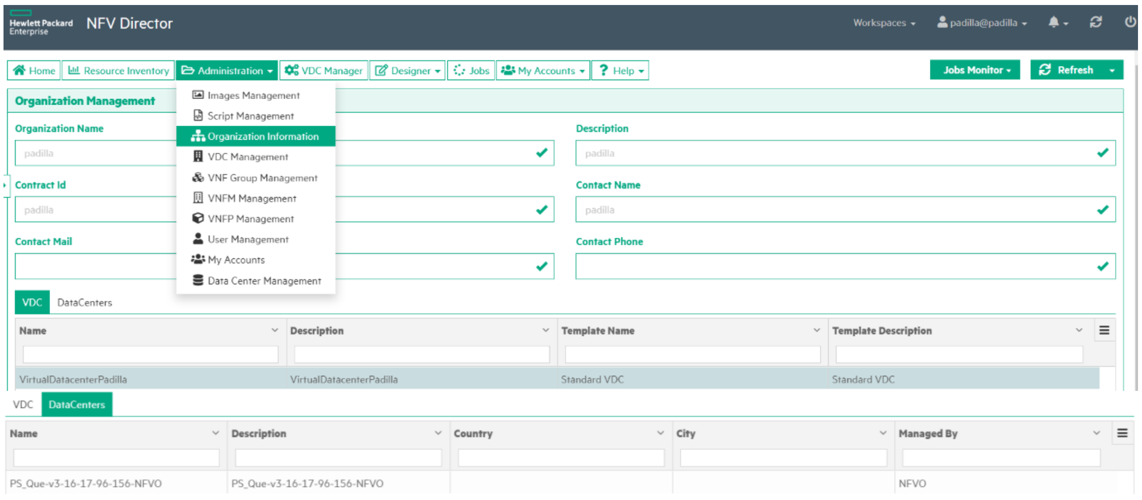


Fig. 31 Resource inventory views

From *Administration* dashboard, Fig. 32 illustrates the information of different elements to be managed. In the following, the three most relevant elements are further developed:

a) In the field *Organization Information*, it is possible to check the basic information of the organization, the VDCs and DCs assigned, as shown in Fig. 32.



Organization Management

Organization Name: padilla

Description: padilla

Contract Id: padilla

Contact Name: padilla

Contact Mail:

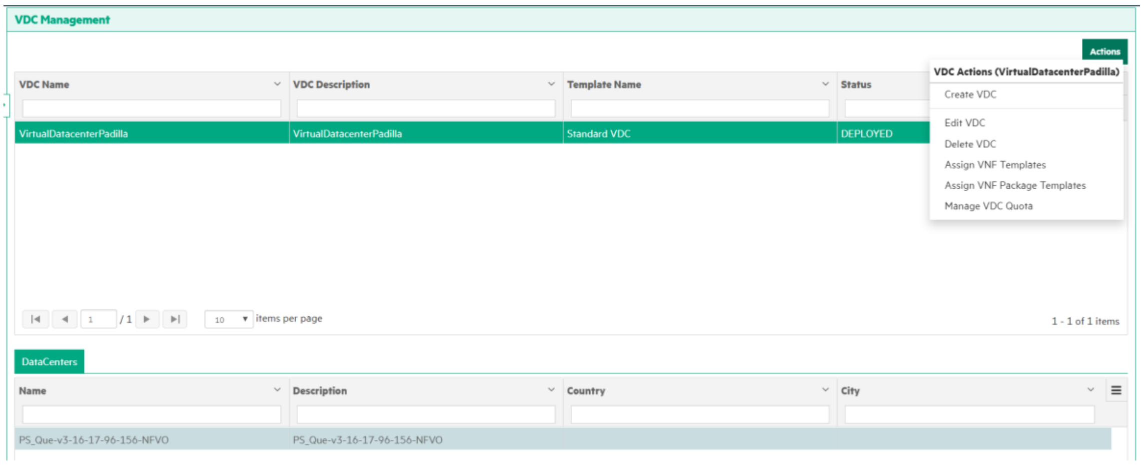
Contact Phone:

Name	Description	Template Name	Template Description
VirtualDatacenterPadilla	VirtualDatacenterPadilla	Standard VDC	Standard VDC

Name	Description	Country	City	Managed By
PS_Que-v3-16-17-96-156-NFVO	PS_Que-v3-16-17-96-156-NFVO			NFVO

Fig. 32 NFV Director Administration dashboard

b) In the *VDC Management* view (Fig. 33), actions over VDCs can be taken, which include editing, deleting, assigning VNF and packages templates or managing the VDC Quota:



VDC Management

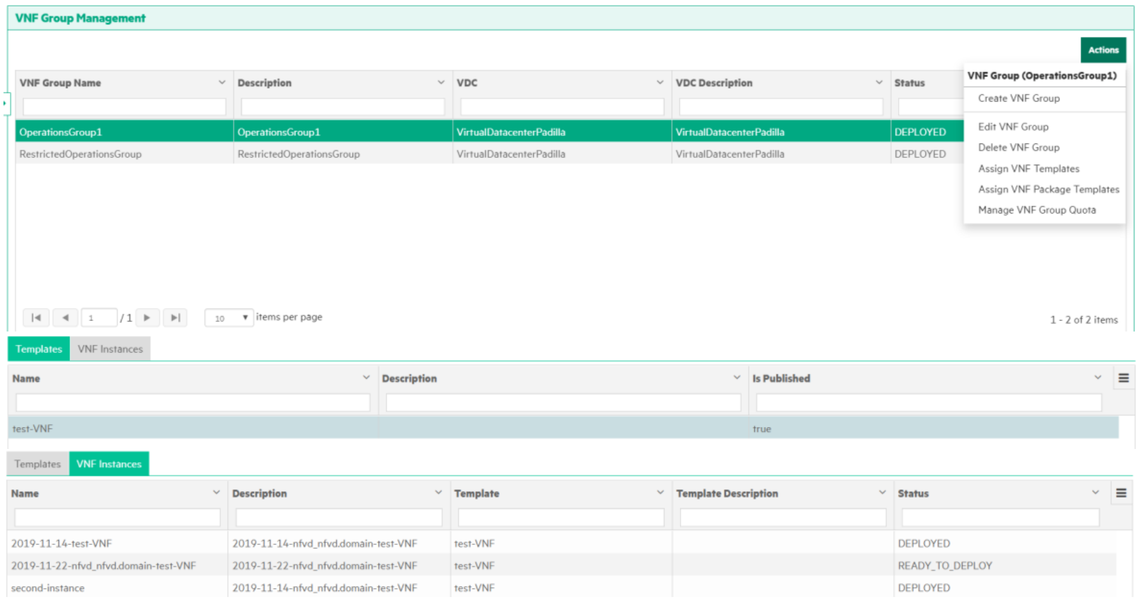
Actions

VDC Name	VDC Description	Template Name	Status
VirtualDatacenterPadilla	VirtualDatacenterPadilla	Standard VDC	DEPLOYED

Name	Description	Country	City
PS_Que-v3-16-17-96-156-NFVO	PS_Que-v3-16-17-96-156-NFVO		

Fig. 33 VDC Management view

c) In the *VNF Group Management* view (Fig. 34), actions over VNF Groups can be taken and the list of templates and VNF instances associates to each VNF Group can be seen:



VNF Group Management

Actions

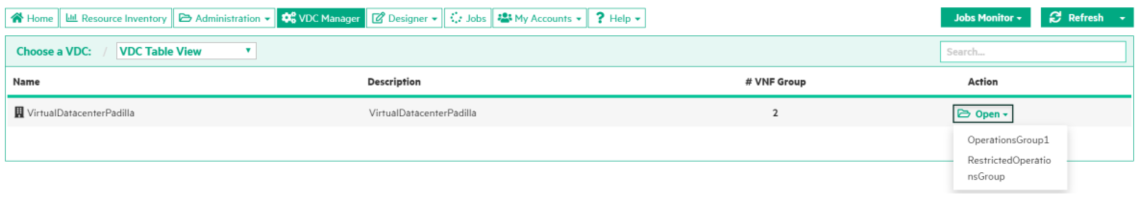
VNF Group Name	Description	VDC	VDC Description	Status
OperationsGroup1	OperationsGroup1	VirtualDatacenterPadilla	VirtualDatacenterPadilla	DEPLOYED
RestrictedOperationsGroup	RestrictedOperationsGroup	VirtualDatacenterPadilla	VirtualDatacenterPadilla	DEPLOYED

Name	Description	Is Published
test-VNF		true

Name	Description	Template	Template Description	Status
2019-11-14-test-VNF	2019-11-14-nfvd_nfvd.domain-test-VNF	test-VNF		DEPLOYED
2019-11-22-nfvd_nfvd.domain-test-VNF	2019-11-22-nfvd_nfvd.domain-test-VNF	test-VNF		READY_TO_DEPLOY
second-instance	2019-11-14-nfvd_nfvd.domain-test-VNF	test-VNF		DEPLOYED

Fig. 34 VNF Group Management

The following dashboard is the *VDC Manager* Dashboard, which shows the available VDCs and allows to open the VNF Groups associated to it. It is possible to visualize this information in a Datacenter graph (Fig. 36) view instead of Table View (Fig. 35):



Name	Description	# VNF Group	Action
VirtualDatacenterPadilla	VirtualDatacenterPadilla	2	Open

Fig. 35 VDC Manager VDC Table View

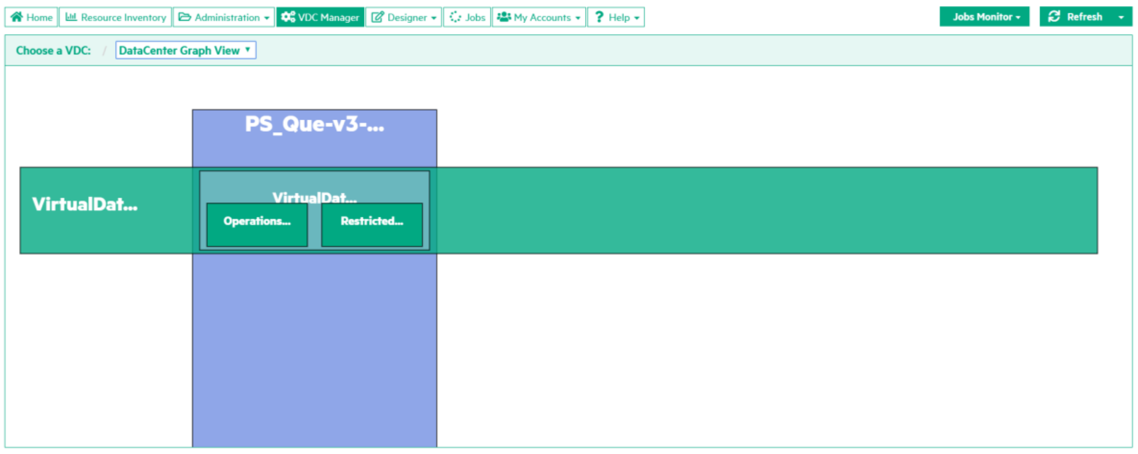
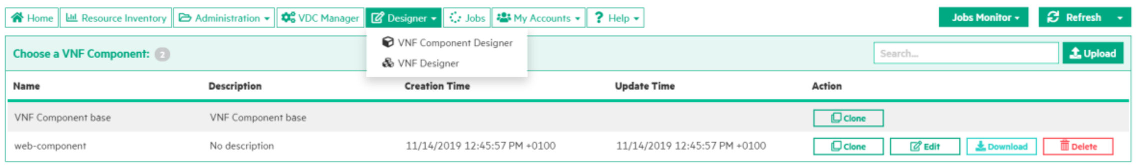


Fig. 36 VDC Manager Datacenter Graph View

Finally, the *Designer* Dashboard allows designing VNF components and VNFs:



Name	Description	Creation Time	Update Time	Action
VNF Component base	VNF Component base			Clone
web-component	No description	11/14/2019 12:45:57 PM +0100	11/14/2019 12:45:57 PM +0100	Clone Edit Download Delete

Fig. 37 NVF Director Designer

5.2.2 Onboarding of a generic VNF

Typically, the VNF onboarding process is performed as a collaboration between the VNF vendor and the telco provider. Previous to a VNF onboarding, the telco provider requires the VNF configuration information regarding the VNF format, number of networks and connectivity required, routing and security policies, IP ranges and performance requirements from the VNF vendor. In this onboarding, VNF vendor information was not available so, as telco providers, the deployment of a generic VNF without specific requirements has been performed. The steps to onboard a VNF are detailed in the flow chart of Fig. 38. Furthermore, a detailed description of the process is included.

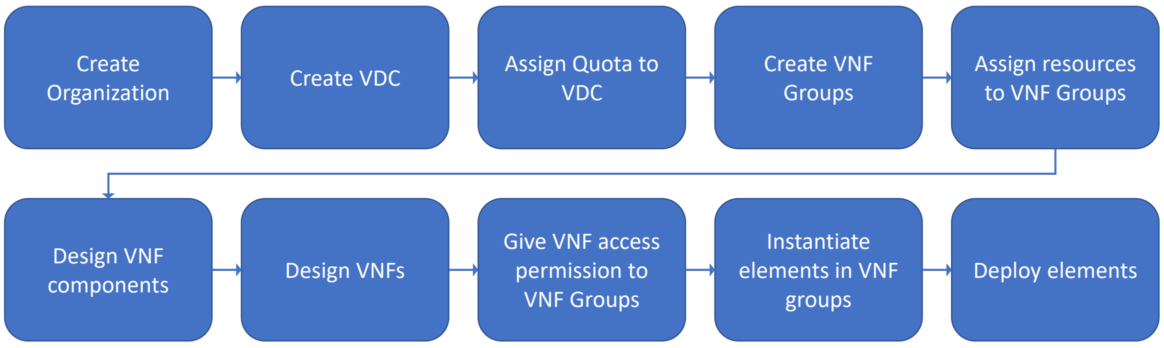
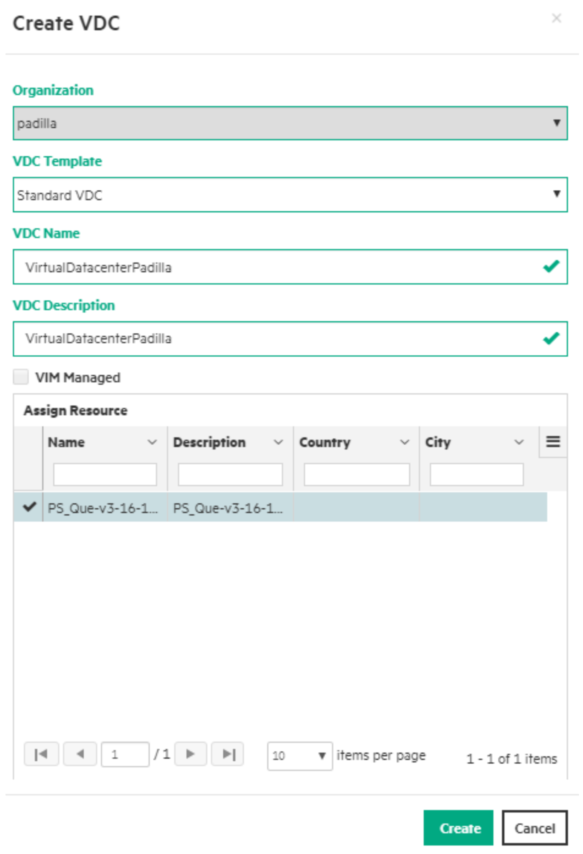


Fig. 38. Process of VNF onboarding

The first step in the process of Fig. 38, is the creation of the organization. For this purpose, HPE as a Domain administrator user of the platform has created an Organization called “padilla” that represents a telco provider, to whom it has assigned a DC, denoted PS_QUE-v3, from which it can consume resources in a cloud base model (i.e. pay-per-use).

The second step in Fig. 38 is the creation of a virtual data center. Inside “padilla” organization, a VDC called “VirtualDataCenterPadilla” is created, which will consume resources from the DC assigned to the organization. The next step to follow after creating the VDC is assigning a quota to it. In this case, the quota consists of 4 Virtual Cores, 32GB of Virtual Memory and 83GB of Virtual Disk. Fig. 39 and Fig. 40 show the process described above.


Create VDC

Organization: padilla

VDC Template: Standard VDC

VDC Name: VirtualDatacenterPadilla

VDC Description: VirtualDatacenterPadilla

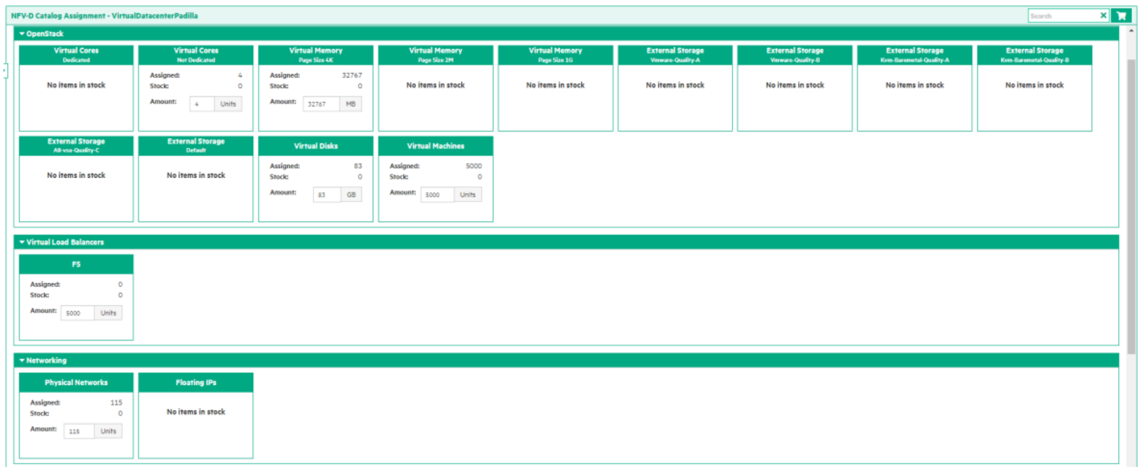
VIM Managed

Assign Resource

	Name	Description	Country	City
✓	PS_Que-v3-16-1...	PS_Que-v3-16-1...		

1 /1 · 10 items per page · 1 - 1 of 1 items

Fig. 39 VDC Creation


NFV-D Catalog Assignment - VirtualDatacenterPadilla

OpenStack

Virtual Cores Dedicated: No items in stock

Virtual Cores Not Dedicated: Assigned: 4, Stock: 0, Amount: 4 Units

Virtual Memory Page Size 4K: Assigned: 32767, Stock: 0, Amount: 32767 MB

Virtual Memory Page Size 2M: No items in stock

Virtual Memory Page Size 1G: No items in stock

External Storage Vmware-Quality-A: No items in stock

External Storage Vmware-Quality-B: No items in stock

External Storage Kvm-Baremetal-Quality-A: No items in stock

External Storage Kvm-Baremetal-Quality-B: No items in stock

External Storage All-vsa-Quality-C: No items in stock

External Storage Default: No items in stock

Virtual Disks: Assigned: 83, Stock: 0, Amount: 83 GB

Virtual Machines: Assigned: 5000, Stock: 0, Amount: 5000 Units

Virtual Load Balancers

F5: Assigned: 0, Stock: 0, Amount: 5000 Units

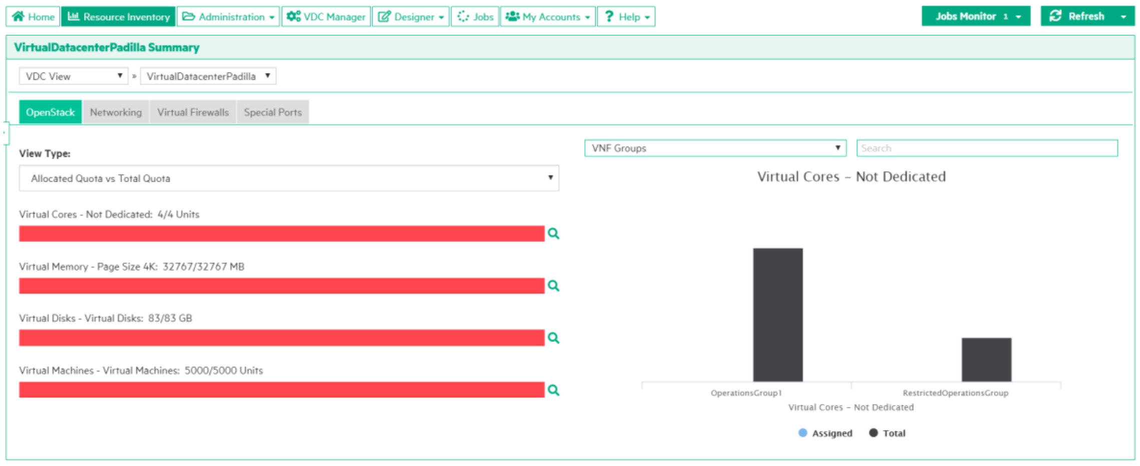
Networking

Physical Networks: Assigned: 115, Stock: 0, Amount: 115 Units

Floating IPs: No items in stock

Fig. 40 VDC Quota management

To complete steps 4th and 5th of the process in Fig. 38, inside the VDC two VNF Groups are created, which can be related to two different tenants or service providers called “OperationsGroup1” and “RestrictedOperationsGroup”. Each VNF group is assigned with a quota in order to distribute the resources of the VDC. As shown in bar graph of Fig. 41, two thirds of the resources are assigned to “OperationsGroup1” and one third to “RestrictedOperationsGroup”.



VirtualDatacenterPadilla Summary

VDC View » VirtualDatacenterPadilla

View Type: Allocated Quota vs Total Quota

Virtual Cores - Not Dedicated: 4/4 Units

Virtual Memory - Page Size 4K: 32767/32767 MB

Virtual Disks - Virtual Disks: 83/83 GB

Virtual Machines - Virtual Machines: 5000/5000 Units

VNF Groups

Virtual Cores – Not Dedicated

Fig. 41 VirtualDatacenterPadilla Resource Inventory

The following step is to design and deploy the generic VNF. The design is performed in the *VNF Component Designer* dashboard. Firstly, the base VNF Component, which is a generic VNF Component workspace, is cloned. Inside the VNF Component workspace, we can add VMs and configure them by assigning a flavor, which are the resources needed by the VM to run the application, and loading the image containing the VNF component application.

In our case, our VNF only requires one component, so in the *Component Designer* we drag a single VM to the workspace and proceed to configure it. We name the VM “test”, the selected flavor has 1 core, 512MB of RAM and 1GB of disk, and the uploaded image is a cirros image containing a lightweight Linux. After configuring the VM, we click on the workspace and we name that component “web-component”, considering that it is an Apache server. At the right side of the workspace, a list with all the parameters of the web-component can be visualized. All this process is included in Fig. 42.

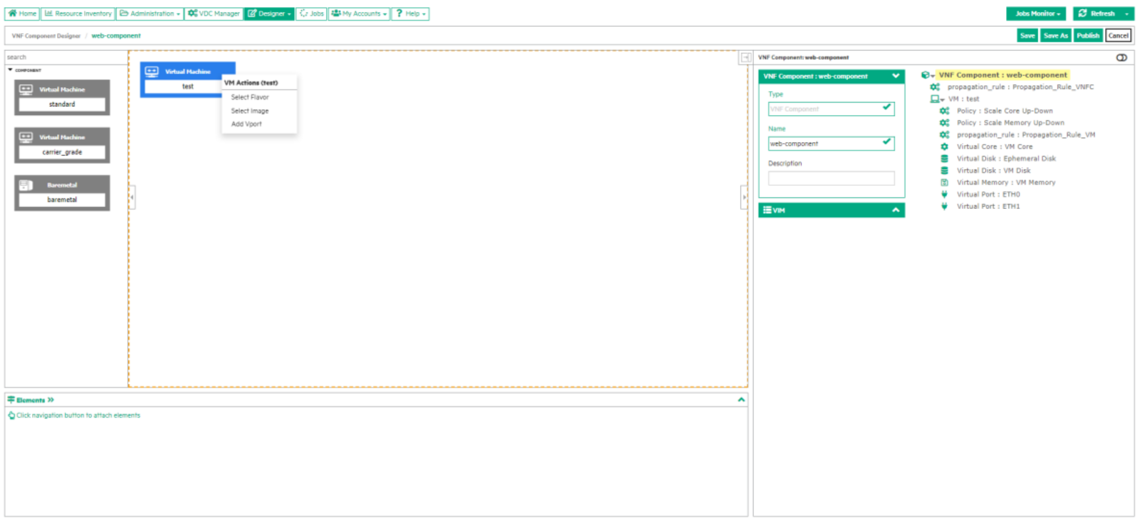


Fig. 42 VNF Component Designer workspace

Once the VNF components are created, they are all available in the *VNF Designer* dashboard (Fig. 43) catalogue. Then, a new VNF can be created by adding all the needed VNF components to the VNF Designer workspace. Moreover, network connections can be also added to the VNF at this stage.

In our case, the web-component is dragged to the workspace to create a new VNF called "test-VNF". In case the VNF would require it, we could use the same component as many times as necessary but, in order to keep the demo simple and taking into account the reduced availability of computing resources, it is just added once. Furthermore, as can be seen in the bottom part of Fig. 43, two networks connections have been added to the VNF: one external through the ETH1 port and another for management through the ETH0 port.

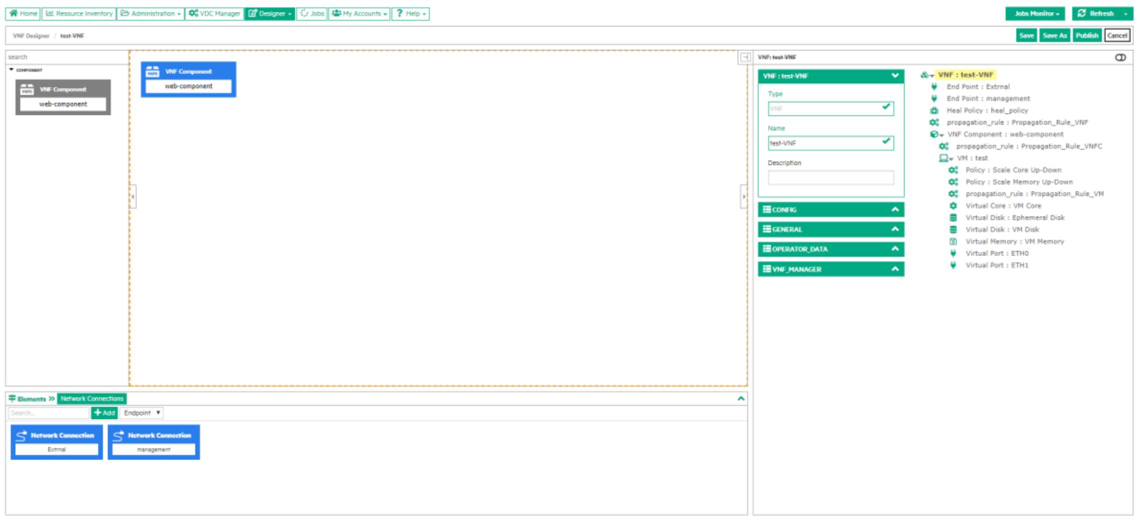


Fig. 43 VNF Designer workspace

When the configuration of the VNF has been finished, it can be saved and published for its use. Then, following step number 8 of Fig. 38, it is possible to give access to the VNF to the entities or groups that are going to use it. After providing access, the VNF appears in their catalogues.

For this demo, we decide to give permissions to the created VNF to “Operationsgroup1” only. In order to check the given access to each of the groups, we move to the *VDC Manager* dashboard and visualize the VNF Group catalogue.

In Fig. 44, we can see that the “RestrictedOperationsGroup” VNF catalogue is empty, as the “VNFs(0)” in the lateral bar indicates, since we have not allowed this group to access the created VNF. The only thing that group can deploy are the virtual networks “virtual_network_dual_stack” and “virtual_network” available in the VDC Elements, which does not require permissions.

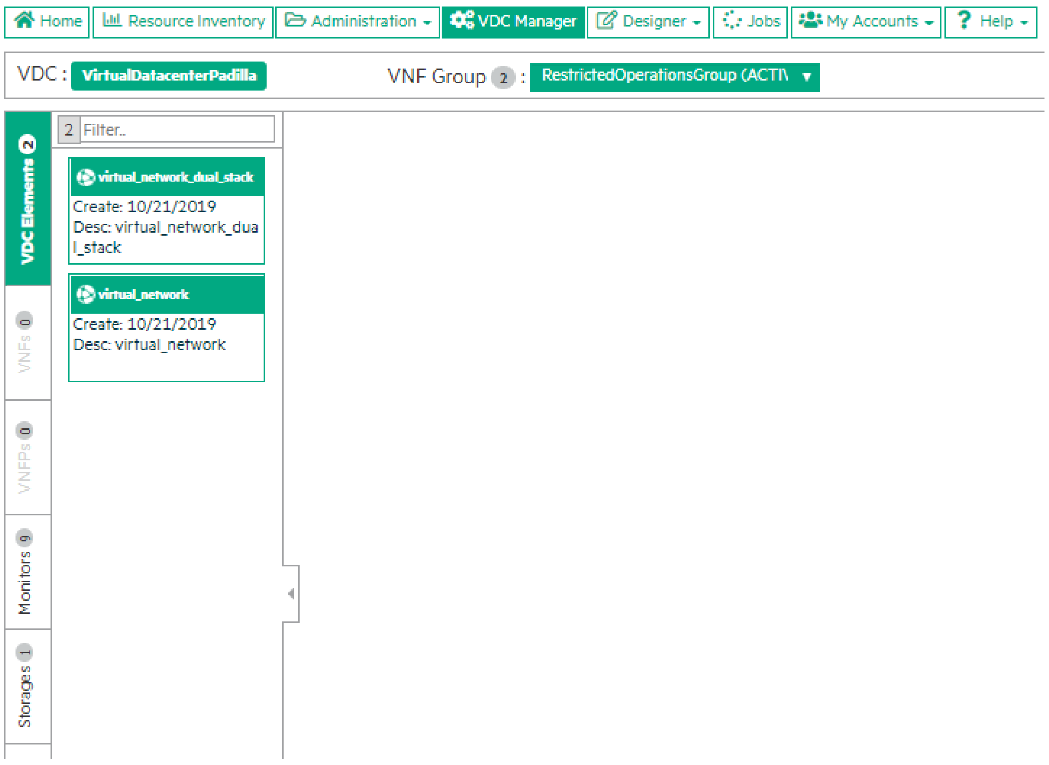


Fig. 44 VNF Group RestrictedOpeationsGroup workspace

Instead, as we have given permission to “Operationsgroup1” to access the VNF, “test-VNF” appears in the catalogue of the workspace to be used, as can be seen in Fig. 45.

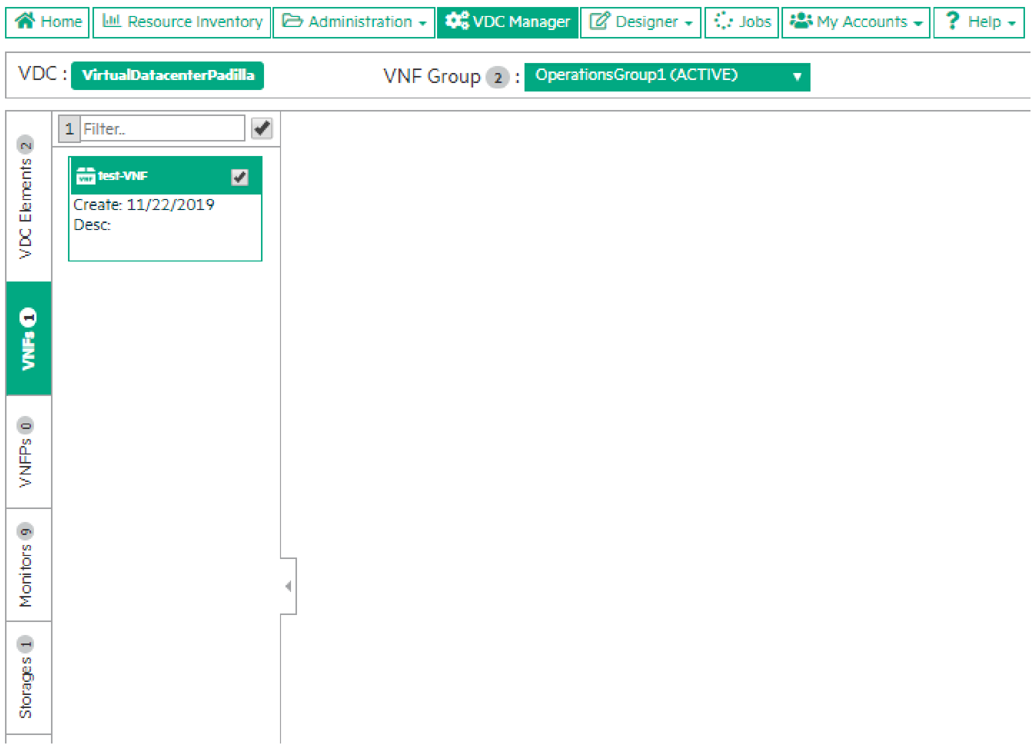


Fig. 45 VNF Group OperationsGroup1 workspace

Then, in the same dashboard, we are going to instantiate a couple of VNFs in “OperationsGroup1” and three networks where we will connect the VNFs, according to the Fig. 46. The first network will be used for the management and the other two will be used for an external connection. One of the external networks will be created using the NFVD while the other using the VIM (OpenStack), which will be then imported to NFVD. The last one is included in order to show that NFVD admits importing elements from the VIM such as networks. This is useful when an already created element for a project in VIM wants to be used in NFVD, which avoids having to create it again in NFVD from scratch.

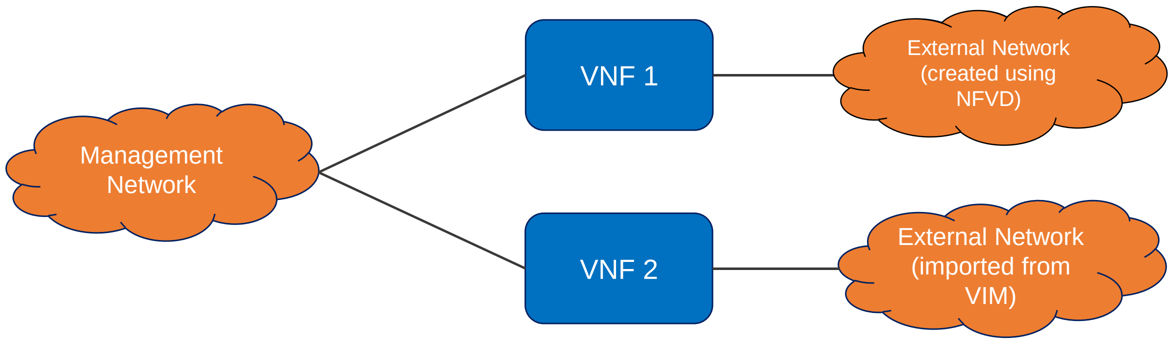
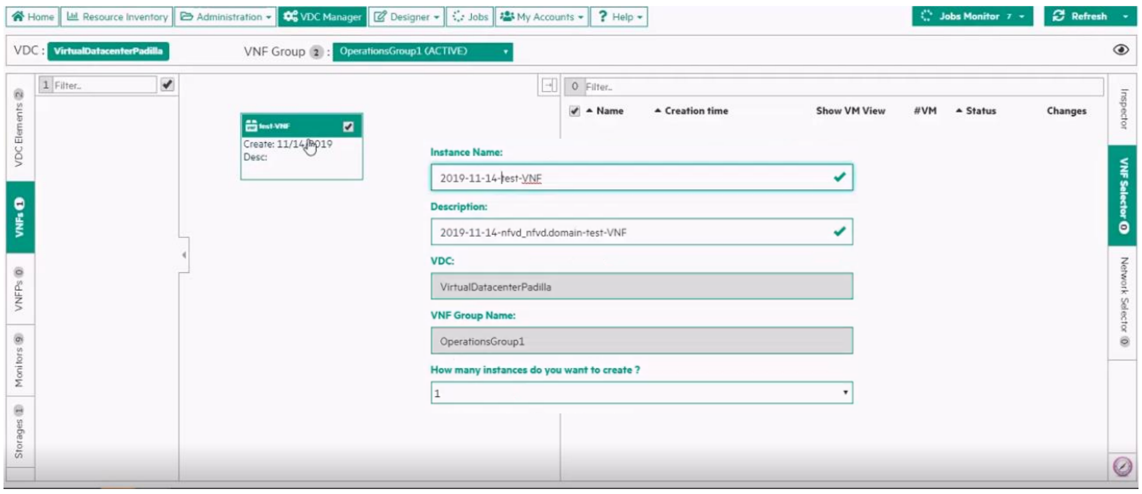

VNF 1
External Network (created using NFVD)
Management Network
VNF 2
External Network (imported from VIM)

Fig. 46. Generic VNFs onboarding scheme

In order to perform step number 9 of Fig. 38, to instantiate VNFs and networks we just have to drag the elements from the catalogue to the workspace. As instantiated VNFs are not still running, we will proceed to click on the “deploy” button to start running them and verify that the necessary resources have been deployed automatically in the VIM (OpenStack).

In our demo, we have instantiated a couple of the previously designed “test-VNF” in OperationsGroup1, each with a different instance name, as shown in Fig. 47 and Fig. 48.



VDC : VirtualDatacenterPadilla   VNF Group 2 : OperationsGroup1 (ACTIVE)

Name	Creation time	Show VM View	#VM	Status	Changes

Instance Name: 2019-11-14-test-VNF

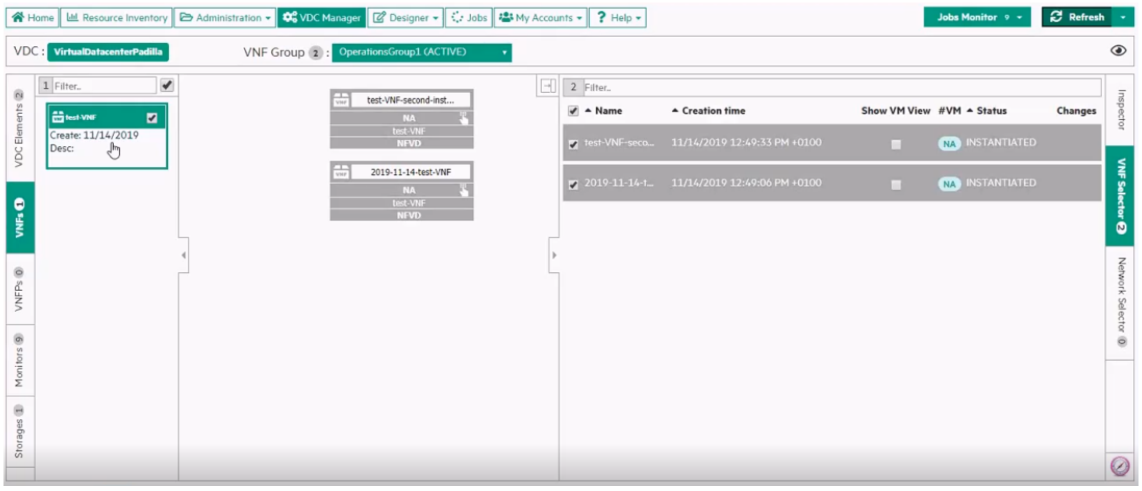
Description: 2019-11-14-nfvd_nfvd.domain-test-VNF

VDC: VirtualDatacenterPadilla

VNF Group Name: OperationsGroup1

How many instances do you want to create ? 1

Fig. 47 VNF instantiation

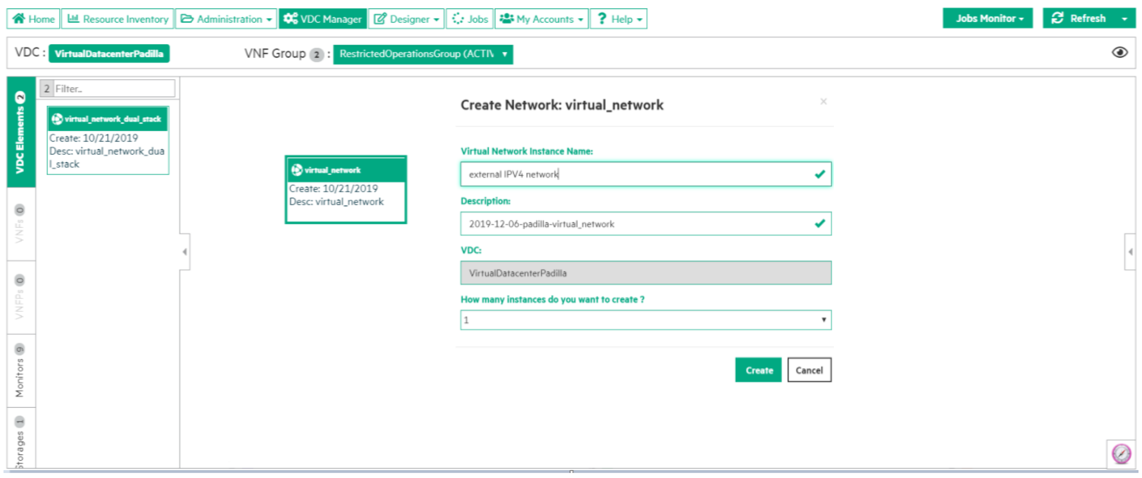


VDC : VirtualDatacenterPadilla   VNF Group 2 : OperationsGroup1 (ACTIVE)

Name	Creation time	Show VM View	#VM	Status	Changes
test-VNF-seco...	11/14/2019 12:49:33 PM +0100		NA	INSTANTIATED	
2019-11-14-t...	11/14/2019 12:49:06 PM +0100		NA	INSTANTIATED	

Fig. 48 OperationsGroup1 with two VNF instances

In order to create the external network from the NFVD, we can move to the “*RestrictedOperationsGroup”* in the *VDC Manager* dashboard and we can instantiate and deploy an IPV4 network, which we call it external network. The process is shown in Fig. 49 and Fig. 50.



Create Network: virtual_network

Virtual Network Instance Name: external IPV4 network

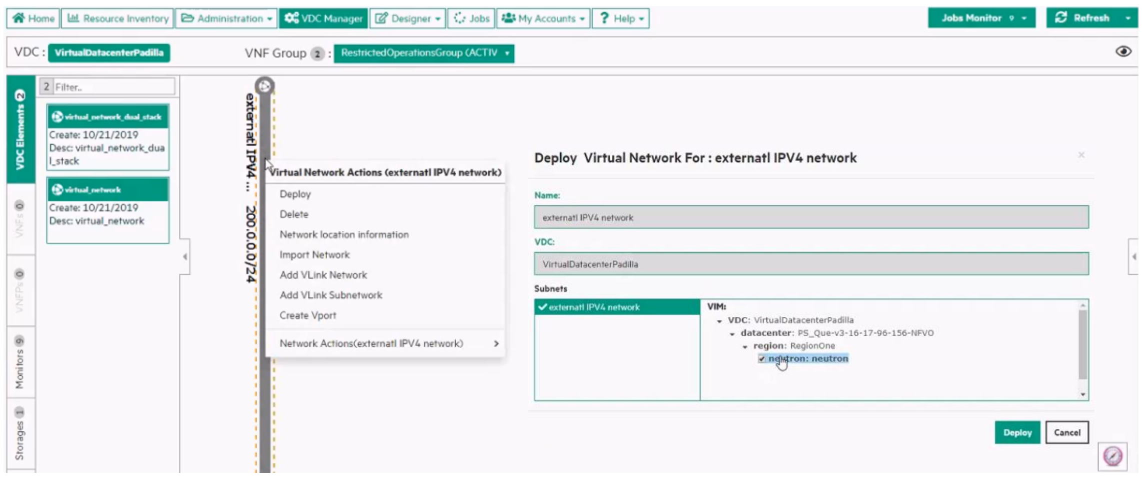
Description: 2019-12-06-padilla-virtual_network

VDC: VirtualDatacenterPadilla

How many instances do you want to create ? 1

Create Cancel

Fig. 49 Virtual network instantiation



Deploy Virtual Network For : externatl IPV4 network

Name: externatl IPV4 network

VDC: VirtualDatacenterPadilla

Subnets

✓ externatl IPV4 network

VIM:
- VDC: VirtualDatacenterPadilla
  - datacenter: PS_Que-v3-16-17-96-156-NFVO
    - region: RegionOne
      - ✓ neutron: neutron

Deploy Cancel

Fig. 50 Virtual network deployment

Until this point, we have seen how to instantiate VNFs and networks. Now, we are going to instantiate a second virtual network for management by following the aforementioned procedure. The virtual network for management is dual stack, which means that it will support IPV4 and IPV6. After deploying the virtual networks, we can right click on VNFs instances and connect them to the virtual networks as shown in Fig. 51.

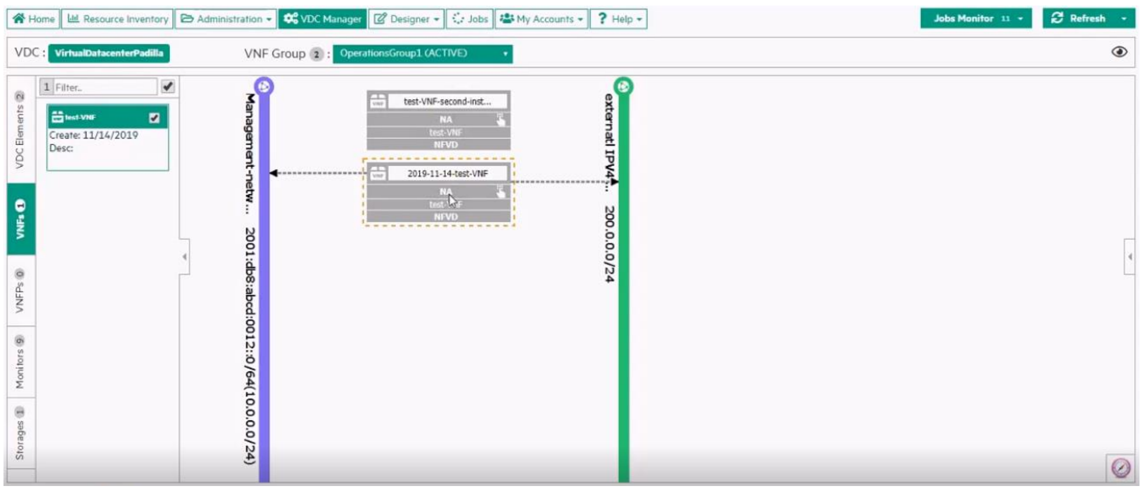


Fig. 51 Connecting VNF instances to virtual networks

If we go to OpenStack portal, which is the VIM, we can see that the virtual networks have been deployed.

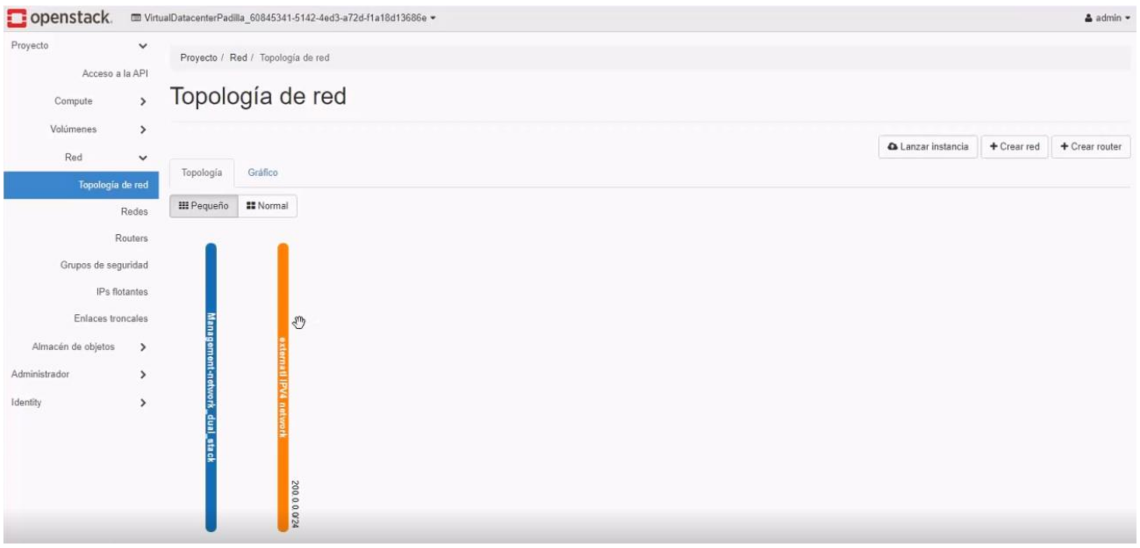
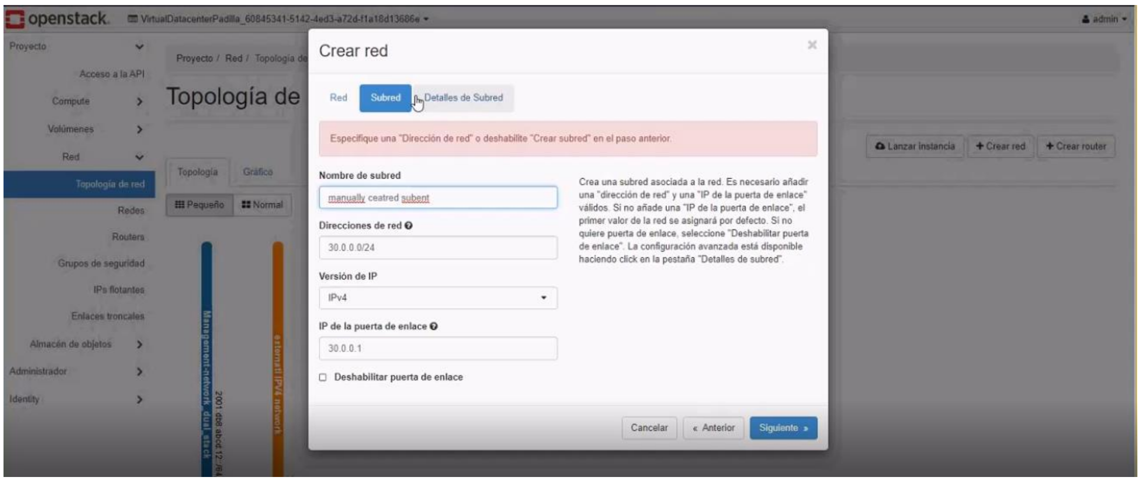


Fig. 52 OpenStack network topology

Taking advantage of the fact that we are in the VIM, we create a network manually, called “manually created net”, which we will later import into NFVD to show this functionality.



Especifique una "Dirección de red" o deshabilite "Crear subred" en el paso anterior.

Nombre de subred: manually ceatred subent

Direcciones de red: 30.0.0.0/24

Versión de IP: IPv4

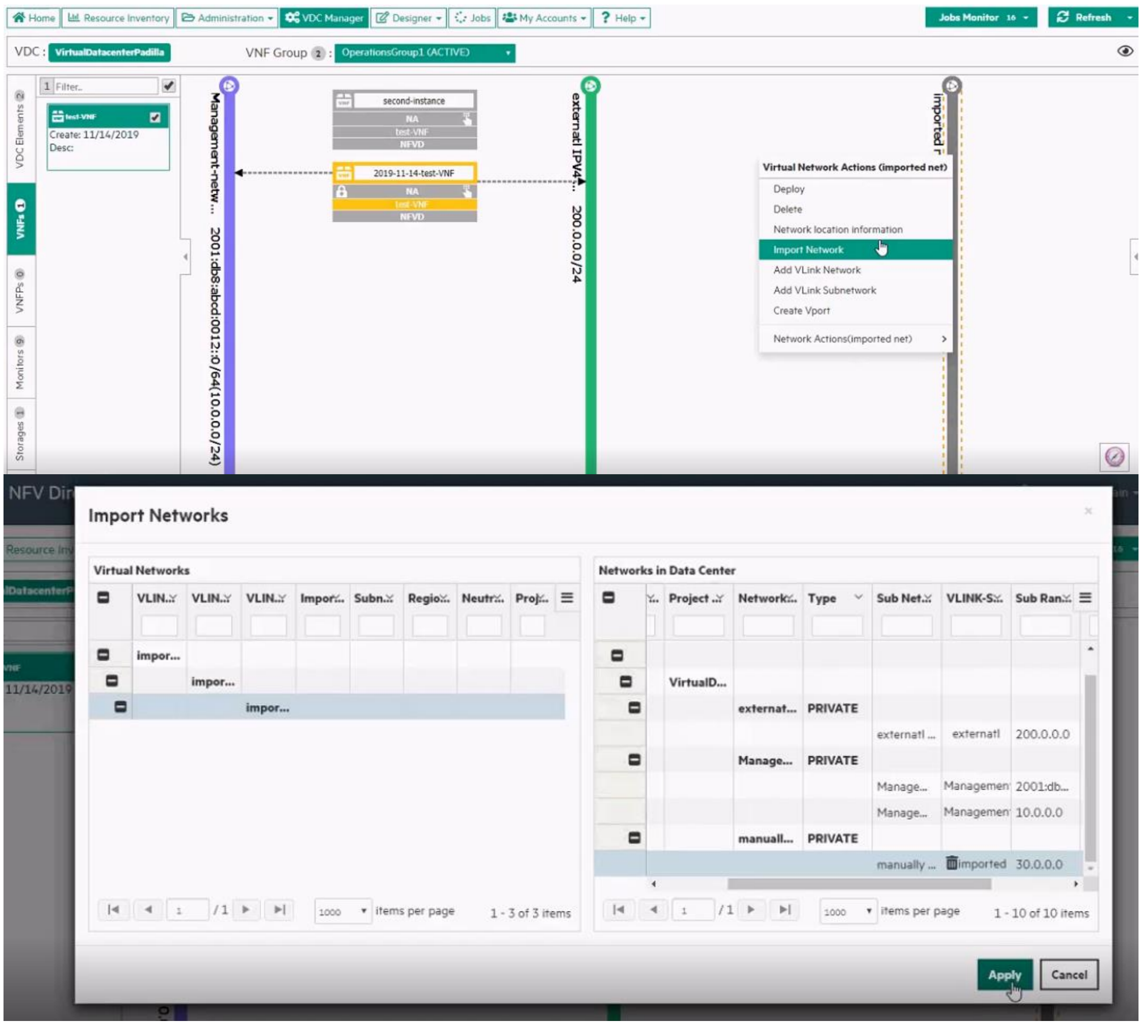
IP de la puerta de enlace: 30.0.0.1

☐ Deshabilitar puerta de enlace

Crea una subred asociada a la red. Es necesario añadir una "dirección de red" y una "IP de la puerta de enlace" válidos. Si no añade una "IP de la puerta de enlace", el primer valor de la red se asignará por defecto. Si no quiere puerta de enlace, seleccione "Deshabilitar puerta de enlace". La configuración avanzada está disponible haciendo click en la pestaña "Detalles de subred".

Fig. 53 Creating network in OpenStack

Back to NFV Director we are going to import the created virtual network in OpenStack, connect the VNF instances to the networks and proceed to deploy them. Fig. 54 shows the procedure to import a network in NFVD from the VIM.


Home
Resource Inventory
Administration
VDC Manager
Designer
Jobs
My Accounts
Help
Jobs Monitor 16
Refresh
VDC : VirtualDatacenterPadilla
VNF Group 2 : OperationsGroup1 (ACTIVE)
second-instance
2019-11-14-test-VNF
Management-netw... 2001:db8:abcd:0012::0/64(10.0.0.0/24)
externatl IPV4... 200.0.0.0/24
imported
Virtual Network Actions (imported net)
Deploy
Delete
Network location information
Import Network
Add VLink Network
Add VLink Subnetwork
Create Vport
Network Actions(imported net)
Apply
Cancel

Import Networks

Virtual Networks

VLIN...	VLIN...	VLIN...	Impor...	Subn...	Regio...	Neutr...	Proj...
impor...							
	impor...						
		impor...					

1 /1 1000 items per page 1 - 3 of 3 items

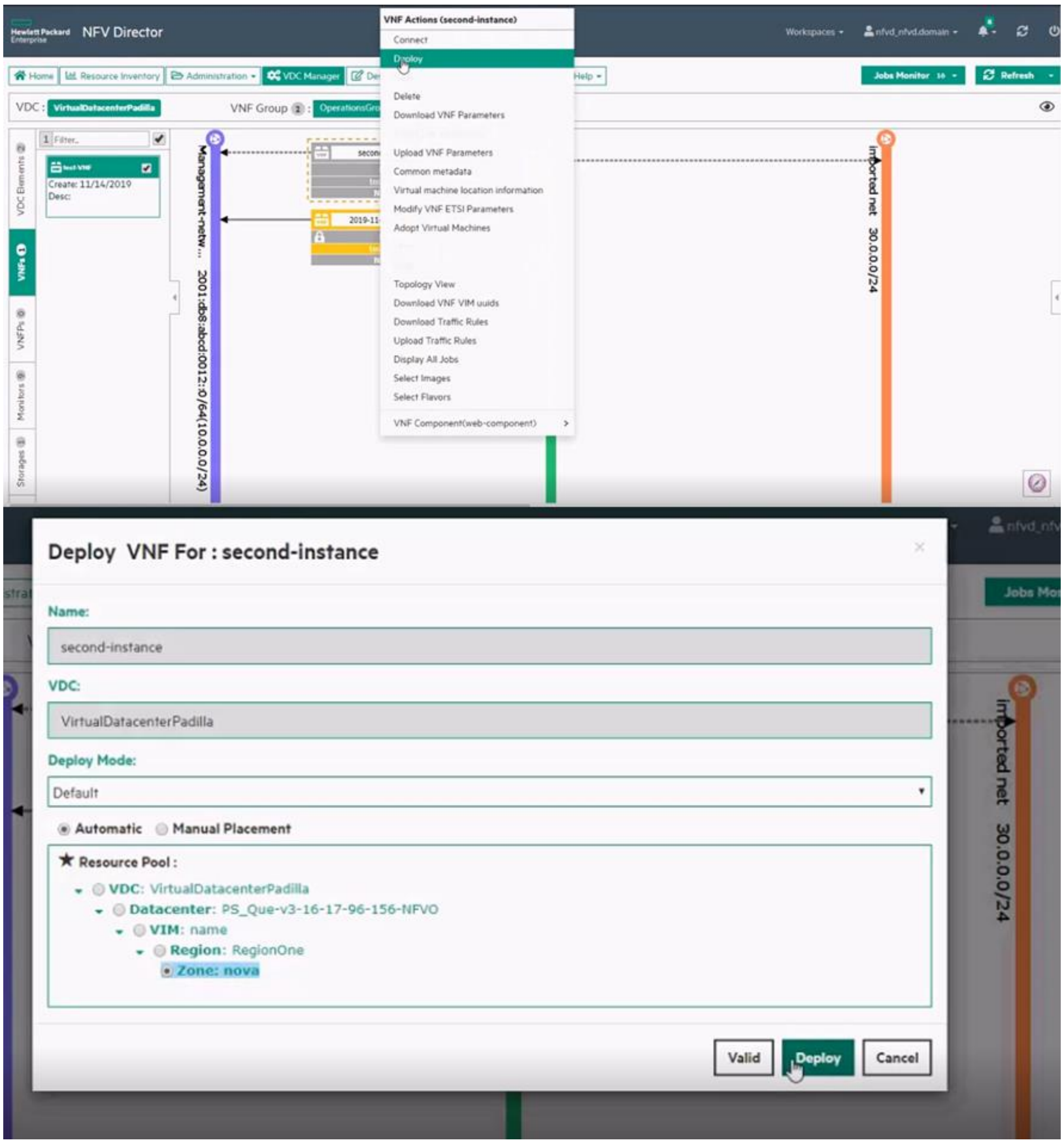
Networks in Data Center

...	Project ...	Network...	Type	Sub Net...	VLINK-S...	Sub Ran...
	VirtualD...					
		externat...	PRIVATE			
				externatl ...	externatl	200.0.0.0
		Manage...	PRIVATE			
				Manage...	Managemen	2001:db...
				Manage...	Managemen	10.0.0.0
		manuall...	PRIVATE			
				manually ...	imported	30.0.0.0

1 /1 1000 items per page 1 - 10 of 10 items

Fig. 54 Importing Networks

Fig. 55 shows the procedure to deploy a VNF once all the VNFs and network connections have been instantiated. This allows the VNF to start running.



Deploy VNF For : second-instance

Name: second-instance

VDC: VirtualDatacenterPadilla

Deploy Mode: Default

[✓ Automatic] Manual Placement

★ Resource Pool :
- VDC: VirtualDatacenterPadilla
  - Datacenter: PS_Que-v3-16-17-96-156-NFVO
    - VIM: name
      - Region: RegionOne
        - [✓ Zone: nova]

Fig. 55 VNF Instance Deployment

Fig. 56 shows the deployed VNFs and networks in NFV Director and in Fig. 57 we can see the same deployment in OpenStack.

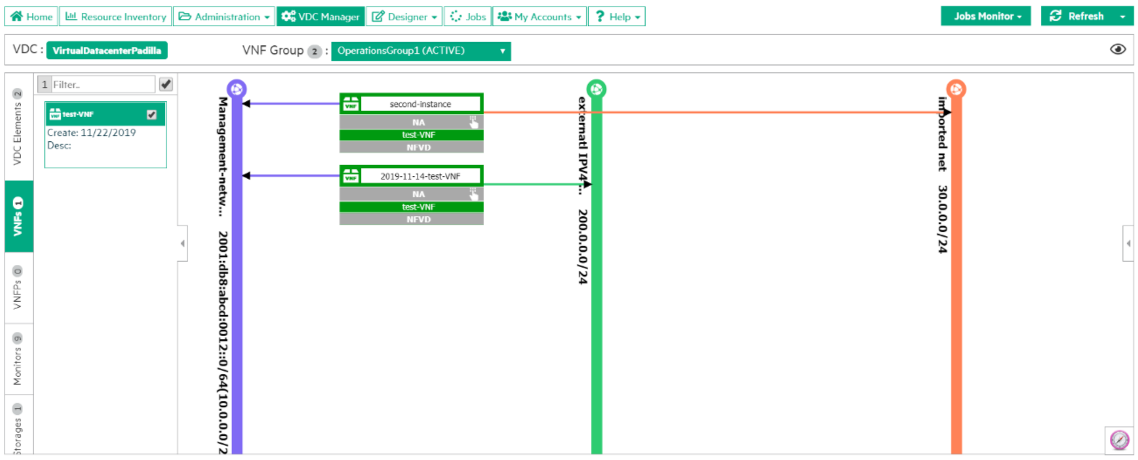


Fig. 56 NFV Director VNF deployment view

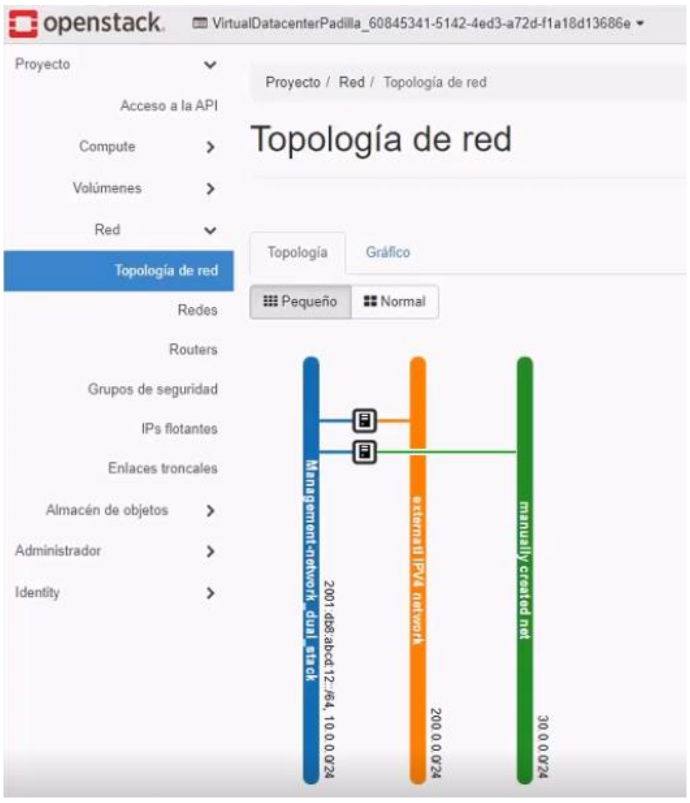
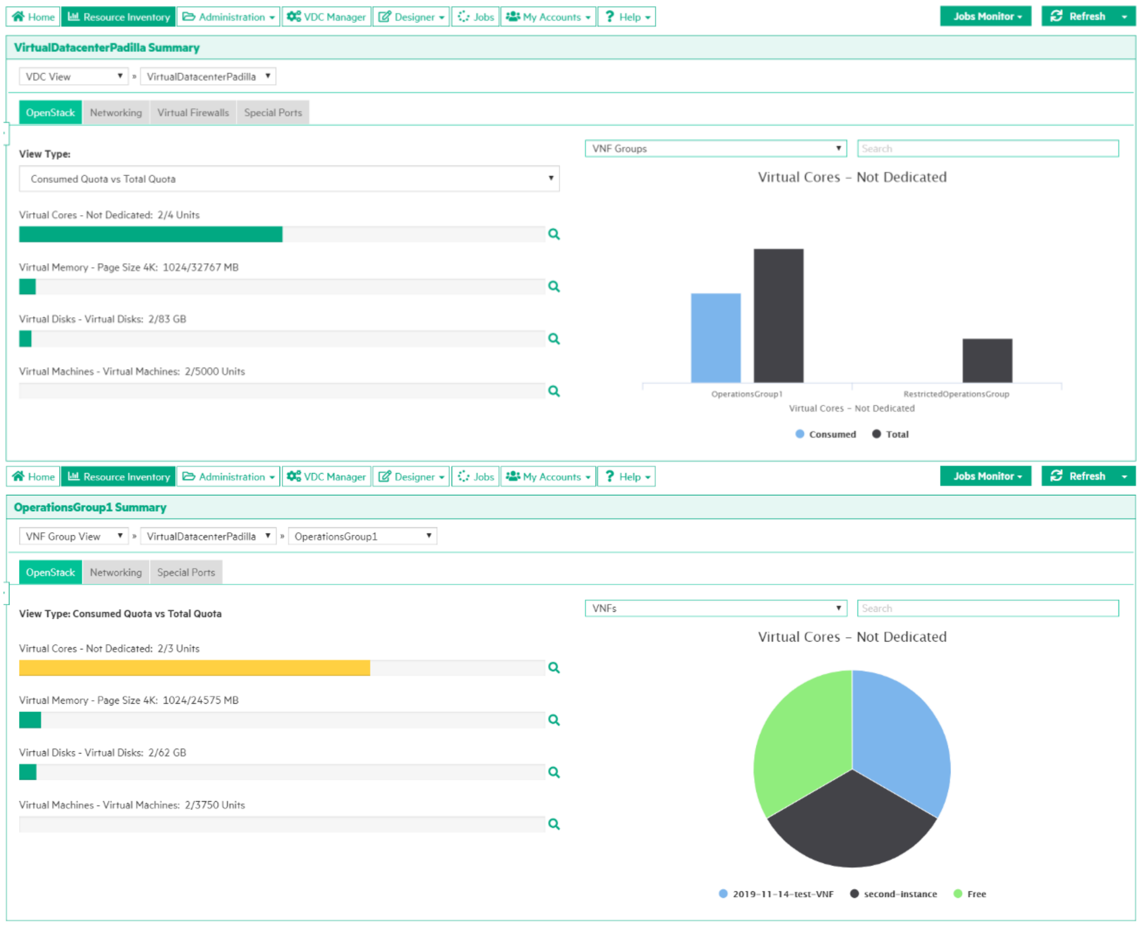


Fig. 57 OpenStack VNF deployment view

Finally, if we go to the *Resource Inventory* dashboard in NFVD, we can see how the status of the quota decreases after the onboarding of two VNFs in the VNF Group OperationsGroup1, as they use the configured allocated resources.



VirtualDatacenterPadilla Summary

VDC View » VirtualDatacenterPadilla

View Type: Consumed Quota vs Total Quota

Virtual Cores - Not Dedicated: 2/4 Units

Virtual Memory - Page Size 4K: 1024/32767 MB

Virtual Disks - Virtual Disks: 2/83 GB

Virtual Machines - Virtual Machines: 2/5000 Units

OperationsGroup1 Summary

VNF Group View » VirtualDatacenterPadilla » OperationsGroup1

View Type: Consumed Quota vs Total Quota

Virtual Cores - Not Dedicated: 2/3 Units

Virtual Memory - Page Size 4K: 1024/24575 MB

Virtual Disks - Virtual Disks: 2/62 GB

Virtual Machines - Virtual Machines: 2/3750 Units

Fig. 58 Resource Inventory after VNFs Onboarding

5.3 <u>Onboarding of the considered 5G scenario</u>

The considered scenario in Fig. 28 with the two slices and the related VNFs has been deployed and orchestrated by means of the HPE NFVD over the modality of bare metal outside the cloud. The external DNs are not deployed since they would be outside the 5G core slice of the tenant. However, an interface connection to the DN is instantiated. The available computing resources are the same as in the previous demo.

In order to create the two network slice entities associated to “Tenant1” and “Tenant2”, we have created two VNF groups in the NFVD named “Slice1” and “Slice2” by following the procedure previously explained. The available resources are equally split between the groups.

Afterwards, the UPF, SMF and AMF VNFs have been also created. Given that the real 5G VNFs requirements are not available to the general public, their requirements in terms of CPU, memory, disk and connectivity have been adapted to the available resources in the NFVD tool.

For the deployment of the scenario, the previously described procedure is developed to create each of the 5G core VNFs. Firstly, the UPF, SMF and AMF VNFs components have been created, each with 1 core, 512MB of RAM and 1GB of disk. In this case, a generic Linux image has been loaded. However, in a real scenario, the image loaded would correspond to a vendor provided VNF image providing each of the NF operation.

In second place, the 5G core VNFs have been created by adding the corresponding components and interfaces connections and, finally, publishing them. Finally, the VNFs groups have been given permission to access all the created VNFs in order that they appear in the "Slice1" and "Slice2" catalogues.

In the following, the description of the steps to deploy the desired scenario is described by instantiating firstly the VNFs for "Slice1" and secondly for "Slice2", with the particularity that the AMF VNF is shared between them.

5.3.1 Single slice deployment

For deploying the first slice, the same steps described previously for the onboarding of generic VNF have been followed. Firstly, we go to *VDC Manager Dashboard* and select the VNF Group "Slice1". At this point, VNFs are already designed and presented in the catalogue so we proceed to drag and connect all the necessary resources. For "Slice1", we drag to the workspace a single instance of each VNF: UPF, SMF and AMF. Then, two networks are instantiated: a Control Plane Network with IP 200.0.0.0 where all the VNFs are connected and a Data Plane Network with IP 200.0.0.1, where the UPF is connected following the scenario scheme. These networks are related to the illustrated in Fig. 5, which correspond to the service-based model interface of 5G core NFs (i.e., Namf and Nsmf), the N4 interface for the SMF to control the UPF and the N6 interface that connects the UPF to the external DN.

Fig. 59 shows the result of the above described process for "Slice1" deployment.

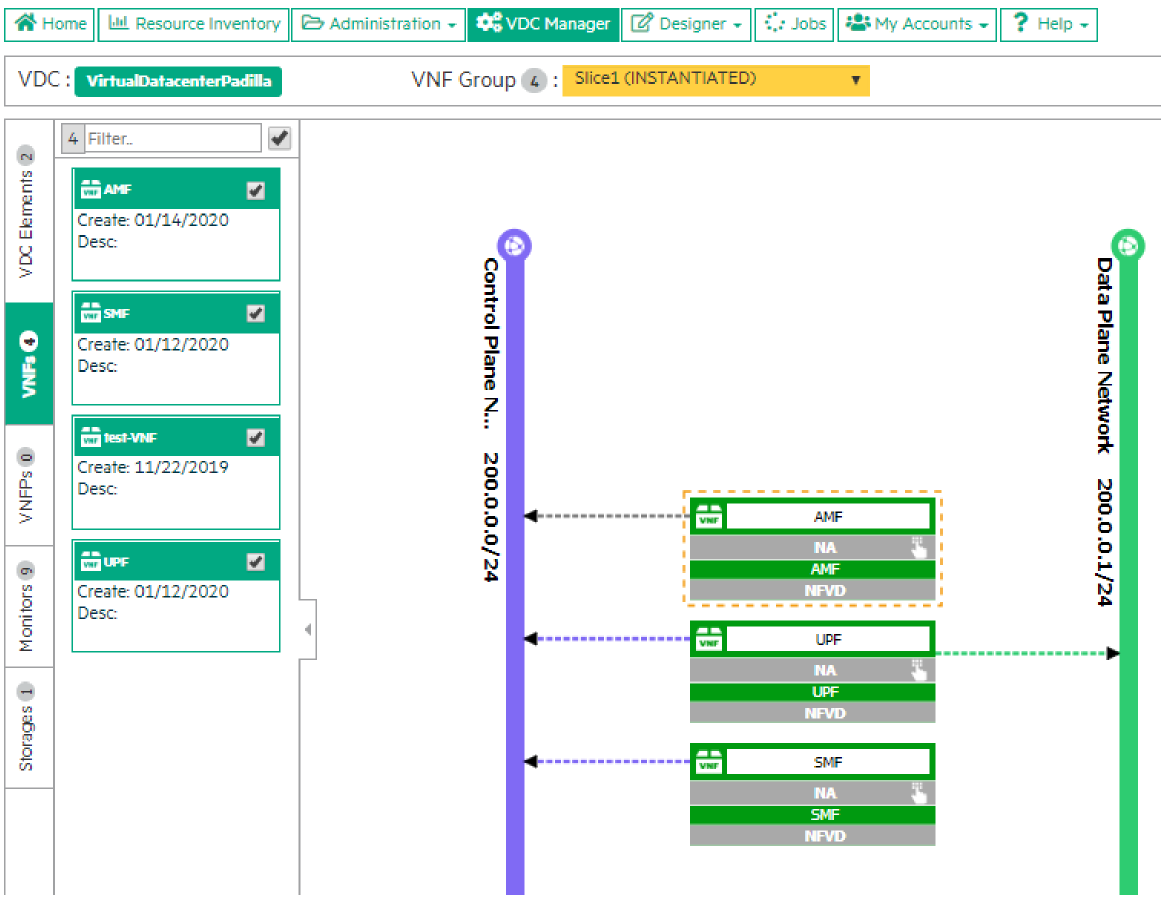


Fig. 59 Slice 1 Deployment

5.3.2 Addition of a new slice

When adding “Slice 2” in the scenario, the steps to follow are similar to the previous one, but with the particularity that a second AMF VNF does not need to be instantiated. In this sense, SMF and UPF are added to the workspace of VNF Group “Slice2”. Then, the control plane and data plane networks are also created with the IP 200.0.0.2/24 and 200.0.0.3/24, respectively. Finally, the SMF and UPF are connected to these networks as before, as shown in Fig. 60.

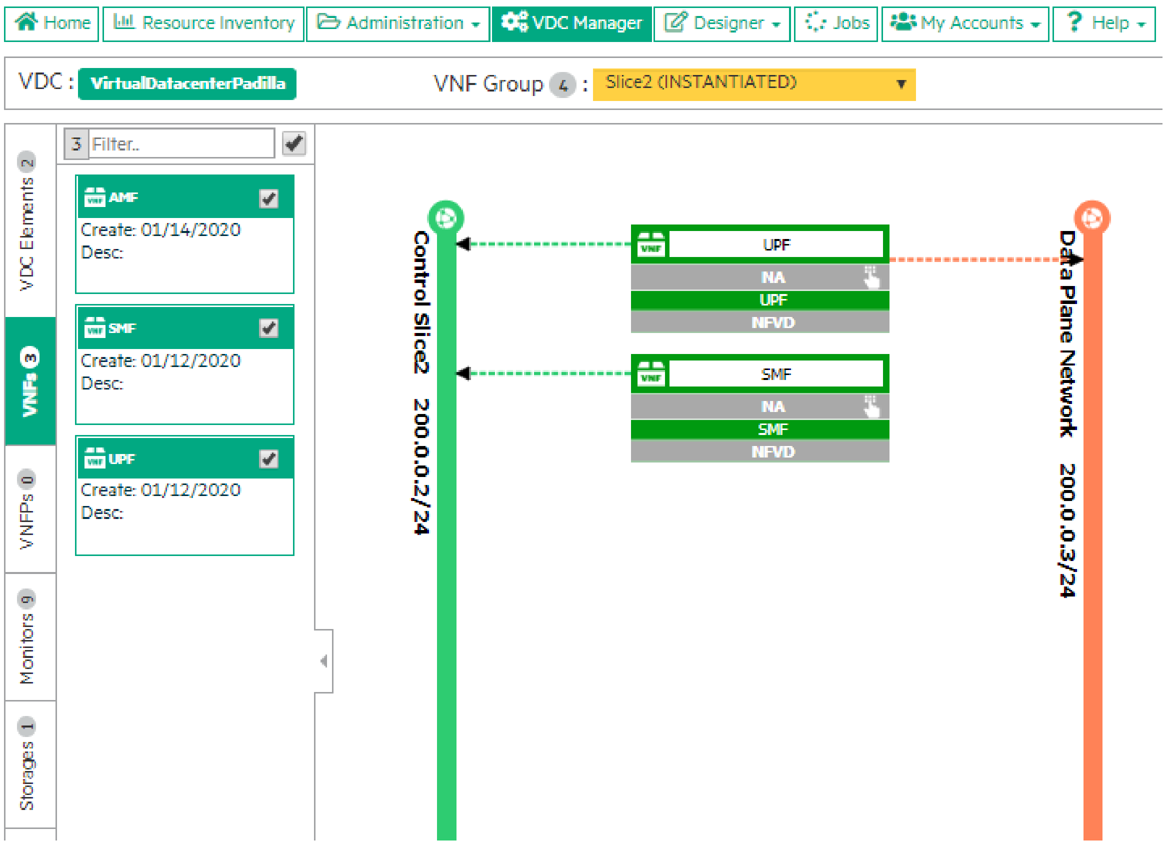


Fig. 60 Slice 2 Deployment

Note that it is not necessary to instantiate a second AMF VNF in “Slice2”, since both slices share the same AMF NF. In order to share the AMF, the AMF VNF of “Slice1” has to be connected to the control plane network of “Slice2”. Given that all the created networks are visible from all VNF group workspaces, this connection can be performed in “Slice1” workspace. Fig. 61 shows the workspace of “Slice1” resulting deployment.

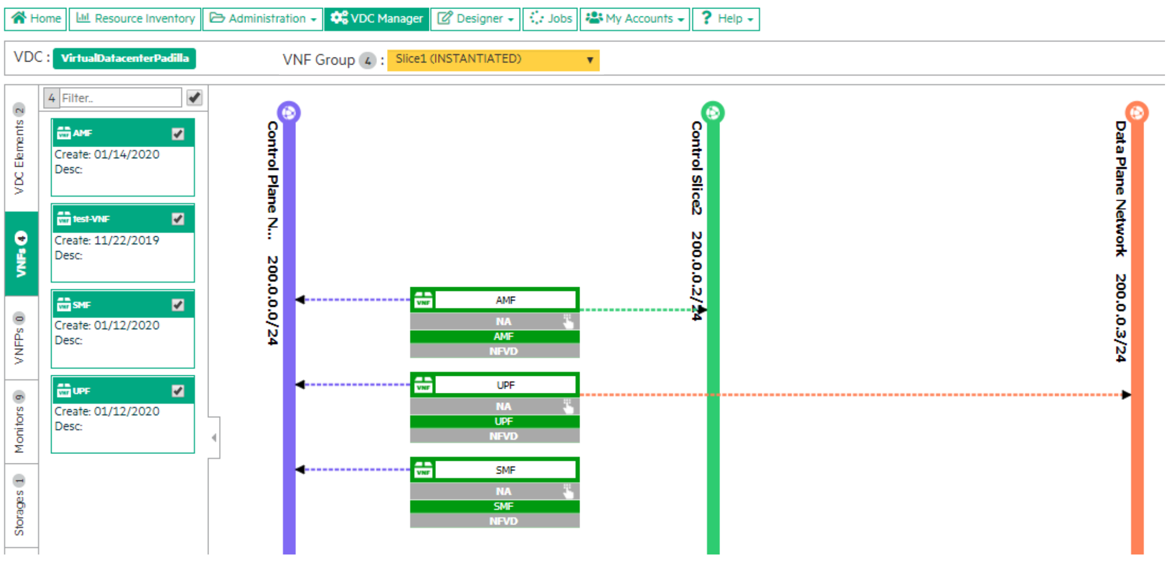
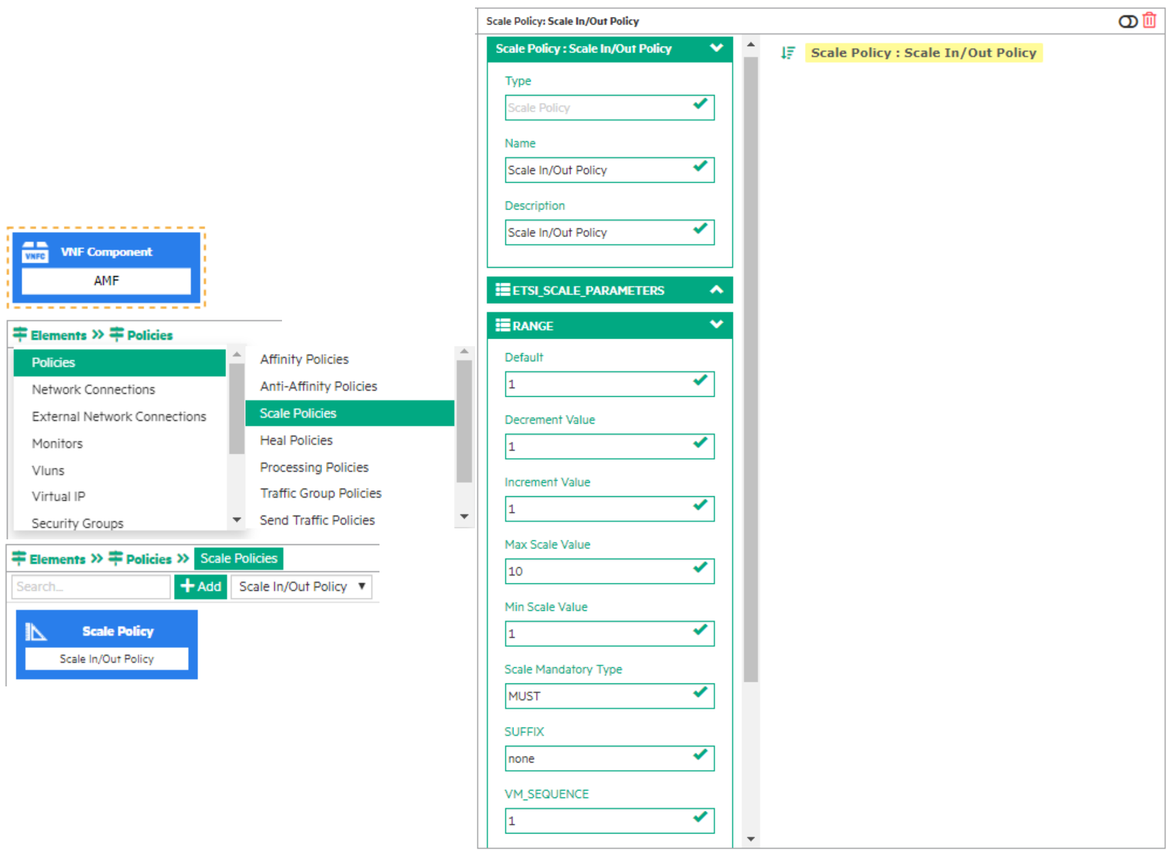


Fig. 61 Slice1 resulting deployment

At this point, it is necessary to re-configure the deployed AMF VNF in order to support the operation of “Slice1” and “Slice 2”. Initially, the AMF VNF was configured to support a limited number of users restricted to “Slice1”, so the virtual resources of AMF need to be increased to support a higher number of users. This can be performed by means of two kinds of procedures: the scale up/down procedure and the scale in/out procedure. While the first allows increasing or decreasing the number of virtual cores and virtual RAM, the second allows adding or removing virtual machines to the VNF. In operation terms, the scale up/down requires to reboot the VNF while scale in/out can be performed while the VNF is running.

For both procedures, the scaling specification (i.e., number of virtual resources or VMs) must be specified in the VM scaling polices field when designing the VNFs. This allows the automation of the scaling process.

In the following, the re-configuration of the AMF VNF is going to be performed by following both procedures. For this purpose, the AMF VNF is designed to incorporate two VM scaling polices called “scaling up/down” and “scaling in/out” policies. While the “scaling up/down” policy is included as default, the “scaling in/out” policy needs to be created and added to the AMF VNF from the “VNF Designer dashboard”. The procedure to include the second policy is shown in Fig. 62 as well as the policy range values, which specify the value of increasing and decreasing in each scale in/out step and the maximum and minimum number of allowed VMs. In this case, each time scale in/out is performed, the number of VMs will be increased/decreased by 1 and the number of VMs has to be minimum 1 and maximum 10.


VNF Component
AMF

Elements » Policies

Policies	
Policies	Affinity Policies
Network Connections	Anti-Affinity Policies
External Network Connections	Scale Policies
Monitors	Heal Policies
Vluns	Processing Policies
Virtual IP	Traffic Group Policies
Security Groups	Send Traffic Policies

Elements » Policies » Scale Policies
Search... + Add Scale In/Out Policy

Scale Policy
Scale In/Out Policy

Scale Policy: Scale In/Out Policy

Scale Policy : Scale In/Out Policy

Type	Scale Policy
Name	Scale In/Out Policy
Description	Scale In/Out Policy

ETSI_SCALE_PARAMETERS

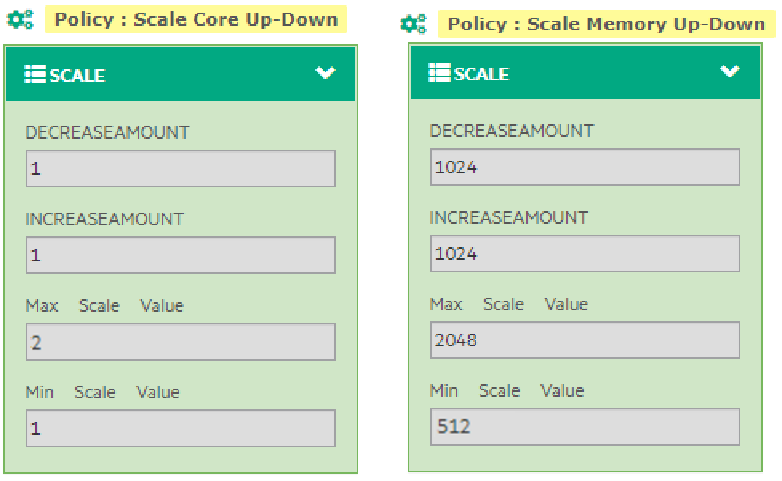
RANGE

Default	1
Decrement Value	1
Increment Value	1
Max Scale Value	10
Min Scale Value	1
Scale Mandatory Type	MUST
SUFFIX	none
VM_SEQUENCE	1

Scale Policy : Scale In/Out Policy

Fig. 62 Adding VNF Scaling In/Out policy

Moreover, default scaling up/down policy values are shown in Fig. 63. If desired, these values can be modified. The same features as in the scale in/out case need to be fulfilled (i.e., increase/decrease amount and maximum/minimum values). In the case of the scale core up/down policy, the number of cores will be increased/decreased by 1 for each scale up/down step and the number of cores has to be minimum 1 and maximum 2, so just one core can be added. In the case of the memory scale up/down policy, each time the scale up/down is selected, the memory will be increased/decreased by 1024 MB, and the minimum of memory capacity is 512MB and the maximum 2048MB.


Policy : Scale Core Up-Down

SCALE

DECREASEAMOUNT	1
INCREASEAMOUNT	1
Max Scale Value	2
Min Scale Value	1

Policy : Scale Memory Up-Down

SCALE

DECREASEAMOUNT	1024
INCREASEAMOUNT	1024
Max Scale Value	2048
Min Scale Value	512

Fig. 63 Scaling up/down policy default values

In order to perform the scaling procedures, it is required to move to the "Slice1" workspace and click on the AMF VNF. As shown in Fig. 64, the options to apply the above mentioned policies appear.

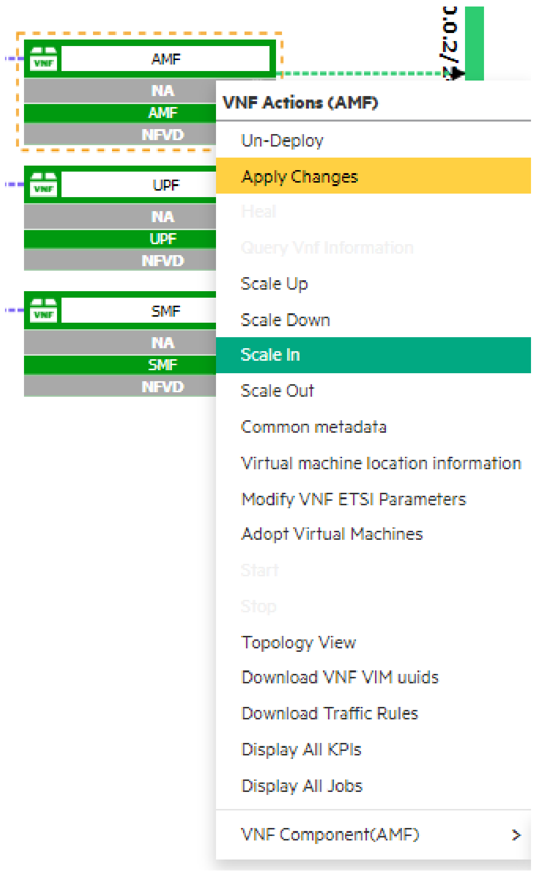


Fig. 64 VNF scaling up/down in/out options

When "Scale in" option is selected a prompt appears to confirm the operation. After confirming the process, if we open the VNF inspector at the right of the screen, we can see that a VNF Component of the same type has been added to the VNF AMF. Fig. 65 shows the result of the described process.

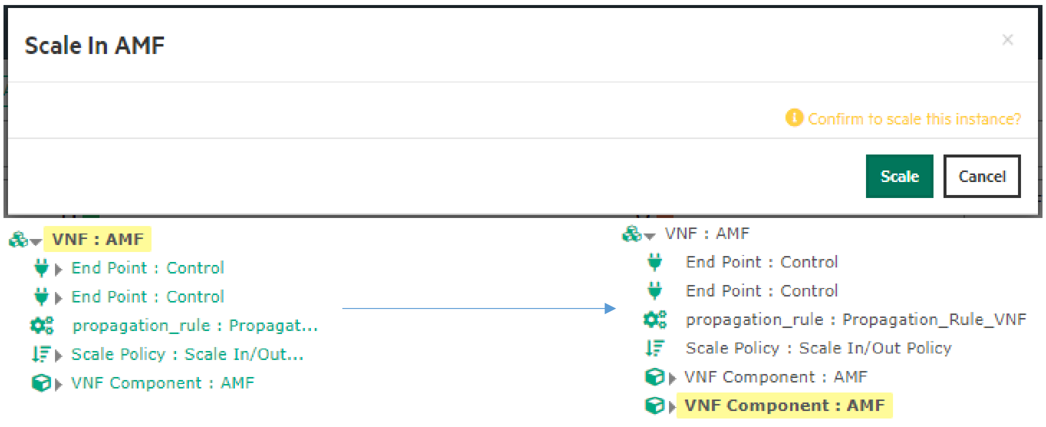
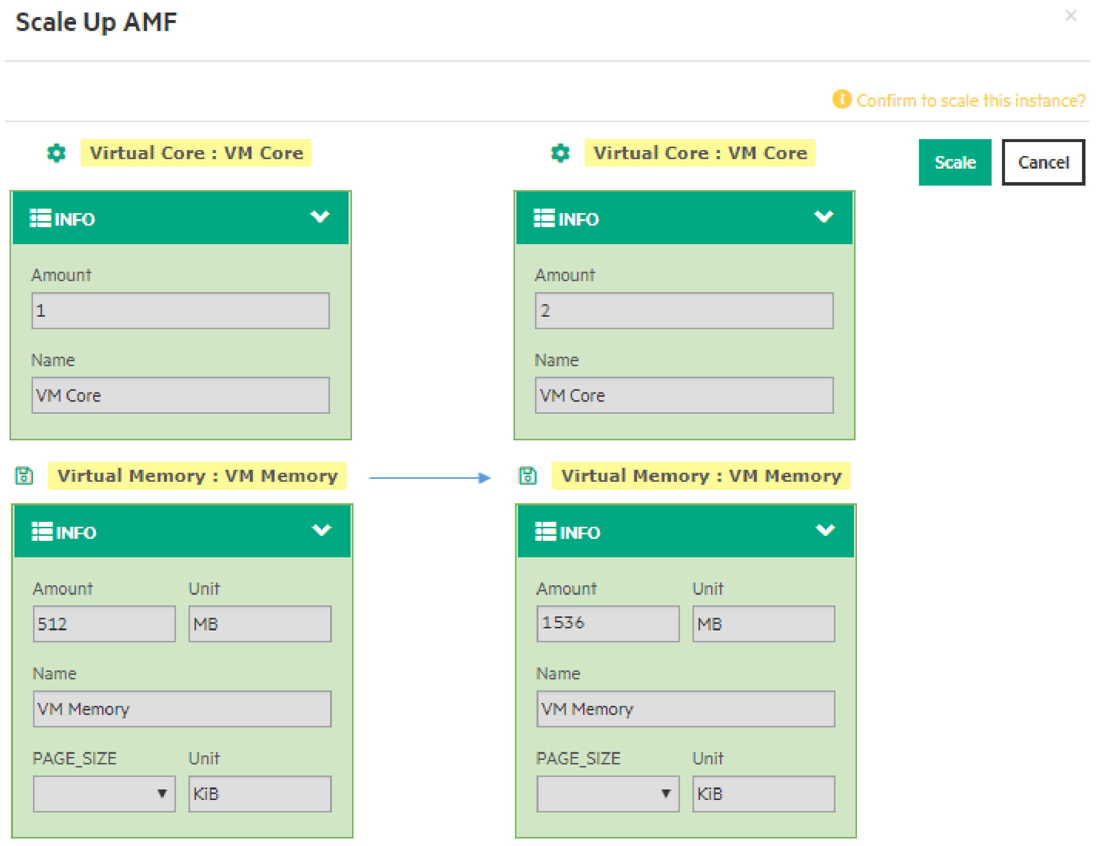


Fig. 65 Scaling in result

If instead of performing a scale in, we want to scale up the VNF, following the same procedure we select the "scale up" option and a prompt appears to confirm the operation.

Then, following the previously defined rules for that policy, the VNF resources are increased. The result is shown in Fig. 66.


Scale Up AMF

Confirm to scale this instance?

Virtual Core : VM Core

INFO

Amount: 1

Name: VM Core

Virtual Memory : VM Memory

INFO

Amount: 512 Unit: MB

Name: VM Memory

PAGE_SIZE: Unit: KiB

Virtual Core : VM Core

INFO

Amount: 2

Name: VM Core

Virtual Memory : VM Memory

INFO

Amount: 1536 Unit: MB

Name: VM Memory

PAGE_SIZE: Unit: KiB

Fig. 66 Scaling up results

5.4 <u>Solution assessment and comparison with other tools</u>

The HPE proposal for NFV-MANO has proved to be a very powerful and complete tool to deploy our 5G scenario with two slices. Although during the completion of this project the resources we had were quite limited, we have been able to perform onboardings and deployments of VNFs that have allowed us to observe the orchestration and management processes and the interaction with external elements like the VIM, which is Openstack in this case.

The process that takes more time is the design of the VNFs as they can become very complex. Although in the scope of the project we wanted to deploy 5G core vendor VNFs in the scenario, the deployed ones are very simple due to the fact that real 5G VNFs are being developed by specific vendors and are not public to its use yet. Furthermore, the available resources for this demo were too small considering the resources used in recent deployments of 5G VNFs, such as the carried out by Intel and SK Telecom [32], where in order to deploy a virtualized UPF a 28-core CPU is used and 10 VMs are deployed. Despite of the fact that HPE tool is ready to deploy, orchestrate and manage VNFs of that magnitude, the provided resources for the demo license would not have been enough to test them.

Specifically, focusing on VNF designing tool, we have checked that is very complete, allowing to develop designs in great detail. This has been achieved by dividing the design process into two parts: VNF component designer and VNF designer. Moreover, different policies applicable in the designs of the VNFs allow managing large projects in a simple manner. Once the VNFs are designed, only simple actions, such as drag and drop into the different workspaces, are required to deploy an entire 5G Network in matters of minutes.

The tool has worked smoothly and quickly throughout the time we have had access to it. No lags or bugs have been experienced for the deployed scenarios, which are reduced in terms of demands on the consumption of resources. However, the performance of the tool when deploying larger and more complex scenarios may be affected.

The graphical interface of the tool is very intuitive, which allows a smooth and fast learning process. Regarding the visualization of projects, the elements and their connections are clearly presented. In turn, the visualization of network slicing environments can be a bit confusing as more levels of abstraction are added. Fortunately, HPE is working in a specific visual environment for network slicing that will improve this point.

One interesting feature of NFV Director is its job tracking monitor, which allows controlling and keeping track of all the processes executed in the tool. The provided information consists of the running times and users that have been involved. This feature is really useful to follow-up and control that the different procedures are executed satisfactorily and, in the case that an error or malfunction occurs, to be able to find its source.

In terms of the virtualization technologies available for the NFVD version used during this project, only VMs are supported. The other technologies introduced in the initial part, which are the unikernels and software containers, are not available, so we have not been able to test them. However, in the new version of the tool, which has been recently released, software containers are supported.

In general, the possibility to use the NFVD has been a very positive experience, both for the opportunity to operate a commercial tool and to have a better and practical understanding of the theory related to NFV.

Finally, a comparative table of the mentioned NFV-MANO solutions in section 4.5 is included to contrast their current situation in the market, considering aspects such as the telco support, their market coverage or the alignment with the ETSI standard.

Table 6 NFV-MANO solutions comparison

	OSM	**ONAP**	**HPE NFVD**
CSPs using the tool.	BT, Telefónica	AT&T, China Mobile	SKT, Verizon, Dtag, NTT
Market coverage by region	Europe	North America, Asia	North America, Europe, Asia
Network vendors of VNFs	ZTE	Ericsson, Nokia, Cisco, Huawei	Ericsson, Nokia, Samsung
Development model	Open Source	Open Source	Proprietary
ETSI NFV MANO compliance	Yes	Yes	Yes
GUI ease to use	Low	Medium	High
Include support	No	No	Yes
Supports containers	No	No	Yes (rel.5)

In order to evaluate the GUI ease to use of the different tools, the VNF onboarding process has been taken as a reference for the three NFV-MANO solutions. Note that for OSM and ONAP this feature has been evaluated considering the onboarding tutorials in [33] and [34], respectively, while for NFVD the VNF onboarding process has been carried out in the above sections. The followed criteria considers the visual and organizational aspects of the tools as well as the complexity of the steps to carry out the same task.

Another fact to mention is that despite OSM and ONAP have been tested and used in 5G research projects, they have not been deployed in real production networks whereas HPE NFVD has been in production for different clients around the world for more than 3 years, according to the available information from HPE.

6 Budget

This chapter includes an estimation of the costs of the project. The different costs are composed of labor cost and development tools cost.

Considering a Junior Engineer receiving 10,30€/hour and a 30% of fees and social insurance, the labor costs considering an hour remuneration of 13,30€/hour are detailed in Table 7.

Table 7. Labor costs of the project

Project stage	Hours	Cost
Formation and research	400h	5320€
Scenario deployment	200h	2660€
Thesis Documentation	200h	2660€
Total	**800h**	**10640€**

The employed tools used for the development of the project are summarized in Table 8, which includes a certain amortization time (if relevant) and considers that the project has lasted 6 months. The HPE NFVD tool has been provided as a free trial version with no costs.

Table 8. Development tools costs of the project

Project stage	Price	Amortization time (months)	Project cost (6 months)
Development PC	900€	36 months	150€
Microsoft Windows 10 Pro License	259€	60 months	27,90€
Microsoft Office Professional plus 2013	149€	36 months	24,80€
Total			**202,70€**

By considering the above, the total costs are included in Table 9.

Table 9 Total project costs

Concept	**Cost**
Labor	10640€
Development tools	202,70€
Total	**10842,7€**

7 Conclusions and future development

This Thesis has presented the state of the art of the 5G systems architecture. This new generation in mobile technologies brings up a revolution in the way cellular systems are conceived, given its orientation as a service based model. In 5G, the architectural elements of the networks are defined as network functions that offer and consume services between them. This novelty introduces a new level of abstraction that allows the disaggregation of the network elements from a hardware specifically built to perform these functions. Usually, this specific network hardware is expensive and costly to maintain by telecommunications service providers, which implies huge investments in both time and money when improving the offered services and upgrading from a mobile generation to the next one. However, the new 5G service-based model, together with the technologies that make it possible, have achieved the necessary level of abstraction to make the deployment of networks in COTS equipment that can be physically allocated anywhere. This allows reducing operator's costs and bringing new possibilities like MEC and Network Slicing, which enable to offer new services and improve the already deployed in the market.

All these improvements that 5G brings with it would not be feasible without the technologies that support its development. These technologies have been presented in this Thesis, which include virtualization technologies and virtualization technologies applied to networks, NFV.

Regarding virtualization technologies, three different approaches have been presented, two of them very well-known and with widespread use in current market systems, such as VMs and software containers and, the third one, unikernels, which is still in a phase of research and development. As we have seen in the Thesis, although the three technologies may seem similar, their intrinsic characteristics make their use to be conditioned on the final needs of the application to be hosted. For example, VMs present a higher level of isolation against containers, but they are heavier and take more time to instantiate than containers. Therefore, we cannot say that one technology is better than another but that it is recommended the use of the technology that better suits the needs of the application in every case.

Then, we have studied the state-of-the-art of NFV to see how virtualization technologies are applied to networks. The standardization of this technology has been carried out by ETSI, and all the elements, functional blocks and their relationships have been defined in order to establish a series of rules when developing tools and applications for virtualized network. In this Thesis, a special focus in the MANO elements of the ETSI Framework has been developed. In this sense, three different market MANO solutions have been presented: OSM, ONAP and HPE NFV Director.

In order to better understand the presented market MANO tools, HPE offered us a license of its MANO tool, NFV Director. Taking advantage of this opportunity, the tool has been tested by deploying a scenario comprised of the 5G core VNFs associated to two different network slices. Although the computational resources in the tool were limited and 5G vendors VNFs were not available, we managed to deploy the designed 5G scenario and assess the possibilities that the provided tool offers, as well as its ease of use.

In conclusion, thanks to HPE NFV Director, it has been possible to analyze the state-of-the-art of one of the market MANO leading tools as well as the possibility to delve deeper into the technologies that make possible the deployment of 5G networks.

7.1 Future Development

This Thesis offers the possibility of continuing using NFV-MANO tools like HPE NFV Director to study more aspects related to NFV applied to cellular systems. In the following, different proposals to keep extending the research are given:

- Program a 5G NF to specify how to establish the necessary resources to be deployed as a VNF.
- Test different MANO tools like OSM or ONAP in order to compare them with HPE proposal.
- Perform the onboarding of VNFs using software containers instead of VMs in the new version of HPE NFVD, in order to study the advantages or disadvantages
- Perform the complete deployment of a virtualized 5G network, including commercial 5G VNFs or self-developed ones.

Bibliography

[1] J. Pérez-Romero, “5G System Architecture”. Eines i tecnologies de la xarxes móbils 5G.

[2] F. Mademann, “The 5G System Architecture,” in *Journal of ICT Standardization*, vol. 6, no. 5, pp. 77–86, May, 2018, Available at: <https://www.3gpp.org/ftp/Inbox/Marcoms/ICT_6_1-2.pdf>.

[3] Cisco White paper. “Cisco Ultra 5G Packet Core Solution”. Available at: <https://www.cisco.com/c/dam/en/us/products/collateral/routers/network-convergence-system-500-series-routers/white-paper-c11-740360.pdf>

[4] 3GPP TS 23.501 V15.4.0, “System architecture for the 5G system; Stage 2 (Release 15),” Dec. 2019.

[5] ETSI, “MEC in 5G networks”, June, 2018. Available at: <https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf>

[6] B. Bertenyi, R. Burbidge, G.Masini, S. Sirotkin and Y. Gao. “NG Radio Access Network (NG-RAN)”, May 2018.

[7] Graziano, Charles. "A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project".

[8] vSphere Documentation Center. (2019). Retrieved 5 October 2019, < https://pubs.vmware.com/vsphere-50/index.jsp >

[9] R. P. Goldberg, “Survey of virtual machine research,” Computer, vol. 7, no. 6, pp. 34–45, Jun. 1974

[10] Virtualization Technology & Virtual Machine Software: What is Virtualization?. (2019). Retrieved 5 October 2019, <https://www.vmware.com/solutions/virtualization.html>.

[11] P. Lourdes Salas, “El uso de software libre en la minimización de costos en centros de tecnologia de información en una universidad peruana ,” Universidad Nacional de Ingeniería Perú, 2014. Available at < http://cybertesis.uni.edu.pe/bitstream/uni/4372/1/salas_cp.pdf>.

[12] M. Jones, “La anatomía de un hipervisor Linux» “. IBM, May 2009. Available at: < https://web.archive.org/web/20160806114016/http:/www.ibm.com/developerworks/ssa/library/l-hypervisor/index.html>

[13] A. U. Rehman, R. L. Aguiar and J. P. Barraca, "Network Functions Virtualization: The Long Road to Commercial Deployments," in *IEEE Access*, vol. 7, pp. 60439-60464, 2019.

[14] What is Hypervisor and what types of hypervisors are there?. (2019). Retrieved 6 October 2019, from https://vapour-apps.com/what-is-hypervisor/

[15] Fundamental shifts in the virtualization market. (2019). Retrieved 6 October 2019, from https://www.redhat.com/en/blog/fundamental-shifts-virtualization-market

[16] Virtualization Market Share Report | Competitor Analysis | VMware vSphere, Microsoft Hyper-V, VMware ESX Server. (2019). Retrieved 6 October 2019, from https://www.datanyze.com/market-share/virtualization

[17] What is a Container? | Docker. (2019). Retrieved 6 October 2019, from https://www.docker.com/resources/what-container

[18] Docker overview. (2019). Retrieved 6 October 2019, from https://docs.docker.com/engine/docker-overview/

[19] V. Riccobene, "Deliverable D4.1 - Optimisation of virtualisation, orchestration, and resource allocation", 5G ESSENCE project, May 2018.

[20] Projects | Unikernels. (2019). Retrieved 6 October 2019, from http://unikernel.org/projects/

[21] notes, e. (2019). What is NFV: Network Functions Virtualization Basics » Electronics Notes. Retrieved 6 October 2019, from https://www.electronics-notes.com/articles/connectivity/data-networks/nfv-what-is-network-functions-virtualization-basics.php

[22] Mulligan, U. (2019). ETSI - Standards for NFV - Network Functions Virtualisation | NFV Solutions. Retrieved 6 October 2019, from https://www.etsi.org/technologies/nfv

[23] ETSI GS NFV 002 v.1.1.1, " Network Function Virtualisation (NFV); Architectural Framework", October, 2013.

[24] Khan, F. (2019). A Cheat Sheet for Understanding "NFV Architecture" – TelcoCloud Bridge. Retrieved 6 October 2019, from https://www.telcocloudbridge.com/blog/a-cheat-sheet-for-understanding-nfv-architecture/

[25] R. Chayapathi, at al. "Network Functions Virtualization (NFV) with a Touch of SDN", Addison-Wesley, 2017.

[26] ETSI GS NFV-MAN 001 v.1.1.1, " Network Function Virtualisation (NFV); Management and Orchestration", December, 2014.

[27] OSM. (2019). Retrieved 15 October 2019, from https://osm.etsi.org/

[28] ONAP. (2019). Retrieved 15 October 2019, from https://www.onap.org/

[29] Gartner Names HPE a Leader in Magic Quadrant for Operations Support Systems. (2019). Retrieved 15 October 2019, from https://www.hpe.com/us/en/newsroom/blog-post/2018/03/gartner-names-hpe-a-leader-in-magic-quadrant-for-operations-support-systems.html

[30] Research - OSM Public Wiki. (2020). Retrieved 29 January 2020, from https://osm.etsi.org/wikipub/index.php/Research

[31] Open Network Automation Platform (ONAP) architecture Overview The Linux Foundation, from https://wiki.onap.org/download/attachments/3245268/ONAP%20Architecture%20Overview%20_Final-1.pdf?version=1&modificationDate=1492047855000&api=v2

[32] Intel briefs. "Accelerating the Virtualized User Plane for 5G Core Network Readiness". Available at: < https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/accelerating-virtualized-user-plane-for-5g-core-network-readiness-brief.pdf>

[33] OSM Release FOUR - OSM Public Wiki. (2020). Retrieved 29 January 2020, from https://osm.etsi.org/wikipub/index.php/OSM_Release_FOUR

[34] vCPE Use Case Tutorial: Design and Deploy based on ONAP - Developer Wiki - Confluence. (2020). Retrieved 29 January 2020, from https://wiki.onap.org/display/DW/vCPE+Use+Case+Tutorial%3A+Design+and+Deploy+based+on+ONAP#vCPEUseCaseTutorial:DesignandDeploybasedonONAP-VNFOnboarding

Glossary

HPE: Hewlett-Packard Enterprise

NF: Network Function

VM: Virtual Machine

NFV: Network Function Virtualization

CAPEX: Capital expenditure

OPEX: Operational expenditure

RAN: Radio Access Network

IT: Information Technology

MEC: Multi-access Edge Computing

NG-RAN: Next Generation-Radio Access Network

NFV-MANO: NFV Management and Orchestration

OSM: Open Source Mano

ONAP: Open Network Automation Platform

HPE NFVD: HPE Network Function Virtualization Director

UE: User Equipment

eNB: E-UTRAN Node B

S-GW: Serving Gateway

MME: Mobility Management Entity

P-GW: Packet Data Network (PDN) Gateway

5GC: 5G Core Network

CN: Core Network

LTE: Long Term Evolution

EPC: Evolved Packet Core

CUPS: Control and User Plane Separation

NRF: Network Repository Function

PLMN: Public Land Mobile Network

UPF: User Plane Function

SMF: Session Management Function

AMF: Access and Mobility Management Function

PCF: Policy Control Function

DN: Data Network

QoS: Quality of Service

LADN: Local Area Data Network

AF: Application Function

NEF: Network Exposure Function

UDM: Unified Data Management

AUSF: Authentication Server Function

NSSF: Network Slice Selection Function

UI: User Interface

VMM: Virtual Machine Monitor

I/O: Input/Output

OS: Operating System

API: Application Programming Interface

OTT: Over The Top

NFVI: Network Function Virtualization Infrastructure

VNF: Virtual Network Function

EM: Element Manager

VNFM: VNF Manager

VIM: Virtualized Infrastructure Manager

NFVO: NFV Orchestrator

CSP: Cloud Service Provider

ICT: Information Communication Technologies

CPE: Customer Premises Equipment

HSS: Home Subscriber Service

GUI: Graphical User Interface

DC: Data Center

VDC: Virtual DC